DESIGN AND IMPLEMENTATION OF WORKFLOW FOR CONTENT MANAGEMENT SYSTEM

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Kanika Maheshwari
Spring 2013
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Kanika Maheshwari

Design and Implementation of Workflow for Content Management System

Carl Eckberg, Chair
Department of Computer Science

Alan Higgins
Department of Computer Science

Steve Kirschvink
Department of Mathematics and Statistics

Feb 25, 2013
Approval Date
DEDICATION

This thesis is dedicated to my parents for the love, support and encouragement provided since the beginning of my studies.
A Content Management System (CMS) is a software application that allows users to create dynamic web pages for easy viewing, editing, and publishing information such as documents and other digital assets in a transparent way. The author is responsible for creating the content, having it reviewed by the editor for content tuning and adjustment, and finally the publisher or administrator publishes the content or rejects the content. The workflow aims to establish the process and the roles within different departments. The critical aspect of CMS is to maintain and manage version history for the contents, providing the user of CMS the ability to revert back to an earlier version. In 2008, a custom content management system was deployed called Toolbox, which can be accessed via https://newscenter.sdsu.edu/toolbox/. The toolbox provides an interface to help departments develop websites.

The objective of the thesis project is to modify and improve this CMS, particularly with regard to the version history feature and other aspects of workflow. In addition to the critical version history feature, email notification, easy management for updating user details, and department user roles will be discussed.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>x</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Overview of Existing CMS</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Working of Existing CMS</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Functions of Existing CMS</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 Limitations</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Motivation</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Introduction About Workflow</td>
<td>6</td>
</tr>
<tr>
<td>1.3.1 Functions Added to Existing CMS</td>
<td>7</td>
</tr>
<tr>
<td>1.3.2 Functions of Workflow</td>
<td>9</td>
</tr>
<tr>
<td>1.3.3 Working of Updated CMS</td>
<td>10</td>
</tr>
<tr>
<td>2 TOOLS AND TECHNOLOGIES</td>
<td>12</td>
</tr>
<tr>
<td>2.1 C# (Programming language) and ASP.NET (Framework)</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Microsoft SQL Server 2008 (Database)</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Cascading Style Sheets</td>
<td>14</td>
</tr>
<tr>
<td>2.4 Microsoft Visual Studio Express 2012</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Comparison of CMS With Open Source CMS Tools</td>
<td>15</td>
</tr>
<tr>
<td>3 OVERVIEW OF DESIGN</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Database Architecture</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Stored Procedures</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Class Diagrams</td>
<td>22</td>
</tr>
<tr>
<td>3.3.1 User Management</td>
<td>22</td>
</tr>
<tr>
<td>3.3.2 Email Notification</td>
<td>23</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1.</td>
<td>Version History of Content Updates</td>
<td>19</td>
</tr>
<tr>
<td>Table 3.2.</td>
<td>Version History of Layout Updates</td>
<td>20</td>
</tr>
<tr>
<td>Table 3.3.</td>
<td>Toolbox Users Information</td>
<td>21</td>
</tr>
<tr>
<td>Table 3.4.</td>
<td>Department User Roles Information</td>
<td>21</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1. Existing CMS overview. .................................................................2
Figure 1.2. Existing CMS flowchart. ...............................................................3
Figure 1.3. Existing CMS home page and user management .......................4
Figure 1.4. Existing CMS department page ...................................................4
Figure 1.5. Updated CMS functionality snapshot 1. ....................................7
Figure 1.6. Updated CMS functionality snapshot 2 ....................................8
Figure 1.7. Updated CMS functionality snapshot 3 ....................................8
Figure 1.8. Updated CMS functionality snapshot 4 ...................................9
Figure 1.9. Updated CMS overview ............................................................10
Figure 1.10. Updated workflow flowchart .................................................11
Figure 3.1. User management class diagram .............................................23
Figure 3.2. Email notification class diagram .............................................24
Figure 3.3. Department user roles class diagram .....................................25
Figure 3.4. Versioning class diagram .........................................................26
Figure 3.5. Page layout history class diagram .........................................27
Figure 3.6. Sub-versioning class diagram ...............................................28
Figure 4.1. Toolbox login page .................................................................30
Figure 4.2. Add user ..................................................................................30
Figure 4.3. Update user ...........................................................................31
Figure 4.4. Delete user ............................................................................31
Figure 4.5. Add user permissions for department ...................................32
Figure 4.6. Setting department user roles ...............................................34
Figure 4.7. Version history of content for department ..............................35
Figure 4.8. Versioning at page level for department ..................................36
Figure 4.9. Version history of layout for department ...............................37
Figure 4.10. Email notification to departments .........................................38
Figure 4.11. Email notification to admin ..................................................39
ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude to the CMS team at San Diego State University. Especially, indebted to Aaron Hoskins, Martin Murphy and Joel Avendano for their invaluable suggestions, encouragement and constant guidance which has contributed a great deal in the success of the thesis. In spite of their busy schedules, they participated in the discussions and presentations about the thesis, offering valuable suggestions regarding the quality and the standard of the project.

Prof Carl Eckberg, gave me the opportunity to work on this project and make contribution to the system. Lastly, a special thanks goes to other committee members, Prof Alan Riggins and Prof Steve Kirschvink for their guidance and advice on all aspects of this thesis.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF EXISTING CMS

A Content Management System (CMS) is a software that provides users with tools to manage and control the development of website in an organization. It allows for creating, modifying and maintaining a website in a centralized form while empowering them to publish high quality content onto web [1].

A Content Management System enables the users who are familiar with regular word processing to create and publish the content on the internet themselves, without the intervention of specialists. This feature has accelerated the publishing process. However, the CMS has to perform a wide array of business processes to accomplish these goals. Implementing a good workflow algorithm is a key feature in developing a CMS.

The CMS has the following parts:

- Administrator of CMS: admin of a content management system manages the departments and users who will be using the application to create their websites. The department contacts the administrator to get permission to create their department’s website, and then new users can login to create and edit contents for the website. Only the administrator has the ability to remove or add a new template (i.e. design strategies and styles for creating web pages). Only the administrator can add or remove the departments/users. Figure 1.1 shows the various functions of an administrator module.

