

Case Studies in Data Management on the Web

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

ABSTRACT

The Highway Safety Research Center(HSRC) at the University of North Carolina in Chapel Hill uses the data management capabilities available through SAS/IntrNet® CGI Tools for several projects. This paper will examine three applications. The PBCAT Order application captures information, forwarding relevant fields to a fulfillment house. The Walk to School application collects data from the web and quickly resurfaces that data after human scrutiny. The PedBike Information System is similar to a problem tracking system. Questions are directed to a team of experts. Both questions and answers are entered into the system, creating a searchable database. This paper describes the structure of these projects, which incorporate base SAS®, SAS/SHARE® and two SAS/IntrNet components (htnSQL and the Application Dispatcher) in a Solaris environment.

INTRODUCTION

For each case study we begin with how the application behaves through a web interface. The parameter section describes project requirements that give direction to the system design. The solution overview describes the general organization and dynamics of the system. The data management section highlights coding techniques.

PBCAT ORDER SYSTEM

This software available free of charge on CDROM helps a customer categorize pedestrian and bicycle crashes. The order form is available on the virtual Web site hosted at Highway Safety (Figure 1).



Figure 1

The process begins when a customer submits an online order (Figure 2).

A screenshot of the PBCAT order form. The form is titled 'bicycle crashes' and 'PBCAT: order online'. It includes a description of the PBCAT software, a list of crash types, and a form for ordering the CD-ROM. The form fields include 'First Name', 'Last Name', 'Organization', 'Phone', 'Fax/Email', and 'Email'. A CD-ROM image is also visible on the right side of the page.

Figure 2

The order arrives at the fulfillment house in an email message:

Subject: PBCAT CD-ROM Order 868796
Date: Mon, 28 May 2001 12:45:53 -0400 (EDT)
From: somewhere@server.unc.edu
To: fulfillment@thatplace.com

Please send a PBCAT CD-ROM to the following person:

Ms Carol Martell
UNC HSRC
730 Airport Rd CB# 3430
Chapel Hill NC 27599-3430
USA
phone 919-962-2202
email carol_martell@unc.edu

This order was placed on 28MAY01

The customer receives acknowledgement email:

Subject: PBCAT CD-ROM Order 868796
Date: Mon, 28 May 2001 12:45:53 -0400(EDT)
From: somewhere@server.unc.edu
To: carol_martell@unc.edu

Thank you for ordering the PBCAT CD-ROM.
We will mail it to:

Ms Carol Martell
UNC HSRC
730 Airport Rd CB# 3430
Chapel Hill NC 27599-3430
USA
phone 919-962-2202
email carol_martell@unc.edu

This order was placed on 28MAY01

Please wait 15 business days for delivery.
If you do not receive your cd within that time, please send email to:
contact@unc.edu and reference order number 868796.
If the customer does not receive the software, they may follow up

with the contact provided, who has access to a dynamic web page showing all orders placed (Figure 3). Each order has links allowing address correction and reordering:

24MAY01 USA	IL	ungnd	weitenbach	update record	delete record	send email	reorder
25MAY01 USA	ID	Christopher	Bates	update record	delete record	send email	reorder
25MAY01 USA	NI	Fran	Sanchez	update record	delete record	send email	reorder
28MAY01 USA	NC	Carol	Martell	update record	delete record	send email	reorder

Figure 3

PARAMETERS

The fulfillment house is external to HSRC, so orders must to be forwarded to them. Their legacy system only accepts manual input, so we could not code something to automatically feed into their system. Instead, order requests are submitted to them by email. Because HSRC plans email notification to previous recipients of future software releases, order information is retained.

SOLUTION OVERVIEW

A single SAS table houses the order data. New information is added to the table through the Web form in Figure 2. Unacceptable mailing addresses, obviously invalid email addresses, and inconsistent zip codes are returned to the browser for correction and resubmission. Email to order the software is automatically sent to the fulfillment house when a record is added. At the same time, an email message acknowledging the order is sent to the customer. In-house maintenance is available through the dynamic web page (Figure 3) that lists each record with links enabling various actions. From this page we can, for instance, correct an address and resubmit the order for a customer.

