Chapter 16

A Case Study in Database Organization
Learning Objectives

• Explain the relationship between XML and XSL
• Describe how to express metadata using XML
• Demonstrate the incremental creation of a database
• Explain the relationship between tags and templates
• Show how to use tag attributes to display images
• State how information is hidden in XML databases
Thinking About a Database

- Regular Versus Irregular Data
  - Relational databases can be tables with regular rows, attributes, keys, relationships
  - Allows for queries where computers do all the organizing and displaying the
  - Data needs to be “regular” following a rigid structure imposed on it.
  - Relational databases may be powerful, but very often the information to record isn’t regular
iDiary Preliminaries

• iDiary will hold information that is found daily:
  – text, photos, URLs of interesting sites, animations, poems, videos, etc.

• Use XML to specify the metadata
  – The database will be an XML tree

• For iDiary, we will follow the Identity, Affinity, and Collection rules (see Chapter 15)
iDiary Preliminaries

• Collecting information is not necessarily organization
• Data needs structure
• Organizing the data helps us keep track of what we have, and it also helps the computer display it
• For iDiary, the database will be organized using time/date sequences
Physical Versus Logical

• The XML tree is the physical database, it is not a relational database
• The logical database is our view of iDiary
• For querying iDiary, XSL (a tagging language) is used
  – XSL stands for Extensible Stylesheet Language
  – It is an integral part of XML and standardized Web databases
iDiary Preparation

• When creating, start small and add as you go
• If the small database is working, and a new feature is added and the database no longer works, then the new feature has an error
• This is part of debugging
iDiary: Journal of Interesting Stuff

This is a record of the most interesting thing I found out each day that’s worth remembering. There’s personal stuff in this database, too, but it’s not gonna be displayed!

19 September 2011

The Digital Diary

This will be fun! I start my digital journal today. So, to launch it with the right sentiment, I looked up what Henry David Thoreau wrote on the first day of his famous *Journal*.

> What are you doing now? he asked. Do you keep a journal? So I make my first entry today.

Which, I guess, is pretty much what I just said. Great minds ... !

20 September 2011

Kepler-16b, *Star Wars* Planet

NASA found Tatoine! It’s a planet orbiting a binary star system, just like Tatooine in the *Star Wars* movies. It’s only 200 lightyears away –– Let’s Go!!! –– and is about the size of Saturn. According to

Figure 16.1 Part 1: An example of the planned iDiary.
The XML Definition

• Entries in the database will be a list of countries
• Each will have a name and a tour that contains a list of sights
• The name of the file contains the country’s flag as a tag attribute for the <name> tag
XML Definition

```xml
<country>
  <name flag="file.gif"> Country name </name>
  <tour>
    <sight> Sight name </sight>
    ...
    <sight> Sight name </sight>
  </tour>
</country>
```
XML Definition

• Use a standard text editor to enter the data
• Add <tags> as you go
• The file must have the .xml file extension AND include:

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

as the first line of the file
XML Definition

• The root element is the Collective tag enclosing all items in an XML file
• <travels> is the root element of this tree
• Countries will be listed within using this structure
• Save the file as travels.xml
Direct Check of XML

- The file is shown in a browser
- The browser uses a stylesheet to show information
- With XML, and the second line commented out, the browser shows the XML tree
- Color coding helps check that the structure is right
Direct Check of XML

• The display is *active*, meaning that tags can be opened and closed in the *Affinity* or *Collection* manner
  – Close the `<tour>` tags by clicking the minus (`−`) signs
• Closing parts of the database allows us to see the some tags without the clutter of other tags
Figure 16.3  The display of the travels.xml file without a styling (XSL) file specified.
Displaying with XSL

- Style information tells the browser how to display a markup language like XML
- Tags are eliminated when using the style information
- Information is then displayed based on the style description
Figure 16.2 The display of the travels.xml file using the travelSS.xsl style information.
XML with Style

• XML contains style information much like Cascading Style Sheets.
  – Cascading Style Sheets are separate files with a .css extension
  – XML sheets use an .xsl extension

• The files are connected because
  `<?xml-stylesheet type="text/xsl" href="travelSS.xsl"?>` is listed as second line

• The line must be exactly as shown with the correct file name
The Idea of XSL

• XSL formatting:
  – The .xsl file contains a series of rules on how to format the information enclosed in XML tags
  – There is only one rule or template per tag
  – The template uses XHTML (remember Chapter 4?)
The Idea of XSL

