```
if { $list_type == "snail_too" || $list_type == "snail_plus_demographics" } {
    # this domain wants a more complicated list; add Snail Mail fields
    append form_fields "<tr><td>Address Line 1<td><input name=line1 type=text
size=35>
<tr><td>Address Line 2<td><input name=line2 type=text size=35>
<tr><td>City, State, Postal Code<td><input name=city type=text size=12>
<input name=state type=text size=6>
<input name=postal_code type=text size=8>
<tr><td>Country<td><input name=country type=text size=3 limit=2> (ISO Code, e.g.,
\"us\", \"fr\", \"ca\", \"au\", etc.)"
}

if { $list_type == "snail_plus_demographics" } {
    append form_fields "<tr><td>Birthday<td><input name=birthday type=text
size=12> (YYYY-MM-DD format must be exact)
<tr><td>Sex<td><input name=sex type=radio value=M CHECKED> Male
<input name=sex type=radio value=F> Female
"
}

# finally return the page to the user
ns_return $conn 200 text/html "<html><head><title>Add Yourself to the Mailing
list</title></head>
<body bgcolor=#ffffff text=#000000>
<h2>Add Yourself</h2>

to <a href=\"home.tcl?domain=[ns_urlencode $domain]\">the mailing list</a>
for <a href=\"$backlink\">$backlink_title</a>

<hr>

<form method=post action=add-2.tcl>
<input type=hidden name=domain value=\"$domain\">
<table>
$form_fields
</table>
<input type=submit value=Submit>
</form>

<hr>
<address>
<a href=\"mailto:$maintainer_email\">
$maintainer_name ($maintainer_email)
</a>
</address>
</body>
</html>
"
```

Note how the final HTML page is strewn with values from the database, such as $maintainer_email, $backlink, and $backlink_title. Now it looks to all intents and purposes like it is part of your friend's service and you won't be getting e-mail from the confused. See Figure 13.3 for an example.

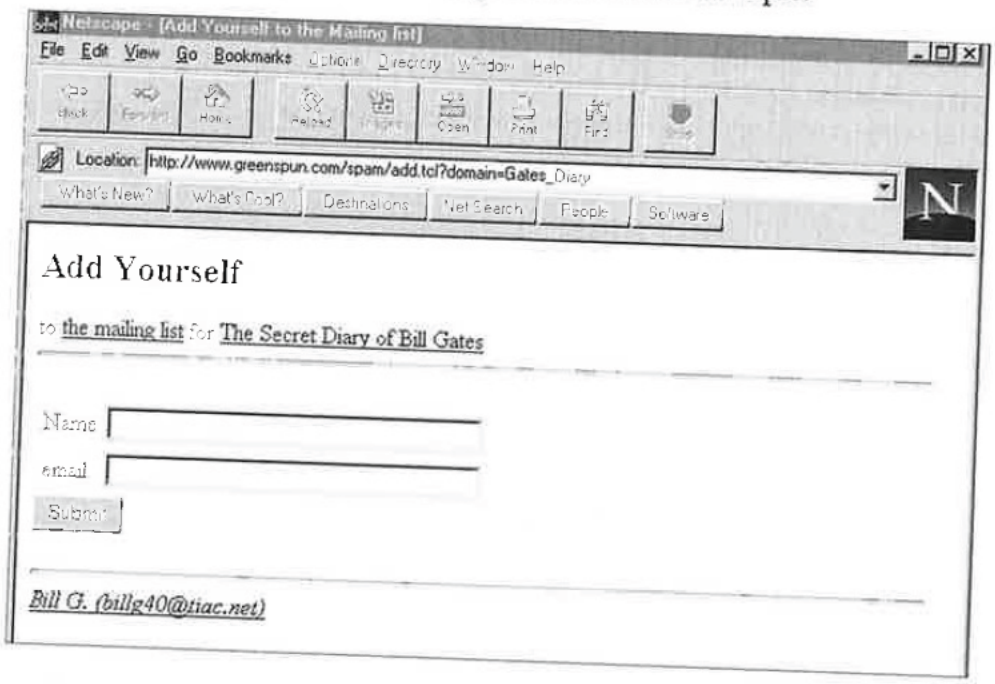


Figure 13.3: My mailing list system being used by another Web publisher (billg40@tiac.net). If users were to look carefully at the location box, they might notice that they were being bounced from www.tiac.net to www.greenspun.com after clicking the "join mailing list" link. Billg40 can keep his Secret Diary of Bill Gates at Tiac, where they don't run an RDBMS, and yet look just like a high-tech Web publisher with a staff of programmers and a database administrator.