DATA MANAGEMENT

The Application Dispatcher is used for data entry. All the information from the web form is passed in to a SAS program as macro variables. In the SAS code, a temporary, one-observation table is created for the order. Data is checked for completeness and validity. The following sample code shows email address screening:

```
email="&email";
if email = '' then
do;
str='We need your email address';
goto errorrm;
end;
if index(email, '@')=0 or
index(email, '.')=0 then
do;
str='We need a viable e-mail address';
goto errorrm;
end;
```

If problems are encountered, the customer is prompted to correct the problem and resubmit. Further processing must be prevented to avoid sending empty email messages. This is accomplished by using a macro variable (&quit):

```
errorrm: do;
file _webout;
put 'Content-type: text/html' ; put;
put '<h1> Please provide all required
information: </h1>';
put str "<br>";
put "Please provide the missing information
and resubmit <br>";
```

```
put "Thank you </html>";
call symput('quit', 'yes');
stop; end;
run;

data _null_;
x=symget('quit');
if x='yes' then do;
abort abend;
end;

run;
```

If all required data items are present and acceptable, the temporary table is appended to the main table. Next, the order is sent:

```
data;
set temporarytable;

filename sendit email
"fulfillment@thatplace.com"
subject="PBCAT CD-ROM Order &order";

file sendit;
put 'Please send a PBCAT CD-ROM to the
following person:';
put ' /salutation first_name last_name /
organization;
...
```

The acknowledgement message is composed similarly. The wording differs and the email filename statement contains the customer's email address in place of the fulfillment house email address.

Customer inquiries about an order are resolved through the in-house maintenance htmSQL page (Figure 3). This page lists each order record with four links to other htmSQL pages. The six-digit order reference number, created with a random number generator, is used to specify the record in the links. The four actions available for a record are update, delete, email and reorder. We examine the entire htmSQL code for the main page. We begin with the opening {query} directive pointing to the SAS/SHARE server:

```
{query server="myshareserver"}
```

Next we select all records from the table:

```
{sql}
select * from mylib.cdrom
order by date
{/sql}
```

We will present the results in a single HTML table, so we must first open the table and provide header information:

```
<font size=+2>
PBCAT Order List for Maintenance</font>
<table border=0>
<tr>
<td>order date</td>
<td>country</td>
<td>state</td>
<td>first name</td>
<td>last name</td>
<td>update record</td>
<td>delete record</td>
<td>email customer</td>
<td>reorder for customer</td>
</tr>
```

We want an HTML table row for every observation. We specify in

the {eachrow} section the HTML formatting and variable display for a single observation. This specification will be applied to each row returned from the query:

```
{eachrow}
<tr>
  <td>{&orderdate}</td>
  <td>{&country}</td>
  <td>{&state}</td>
  <td>{&firstname}</td>
  <td>{&lastname}</td>
  <td> <a href="update.hs1?o={&ord}">
    update record</a></td>
  <td><a href="remove.hs1?o={&ord}">
    delete record</a></td>
  <td><a href="email.hs1?o={&ord}">
    send email</a></td>
  <td><a href=
    "...broker8?_program=a.reord.sas&o={&ord}">
    reorder for customer</a></td>
</tr>
{/eachrow}
```

The first five HTML columns above display variable values. The next three build the links with accompanying name/value pair. The last column is an Application Dispatcher call to a program that sends a reorder request to the fulfillment house. Having composed the row, we close the {eachrow} section. Next we close the table and the {query} section:

```
</table>
{/query}
```

The update.hs1 page is exactly like the original order form except that it is already populated with the observation's values. We can easily accomplish this by making a copy of the order form, naming it 'update.hs1'. We place a query to select a specific observation at the top.

```
{query server="myserver"}
{sql}
select * from mylib.cdrom where order="{&o}"
{/sql}
```

Then we surround the form with {eachrow} directives, supplying values for each field. For example, where the order form contained an input field for first name with no value supplied:

```
<input name="firstname" value="">
```

the update form contains:

```
<input name="firstname" value="{&firstname}">
```

The order number must be passed along in a hidden field. Finally, we change the _program name to a different SAS program - one that updates an observation's values instead of the one that adds a new observation.

