Walk Our Children to School: an Internet Data Management Application

Carol Martell, Highway Safety Research Center, Chapel Hill, NC

ABSTRACT

This paper describes a complete data management and query system wherein data is collected, maintained and dynamically surfaced on the web. The system integrates SAS/IntrNet Application Dispatcher, htmSQL, the email facility, and an ODS HTML imagemap. Even though all data is manually screened prior to release, the manual effort is minimal and the people screening and maintaining data are not SAS users. Good, old-fashioned data management experience teamed up with common sense can use the power of SAS to take care of the scene behind the web pages.

INTRODUCTION

The fourth annual National Walk Our Children to School Day is scheduled for October 6, 2000. Children, parents and community leaders will walk to school together to promote safety, health, physical activity, and environmental concern. Anticipation is that participation in this event to engender partnerships for change will again dramatically increase as in previous years. The goal of enabling synergy among event providers and of quickly surfacing new information would best be realized with a dynamic web application.

Having encountering various difficulties using other application server products to manage data, our web development group asked whether the SAS System could handle data from an online registration form and incorporate the following requirements. After initial web registration, people would need to update their records as event plans coalesce and more community leaders and organizations agree to participate. All registrants would be asked to update after the event, describing what actually happened in their community. Event information provided would be immediately available at www.walktoschool-usa.org. All content would be ***manually*** screened prior to release.

The project was a litmus test of integrating web developers with SAS programmers, so there were effectively two sets of clients needing special care: the people registering on the web ('registrants') and the in-house people who would manage the data ('users'). The SAS System provides a variety of components that made it possible to build an application designed to keep the users in their native web environment with no learning curve.

SAS Application Dispatcher handles the original registration data. Manual data content screening occurs via email. Registrants are given a userid and password to access and update their records. A SAS/GRAPH drilldown map of the United States is the entry point for dynamic data queries, providing immediate access to new data. A dynamic web page listing all registrants gives the in-house users update access and the means to easily supply lost userids and passwords or to simply communicate with individual registrants. A web form provides a bulk email facility for general announcements to the registrants.

INITIAL REGISTRATION

The SAS Application Dispatcher broker accepts information from the registration form in Figure 1 and executes a SAS program. The program creates a permanent SAS table containing only that registrant's observation and sends email containing those values to the in-house users for screening. The acknowledgement seen in Figure 2 is returned to the registrant's browser window.

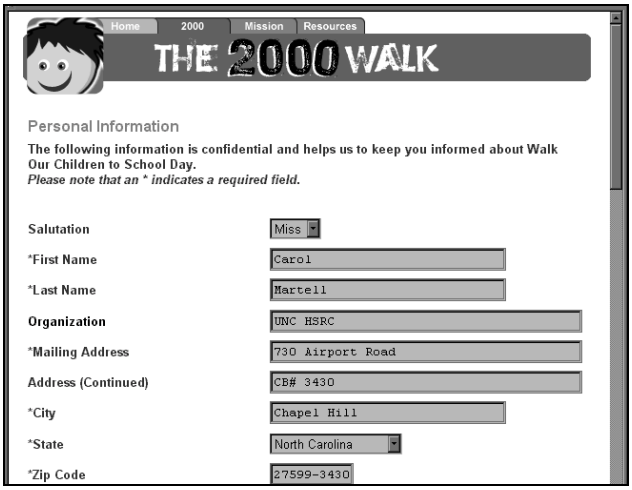

Figure 1

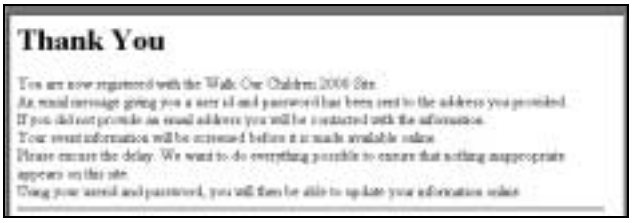

Figure 2

Userid and password variables are also created in the program and are combined to form the SAS table name seen in Figure 3.

```
$ls carol*
carol_martell425595.sas7bdat
$
```