CASE 3: THE BIRTHDAY REMINDER SYSTEM

Olin refused to give Alex (http://photo.net/photo/alex.html) a sample off his plate at brunch. "You're just a dog, Alex," Olin said. We pressed Olin as to his reason for feeling superior to Alex. Olin thought for a few minutes and all he could come up with was "I have a Ph.D. and he doesn't."

Olin demonstrated the practical value of his Carnegie-Mellon computer science degree by turning down Jim Clark's 1994 offer to become Employee #3 at a little start-up called Mosaic Communications (grab http://www.netscape.com

if you want to see how Jim and Co. are doing now). Consequently, his resort to credentialism set off howls of laughter throughout the room.

"Let's see what value society places on your Ph.D., Dr. Shivers," I said. "We'll take both you and Alex to Harvard Square and hang signs around your necks. Alex's sign will read 'Needs home.' Your sign can read 'Needs home. Has Ph.D.' Which one of you do you think would have to sit out there longer?"

"Nooooo contest," opined Olin's girlfriend.

Anyway, Olin clung to his belief that his Ph.D. was worth something despite the fact that the marketplace was crushing him under a burden of poverty to correspond to his burden of ignorance of how to build an RDBMS-backed Web service.

I kept offering to show Olin but he was too busy writing papers for academic journals that even he didn't bother reading. We began to joke that Olin was "afraid to be rich." Then one night Olin came over to my house and said, "Let's jack into this World Wide Cybernet thing."

We sat down to build a toy AOLserver/Illustra-backed birthday reminder system. Sure there are plenty of fancy calendar management systems that incorporate one-time events, recurring events, and reminders. But most of these calendar management programs require you to maintain them on a Macintosh or Windows machine. If you switch from computer to computer, they don't do you much good. We all read our e-mail no matter where we are, so why not build a system that feeds reminders into our e-mailbox? Again, it turns out that there are Web-based calendar management systems that will do just that. But these programs are very complicated. I don't have a job. I don't make appointments. I don't plan in advance. I don't want to invest in learning and using a calendar management program. I just want an e-mail message a week before my friend's birthday so that I can send him a card.

Olin and I sat down at 9 p.m. to build RemindMe. We were finished by midnight. Then we showed the text-only system to Ulla Zang (http://www.ullazang.com) and asked her to do a spiffy graphic design. Now we have a nice public service to offer.

STEP 1: THE DATA MODEL

Note: The syntax is for the Illustra 3.2 RDBMS.

```
--
-- this table has one row for each person using the system
-- the PRIMARY KEY constraint says that there can't be two
-- rows with the same value in the EMAIL column
--
```

```
create table bday_users (
        email           text not null primary key,
        password        text not null
);

--
-- we index bday_users because of the integrity constraint
-- on bday_reminders (so that an insert to bday_reminders
-- doesn't have to do a sequential scan of bday_users)
--

create index bday_users_by_email on bday_users using btree ( email );

create table bday_reminders (
        email                   text references bday_users,
        event_description       text,
        event_date              date,
        remind_week_before_p    boolean default 'f',
        remind_day_before_p     boolean default 'f',
        remind_day_of_p         boolean default 'f',
        last_reminded           date
);
```

The first item of interest in this data model is the integrity constraint that values in the email column of bday_reminders must correspond to values in the email column of bday_users. That's what "references bday_users" tells the database. After a row in bday_reminders is inserted or updated, the RDBMS will check to make sure that this integrity constraint is true. If not, the transaction will be aborted. Also, nobody will be able to delete a row from bday_users if there are still rows in bday_reminders that contain the same e-mail address.

Integrity constraints are critical if you have users typing data into a shell database tool. But here users will only be able to access the RDBMS through our Web pages. Why can't we just write our forms-processing software so that it never allows bad data into the database? Well, we can and we will. But unless you are the rare programmer who always writes perfect code, it is nice to have the RDBMS's integrity constraint system as a last line of defense.