- Front End: the existing users can create and update the content for the website using the CMS user interface. It is an easy to use interface, which allows the non-technical people to enter or edit the content without having HTML knowledge.

- Business Logic or Controller: the business logic serves as an interface between the front-end and the database. For example, information retrieved from the database can be processed and formatted in special ways before being sent back to the client.

- Database or Backend: a database is the key to Content Management Systems. The database acts as a central repository for retrieving the content, templates, graphics, users and metadata. A true CMS does not store flat, static HTML pages. Instead, the system places the content in a relational database capable of storing a variety of binary data and text. The database is capable of storing the author info as well as the creation, publication and deletion dates for the department or the web pages added to the department.
Figure 1.1. Existing CMS overview.

- Templates: templates are the structures that format and display content following a request from a user for a particular web page. Templates ensure a consistent, professional look and feel for all the content on the site. The templates represent the skeleton of a web page where the content of the web page is embedded into it. A master template is chosen for the department’s website by the editor; however the individual web pages can also maintain their own sub templates.

- Permissions: the administrator of the CMS holds the authority to create the new department, add users and add site admin for new department.

A Computer Science Department can easily create its own website, but a Literature Department might find this troublesome. The departments may want to maintain their website, authorize the individual faculty members to manage their own web pages, create the class schedules, post assignments, midterms and project deadlines. They can maintain interactive discussion forum with the students of class. All of these features are motivation for the development of the Content Management System. It is designed as per the requirements of the various departments and organizations at SDSU.

1.1.1 Working of Existing CMS

All new departments contact the administrator to get authorization for the department and various users to use the CMS, as a preliminary step. The administrator creates the department and new toolbox users; the new users are then given permission to be site admins to manage the website for the department using the content management system. The site admins of the website maintain the website, edit, manage and publish the content. Generally departments opt to have one site admin. Figure 1.2 explains the working of the current CMS and the steps a department has to follow to maintain their website.
1.1.2 Functions of Existing CMS

The functions of the Content Management System are -

- Access control: user groups allow to control how registered users interact with the site. A website can be restricted to one or more groups (shown in Figure 1.3).
- Easily editable content: editing and manipulation of content is easier once the website design is separated from content.
- Scalable Expansions: the CMS has the ability to serve as many departments as needed (shown in Figure 1.4).
- Collaboration: the application acts as a collaboration platform, which allows content to be retrieved by more than one authorized user.
- Automated Templates: an automatic template, once created, can be applied to either new or existing content [2].
1.1.3 Limitations

Even though the existing CMS gives users the capability to upload and edit their content automatically, various steps in the process are manual:

- The CMS admin does not have contact information for users of the department in case he wants to communicate with them regarding any inappropriate usage of the CMS toolbox to upload their content. The admin has to determine the department users and contact them by phone or over email which is recorded separately outside the realm of the CMS system.

- The Admin is not aware of distribution of the responsibilities of creation, publication and editing among the department.

- The department’s site admin had to manually set the roles and responsibilities for creating the website for the department using the CMS rather than automatically setting it using the workflow in CMS.

- While creating the content for the department, there are a series of CREATEs, UPDATEs and DELETEs; if the user deletes some content or a web page by mistake,
they have no way to retrieve it as the page or image is permanently deleted from the department folder. That is, in case of any deletion of content by mistake, the department will not be able to restore the web page back to the previous version. They will have to manually replace the data, which can be time consuming in case of large information.

1.2 MOTIVATION

The Content Management System is a set of processes, which manages, edits and publishes the content in any form of media. We can create the content, but if we do not have the process to sustain it and manage it then our strategy will fail. It is necessary that we should establish the roles and responsibilities for the people using the CMS. The departments need the following functionality, which contributes to the need for a content management system, and they are as follows:

- Page history or versioning for the changes made in a particular department’s content.
- Change management, which gives us the ability to revert back to the previous version if needed.
- Establishing the users and editing the user details. The CMS should have the contact info of users and the ability to edit user details if needed.
- Assignment of the roles and responsibilities of the department CMS users before creating the website.
- Sending notification via email from the administrator to the department user with a particular role or communications between department users playing different roles.
- Exception handling and error processing.

The different departments within the university use the CMS system of the San Diego State University. The CMS is used by the departments to set up their own websites. Without the workflow, it is difficult for the departments to interact with the administrator because:

- They have to frequently contact the administrator for help in setting up the website, so instead of manually keeping track of contact details, they need to be able to email the administrator from within the CMS.
- Once the roles and responsibilities of the departments are set, a user can send email notifications within the department to notify the other appropriate users. For example, the editor of Aztec shops can send an email notification to the publisher that the content is ready to be published.
1.3 INTRODUCTION ABOUT WORKFLOW

“Workflow is a term used to describe the tasks, procedural steps, organizations or people involved, required input and output information, and tools needed for each step in a business process” [3].

Workflow aims at the enforcement of the processes rather than the document [3]. It may be seen as an abstract model representation of real work. The flow is a series of sequential steps defined for finalizing or updating the document.

Closely related are the concepts used for describing organizational structure, such as silos, functions, teams, projects, policies and hierarchies. “Workflows may be viewed as one primitive building block of organizations” [4].

A workflow management system automates the paper work required to enforce business processes. The workflow requires notifying the individual for the next task on the completion of the current task. This is the concept of dependencies. A workflow management system reflects the dependencies required for the completion of each task [4].

Workflow for the CMS which we have implemented is as follows:

- Workflow solutions in which users interact.
- Use of predefined sequences to guide and control the processes.
- A system that completely defines, manages, and executes “workflows” through the execution of software, whose order of execution is driven by a computer representation (process definition) of the workflow logic.

The process of publishing content can be complex. Typically, participants pass content between the authors, developers, reviewers, editors, translators, approvers, and publishers. However, workflow automation in a CMS is not solely about using software to facilitate the transfer of content from person to person, but also about tracking and recording the progress of an activity, delivering the work to appropriate users.