The code for the htmSQL page to delete a record could contain an {sql} section as simple as the following:

```
{sql}
delete from mylib.cdrom where order="{&o}"
{/sql}
```

Weaving back and forth between htmSQL and Application Dispatcher allows the developer to take advantage of the strengths of each tool.

WALK TO SCHOOL 2001

This project is for a campaign to identify obstacles preventing children from safely walking to school. HSRC hosts a website (Figure 4) with an online registration application to help event coordinators in the USA plan and publicize their local event. Coordinators provide information and we surface it to the web.



Figure 4

Figure 5

When someone organizing an event registers online (Figure 5), an acknowledgement window appears in their browser window (Figure 6).



Figure 6

The record immediately becomes available for in-house screening (Figure 7).

refresh the list
review registrant file

28MAY2001 21:33:34
• no similar records

Personal Information
The following information is confidential and helps us to keep our children to school day.

Salutation: Ms
First Name: Carol
Last Name: Martell

• no similar records
• no matching login and password

Figure 7

The screener can delete a bogus registration, approve it as is, or correct typos and approve corrected version (Figure 8). Upon approval, the registrant receives a 'welcome' email message (Figure 9). The message includes a link for updating their record. The link brings up a registration form already filled out with their information, ready for additions (Figure 10).

Carol Martell new
28MAY2001 21:33:34

• no similar records
• no matching login and password

28MAY2001 21:33:34

Send welcome message
☒ yes
☐ no

Showcase this event
☐ yes
☒ no

Flag3
☐ yes
☒ no

Flag4
☐ yes
☒ no

Change and Register

OR
[Add As Is \(NO CORRECTIONS\)](#)
OR
[Delete this record \(Bogus or Not New\)](#)

Figure 8

Subject: Walk to School Day - Registration Information
Date: Mon, 28 May 2001 21:46:01 -0400 (EDT)
From: walk to school day@duke.edu
To: carol_martell@nc.rrc.edu

Dear Ms Martell,

WELCOME! THANKS FOR REGISTERING FOR INTERNATIONAL WALK TO SCHOOL DAY IN THE UNITED STATES. You have just joined thousands from around the world who are going to walk to school on October 2, 2001. You will find your login, password and an update link below. Please keep this email for your reference. You have been added to our mailing list so that you can receive email about Walk to School Day news. Be sure to check the site from time to time to see who is walking and access resources available to help you while planning your event.

From: Logistics: CarolMartell
Your Password: wpschool01
The Website: <http://www.walktoschoolusa.com>
Your Update Link: <http://www.walktoschoolusa.com/cgi-bin/update.cgi?login=CarolMartell&password=wpschool01>

WELCOME!!!

FROM: Logistics
WALK TO SCHOOL DAY - USA
(919) 842-7410
www.walktoschoolusa.com

Figure 9

Personal Information
The following information is confidential and helps us to keep you informed about Walk Our Children to School Day.

Salutation: Ms
First Name: Carol
Last Name: Martell
Organization: UNC HSRG
Mailing Address: 730 Airport Rd CB#3430

Figure 10

All previously approved registrations are also available for in-house review in an update mode (Figure 11).

Get date: 28MAY2001 21:33:34

Personal Information
The following information is confidential and helps us to keep you informed about Walk Our Children to School Day.

Salutation: Ms
First Name: Carol
Last Name: Martell
Organization: UNC HSRG
Mailing Address: 730 Airport Rd CB#3430
Address (Continued):

Figure 11

The image map on the web site is the starting point for the public to view event details for all registered events (Figure 12). States with registered events are highlighted. Within ten minutes of registration approval, a new registrant's state becomes highlighted on the map (Figure 13).