Figure 3

With Application Dispatcher, information typed into each input field in a web form is passed to a SAS program. The program handling this form data is called *register.sas* and it resides in a directory defined to Application Dispatcher as *myjobs*. This information is specified in the html source code as:

```
<input type=hidden name="_program"
value="myjobs.register.sas">
```

As example of how data is passed, the html source code for the first name field is:

```
<input type = "text" name = "first_name"
size = "30">
```

The registration form in Figure 1 has 'Carol' in the blank for first name. The field name, *first_name*, is passed to *register.sas* as the macro variable *&first_name* having the value 'Carol'. The program can assign the macro variable's value to the SAS table variable for first name:

```
f_name=symget('first_name');
```

Register.sas also sends an email message to a project email account. The body of the message contains all the data from the registration form for screening as seen in Figure 4.

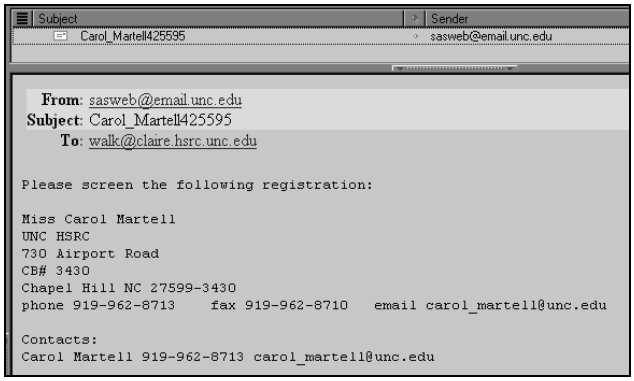

Figure 4

Sending email is accomplished using **filename** and **put** statements. Following the syntax for our Unix host system, the fileref is:

```
filename m email
'walk@www.walktoschool-usa.org'
subject="&userid&pwd";
```

Put statements generate the message body:

```
file m;
put
'Please screen the following registration:';
put salutation f_name l_name;
```

MANUAL SCREENING

Content screening is carried out using email. A reviewer reads the email message to determine whether to keep or delete the record. The end of each message contains three hypertext links as seen in Figure 5. The URL for each link includes parameters equivalent to the fields in a web form.

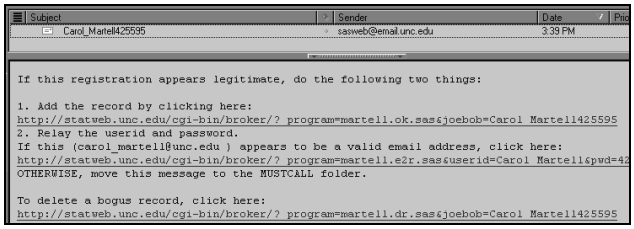

Figure 5

Each link executes a different SAS Application Dispatcher program. One link adds a valid registration to the main registration table. Another link deletes a bogus registration. The third link sends email to the registrant acknowledging registration and conveying the userid and password. Figure 6 shows the bottom of that email message for our example registration.

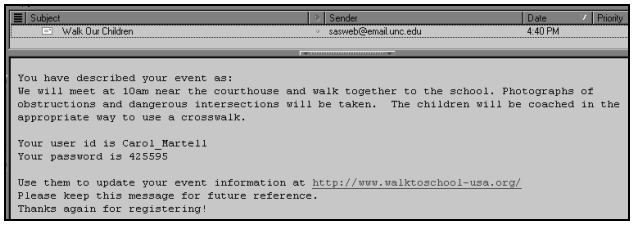

Figure 6

SURFACING DATA

The entry point for data presentation on the web is an imagemap of the United States (Figure 7.) Since the map shows states with registered events, it must be recreated whenever someone registers from a new state. When a record is added, if the state of residence was not already in the main table, code is executed to recreate the imagemap with the new state highlighted.