Thomas Jefferson did not say, "Eternal vigilance is the price of liberty." That's because he wasn't John Philpot Curran (Irish statesman who never set foot in the United States). Nor did Jefferson say, "Eternal sluggishness is the price of integrity." That's because he wasn't an RDBMS programmer.

Let's focus a bit on "values in the email column of bday_reminders must correspond to values in the email column of bday_users." It is not magic fairies who maintain integrity but rather plodding, stupid computer programs.

Oracle Exhibit 1011
Page 108 of 171

During every attempt to insert a row into bday_reminders, the plodding, stupid relational database management system will grind through every row in bday_users to see if the proposed value for the EMAIL column matches one of those rows. This is an "order *n*" operation, requiring time proportional to the number of rows in the bday_users table. If, however, we create an index on the bday_users table thusly

```
create index bday_users_by_email on bday_users using btree ( email );
```

then the plodding, stupid RDBMS can check the index instead of the full table. For the binary tree index we just created, this takes "order log *n*" time. If you have 64,000 rows, *n* is 64,000 and log *n* is about 16.

STEP 2: LEGAL TRANSACTIONS

You'd think that the most obvious legal transaction would be "add user to bday_users table." However, I decided to not make that one of the legal transactions. I don't want a table full of e-mail addresses for people who aren't really using the system. Thus it is only legal to add a user atomically with at least one reminder:

```
begin transaction;
insert into bday_users
(email, password)
values
('philip@greenspun.com','hairysamoyed');
insert into bday_reminders
(email, event_description,
event_date,remind_week_before_p,remind_day_before_p,remind_day_of_p)
values
('philip@greenspun.com','remember to finish PhD',
'1997-09-01','t','t','t');
end transaction;
```

This transaction inserts the user "philip@greenspun.com" with password "hairysamoyed" and a reminder to finish his Ph.D. by September 1, 1997. Reminders will be sent a week before, a day before, and the day of.

Suppose the user doesn't want to be reminded the day of? That's another legal transaction:

```
update bday_reminders
set remind_day_of_p = 'f'
where oid = '2056.2017'
```

OID? That's a non-standard Illustra hidden column, the "object identifier." It uniquely identifies a row. Pages leading up to this transaction can get all the

columns of bday_reminders plus the oid by querying "select oid,* from bday_reminders." Mind you, if I were a good database programmer, bday_reminders would have a unique key. Either that or I would demand that the triple of EMAIL, EVENT_DESCRIPTION, and EVENT_DATE be unique and then use those as the key. Or I would add a REMINDER_ID column to the table and then assign unique values to this column from a sequence generator.

It is possible to index the table by EMAIL, EVENT_DESCRIPTION, and EVENT_DATE, then use a triple as a key. However, it makes for rather unwieldy looking update statements. Had I been using Oracle, I would have used a REMINDER_ID column and then inserted "reminder_id_sequence .NextVal" for the value. This takes advantage of a built-in Oracle sequence generation facility. Illustra doesn't have anything like this so I'd have to define a separate sequence generator table and complicate my transactions.

Where do I draw the line at vendor-specific kludges like this? On the client/server boundary. If other tables needed to refer to rows in bday_reminders, I'd create a REMINDER_ID column. I would never use an OID as a pointer from one database row to another because then my data model wouldn't be portable to Oracle or Sybase. However, in this case I'm just going to use the OID temporarily in my client Tcl program. If I move the system to another RDBMS, I'll add a key column to the data model and rewrite the two Tcl pages in question.

A reminder system with only one reminder isn't anything to write home about. We ought to be able to add new events:

```
insert into bday_reminders
(email, event_description,
event_date, remind_week_before_p, remind_day_before_p, remind_day_of_p)
values
('philip@greenspun.com','Wash dog whether he needs it or not',
'1997-12-01','t','f','t')
```

A week before December 1, I'll be reminded to buy shampoo for my Samoyed (he usually requires an entire bottle). I disabled the day-before reminder so that's "f," but I'll be reminded on December 1 itself.

One last legal transaction: deleting a reminder. Suppose that I finish my Ph.D. thesis (to which supposition my friends invariably respond, "Suppose the sun falls out of the sky"):

```
delete from bday_reminders where oid = '2056.2017'
```

Again, we're using the OID but just for the user interface.