Figure 12



Figure 13

Clicking on the state brings up a list of communities with registered events (Figure 14), and clicking on a community name brings up event descriptions for that community (Figure 15)



Figure 14

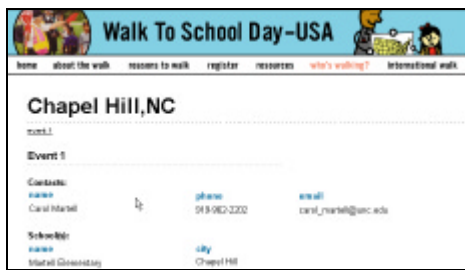


Figure 15

PARAMETERS

This application is streamlined and otherwise improved from the previous year's registration application. Goals included avoiding problems that cropped up the previous year. Registrants reluctant to navigate a web site or to supply login and password can take advantage of the update link provided in the welcome email message. Those who mistakenly register anew instead of updating are flagged on the screening page, minimizing the chance of duplicating records. Registrants are allowed to choose their own passwords. The code to refresh the image map and the code to send the welcome email message is placed in frequently run scheduled batch SAS jobs. No Application Dispatcher is used in the system; it is written entirely in htmSQL and scheduled batch jobs. Presenting a record to be screened in an update form allows for on-the-spot typo correction. We introduce the use of frames to simplify navigation for data maintenance.

SOLUTION OVERVIEW

A single SAS table houses the screened registration data. Each unscreened registration or update is a separate table that is deleted after screening. To flag instances where the customer mistakenly registers anew rather than submitting an update, the dynamic screening page provides links to display similar preexisting registrations. Since passwords are user-defined, duplicate logon information is also displayed. New submissions and preexisting records are displayed side-by-side on the screening page for easy comparison. Both new and preexisting records are displayed in a populated form; each can be accepted, modified or deleted.

DATA MANAGEMENT

Record flags are used to accomplish tasks requiring a data step. A batch job is scheduled to run periodically that checks for the welcome message flag. All new registrations have this flag set. The job sends a customized message to the email address in each record having the flag set and then resets the flag to 0. A batch job publishes the image map of the US every 10 minutes. Other tasks are accomplished using htmSQL. The screening page frameset defines the layout seen in Figure 16.

```
<frameset cols="20%,40%,40%">
  <frameset rows="10%,90%">
    <frame src="menu.html">
    <frame src="screen.hsqli"
      name="thelist">
  </frameset>
  <frame name="incoming">
  <frame name="existing">
</frameset>
```

This code divides the page into three vertical columns (frames), the first of which is also divided horizontally. Each frame given a name can be the target of a link from elsewhere on the page. We use the convention of always displaying new data in the center

column, and always displaying preexisting data in the far right column...hence the names 'incoming' and 'existing'. The bottom left frame displays screen.hsqli, listing all new records for screening.

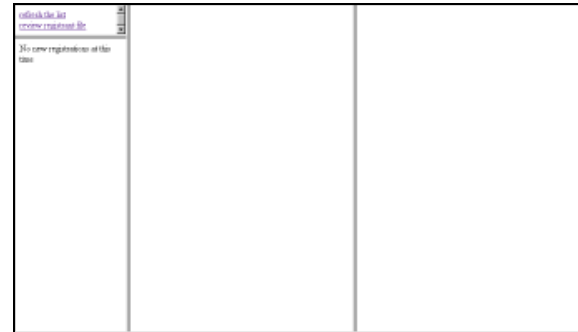


Figure 16

We examine the code for screen.hsqli. We query the dictionary tables to find all tables other than the main registration table. If there are none, we display a message to that effect:

```
{sql}
select * from
dictionary.tables where libname="WOCSLIB"
and memname ne "REGISTERED"
{/sql}
{norows}No new registrations at this time
{/norows}
```