Figure 7

Imagemaps are generated using SAS/GRAPH and the Output Delivery System (ODS).

```
ods listing close;
filename carol 'mypath';
ods html  file='us2.html' path=carol;
goptions device=gif  noborder;
pattern1 color=CXFFCC00 value=msolid ;
proc gmap map=maps.us imagemap=mapds all
data=states;
  id state;
  choro j/
  coutline=black cempty=black
  html=st name="us"  nolegend ;
run;
ods html close;
quit;
```

Highlighted states, those with registrants, are drillable. Every link in the map is to the same htmSQL page. A state parameter customizes the resulting page. The links are set to the value of the variable named in the *html=* parameter in the *choro* statement. That variable was earlier constructed using SAS SQL.

```
create table states as
   select
     1 as j,
     stfips(state) as state,
     'href="citylist.hsql?st='||state||'"'
        as st
   from
   (select distinct state from r.registrants)
   order by state;
```

The North Carolina link, for example, is the following:

```
citylist.hsql?st=NC
```

When clicked, the htmSQL page queries the registration table for all records from the state of North Carolina and return a list of communities with registered events (Figure 8).

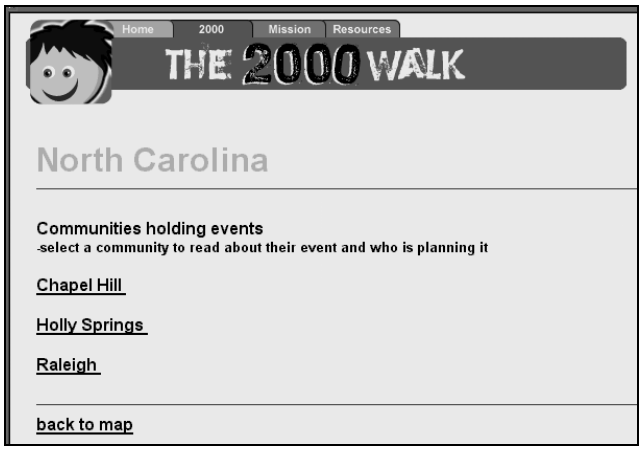

Figure 8

Figure 9

SAS htmSQL resembles other application server packages in that the code for the page contains one or more query sections with corresponding sections to html-encode the query results for the browser window. The browser window shows only the formatted query results. The parameters passed in as well as the columns selected in the SQL query are treated as macro variables syntactically referenced inside brackets: *{&var}*.

A SAS SHARE server provides access to the data and is identified in the query tag.

```
{query server="host.myserver"}
```

The *citylist.hsql* query is complicated by the fact that a single registration can encompass events in up to five schools that may be in different communities. The following *sql* section selects a distinct list of communities in a given state.

```
{sql}
select distinct * from
(
  (select city1 as city,
    translate(trim(city1),'+',' ') as cityp
    from wr.registrants where state="{&st}")
  union
  ...
  union
  (select city5 as city,
    translate(trim(city5),'+',' ') as cityp
    from wr.registrants where state="{&st}")
)
{/sql}
```

The *eachrow* section formats the results of the query. Each community returned from the above *sql* section is html-encoded to be a link with appropriate parameters. This enables the browser to drill down to another level of information.

```
{eachrow}
 <p>
 <a href=getcity.hsql?sta={&st}&cit={&cityp}>
      {&city}</a></p>
{/eachrow}
{/query}
```

Clicking on Chapel Hill in Figure 8 drills down to reveal the event used in our illustrations. *Getcity.hsql*?sta=NC&cit=Chapel+Hill dynamically creates Figure 9.

REGISTRANT UPDATES

Registrants may update information using their userid and password in the login web form seen in Figure 10.