The basic roles and responsibilities, which I have set up for the CMS at SDSU are:

- Author: responsible for creating and modifying content.
- Editor: responsible for tuning the content message, setting the style and layout of a department’s website
- Publisher: responsible for releasing the web content for use.
- Site Admin: responsible for setting the roles and responsibilities within the department and monitoring the progress of the departments page [5].
Administrator of CMS: responsible for managing and monitoring the content uploaded by the departments using the CMS.

1.3.1 Functions Added to Existing CMS

The functionality added to the existing content management systems by this thesis are:

- **Versioning:** Versioning is useful for content creating and updating over time, and it may be necessary to go back to or reference a previous copy (Shown in Figure 1.5 and 1.6).

- **Delegation:** CMS software allows the restricted access for the users over the website content and distributing the responsibility of content management (Shown in Figure 1.7).

- **Workflow Management:** Workflow is the process of creating a series of sequential steps defined to accomplish tasks in the CMS. For example, the author can submit a story, but it is not published until the copy editor cleans it up and the publisher approves and publishes it [2].

- **Email Notification:** CMS software allows the departments to communicate within the department or with the administrator of the content management system by using email updates (Shown in 1.8).

- **User Management:** CMS software now requires obtaining email addresses when creating new users; a user can contact the administrator to update the email address (Shown in Figure 1.7).

![Figure 1.5. Updated CMS functionality snapshot 1.](image)
Figure 1.6. Updated CMS functionality snapshot 2.

Figure 1.7. Updated CMS functionality snapshot 3.
1.3.2 Functions of Workflow

Some of the basic functions of the workflow for content management system include the establishment of users, assignment of work items, exception and error processing, event generation and notification, auditing, tracking and reporting of results, tracking and reporting of statistics, versioning and change management. Versioning is the most critical aspect of content management; it manages the versions of the content as the website content gets updated. Authors and editors often need to restore older versions of edited products due to a process failure or an undesirable series of edits. It is also important to keep track of the layout used for the CMS; a user might prefer using the old template with the new content. There is also page template versioning, which keeps track of the layouts changed by the department and allows the department to restore the previous layout. Another equally important aspect of content management involves the creation, maintenance, and application of review standards. Each member of the content creation and review process has a unique role and set of responsibilities in the development and/or publication of the content. Each review team member requires clear and concise review standards, which must be maintained on an ongoing basis to ensure the long-term consistency and health of the knowledge base. As the roles are being assigned, each member of the team is sent an email about the roles they have been assigned for the department.

Figure 1.9 shows the updated functionality of the administrator. The existing users can update their contact details with the administrator. The communication between the departments and administrator can be established using the email notification feature.
1.3.3 Working of Updated CMS

All new departments contact the administrator to get the department and user created as a preliminary step. The administrator creates the department and new toolbox user, the new user is then given the permissions as the site admin to manage the website for the department using the content management system. The content management system starts maintaining the version history for content and page template updates for the department. The site admin specifies the user roles and permissions for creating, editing and publishing the content. The author creates the content and sends the email notification to editor that content is ready for being formatted. The editor does the formatting of the content and notifies the publisher that website content is ready to be published. The publisher either publishes the contents or rejects the content and sends the notification back to author or editor. Figure 1.10 explains the working of updated CMS and the steps a department has to follow to maintain their website.
Figure 1.10. Updated workflow flowchart.
CHAPTER 2

TOOLS AND TECHNOLOGIES

This chapter furnishes details about the main technologies used in the design and implementation of the project.

2.1 C# (PROGRAMMING LANGUAGE) AND ASP.NET (FRAMEWORK)

Started within its .NET initiative for Common Language Runtime [6], C# has now become one of the most popular of all .NET languages. Described as “a simple, modern, object-oriented, and type-safe programming language” [6], it provides developers with rapid application development tools combined with the power of C/C++. Lead by Anders Hejlsberg at Microsoft, the language was initially called Cool, which stood for ‘C like Object Oriented Language’ [7]. Later on it was changed to C# (pronounced C sharp) as it was above C-style languages and “#” represented a semi-tone above the note in music theory. Supporting features like exception handling, multiple types of polymorphism, separation of interface and implementation, generics and backed by an extensive class library, C# is well suited for developing various types of software’s including internet applications.

ASP.NET is a compiled, server-side web-application framework designed for .NET. It is used to build dynamic and enterprise-class websites. Built on the Common Language Runtime, applications can be developed using any of the compatible .NET languages including, Visual Basic .NET, C#, and JScript .NET. The framework provides powerful features such as state management, debugging support, extensible designer environment, etc. making it an excellent choice for developing the CMS [8].

Unlike static websites, ASP.NET pages contain extra directives or special instructions that tell the framework how to process the page. When a web browser requests a web page, the request is forwarded from the web-server (IIS) to the framework depending on the page extension. ASP.NET is used to process .aspx pages where as .html or .htm are not [9]. Based on the optional @Page directive, the final web page document is generated and is sent back to the client where it is shown in the browser.
2.2 MICROSOFT SQL SERVER 2008 (DATABASE)

According to Microsoft's VP of Server Application, Paul Flessner, the best advantage of using SQL Server 2008 is that it maintains the data through its self-tuning technology for support of various structured and semi-structure data [10]. This allows content management systems to leverage a great deal of data formats and types such as XML, spatial data, email, time/calendar, files, and digital media such as images, video, and sound. According to Chad Boyd, a known SQL Server architect in the IT industry, the features within SQL Server 2008 have benefits across a wide range of roles for both IT and non IT users [11]. Some of these benefits include security, availability and reliability, performance, and reporting.

In terms of performance tools, the greatest asset for a DBA is the activity monitor within SQL Server 2008 that allows for real-time information of processes such as SQL statements, packages, endpoints, and other various components in visual charts and drill down reports [12]. DBA’s can pause and take a snapshot of performance for a baseline or to investigate further without data being overwritten. Other great solution is the data compression capabilities that offer more efficient ways of data storage than traditional means, through the vardecimal option [11]. Available features include supporting pluggable CPUs – addable automatically, as well as database mirroring to allow for compression of mirror streams, and increased performance output of the database engine through automatic page-level repair.