Oracle Exhibit 1011
Page 110 of 171

STEP 3: MAPPING TRANSACTIONS ONTO WEB FORMS

My general philosophy is to have as few pages as possible. This keeps the user interface simple. Figure 13.4 shows the interaction flow for the RemindMe system. Oftentimes, the system uses redirects to pages that reflect current status rather than separate confirmation pages. For example, after a user disables a week-before alert, there is no confirmation page. Instead, after doing the database update, the Tcl script issues a 302 redirect back to the reminder summary page where the alert is shown to be disabled. Come add yourself as a RemindMe user (the system will be linked from http://webtools.com/wtr/ and http://arsdigita.com/) and then let me know if you think the interaction design is a success.

Figures 13.5 and 13.6 illustrate the evolution of our user interface.

STEP 4: WRITING CODE TO PROCESS THE FORMS

Here is the .tcl page that summarizes a user's reminders (see Figures 13.7 and 13.8):

```
set_form_variables
set_form_variables_string_trim_DoubleAposQQ

# email, password

set db [ns_db gethandle]
if { [database_to_tcl_string $db "select unique upper(password)
from bday_users
where upper(email) = upper('$QQemail')"] != \
     [string toupper $password] } {
    ns_returnredirect $conn \
                      "bad-password.tcl?email=[ns_urlencode $email]"
    # we've returned a 302 redirect to the user, now exit this thread
    return
}

# if we got here it means that the password checked out OK

# we're going to use this a bunch of times so let's save some work

set emailpassword \
    "email=[ns_urlencode $email]&password=[ns_urlencode $password]"

# we return the headers and top of the page before doing a query
# into the BDAY_REMINDERS table; that way the user isn't staring at
# a blank screen

# note: ReturnHeaders is one of my magic functions from utilities.tcl,
# available from http://demo.webho.com
```