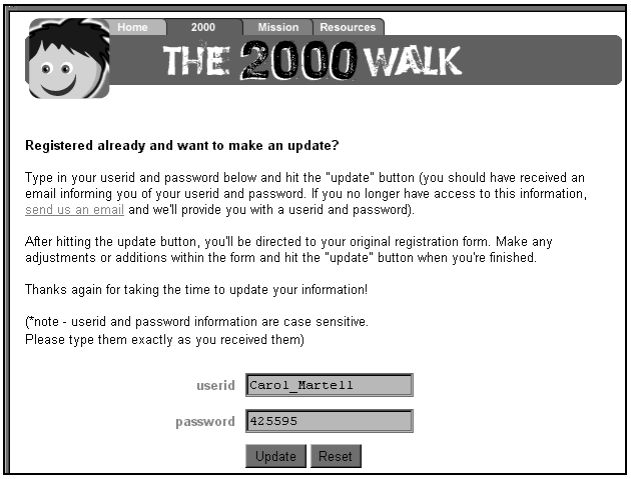

Figure 10

Clicking the update button executes an htmSQL query which returns what appears to be the original registration form (Figure 11) with the current information already typed into the fields.

Figure 11

Populating the registration form with data from the registrant table using SAS htmSQL is simple. The html code for the original form is sandwiched into the *eachrow* section. The preceeding *sql* section selects the correct record. Macro variables containing

values for that record are inserted into each input field using the *value*= html tag parameter.

```
{sql}
  select *
  from wr.registrants
  where userid="{&u}" and pwd="{&p}"
{/sql}

{eachrow}
  …
  <input type="text" name="first_name"
    size="30" value="{&f_name}">
  …
{/eachrow}
```

Processing registrant updates follows a path nearly identical to that for the original registration. Email reflecting the new version of the registration is sent to the reviewer, who again clicks an appropriate link to either accept or reject the changes. If accepted, an update data step replaces the old record.

MAINTENANCE

An htmSQL page (Figure 12) provides access for in-house users. The query lists all registrants, providing the options to update (update.hsql) or delete (remove.hsql) each record or to send an email message to the registrant (emailem.hsql). Each option is available as an htmSQL link with parameters to select the specific record. The eachrow section from the htmSQL page to list all registrants follows:

```
{eachrow}
  <tr>
    <td>{&st}
    <td>{&ct}
    <td>{&f_name}{&l_name}
    <td><a href=
       "update.hsql?u={&userid}&p={&pwd}">
       <font color="#00FF00">update record
       </font></a>
    <td><a href=
       "remove.hsql?u={&userid}&p={&pwd}">
       <font color="#CC0000">delete record
       </font></a>
    <td><a href=
       "emailem.hsql?u={&userid}&p={&pwd}">
       <font color="#0000cc">email registrant
       </font></a>
  </tr>
 {/eachrow}
```

DC	Washington	SteveWaters	update record	delete record	email registrant
FL	Boca Raton	JoyPuerta	update record	delete record	email registrant
FL	Deland	BarryWall	update record	delete record	email registrant
FL	Ft. Lauderdale	JeffAndrews	update record	delete record	email registrant
FL	Hollywood	PamelaMattingly	update record	delete record	email registrant
FL	Palm Bay	SallyWhite	update record	delete record	email registrant
IN	Greenfield	RonNorris	update record	delete record	email registrant
LA	New Orleans	KerryChausmer	update record	delete record	email registrant
MA	Cambridge	EllenKramer	update record	delete record	email registrant
MA	West Brookfield	PamelaChristiansen	update record	delete record	email registrant
MD	Silver Spring	WilliamSmith	update record	delete record	email registrant
MD	Westminster	KimberlyJones	update record	delete record	email registrant
MT	Helena	JenniferDalrymple	update record	delete record	email registrant
MT	Helena	KathyHarris	update record	delete record	email registrant
NC	Chapel Hill	CarolMartell	update record	delete record	email registrant

Figure 12

The email htmSQL page creates a customized web form. This dynamic form uses SAS Application Dispatcher and the email facility to send a message to a specific registrant. For illustration, the form seen in Figure 13 is completed and sent. Figure 14 shows the message received by the registrant.