For each registration found, we must look for similar names and watch out for duplicate name/password combinations. We use the nesting capabilities of htmSQL to accomplish our various tasks:

```
{eachrow}
{sql}
select fname as f, lname as l, type as t,
uid as unew, pword as pnew,
"{&memname}" as filename
from wocslib.{&memname}
{/sql}
```

We build a link targeting the center column to display the new data. The variable type, aliased as t, specifies whether the record is a new registration or an update, taking on values 'new' and 'upd'. The link will resolve to either viewnew.hsqli or viewupd.hsqli. They must differ because if the data is new it should be added while updates replace an existing observation.

```
{eachrow}<a href=
  "view{&t}.hsqli?f={&filename}"
  target="incoming">
  {&f} {&l} {&t} {&crdate}
</a>
```

We now look for similar preexisting data, comparing the names using soundex and spedis functions. If we find none, we display a message to that effect. For each record found we construct a link to display the existing data in the far right column:

```
{sql} select uid as u, pword as p,
first_name as fn, last_name as ln
from wmain.registrants
where
(
  soundex(first_name)=soundex("{&f}")
and
```

```

        soundex(last_name)=soundex("{&l}")
    )
    or
    (
        spedis(first_name,"{&f}")
        +
        spedis(last_name,"{&l}")
        <50
    )
} /sql}
{norows}no similar records{/norows}
similar records already registered
{eachrow}<a href=
    "viewold.hs1?u={&u}&p={&p}"
    target="existing">
    {&fn} {&ln}
    </a>
{/eachrow}

```

We perform another search for exactly matching preexisting logon information (the login has been constructed by compressing together first and last names). Again we either announce that there are no matches or build links to these similar records, targeting the far right frame. We close the encompassing eachrow sections:

```

{sql} select uid as ux, pword as px,
    first_name as fn, last_name as ln
from wmain.registrants where
    ux="{&unew}" and px="{&pnew}"
{/sql}
{norows}no matching login and password
{/norows}
already registered with login/password
{eachrow}<a href=
    "viewold.hs1?u={&ux}&p={&px}"
    target="existing">
    {&fn} {&ln}
    </a>
{/eachrow}
{/eachrow}
{/eachrow}

```

HTML formatting to display these results is not included in the code above. The formatting we use organizes the gathered information about each new record into a table with a border. The similar names and matching login/password records are displayed as unordered lists. Consequently, the person screening data sees a box for each new record, with bulleted lists of preexisting data that should be used for comparison. Data screeners are very happy with this solution. Providing access to all relevant information in a single Web page has proved to be extremely advantageous.

PEDBIKE INFORMATION SYSTEM

The use of frames for the Walk to School application was applied to another Web data management project. This project functions like a problem tracking system. A panel of experts answers questions posed by the public through a variety of avenues. Questions and answers are logged into the system through the web. The single Web page for this application is divided into five frames (Figure 17). The top left frame provides a menu. We will examine the avenues available there. Suppose someone named David Harkey poses a question. The first step is to determine whether or not he is already in the system. Entering a portion of

his name, 'hark', and clicking 'find' (Figure 18) sends a list of matching names to the lower left frame. We see that a record is found for David Harkey (Figure 19).