Oracle Exhibit 1011
Page 111 of 171

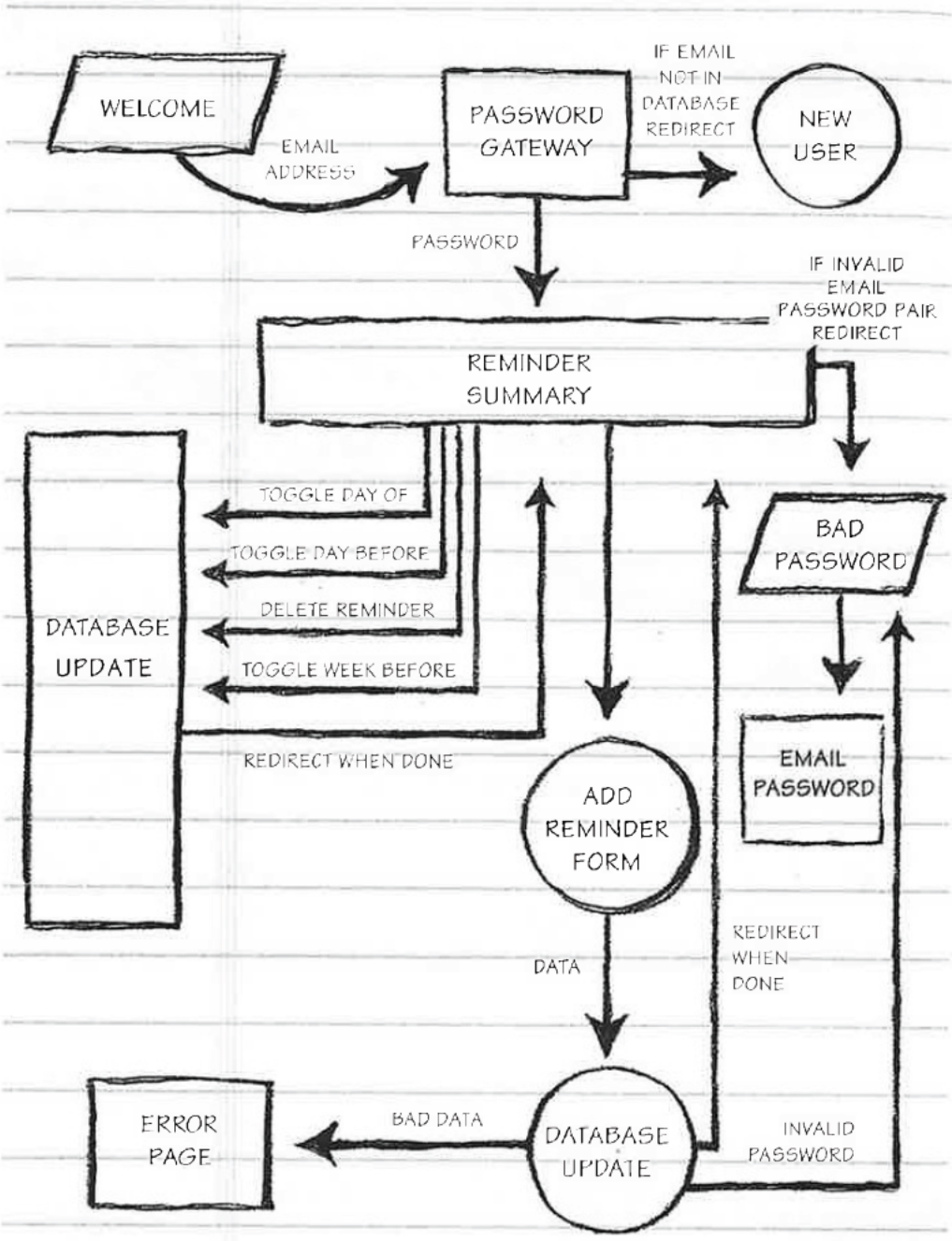


Figure 13.4: Interaction for the RemindMe system. Note the extensive use of redirects when bad input is received and/or after a simple update.

Oracle Exhibit 1011
Page 112 of 171

Welcome to RemindMe

Please enter your email address:

remindme@greenspun.com

Figure 13.5: The text-only welcome page, built by me and Olin



Figure 13.6: The welcome page, design by Ulla Zang; we saved a lot of time and heartache by completing our interaction design with a text-only site before consulting a graphic designer. Ulla turns out to be one of the few graphic designers I've worked with who is also an excellent interaction designer. Nonetheless, by having finished the programming beforehand, we were able to use Ulla's time to maximum advantage.

Oracle Exhibit 1011
Page 113 of 171

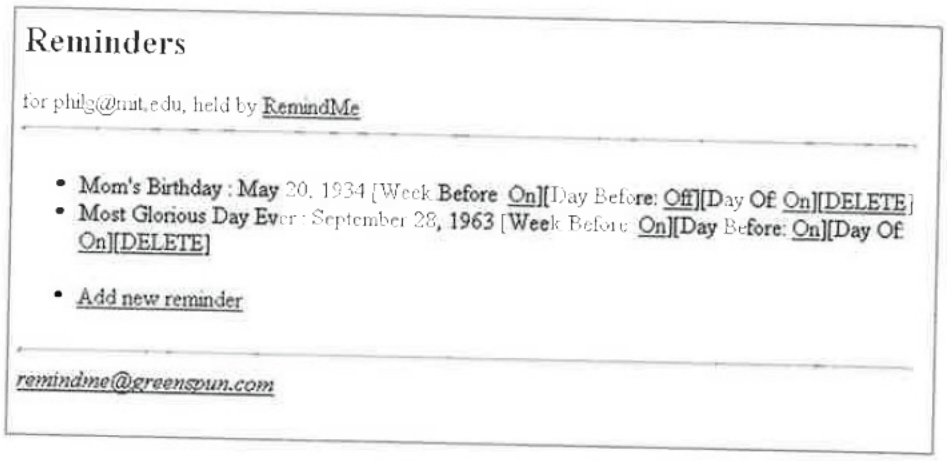


Figure 13.7: The reminders summary page, core of the user interface. Olin and I weren't satisfied with this design but decided to dump the user interface issue onto Ulla.