• When the XML file is opened in a browser:
  – The database (DB) and stylesheet (SS) are input to a transformer
  – The transformer “walks” the XML tree, converting all of the tags to HTML according to the template
  – When the “walk” is finished, the HTML page is displayed
Figure 16.4 Schematic diagram showing how the XML database tree and the XSL style information are merged to produce HTML; the final HTML result is displayed by the browser.
XSL Templates

• When reviewing XSL templates:
  – Notice that there are tags everywhere
  – XSL is XML
  – The first line is the required first line of any XML file
  – The second line is also required
XSL Templates

• There are five different tags used in the XML tree
• There are five templates in the XSL file, one for each tag
• The templates have a standard form specifying how to display the tags in XHTML
Creating the Travelogue Display

between the<br>tags are XHTML tags

Start of the HTML page

End of the HTML page

Copyright © 2013 Pearson Education, Inc. Publishing as Pearson Addison-Wesley
Visualizing the iDiary Database

• The template includes the heading and body tags
• These tags are <table> tags, because the content of the travels.xml file is going to be displayed as a table:

<table>
<thead>
<tr>
<th>Info for &lt;name&gt; tag</th>
<th>&lt;sight&gt; entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag display here</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>&lt;sight&gt; entry</td>
</tr>
</tbody>
</table>
Visualizing the iDiary Database

- Each `<country>` will be a row in the table
- There are two items in the first cell of the table: the name and the flag image
- The second cells contains a list of items of sights that may been seen on a tour
Summary of the XML tags style roles used for iDiary

<table>
<thead>
<tr>
<th>XML Tag</th>
<th>XSL Template Task for Displaying the Tag’s Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;travels&gt;</td>
<td>Set up the page, start and finish, including the tags for a table.</td>
</tr>
<tr>
<td>&lt;country&gt;</td>
<td>Set up a table row.</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>Set up the table data tags for the first cell of a row, place the name, skip to the next line, and place the image of the flag.</td>
</tr>
<tr>
<td>&lt;tour&gt;</td>
<td>Set up the table data tags for the second cell.</td>
</tr>
<tr>
<td>&lt;sight&gt;</td>
<td>Break to a new line and display the sight.</td>
</tr>
</tbody>
</table>

Each XML tag plays a stylistic role in the overall creation of the Web page.
The **apply** Operation

- `<xsl:apply-templates/>` tag:
  - This is a stand-alone tag with no mate
  - The tag is included once in each template
  - The meaning of the tag is that whatever is inside the tag should be processed
  - means, “now process whatever is inside this tag”
The **apply** Operation

- **Example (in English):**
  - “When encountering a `<tour>` tag:
    - Place a `<td>` tag in the accumulating XHTML definition
    - Process the items found within the `<tour>` tag (which will be a bunch of `<sight>` tags)
    - Place the `</td>` tag to complete the table

```xml
<xsl:template match="tour">
  <td>
    <xsl:apply-templates/>
  </td>
</xsl:template>
```
Tag Attributes

• To display an image requires the `<img src ... />` tag
• Note the form, it includes `{@flag}
• `{@flag} refers to the value of the tag attribute of the `<name>` tag (which gives the file name `flag.gif`
Tag Attributes

• Placing a tag attribute reference in braces in XSL:
  – Causes the tag attribute’s value from the XML to be placed inside the quotes specifying the file source name
Summary of XSL.

1. Open the `.xml` file with a browser
2. It finds the specified `.xsl` file
3. The browser’s transformer begins to process the XML tree
4. As it finds the tags, it checks for a template in the `.xsl` file
Summary of XSL.

6. The browser then does what the template says and places the XHTML
7. The browser then process the other information within the tags
8. When it’s done, it appends remaining XHTML tags to the HTML definition
9. Finally, it displays the resulting Web page
The iDiary Database

• When creating, we’ll use an incremental approach that will naturally guide you:

1. Getting started
2. Creating a first entry
3. Thinking about the nature of things
4. Developing tags and templates
1. Getting started

• First concern is building the XML database
• XML allows us to think up the tags and enables us to have any structure we want!
• That means, we have a design task:
  – What are our needs?
  – What kind of structure will meet those needs?
  – We will be storing information about “things”
  – There will be a sequence of entries with a date entry and information to be stored
1. Getting started

• Creating the XML Database:
  – First, decide on the *Affinity* tag (use `<entry>`)
  – Second, decide on the *Collection* tag (use `<idiary>`)

• Creating the XSL Stylesheet:
  – The XSL stylesheet will need to recognize the two tags
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- <xml-stylesheet type = "text/xsl" href = "iDiarySS.xsl" -->
<idiary>
  <entry> This is the first entry </entry>
</idiary>
This contains the setup for the Web page: the title, heading, and italicized comment at the start of the page.
iDiary: Journal of Interesting Stuff

This is a record of the most interesting thing I found out each day that’s worth remembering. There’s personal stuff in this database, too, but it’s not gonna be displayed!