Security is very important especially governing student information systems integrated at SDSU such as content management systems. Thus, it is imperative that the database management system should have powerful capabilities in security. SQL Server 2008 provides these functionalities such as transparent data encryption and external key management. Transparent data encryption allows for encrypting the data on the disk while being hidden from other applications, while key management keeps a handle on permissions within integration of applications such as with Crystal Reports and Microsoft Visual Studio Express for the web [11]. However, Crystal Reports is not the only means of getting information to end users within the content management system; SQL Server 2008 provides reports through reporting services deployment. Reporting services deployment does not require IIS to run reports that provide rich-text and visualization via graphs while being rendered through MS Word or Excel for data mining. In conclusion, SQL Server 2008
provides the best database management tool for the content management system through its security, availability and reliability, performance, and reporting [11].

### 2.3 Cascading Style Sheets

“CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers” [13]. It is a declarative language to describe styles to be applied to the presentation of elements. We use this in expert capacity to determine the kinds of devices, the UI requirements, and then write new rules for display of new and novel content. One of the important feature of using CSS is that it allows styling to be completely separate from content.

Although it is designed to be used with any XML language, in practice it is used with HTML and XHTML. CSS works by overriding browsers default style settings on how it interprets the tags and their display method. It lets you use any HTML element indicated by an opening and closing tag to apply style attributes defined either locally in the HTML/XHTML document or in a style sheet. “Style Sheets contain rules, composed of selectors and declarations that define how styles will be applied. The selector (a redefined HTML element, class name, or ID name) is the link between the HTML document and the style” [14].

### 2.4 Microsoft Visual Studio Express 2012

According to Microsoft, Visual Studio Express 2012 for web (VSW) empowers developers and beginners with the capability to create full n-tier web applications and sites with ease. VSW allows for a quick and easy way to implement a development environment. Before VSW can be installed, .NET Framework and IIS Express must be on the local system. Nonetheless, capabilities described below enable developers to create complex web applications.

Websites & Web application deployment provides three methods to create a development environment. These include file system, local IIS, and ftp-deployed web applications/sites. The file system allows for local or network share of storing source files while using IIS express to create a web server to run the web application. The local IIS
creates a web server on the local system while the ftp deployments let developers make immediate changes to web pages and publish web applications.

Web Pages & Server controls components are powerful in the capability of developing dynamic web applications/pages in ASP.NET. There are two primary views: source and design. In the source mode, IntelliSense and other tools allow for easy lookup on namespaces, attributes, functions, and other components while creating well-structured code and validating against schemas. On the other hand, in the design view the abilities allow for edits within HTML code, CSS, and visual layout through ASP.NET and HTML web controls [15].

2.5 COMPARISON OF CMS WITH OPEN SOURCE CMS TOOLS

There are numerous open source and commercial web based content management systems available in the market with WordPress, Joomla and Drupal considered the top-3 [16]. Joomla and WordPress are all developed using PHP programming language and only official support for MySQL database is provided. In many cases, use of SQL Server is possible by setup of external modules [17]. Even though WordPress has made big strides in becoming a fully-fledged CMS, it is primarily geared towards blogging websites with basic CMS features added as an afterthought. While WordPress is more popular for blogging websites. Joomla has more CMS features and flexibility than WordPress. Drupal is a lot less friendly than WordPress and Joomla to set up, configure and maintain. The site administration is also very complex and not very user friendly, especially for a non-technical person. The department will need an advanced programmer in order to customize and add features [18].

The main purpose of a content management system is to make the website development simple and easier to maintain. These are lots of CMS’s available in the market, but the harder part is deciding which content management system is more suitable for our requirements and budget. For using an open source CMS one has to research on the CMS or get feedback from the experienced programmers who have contributed to these tools and can help in configuring them according to our needs.

The existing Content Management System, first deployed at the campus server in 2008, has been developed from scratch and does not use any existing CMS creation tools.
One of the main reasons for developing it in-house is so that we can implement it based on requirements and functionality of departments and organizations across San Diego State University. This thesis aims to contribute in adding critical features like versioning and workflow to enhance its usability and management.
CHAPTER 3

OVERVIEW OF DESIGN

This chapter discusses the design of the various features implemented as part of the workflow development.

3.1 DATABASE ARCHITECTURE

Database can be defined as “A database is a collection of information that is organized so that it can easily be accessed, managed, and updated” [19]. There has been ongoing research on how to effectively store data from 1960s and presently, there are more than 10 different types of databases [20]. Even with the plethora of options, most content management systems use traditional (or relational) model because it provides a reasonable and affordable choice for most. Based on concept of relational algebra, it represents the dataset in tabular format with each table representing an entity and each column in the table representing it properties or attributes. Our selected database tool MS SQL Server falls under traditional database category. MS SQL Server allows use to easily retrieve data using ODBC/SQL and displayed in websites developed using standard ASP.NET [21]. Traditional database excel and are beneficial when the data is strictly structured and less likely to change. For example, the structure of table JOBS in the traditional SQL approach is

<table>
<thead>
<tr>
<th>JobId</th>
<th>JobTitle</th>
<th>CompanyName</th>
<th>JobDescription</th>
</tr>
</thead>
</table>

Now if the Human Resources (HR) needs to add more details e.g. salary paid for the job, we will have to perform multiple steps to have a correct working system:

- Add a new column *JobSalary* [21].
- Modify the sql queries for saving salary.
- Modify the code used for displaying the salary.
The second approach is to save the information in form of XML, and saving it in the database as XML type attribute field. Adding a new element, does not change the structure of the database table, it is always mapped to XML data with a new or updated tag element. If we were using XML approach, the above example will just necessitate updating one row of XML data with the added attribute details.