Figure 13

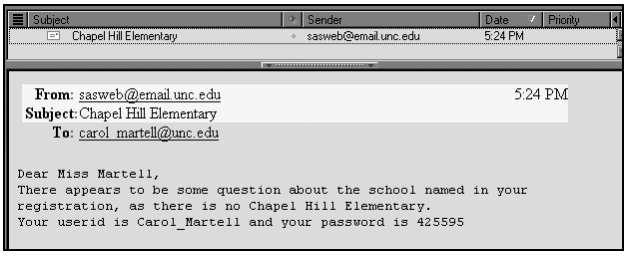

Figure 14

All correspondence is logged with email messages to the reviewer email account. Subject lines are constructed using userid and password so that when messages are sorted, all activity for a registrant appears together. The original registration subject has no suffix. Update subject lines contain the suffix 'update'. Figure 15 shows a documentary email message and subject line with the 'msg' suffix to show there was individual correspondence.

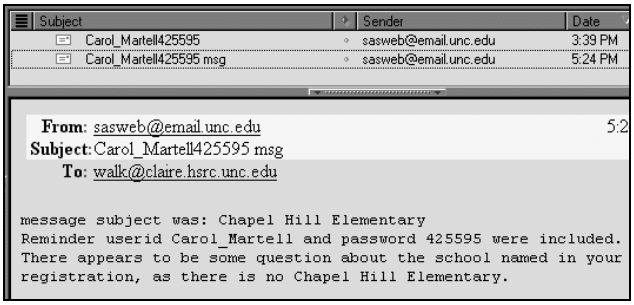

Figure 15

BULK MAIL

A bulk mail form (Figure 16) allows in-house users to send announcements and reminders to registrants. The users are encouraged to send and examine a trial message before clicking 'back' and changing the choice to 'all registrants'. Documentary copies of bulk email messages are sent to the reviewing mailbox with a subject line prefix of 'Bulkmail'.

Figure 16

SUMMARY

As illustrated in Figure 17, this dynamic data collection and retrieval system focuses around the registration table. The event participants generate new records and update existing ones. In-house users update or delete records. Data-triggered updates keep the imagemap current. Dynamic data retrieval occurs when visitors to the site drill down on the imagemap and subsequent pages, when registrants login to update their information, or when in-house staff use the maintenance utility. Email serves as documentation in addition to providing a means to communicate with registrants.

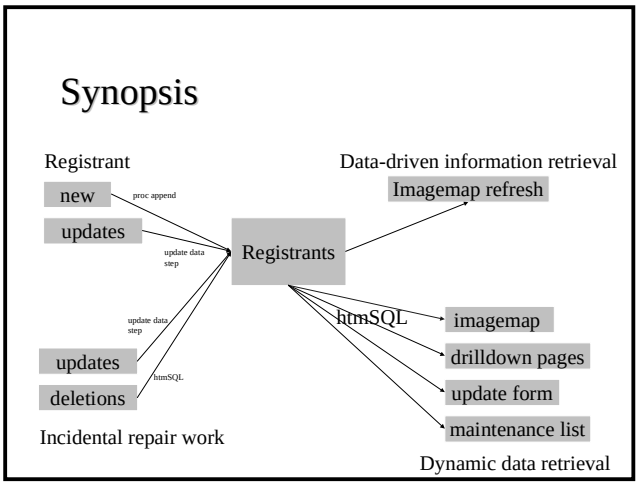

Figure 17

This application was built using SAS Application Dispatcher, htmSQL, SAS/GRAPH, the SAS Output Delivery System and the email facility. The only machine which actually runs The SAS System is the application server housing the data. Neither the registrants nor the in-house monitors need SAS running on their desktops because everything is processed using web technologies. The components we have used represent only the basic SAS internet technologies. The facilities exist to do much more, and we will continue to incorporate them as they are implemented at our site.

ACKNOWLEDGMENTS

The US Department of Transportation (DOT) provides funding for www.walktoschool-usa.org through the Pedestrian and Bicycling Information Center (www.walkinginfo.org).

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Carol Martell
UNC Highway Safety Research Center
730 Airport Rd, CB# 3430
Chapel Hill, NC 27599-3430
Work Phone: 919-962-8713
Fax: 919-962-8710
Email: carol_martell@unc.edu