This is the first entry
2. Creating a first entry

• Consider what goes inside of each <entry> tag:
  – Start with the September 19 entry
  – Since the entry is the entity that is going in the database, what items should be enclosed by it?
    • The date is one part (use <date>)
    • Content is the other part (use <cool>, since the content will include cool stuff
2. Creating a first entry

• Date Tagging:
  – This refers to the decisions surrounding the metadata for calendar dates
  – Since data in databases is atomic, we cannot refer to the day, month, and year separately if it is written:
    <date>September 19, 2011</date>
  – This, however, would be a lot to type:
    <date><month>September</month><day>19</day><year>2011</year></date>
<entry>
  <date>19 September 2011</date>
  <cool><title>The Digital Diary</title>
      This will be fun! I start my digital journal today.
    ...
    Great minds ... !
  </cool>
</entry>
2. Creating a first entry

• Now add templates to the iDiary.xsl file

• The two tags will both be displayed as list items with <li> tags, with different “looks”

• Use CSS properties to identify how each <li> tag will look
iDiary: Journal of Interesting Stuff

This is a record of the most interesting thing I found out each day that's worth remembering. There's personal stuff in this database, too, but it's not gonna be displayed!

19 September 2011
The Digital Library This will be fun! I start my digital journal today. So, to launch it with the right sentiment, I looked up what Henry David Thoreau wrote on the first day of his famous Journal. 'What are you doing now?' he asked. 'Do you keep a journal?' So I make my first entry today. Which, I guess, is pretty much what I just said. Great minds ... !
2. Creating a first entry

• Class Styling:
  – The styling for the `<li>` tags uses some unfamiliar CSS
  – The features are straightforward, but we haven’t seen them before
  – For more information on these types of features, see the CSS tutorial at
    www.w3schools.com/cssref/default.asp
Figure 16.6  CSS styling for the two classes of list items: date and entry.
3. Thinking about the nature of things

- When creating an entry, all of the information must be captured digitally
- Data can take many forms
- These forms will affect both the XML and the XSL definitions
3. Thinking about the nature of things

• Recognizing the Need for Specific Tags
  – When considering the design of the XML, notice that we must specify different data for each type of content

<table>
<thead>
<tr>
<th>Type</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>URL and anchor</td>
</tr>
<tr>
<td>Image</td>
<td>Source file URL, and possibly width</td>
</tr>
<tr>
<td>Text</td>
<td>Written directly into the file, possibly with italic</td>
</tr>
<tr>
<td>Video</td>
<td>URL, player dimensions, other parameters</td>
</tr>
</tbody>
</table>
3. Thinking about the nature of things

- Recognizing the Need for Specific Tags
  - Each of tag requires that we specify different information
  - They also require different formatting.
  - The text within each tag can take several different forms

<table>
<thead>
<tr>
<th>Type</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titles</td>
<td>Centered with enlarged size</td>
</tr>
<tr>
<td>Quotes</td>
<td>Intended with colored background</td>
</tr>
<tr>
<td>Poems</td>
<td>Title, author, and line breaks at specific points</td>
</tr>
</tbody>
</table>
3. Thinking about the nature of things

• Choosing Specific Tags.
  – Knowing that different kinds of data need different tags (each database attribute requires its own tag), a new tag is assigned to each kind of data stored in iDiary

<table>
<thead>
<tr>
<th>Tag</th>
<th>Encloses</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;title&gt;</td>
<td>Text to be centered; font is enlarged</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>Anchor text; the URL is a tag attribute</td>
</tr>
<tr>
<td>&lt;pic&gt;</td>
<td>Stand-alone tag with file name and width tag attributes</td>
</tr>
<tr>
<td>&lt;quote&gt;</td>
<td>Text to be indented and given background color</td>
</tr>
<tr>
<td>&lt;poem&gt;</td>
<td>Groups &lt;p_title&gt;, &lt;poet&gt;, and &lt;lines&gt; tags</td>
</tr>
<tr>
<td>&lt;nasavideo&gt;</td>
<td>Stand-alone tag with URL of video</td>
</tr>
</tbody>
</table>
3. Thinking about the nature of things