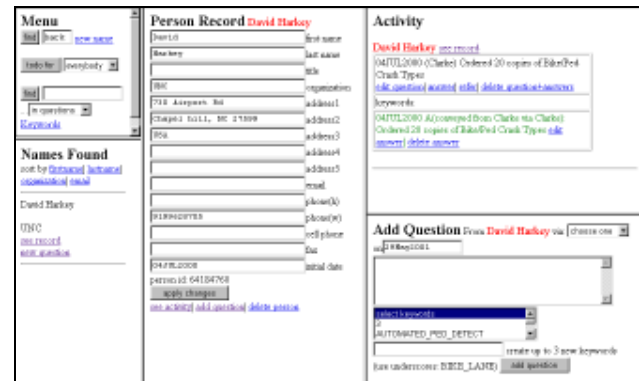


Figure 17



Figure 18

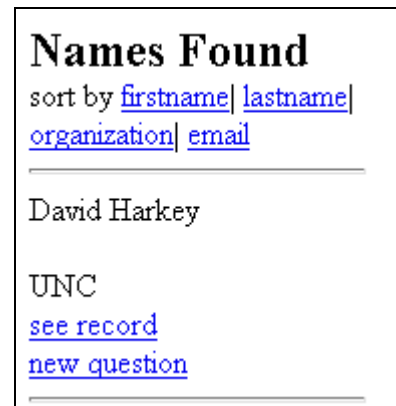


Figure 19

Clicking 'see record' sends his personal information record to the middle frame (Figure 20). It is displayed in a form so that information can be easily added or corrected.

Person Record **David Harkey**

David	first name
Harkey	last name
	title
UNC	organization
730 Airport Rd	address1
Chapel hill, NC 27599	address2
USA	address3
	address4
	address5
	email
	phone(h)
9199628705	phone(w)
	cell phone
	fax
04JUL2000	initial date

person id: 64184760

[see activity](#) | [add question](#) | [delete person](#)

Figure 20

Clicking 'see activity' sends a list of David's questions with the answers and outstanding referrals to the top right frame (Figure 21). Clicking 'add question' sends a form to the bottom right frame (Figure 22). Clicking 'answer' in the activity frame sends a form to the bottom right frame (Figure 23). Clicking 'refer' in the activity frame sends yet another form to the bottom right frame (Figure 24).

Activity

David Harkey [see record](#)

04JUL2000 (Clarke) Ordered 20 copies of Bike/Ped Crash Types
edit question answer refer delete question+answers
keywords:
04JUL2000 A(conveyed from Clarke via Clarke): Ordered 20 copies of Bike/Ped Crash Types edit
answer delete answer

Figure 21

Add Question From **David Harkey** via:

on:

3
AUTOMATED_PED_DETECT

create up to 3 new keywords
(use underscores: BIKE_LANE)

Figure 22

Answering **David Harkey** re: Ordered 20 copies of Bike/Ped Crash Types...

Answer source: Conveyed by:

on: Status:

Figure 23

Referring **David Harkey** Clarke question: Ordered 20 copies of Bike/Ped Crash Types...

Date: From: To:

Comment:

Figure 24

Dynamic select lists used throughout the forms provide choices for items such as the expert's name or the question source (Figure 25).

choose one	choose one
Clarke	800 number
Suttles	other phone
Harkey	PBIC email
Zegeer	other email
James	mail
Marchetti	verbal
search	other

Figure 25

To explore another avenue, we return to the menu frame (Figure 18) and click 'todo'. A list of four followup categories appears in the bottom left frame (Figure 26).

ToDo Page

Referrals: none
 Phone calls to make: [1](#)
 Faxes to send: none
 Letters to send: none

Figure 26

Beside each category is a numbered link for every item requiring attention. We click the link to see a not-yet-conveyed answer in the top right frame (Figure 27). The answer not yet conveyed is displayed as a form with conveyance marked in red. Once the answer is conveyed, the expert clicks the 'completed' button and the item will no longer show up in the 'todo' list.

John Fcyga [see record](#)

04JUL2000 (Clarke) Wants trail design information; waiting for call back. He talked with Christopher and got what he needed.
[edit question](#) | [answer](#) | [refer](#) | [delete question+answers](#)

keywords:

04JUL2000 A(conveyed from Clarke via Clarke): Wants trail design information; waiting for call back. He talked with Christopher and got what he needed. [edit answer](#) | [delete answer](#)

18MAY2001 A(phone from via): blah [edit answer](#) | [delete answer](#)

person conveying answer

Figure 27

The menu (Figure 18) offers the ability to search questions or answers for a word or phrase. We enter the search term, choose the file to search, and click 'find' (Figure 28). Results are displayed in the lower right frame (Figure 29). Each matching item has a numbered link. Clicking the link would display the specific question and answer(s) in the top right frame.