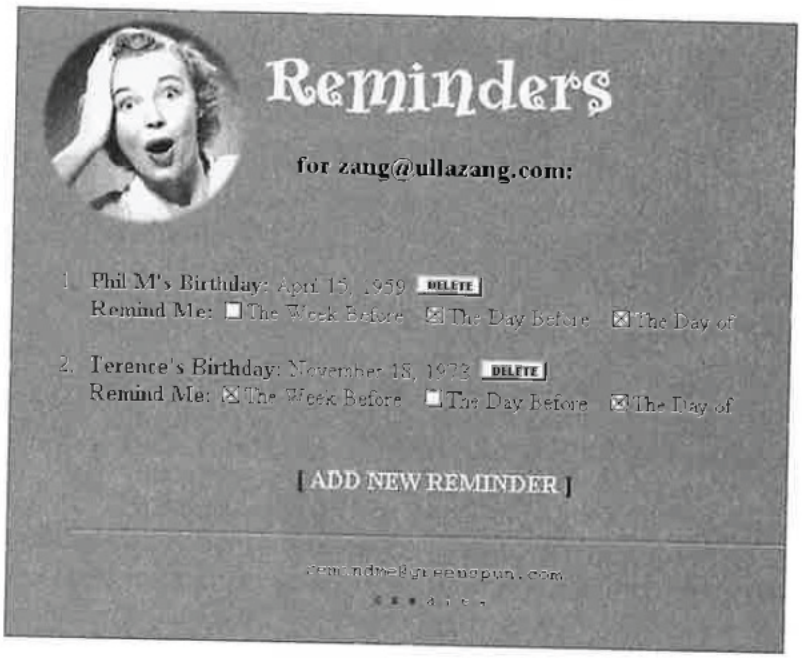


Figure 13.8: Ulla's reminders summary page. Much bigger than our text-only version, but also much cleaner. We'll be able to take her simplifications and translate them back into the text-only site as well. (That's another advantage of doing a full text-only site first; you always have something to satisfy the 28.8 modem crowd even if your graphic designer goes wild with huge images.)

```
ReturnHeaders $conn

ns_write $conn "<html>
<head>
<title>Reminders for $email</title>
</head>

<body bgcolor=#ffffff text=#000000>
<h2>Reminders</h2>

for $email

<hr>

<ul>
"

set selection [ns_db select $db \
"select *,
        oid,
        extract(month from event_date) as event_month,
        extract(day from event_date) as event_day
from bday_reminders
where upper(email) = upper('$QQemail')
order by event_month, event_day"]

while {[ns_db getrow $db $selection]} {
    set_variables_after_query
    ns_write $conn "<li>$event_description : [util_IllustraDatetoPrettyDate
$event_date] "
    # it would have been cleaner to think more and come up with
    # a general-purpose action.tcl function, but I think it is
    # also OK to do what we've done, use a separate .tcl page for
    # each kind of action

# for each reminder, we test to see if it is already set,
    # then present an appropriate current status hyperlinked to
    # a URL that will toggle the state of that reminder
    # note that we're using the Illustra OID as a key
    if { $remind_week_before_p == "t" } {
       ns_write $conn "\[Week Before: <a href=\"week-before-
off.tcl?oid=[ns_urlencode $oid]&$emailpassword\">On</a>\]" } else {
       ns_write $conn "\[Week Before: <a href=\"week-before-
on.tcl?oid=[ns_urlencode $oid]&$emailpassword\">Off</a>\]" }
    if { $remind_day_before_p == "t" } {
       ns_write $conn "\[Day Before: <a href=\"day-before-
off.tcl?oid=[ns_urlencode $oid]&$emailpassword\">On</a>\]" } else {
```

```
        ns_write $conn "\[Day Before: <a href=\"day-before-
on.tcl?oid=[ns_urlencode $oid]&$emailpassword\">Off</a>\]" }
    if { $remind_day_of_p == "t" } {
        ns_write $conn "\[Day Of: <a href=\"day-of-off.tcl?oid=[ns_urlencode
$oid]&$emailpassword\">On</a>\]" } else {
        ns_write $conn "\[Day Of: <a href=\"day-of-on.tcl?oid=[ns_urlencode
$oid]&$emailpassword\">Off</a>\]" }

    ns_write $conn "\[<a href=\"delete.tcl?oid=[ns_urlencode
$oid]&$emailpassword\">DELETE</a>\]"

}

ns_write $conn "

<P>

<li>

<a href=\"add-reminder.tcl?$emailpassword\">Add new reminder</a>

</ul>

<hr>
<a href=\"mailto:[bday_system_owner]\">
<address>[bday_system_owner]</address>
</a>
</body>
</html>"
```

Most of the interesting points about this procedure are documented in the comments above. The only thing worth stressing is that this is the meat of the user interface. The links from this page mostly just update the RDBMS and then redirect back to this page. For example, here's week-before-off.tcl:

```
set_form_variables

# email, password, oid

set db [ns_db gethandle]

# do a password check here as in the above procedure (code omitted)

ns_db dml $db "update bday_reminders
set remind_week_before_p = 'f'
where oid = '$oid'"
```

```
ns_returnredirect $conn "domain-top.tcl?email=[ns_urlencode
$email]&password=[ns_urlencode $password]"
```

This procedure just checks the password, updates the bday_reminders row, then redirects back to the reminder summary page.