For example, the structure of the JOB table using XML will be:

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobId</td>
</tr>
<tr>
<td>JobDocumentXML</td>
</tr>
</tbody>
</table>

The initial XML document is as follows:

```xml
<Job>
    <JobTitle>Software Engineer</JobTitle>
    <CompanyName>Amazon</CompanyName>
    <JobDescription>Software developers or software developers who focused on imaging / vision.</JobDescription>
    <JobLocation>Seattle, WA</JobLocation>
</Job>
```

If the HR manager wants to publish the salary for the job, then the XML document will be updated as follows:

```xml
<Job>
    <JobTitle>Software Engineer</JobTitle>
    <CompanyName>Amazon</CompanyName>
    <JobDescription>Software developers or software developers who focused on imaging / vision.</JobDescription>
    <JobLocation>Seattle, WA</JobLocation>
    <JobSalary>95000</JobSalary>
</Job>
```

The existing CMS uses the traditional SQL approach to store the content of the departments website. The content is distributed across different tables in database. The
information about the different departments is stored in the DEPARTMENTS table. Appendix A is an example of DEPARTMENTS and PAGES table used for traditional SQL.

This thesis does not aim to alter the traditional SQL database structure used by present CMS. The XML approach is used in implementing the new feature of versioning or change management for the CMS. The new tables CONTENTSUPDATEHISTORY and PAGE_HISTORY save the updated content as XML format in the Contents and Layout_Contents fields. Appendix B shows the example of updated contents stored as XML in database.

The CONTENTSUPDATEHISTORY table is added to store the information of the contents updated by all the departments in content management system. The attribute Contents is the XML and it stores the content updated for the department in XML format, VersionId stores the versions for content updated for the specified DepartmentName. The VersionId is unique for each department but not for table as a whole. The attribute ArtifactVersionId refers to the version at the page level for the content update of department. The ArtifactVersionId is unique for a particular table but not for whole department (see Table 3.1).

<table>
<thead>
<tr>
<th>CONTENTSUPDATEHISTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>VersionId</td>
</tr>
<tr>
<td>ArtifactVersionId</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Comment</td>
</tr>
<tr>
<td>Author</td>
</tr>
<tr>
<td>Contents (XML)</td>
</tr>
<tr>
<td>DepartmentName</td>
</tr>
<tr>
<td>TableName</td>
</tr>
<tr>
<td>TableAction</td>
</tr>
</tbody>
</table>

The PAGE_HISTORY table is added to store the information of the layout updated by all the departments in content management system. The attribute Layout_Content is the
XML and it stores the content updated for the department in XML format, VersionId stores the versions for content updated for the specified DepartmentName. The VersionId is unique for each department but not for table as a whole. The attribute Layout_Name stores the template updated by the department in the current version. The PAGE_HISTORY stores the author and date for the specified version (see Table 3.2).

**Table 3.2. Version History of Layout Updates**

<table>
<thead>
<tr>
<th>PAGE_HISTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>VersionId</td>
</tr>
<tr>
<td>DepartmentName</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Comment</td>
</tr>
<tr>
<td>Author</td>
</tr>
<tr>
<td>Layout_Name</td>
</tr>
<tr>
<td>Layout_Content (XML)</td>
</tr>
<tr>
<td>TableName</td>
</tr>
<tr>
<td>TableAction</td>
</tr>
</tbody>
</table>

The TBXUSERS table existed in the present CMS; it stores the details of the toolbox users. One more point to note is that the current user management functionality does not store the email information for the user. The field is present for historical reasons and the value for all the rows is set to NULL. Along with the ID field, username attribute is also unique to the user. This is the user facing value that is used during logging in to the system. The attribute SiteAdmin determines if the user is the administrator of CMS. Other attributes of the table, keeps record of the date when user is created, deleted when they last linked to toolbox in the three Linked* columns (see Table 3.3).

The DEPARTMENTUSERROLES table is added to store the department user roles with DepartmentId as primary key. The attributes CreatedBy and UpdatedBy store the username of user creating of updating the department user roles, created on and updated on store the date when the department user roles are created or updated. The attributes Author, Editor and Publisher store the usernames of the users specified by site admin (see Table 3.4).
### Table 3.3. Toolbox Users Information

<table>
<thead>
<tr>
<th>TBXUSERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (Primary Key)</td>
</tr>
<tr>
<td>Username</td>
</tr>
<tr>
<td>SiteAdmin</td>
</tr>
<tr>
<td>Password</td>
</tr>
<tr>
<td>Firstname</td>
</tr>
<tr>
<td>Lastname</td>
</tr>
<tr>
<td>Email</td>
</tr>
<tr>
<td>Phone</td>
</tr>
<tr>
<td>LinkedBy</td>
</tr>
<tr>
<td>LinkedOn</td>
</tr>
<tr>
<td>LinkedTo</td>
</tr>
<tr>
<td>LoginIp</td>
</tr>
<tr>
<td>CreatedBy</td>
</tr>
<tr>
<td>CreatedOn</td>
</tr>
<tr>
<td>DeletedBy</td>
</tr>
<tr>
<td>DeletedOn</td>
</tr>
</tbody>
</table>

### Table 3.4. Department User Roles Information

<table>
<thead>
<tr>
<th>DEPARTMENTUSERROLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DepartmentId (Primary Key)</td>
</tr>
<tr>
<td>DepartmentName</td>
</tr>
<tr>
<td>Author</td>
</tr>
<tr>
<td>Editor</td>
</tr>
<tr>
<td>Publisher</td>
</tr>
<tr>
<td>CreatedBy</td>
</tr>
<tr>
<td>CreatedOn</td>
</tr>
<tr>
<td>UpdatedBy</td>
</tr>
<tr>
<td>UpdatedOn</td>
</tr>
</tbody>
</table>
3.2 STORED PROCEDURES

Stored Procedures (SP), are set of SQL statements that are compiled and stored in data dictionary of database allowing it to be shared across multiple programs [22]. While SQL is a standard, SP is tied to the database vendors but is available in most of the major systems. A stored procedure is a prepared SQL code that we can save, so that the code becomes reusable. If we have a reusable query in code, instead of rewriting the query each time, we can save it as a stored procedure and then just call the stored procedure to execute the SQL code that you saved as part of the stored procedure. All of these features make them valuable enough to be included into the project.

Apart from running the same SQL code over and over again, we also have the ability to pass parameters to the stored procedure, so depending on the needs, stored procedure acts as per the parameter values that are passed [23].