Menu

[new name](#)

...

[Keywords](#)

Figure 28

Search Results

[1](#) With interest we learned about some of your publications. As the subject matter covers the field of our interest, we would be most grateful if you could provide us with paper copies of the following publications: Intersection geometric design and operational guidelines for older drivers and pedestrians. Volume III: guidelines, 1997, FHWA-RD-96-137 Analysis of older drivers on freeways : HSIS summary report, 1997, FHWA-RD-96-035 Development of the bicycle compatibility index : a level of service concept, final report, 1998, FHWA-RD-98-072 and Engineering solutions to enhance the safety and mobility of older persons. Paper presented at the 30th Annual Human Factors Workshop of the Transportation

Figure 29

Now we choose the 'keyword' link from the menu frame (Figure 18). We see, in the lower right frame, three forms, two of which have select lists (Figure 30). These forms manipulate keywords. The first allows modification of an existing keyword.

Keywords

BIKE_SAFETY
 BIKE_SAFETY_COUNTERMEASURES
 BIKE_STATISTICS

Figure 30

Clicking modify brings up three choices (Figure 31). One might find that two keywords should be collapsed into one, or that a keyword was misspelled and should be replaced with a new, correctly spelled keyword. Keyword deletion is also available.

Changing keyword

Replace BIKE_STATISTICS with this new value and redirect references:

Redirect references to BIKE_STATISTICS to the following keyword and then remove BIKE_STATISTICS from keyword list:

[Delete](#) BIKE_STATISTICS and references without redirecting to another keyword

Figure 31

The keyword search form (Figure 30) allows multiple selection of keywords to find questions flagged with all the selected keywords. The search results (Figure 32) follow the familiar conventions: clicking the result number causes the question to display in the top right frame (Figure 33) with the usual accompanying information and links.

Search Results for questions with keyword BIKE_SAFETY.

1 I am a student majoring in elementary education at Otterbein College. This quarter, I am taking a health class in which I am required to create a lesson on a health issue that faces our children. I decided on a lesson about bike safety and I wondered if you could send me any information that would be useful. Any children's literature, posters, worksheets, etc., that would be useful, would be greatly appreciated! Thanks for your help,

2 I am helping with a WS DOT study on before and after analysis for two future crosswalk locations (through TRAC - Transportation

Figure 32

Questions

Laine Peterson [see record](#)

26SEP2000 (PBIC email) I am a student majoring in elementary education at Otterbein College. This quarter, I am taking a health class in which I am required to create a lesson on a health issue that faces our children. I decided on a lesson about bike safety and I wondered if you could send me any information that would be useful. Any children's literature, posters, worksheets, etc., that would be useful, would be greatly appreciated! Thanks for your help,

[edit question](#) [answer](#) [refer](#) [delete question+answers](#)

keywords: BIKE_SAFETY

15FEB2001 A(conveyed from Suttles via Suttles): First,

Figure 33

PARAMETERS

The system needed to track who is asking questions, what those questions are, who is answering the questions, and what the answers are, and needed to be searchable to allow reuse of previously supplied answers. An expert should be able to refer a question to another expert. All question and answer activity for a customer should be available to help provide the expert with a context. Determining whether someone is new versus already in the database should be extremely easy. A list of keywords to select and associate with questions must be included, along with the ability to add new keywords. Since the experts are scattered around the country, a web application was requested.

SOLUTION OVERVIEW

There are separate tables for the questions, answers, people, keywords, referrals, and experts. Each person, question, answer and referral is assigned a unique id. These are used to link the information for display. Where possible, answers are automatically sent by email or fax, using scheduled batch jobs that search the answer table for flagged records, send the answers, and change the flag values. Keyword management with retroactive action is included.