STEP 5: STEP 5?

Yes, there is a Step 5 for this system: Making it work. If you are an Internet entrepreneur who has just raised $40 million for your WebJunkware System 2001 then Step 5 is optional. You only need a front-end good enough to show to venture capitalists and a screen capture program so that you can insert some screen shots into the initial public offering prospectus. If, however, you are a cringing little engineer whose ego is pathetically dependent upon producing a useful service, you need to write a back-end to send out reminders.

Almost all back-ends require that a function run every day at a set hour. I like to write my back-end code using the same tools as the rest of the system. In the old days, I would write the nightly sweeper or whatever as a dynamic Web page. Then I'd use the Unix cron facility to run a shell script every night (the Windows NT equivalent is the At command). The shell script would call htget (a Perl script) to grab this dynamic Web page.

AOLserver, however, has a built-in cron-like function. I prefer to use it rather than an operating system facility because it means less system administration when moving a service from one physical computer to another. Also, my code is portable across operating systems, and I won't need to install Perl scripts like htget.

It should be easy:

```
ns_schedule_daily 5 0 bday_sweep_all
```

This tells the AOLserver to run the function bday_sweep_all at 5:00 a.m. every day.

The first and most obvious problem with this statement is that the server might be down at 5 a.m. If we are careful to define bday_sweep_all so that it won't send out duplicates, we can just schedule it for a bunch of times during the day:

```
ns_schedule_daily 5 0 bday_sweep_all
ns_schedule_daily 9 0 bday_sweep_all
ns_schedule_daily 13 0 bday_sweep_all
```

The second problem is an AOLserver bug (my opinion) or feature (Doug McKee's opinion; he's one of the server's authors): If these statements are executed multiple times, the function will be multiply scheduled. For example, if

you put these schedule requests in the Tcl directory that gets sourced on server startup and then reinitialize Tcl five times (to test changes in other code), you will find that bday_sweep_all is called six times at 5 a.m.

Here's my workaround:

```
global bday_scheduled_p

if { ![info exists bday_scheduled_p] } {
    ns_schedule_daily 5 0 bday_sweep_all
    ns_schedule_daily 9 0 bday_sweep_all
    ns_schedule_daily 13 0 bday_sweep_all
    set bday_scheduled_p 1
}
```

I tell the Tcl interpreter that the variable bday_scheduled_p is to be global among all the AOLserver threads. If it has not been set yet ("![info exists"), I schedule the sweep at 5 a.m., 9 a.m., and 1 p.m. Then I set the flag to 1 so that subsequent loads of this Tcl file won't result in redundant scheduling.

All we have to do now is write the bday_sweep_all procedure. We can expect the algorithm to be more or less for all three reminder types, so we posit a basic bday_sweep procedure that takes the reminder type as an argument:

```
proc bday_sweep_all {} {
    bday_sweep "day_of"
    bday_sweep "day_before"
    bday_sweep "week_before"
}
```