In our project, we use stored procedures to update and restore (select) back data from the version. Appendix C gives the code for the stored procedure used in the project.

3.3 CLASS DIAGRAMS

The class diagrams for the features implemented for workflow are explained in the following sections.

3.3.1 User Management

Users with the administrator permissions can manage other users. The permissions to view and edit the user details are given to the administrator of Content Management System (see Figure 3.1). “Home.aspx” page is displayed when administrator logs into toolbox. On clicking the “Manage Users” button, it loads the “User_Management.aspx” page. The user gets the option of:

- Add New User: the “OnClick_AddUser” function creates a new user and saves the details of the user into TBXUSERS table. The new toolbox user requires a unique username, password and a valid email address.
- Update User Contact Info: the “OnClick_UpdateEmail” function is to:
  - Update the email address of the existing toolbox users whose contact information has not been saved.
  - Update the email address of toolbox users to the recent one.
3.3.2 Email Notification

The departments and administrator can establish effective communication using email notification. It also helps in increasing the communication within the departments. This functionality is also used to schedule a new event (see Figure 3.2). “Home.aspx” page is displayed when a toolbox user logs into the system. Clicking on the “Notification” button loads the “Notify_User.aspx” page. The user gets two options for sending the notification:

- Notification to the Administrator of CMS: the “submitAdmin” function is called when user selects the “Notification to Administrator” tab, selects the department, user roles, updates the subject, writes the detailed message and clicks submit. The “SendMail” function in the Email.cs class sends the email notification with the message to administrator’s inbox.

- Notification within the Department: the “submit” function is called when user selects the “Notification to departments” tab, adds department, user roles, adds other toolbox user to be updated about the message, updates the subject and writes the detailed message and clicks submit. The “SendMailMultipleRecipients” function in the Email.cs class sends the notification with the message to:
  - Department user role selected in “Notify To” list.
  - Toolbox users added in “Notify Others” textbox.
  - Sends the notification to administrator of CMS.

- The “Notification to Departments” tab can also be used by the administrator of CMS to contact the departments. It gives an option to the administrator to select the user role as per updates or the problem he wants to discuss.
3.3.3 Department User Roles

The new departments need to set the department user roles after the department is created and the permission to read, write and edit is given to site administrator of the department. The permission to edit and update the content for the department is denied till the author, editor and publisher for the department is specified. Once the roles are defined, automatic email notifications are sent to the users with the specified roles and the departments (see Figure 3.3). The site admin accesses the “Home.aspx” page after logging into the toolbox; the department’s settings page is loaded on clicking the department folder. The user can add or update the permissions by selecting the “Department User Roles” tab:

- Author toolbox username, editor toolbox username and publisher toolbox username are blank in case the department roles have not been setup. Site admin adds the department roles and clicks submit button. The “OnClick_SaveDepartmentRoles” function checks the database and inserts new row in DEPARTMENTUSERROLES table by calling “addDepartmentUserRoles” function and sends an email to the specified users.

- The editor, publisher and site admin have the permission to update the existing department roles. On clicking submit, the “OnClick_SaveDepartmentRoles” function checks the database and updates existing row in DEPARTMENTUSERROLES table by calling “updateDepartmentUserRoles” function.
3.3.4 Change Management

The content management system aims to separate the layout from the content. Change management also separates the content versioning from layout versioning. The user can have old content with new layout.

3.3.4.1 Versioning of Content

Versioning of the content saves update history for the following:

- Page updates like add a new page, update existing page, delete a page.
- Restore to the previous version in case of page is deleted.
- Content history for adding, updating or restoring department roles to previous version.
- Content history for adding or deleting of images.
- Restore the deleted image.

The “User Permissions” table is updated when the department roles are added or updated.
Content history updates for adding, updating or restoring the linking groups.

The content updated for that particular version is being saved in the form of xml in CONTENTUPDATEHISTORY table. The “SaveContents” function saves VersionId, ArtifactVersionId, Author, Date, Comment, Contents (in form of xml), TableName (the database table affected by that update), TableAction (i.e. if the row is created, updated or deleted in that particular version). Figure 3.4 shows the class diagram for versioning.

**Figure 3.4. Versioning class diagram.**

### 3.3.4.2 VERSIONING OF PAGE LAYOUT

The “Page Template Version History” link button on the “Department Settings” tab manages the change of the layout for the department (see Figure 3.5). The departments can roll back to the previous layout if the present layout is not suitable with the contents:

- To view the layout history of the department, the user clicks “Page Template Version History”. It displays different layout updated for the department when “Page_History.aspx” page is loaded by taking the data from PAGE_HISTORY database table.

- Clicking “Restore” for that particular version can do the restore to previous layout. The function “OnClick_Restore” is called and it restores the layout settings in DEPARTMENTS table of the database.
3.3.4.3 VERSIONING AT PAGE LEVEL

Version history is implemented at page level (see Figure 3.6). Also the content history is updated when user updates the contents of the particular page or tab. Page level versioning is useful to the authors while creating, updating and managing the content for web pages. The “Page Content History” link button at the bottom of the page gives user the chance to view and restore the contents on that page instead of searching it in the department level content updates in “Department Settings” tab. The database table associated with the versioning is CONTENTUPDATEHISTORY, the column ArtifactVersionId manages the updates at page level.
Figure 3.6. Sub-versioning class diagram.
CHAPTER 4

APPLICATION OVERVIEW

This chapter gives an overview of the features implemented for workflow of Content Management System. The current thesis aims to meet the following requirements:

- Allows the administrator to create the new users.
- Allows the administrator to update the contact information of existing users.
- The roles for the departments i.e. author, editor and publisher are specified by the site admin and their permissions are updated automatically.
- Author has the permissions to create, edit and manage the content. They cannot change the department settings to set the layout and publish the content.
- The author, editor and publisher are notified via email about the roles they have been assigned.
- Editor can set the layout and style for content but they cannot publish the website.
- The publisher has the permission to monitor the website, accept or reject the content updated by author or editor.
- The email notification is used to communicate within the department or between the department and administrator.
- Version history for the contents and page layout updates, is maintained.