- The `<poem>` tag is an *Affinity* tag grouping three other tags that give the title of the poem, the author, and the lines.
- In addition to `<title>` there is included:
  - a `<p_title>` for the poems
    Here, `<title>` will be centered and larger than the normal text.
3. Thinking about the nature of things

• Because there is a need for different tags for different kinds of information, the role of the `<cool>` tag changes slightly

• Previously it was an *Identification* tag and enclosed the most interesting thing

• Now, the role of the `<cool>` tag is that of an *Affinity* tag
  – It groups together all of the different forms of information
4. Developing tags and templates

• The Title Tag
  – This tag announces the most interesting thing entry:
    
    ```html
    <title> The Digital Diary </title>
    ```
  
  – The text is centered and uses HTML heading tags to enlarge the font:
    
    ```xml
    <xsl:template match="title">
      <h2>
        <xsl:apply-templates/>
      </h2>
    </xsl:template>
    ```
4. Developing tags and templates

• The **Link** Tag
  – The `<link>` tag specifies a Web link
  – The Web link has two parts:
    • The tag encloses the anchor text
    • The URL is specified using the tag attribute
      
      ```html
      <link url="http://apod.nasa.gov"> APOD, </link>
      ```

  – The stylesheet must place an `<a href...>` tag and enclose the tag attribute value in quotes
4. Developing tags and templates

• The Link Tag
  – The template to do that is:

```xml
<xsl:template match="link">
  <a href="{@url}">
    <xsl:apply-templates/>
  </a>
</xsl:template>
```

– The @ symbol is the XSL reference to the tag attribute of the XML tag
4. Developing tags and templates

• The Picture Tag
  – The `<pic>` tag is a stand-alone
  – The tag encodes the file name of the image and its desired display width

  ```html
  <pic file="pix/hela.jpg" width="350"/>
  ```

  – Note that as a stand-alone tag is terminated by the `/>`
4. Developing tags and templates

• The Picture Tag

```xml
<xsl:template match="pic">
  <img src="{@file}" width="{@width}" alt="Picture of Interesting Thing"/>
</xsl:template>
```

– A stand-alone tag does not enclose anything
– There is no need for the `<xsl:apply-templates/>` tag
– This makes the `<pic>` tag slightly different from those we’ve seen
4. Developing tags and templates

• The **Quote** Tag
  – Quotes are an example of text that should be set off and noticeable
  – Use the `<blockquote>` HTML tag and style it as liked:

```html
<quote>'What are you doing ... today?'</quote>
```
4. Developing tags and templates

• The **Quote** Tag
  
  – We can style the implementing `<blockquote>` tag in the `<head>` section of the page, but it is just as easy to style it inline:

    ```xml
    <xsl:template match="quote">
        <blockquote style="background-color : #c89848; color : white; padding : 10px; font-size : small">
            <xsl:apply-templates/>
        </blockquote>
    </xsl:template>
    
    – By classifying different types of text, there is tight control on the format of the page
4. Developing tags and templates

• The Poetry Tags
  – We identify several attributes of poetry (title, author, lines) and assign tags to each.
  – They are enclosed in the Affinity tag `<poem>`

```xml
<poem>
  <p_title>Selecting a Reader</p_title>
  <poet>Ted Kooser</poet>
  <line>First, I would have her be beautiful,</line>
  <line>and walking carefully up on my poetry</line>
  ...
</poem>
```
4. Developing tags and templates

• The **Poetry Tag Templates:**

```xml
<xsl:template match="poem">
  <span style="font-family: century gothic">
    <xsl:apply-templates/>
  </span>
</xsl:template>

<xsl:template match="p_title">
  <h3>
    <xsl:apply-templates/>
  </h3>
</xsl:template>

<xsl:template match="poet">
  <h4><i>
    <xsl:apply-templates/>
  </i></h4>
</xsl:template>

<xsl:template match="line">
  <xsl:apply-templates/>
  <br/>
</xsl:template>
```
4. Developing tags and templates

- **The Video Tag**
  - To include videos in the iDiary, embed a connection to the player
  - Most sites that stream videos have an “embed” button near the video, and clicking it gives you all of the HTML you need to show the video
  - Packaging this information into an XSL template accesses the video
4. Developing tags and templates