DATA MANAGEMENT

We will examine the htmSQL page invoked by clicking the 'activity' button from the person record (Figure 20). The person id has been passed in as 'p'.

```
<h2>Activity</h2>
{query server="myserver"}
{sql}
    select fname, lname from pbic.person
    where pid={&p}
{/sql}
{eachrow}{* display name, person rec link}
    <font size=+1 color="red">
    {&fname} {&lname}
    </font>
    <a href="person.hsql?p={&p}"
    target="tm">
    see record
    </a><br>
{/eachrow}
{sql} {* find questions for person}
    select qid as q, * from pbic.question
    where pid={&p} order by date
{/sql}
{norows}
    <br>no activity
{/norows}
{eachrow}
    <table width=350 border=1>
    <tr><td>{&date} {&source} {&text}<br>
    <a href="editques.hsql?p={&p}&q={&q}"
    target="br">edit question</a>|
    <a href="newansw.hsql?p={&p}&q={&q}"
    target="br">answer</a>|
    <a href="newref.hsql?p={&p}&q={&q}"
    target="br">refer</a>|
    <a href="deleteques.hsql?q={&q}"
    target="br">delete question+answers
    </a>
    </td></tr> {* show keywords}
    <tr><td>keywords: {&keyword}</td></tr>

{query server="myserver"}{* find answers}
{sql}
    select *,
    {* words/fonts vary by status}
    case sendthis
    when 'a' then 'conveyed'
    when 'p' then
    '<font color="red">phone</font>'
```

```

        when 'f' then
            '<font color="red">fax</font>'
        when 'x' then 'autofax'
        when 'e' then 'autoemail'
        when 'm' then
            '<font color="red">snailmail</font>'
        else '<font color="red">unknown</font>'
        end as status,
    { * HTML comment tags to fake out browser }
    case
        when sendthis in( 'a' 'x' 'e')
            then '<!--' else ''
            end as op,
    case when sendthis in( 'a' 'x' 'e')
        then '-->' else ''
        end as cl
    from pbic.answer
    where pid={&p} and qid={&q}
    {/sql}
    { * formatting for each answer follows }
    {eachrow}
    <tr>
    <td>
        <font color="green">{&date} A({&status}
        from {&source} via {&provider}): {&text}
        <a href=
            "editansw.hs1?a={&aid}&p={&p}&q={&q}"
            target="br">edit answer
        </a>|
        <a href="deleteansw.hs1?a={&aid}"
            target="br">delete answer
        </a>
    { * only need form for unconveyed answers }
    { * surrounding HTML comment tags override }
        {&op}
        <form action="sentthis.hs1"
            target="br">
        {&cl}
        {&op}
        <select name="prv">
            <option value="">
        {&cl}
        {&op}
        <input type="hidden" name="p"
            value="{&p}">
        {&cl}
        {&op}
        <input type="hidden" name="a"
            value="{&aid}">
        {&cl}
        {&op}
        <input type="hidden" name="q"
            value="{&q}">
        {&cl}

    {sql}
        select distinct source as pr from
        pbic.sources
    {/sql}
    {eachrow}
    {&op}
    <option value="{&pr}">{&pr}
    {&cl}
    {/eachrow}
    {&op}
    </select>person conveying answer
    <input type="submit"
        value="completed">
    </form>
    {&cl}
    </td></tr>
    {/eachrow}

```

```

    {/query} { * ends answer query }
    { * search for referrals not yet addressed }
    {query server="myserver"}
    {sql}
        select * from pbic.referral
        where pid={&p} and qid={&q}
    {/sql}
    {eachrow}
    <tr><td>{&date} Referred to {&referto} by
        {&source}
        {&comment before=" with comment: "} <br>
    <a href=
        "newanswref.hs1?r={&rid}&q={&q}&p={&p}"
        target="br">respond to referral</a>
    </td></tr>
    {/eachrow}
    </table><br>
    {/query} { * ends referral query }
    <hr>
    {/eachrow}
    {/query} { * ends question query }

```

CONCLUSION

Web data management allows a distributed population to add, modify and query shared data. In this examination of case studies we have seen that a tailored solution can be designed to suit project requirements.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors 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
 Web: www.hsrrc.unc.edu