4.1 INTRODUCTION ABOUT TOOLBOX

Toolbox is a tool for adding, editing and managing the content in the content management system at SDSU. The Toolbox is used for building and maintaining websites for departments across university. This is the last stage when we have already done the following:

- Establish the solid Information Architecture (IA).
- Edited your content according to the IA; edit it according to the web standards and SDSU tutorial guidelines [24].

The toolbox login page looks like Figure 4.1.

4.2 USER MANAGEMENT

The first step is to create an account to access the toolbox and edit the content. The feature has certain security restrictions for user management in that only administrator is
allowed to create new users and edit the user information. Normal users do not have any permission to create or update data about users. If a normal user needs to update their information, they will need to contact the admin and ask him to perform the changes. To create a new account user needs a valid email address, unique username and the password. The toolbox username created is of the format tbx.<username>. The toolbox username and password will be required for the subsequent logins. Figure 4.2 shows the add user page. Figure 4.3 shows the update user page. Figure 4.4 shows the delete user page.

Figure 4.1. Toolbox login page.

Figure 4.2. Add user.
Figure 4.3. Update user.

Figure 4.4. Delete user.
4.3 Creating New Department

The CMS is used by various departments in SDSU to create their website and load their content. If the new department wants to create the website using the CMS, it contacts the site administrator and provides them with the details for creating a new department. A computer science department can easily create their own website but it can be bothersome for the other department. The department can easily create its own page, maintain, manage and update its information as desired. The permission to add a new department is with the administrator. A new department can be added by clicking “Add Dept” button, then the administrator can add the username and update its permissions to read, write and admin by selecting users tab from the tree. The site admin is responsible for updating user roles to create, edit and publish the departments website. Figure 4.5 shows the user permissions page for departments.

![User Permissions Page](image)

**Figure 4.5. Add user permissions for department.**

4.4 Department User Roles

The departments need to set up the user roles for publisher, editor and author of the department before editing the other details and adding the content to the page. The user will
be able to edit the other details only when the department user roles are set or else it will show the message “The department user roles need to be updated”. Updating the department user roles automatically leads to updating the user permissions for their roles:

- Author: read, write, site administrator (Author has a limitation that they cannot update the department settings page but has access to view them)
- Editor: read, write, site administrator (Editor can update the departments settings tab but cannot publish the departments website)
- Publisher: read, write, site administrator (Publisher has full rights to accept or reject the content updated by author and validated by publisher)

The users for the appropriate roles will be notified by email about the role they have been appointed and the department name. Once the department user roles are setup, the administrator of CMS can communicate to the departments through these roles at the start of website creation.

The author creates the content and sends the email to the editor for approval and formatting of the content. The tasks for author includes:

- Adding text, images, forms, form feeds.
- Creating new pages.
- Deleting existing pages [25].

The editor is responsible for:

- Properly formatting the tables, headers, captions and links for the website [25].

The publisher is responsible for finally publishing the edited and formatted content to server. The department user permissions ensure that each user can change the section of content they are responsible for. Figure 4.6 shows the departments user roles for department.

### 4.5 Versioning of Content

Version Control gives the author, editor, publisher and administrator to retrieve the old content. Versioning is the ability to manage versions of content as it evolves. Content Management System contains multiple versions of content. Each time the content is updated, it creates a new version and save that particular update. Authors and editors often need to restore older versions of edited products due to a process failure or an undesirable series of edits. The add, update and delete for all the content is saved for that particular department. The content history for that department can be visible on clicking the “Content Version
History” for that department on the “Department Settings” tab. The department can restore the content to the previous version for that particular update by clicking the “Restore”. Figure 4.7 shows the version history of content for department

**4.6 VERSIONING AT PAGE LEVEL**

Versioning is being implemented for the update done for particular page in the department. The user can view the updates done for the particular page for the “Page Content History” link at the bottom of the page. In the pages tab, the “Page Content History” gives the history of all the pages added, edited or deleted for the particular department. While doing a series of edits, if a page is deleted by mistake, the user can revert back to the version in which the page has been added. The restore facility is available on that same page. The user does not need to go back to the main content history to revert. Figure 4.8 shows the version history at page level for department.
4.7 VERSIONING OF DEPARTMENT LAYOUT

Currently the CMS has just one layout “SDSU 2011 Template”, but there may be a new Layout with style sheets for example “SDSU 2012 Template”. The department may not like the new layout and want to stick to the old layout with the new content. The versioning at the layout level gives the departments option to use old layout with new content. Figure 4.9 shows the version history for layout updates for department.

4.8 NOTIFICATION MANAGEMENT

The notification management has the following features for the user.
4.8.1 Notification and Event Generation (via Email)

Notification is part of workflow management. It is used to create the event and set up a process within the department. When the author has created the content and is ready for the editor to tune the content and add styling, he notifies the editor to review the changes. After the editor edits the content, they send the notification to the publisher who reviews it and publishes the content. In case the content is rejected, then an email notification is sent to author of the department by the publisher stating the reason why the content has been rejected. If the user wants to include multiple users apart from the main department roles, then their toolbox username can be entered in the “Notify Others” textbox separated by
Figure 4.9. Version history of layout for department.

commas. The department list only gives the list of the departments, which have the department user roles setup, and their toolbox users have their email id setup in the database. Figure 4.10 shows the email notifications page for department.

4.8.2 Notification to Administrator (via Email)

The departments can email to the administrator of the site by selecting the:

- Department from the list.
- Their user role in the department which can be author, editor, publisher or others.
- Giving the subject and writing their query in detail.

The notification to admin allows the departments using the CMS to contact the admin in case of the doubts regarding adding and publishing the content. It automates the communication process rather than manually keeping the track of the email or the phone number of the admin. The queries of the departments are directly sent to the mailbox of the CMS admin. Figure 4.11 shows the email notifications to administrator page for department.
Figure 4.10. Email notification to departments.
Figure 4.11. Email notification to admin.
CHAPTER 5

CONCLUSION AND FUTURE EXTENSIONS

In conclusion, content management systems can be used for setting up workflow for interoperability for the various departments at SDSU. This is illustrated through analysis of the current CMS and its features while explaining the workflow, roles, and the underlying technologies used. Chapter 1 talks in brief about the existing CMS, its features and its limitations. Chapter 2 talks about the technologies used in developing the CMS. Chapter 3 explains the design of the functionalities implemented as part of workflow. It gives the high-level overview structure of the classes and the database. An in-depth explanation of working and functionality of the new workflow system is given in Chapter 4.