- The Video Tag

Figure 16.8 Accessing embedding information at YouTube; (a) at the bottom of the video is Share; navigate Share > Embed to reveal the information in (b) which shows the embedding text (<iframe>) plus options; notice that <iframe> information should be selected after picking options.
4. Developing tags and templates

• The Video Tag
  – Develop the XML and XSL as you did for the stand-alone tag: `<pic file=... />`
  – The approach allows for simply providing the URL and getting the formatting automatically

```xml
<youtubestd url="http://www.youtube.com/embed/0MfRo0eC1ks?rel=0"/>
```
4. Developing tags and templates

- The Video Tag
  - For the XSL text, use the <iframe> text format
  - Replace the URL with the reference to the <youtubestd> tag's URL attribute: "{@url}"
4. Developing tags and templates

• The Video Tag
  – Notice there is one additional change from the text that received from YouTube:
    • the last attribute in the `<iframe>` tag is `allowfullscreen`
  – This is not a *legal* attribute since they are to be followed by an equal sign and a “value”
  – All this attribute is saying is that it’s Okay to let the user display at full screen size
    `allowfullscreen = "true"`
4. Developing tags and templates

• Another Video Source
  – Video embedding varies a lot
  – The <youtubestd> tag probably won’t work for other videos
  – You may need to develop a new tag for each video source
  – To make an XSL template for new video tag, do the same thing you did before:
    • Remove the URL
    • Replace it with the "{@url}" reference
    • Set the two attributes to be assigned the value true
Archiving Photos

- As the page was built, the iDiary.xml and iDiarySS.xsl files and photos have been on the desktop
- These need to be placed in a permanent location
- Store the photos in a separate folder within the folder containing the two database files (call the folder imFiles)
Archiving Photos

• There are two choices for specifying the path to these pictures
  – Add the path in the XML file as part of the <pic . . . /> tag (\texttt{<pic file="imFiles/planet.jpg" . . . />})
  – Add it in the XSL file as part of the <pic . . . /> tag template (\texttt{<img src="imFiles/{@file}" . . . >})

• The first solution is the more flexible, allowing references to images stored elsewhere on the Internet as well as other folders
Hiding Information

• All of the information in the database is displayed

• *We do have to tag everything, and we must provide a template to process each tag, but…*
  – But, we don’t have to display it!
Hiding Information

• Add a tag `<personal>` that encloses our personal thoughts.
• Assume the `<personal>` tag is inside of the `<cool>` tag

```xml
<xsl:template match="personal">
  <xsl:apply-templates/>
  <!-- Display personal information-->
</xsl:template>
```

• It’s wise to include a comment to remind ourselves what we’re displaying
Hiding Information

• You need to tell the transformer to “process the information enclosed in the matched tag,” or “leave it out”:

```
<xsl:template match="personal">
  <!--Don't display personal information-->
</xsl:template>
```

• When the transformer gets to a `<personal>` tag, it checks to see what to do
Hiding Information

• With no instructions to apply templates to the enclosed information, it skips the information inside the tags

```xml
<xsl:template match="personal">
  <!--Don't display personal information-->
</xsl:template>
```

• Note that including personal information in the file without enclosing it in tags, or tagging it but not providing a template will result in the information displayed
Entering Data into the Database

- Create a "template" for a new entry in the XML database file
- Use placeholders or dummy file names for URLs, widths, etc.
- To use, copy/paste this "template" and edit into the specific content for the new entry
Entering Data into the Database

<!-- The following tags are available for adding a new entry.
   Change the places containing x's
<entry>
   <date>x-day-x x-month-x x-year-x</date>
   <cool>
      <link url="http://xx">x-anchor_text-x</link>
      <title>x-title_of_entry-x</title>
      <pic file="xxx.jpg" width="xx"/>
      <quote>xxquoted_text-x</quote>
      <poem>
         <p_title>x-poem_title-x</p_title>
         <poet>x-author-x</poet>
         <line>x-line_of_poem-x</line>
      </poem>
   </cool>
</entry>

Edit all places with xx -->
Summary

- From this case study you now understand the following:
  - XML databases can record irregular data that relational databases cannot.
  - An XML database can be directly displayed by opening it in a browser.
  - Adding a stylesheet line to XML and building templates in XSL allows the XML file to be attractively formatted using HTML so that it can be displayed by a browser.
Summary

• From this case study you now understand the following:
  – A complex database can be set up incrementally, adding tags and templates one at a time, and checking that they work as planned.
  – An XML database can optionally hide some of its information, allowing for the selective display of its contents.