The key advantage of content management system can be described as intuitive and user friendly. It is important for the CMS to separate the content from the layout, formatting the content, implementing the workflow for approval of content before it is published to the user. Thus, CMS capabilities were focused more on the following: (1) version control for content (2) sub-version control for content (3) workflow management (4) user authentication (5) email notification. All of these capabilities ensure quality publications of SDSU Newsletter and other content while maintaining a history and notifying students of important materials thus bridging the relationship between department, students, and technology. We have implemented these features as a part of workflow for the Toolbox CMS.

In the future, new functionality will be added such as content checkpoint. The content checkpoint will save the entire content for the website as of that date. It will help the user when trying to edit or entirely change the website. If the user does not like the present content, then they can restore the entire website to the previous checkpoint. The content checkpoint functionality is similar to a system restore to a previous date in Windows OS. Additionally, multilingual support will help faculty, departments, and students get a richer level of knowledge while yet creating a strong bond with future alumni and leaders in an ever growing world.
REFERENCES


APPENDIX A

DEPARTMENT AND PAGES TABLE SCHEMA
APPENDIX A: Department and Pages Table Schema

The DEPARTMENTS table stores the information about different departments using CMS to make their websites. The table assigns a unique id to each department stored in the id attribute. Attribute active states if the website is published or not. The table PAGES stores the details of the web pages added by different departments. Each webpage is given a unique id stored in the id attribute and the attribute DepartmentId signifies the department adding that particular web page.
APPENDIX B

XML EXAMPLE
APPENDIX B: XML Example

Example of the information about new page in form of XML, this information is saved when a new version is created on addition of page. The ContentsUpdateHistory stores version history of content update.

```xml
<PagesInfo>
  <Pages>
    <ROW>
      <id>3128</id>
      <name>Comm1234</name>
      <fileName>comm.aspx</fileName>
      <active>1</active>
      <rightLinkGroupId>0</rightLinkGroupId>
      <footerLinkGroupId>0</footerLinkGroupId>
      <inheritPermissions>1</inheritPermissions>
      <requireApproval>0</requireApproval>
      <layoutID>10</layoutID>
      <dateUpdated>2012-12-04T17:46:45.980</dateUpdated>
      <updatedBy>tbx.kanikam</updatedBy>
      <updatedUserIP>::1</updatedUserIP>
      <departmentID>106</departmentID>
      <bk_tile>1</bk_tile>
      <comments>0</comments>
      <bk_color>FF42AD</bk_color>
      <right_link_title />
      <image_alt />
    </ROW>
  </Pages>
</PagesInfo>
```
APPENDIX C

STORED PROCEDURE
APPENDIX C: Stored Procedure

The Stored procedure used to read parse the XML and update the data in the respective table in database

GO

/****** Object: StoredProcedure [dbo].[prXMLDataUpdate] ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Description: Translate XML to SQL RecordSet for Update
-- =============================================

CREATE PROCEDURE [dbo].[prXMLDataUpdate]
(  
    @XmlData xml
)
AS
BEGIN

SET NOCOUNT ON
SET ARITHABORT ON
SET CONCAT_NULL_YIELDS_NULL ON
SET QUOTED_IDENTIFIER ON
SET ANSI_NULLS ON
SET ANSI_PADDING ON
SET ANSI_WARNINGS ON
SET NUMERIC_ROUNDABORT OFF

DECLARE @hdoc int

-- Prepare XML document
EXEC sp_xml_preparedocument @hdoc OUTPUT, @xmlData

-- Set Raw XML Schema
SELECT *
INTO #xmlDoc
FROM OPENXML( @hdoc, '/*',2)

-- Set Primary Table to use
SELECT DISTINCT Identity(int,1,1) id, rt.localname + '/' + tbl.localname + '/' + col.localname as NodePath,
    tbl.localname as NodeRow
INTO #xml
FROM #xmlDoc rt
INNER JOIN #xmlDoc tbl
    ON rt.id = tbl.parentID and rt.parentID is null
INNER JOIN #xmlDoc col
    ON tbl.id = col.parentID

DECLARE @i int, @NodePath varchar(255), @NodeRow varchar(50), @NodeKeys varchar(255), @NodeCol varchar(4000), @UpdateNodes varchar(4000), @sSql nvarchar(4000)

-- Set id of first row
SELECT @i = min(id) from #xml

-- Begin looping through xml recordset
WHILE @i is not null
BEGIN

SELECT @NodePath = NodePath, @NodeRow = NodeRow FROM #xml WHERE id = @i

-- Get Table Schema for XML data columns
SELECT @NodeCol = [dbo].[fnGetTableSchema](@NodeRow)
SELECT @UpdateNodes = [dbo].[fnGetTableUpdate](@NodeRow)
SELECT @NodeKeys = [dbo].[fnGetPrimaryKeys](@NodeRow)

DECLARE @param NVARCHAR(50)
SET @param = N'@hdoc INT'

/******** This updates xml Recordset on primary keys of a given table ******/
SET @sSql = 'UPDATE ' + @NodeRow + ' SET ' + @UpdateNodes + ' FROM OPENXML( @hdoc, ''' + @NodePath + ''' ,2) WITH (' + @NodeCol + ') as xm INNER JOIN ' + @NodeRow + ' ON ' + @NodeKeys

/******** Execute the query and pass in the @hdoc for update ******/
EXEC sp_executesql @sSql, @param, @hdoc
/***** Movenext *****/
SELECT @i = min(id) FROM #xml WHERE id > @i
END

-- Release @hdoc
EXEC sp_xml_removedocument @hdoc
DROP TABLE #xmlDoc
DROP TABLE #xml

END
GO