Now we just have to write the sweeper per se:

```
proc bday_sweep {message_type} {
    # message_type can be "week before", "day before", "day of"

    switch $message_type {
       week_before { set sql_clause "remind_week_before_p
and event_month=extract(month from (current_date + interval '7' day))
and event_day=extract(day from (current_date + interval '7' day))"
                     set subject_fragment "NEXT WEEK"
                     set body_fragment "next week"
                  }
       day_before { set sql_clause "remind_day_before_p
and event_month=extract(month from (current_date + interval '1' day))
and event_day=extract(day from (current_date + interval '1' day))"
                     set subject_fragment "TOMORROW"
                     set body_fragment "tomorrow"
                  }
       day_of { set sql_clause "remind_day_of_p
```

```
and event_month=extract(month from current_date)
and event_day=extract(day from current_date)"
                set subject_fragment "TODAY"
                set body_fragment "today"
               }

    }

    # grab two database connections, one to look for reminders,
    # and one for updates to mark reminders as having been done
    set db_connections [ns_db gethandle [philg_server_default_pool] 2]
    set db [lindex $db_connections 0]
    set db_sub [lindex $db_connections 1]

    # we're going to keep track of how many reminders we sent and when
    set email_count 0
    set start_stamp [database_to_tcl_string $db "return current_timestamp(0)"]

    # we use our first database connection to look for relevant reminders
    # note that $SQL_CLAUSE is set in the SWITCH statement above
    # note also the final clause that prevents us from sending two
    # reminders on the same day for the same event
    set selection [ns_db select $db "select oid,*,extract(month from event_date)
as event_month, extract(day from event_date) as event_day
from bday_reminders
using (isolation level read uncommitted)
where $sql_clause
and (last_reminded <> current_date or last_reminded is null)"]

    while {[ns_db getrow $db $selection]} {
        set_variables_after_query
        # now all the columns from the DB are accessible as Tcl vars

        # we wrap a Tcl CATCH around the call to sendmail; we don't
        # want one error (e.g., a really badly formed email address
        # stopping the entire sweep)
        if [catch { ns_sendmail $email [bday_system_owner] "$event_description is
$subject_fragment" "Reminder:

$event_description ($event_date)

is $body_fragment.

This message brought to you by [bday_system_name].

If you don't want to receive these reminders in the future,
just visit [bday_system_url].
```

Oracle Exhibit 1011
Page 119 of 171

```
    "
        } errmsg] {
           # failed to send email
           # this AOLserver API call writes an entry into the error log
           ns_log Error "[bday_system_name] failed sending to $email: $errmsg"
        } else {
           # succeeded sending email, mark the row as reminded
           # note that we are using the $DB_SUB database connection
           # so as not to disturb our sweep through the reminders
           ns_db dml $db_sub "update bday_reminders
    set last_reminded = current_date
    where oid = '$oid'"
           incr email_count
        }
     }

     # we're all done, let's log the work
     ns_db dml $db "insert into bday_log (message_type, message_count, start_stamp,
    end_stamp)
    values
    ('$message_type',$email_count,'$start_stamp',current_timestamp(0))"
     # we call these directly because bday_sweep_all calls this fcn
     # three times in succession and otherwise AOLserver won't allow
     # the ns_db gethandle to go through
     ns_db releasehandle $db
     ns_db releasehandle $db_sub

}
```

An interesting highlight of this sweeping function is that we add "using (isolation level read uncommitted)" to the first SELECT. Illustra uses a page locking system rather than the row level locking selectable (per table) in heavier duty RDBMSs such as Informix and Oracle. We want to make sure that we don't deadlock ourselves by trying to read from the table and update it simultaneously. The "read uncommitted" tells Illustra not to grab any locks on the table when doing the SELECT.

Oh yes, in case you were wondering, we do have to define the bday_log table:

```
create table bday_log (
        message_type    text,
        message_count   integer,
        start_stamp     timestamp(0),
        end_stamp       timestamp(0)
);
```

CASE 4: THE BULLETIN BOARD

In 1982, I tried Usenet news for the first time. I'd just graduated from MIT and took up my cubicle at HP Labs in Palo Alto. Though I had access to good operating systems (TOPS-20 and the Lisp Machine), for some reason I decided to ask for an account on the VAX, which was running Unix. I think it was my friend Luigi who signed me up to net.jokes. After two days, I counted one funny joke, five not funny jokes, 20 people complaining that Joke X was inappropriate for the list because it was racist/sexist/not funny/whatever, and 80 people complaining that the complainers should not have complained.

My Concurrency Question

Suppose I have two database connections open.

- Connection 1 does a SELECT * FROM BIG_TABLE; and then is slowly sweeping through it. The value of the FOOBAR column is 56 for all rows.
- Just as Connection 1 is reaching the 1,000th row, Connection 2 does an UPDATE BIG_TABLE SET FOOBAR = 15 WHERE ROW_ID = 2000.

Connection 1 eventually reaches the 2,000th row, maybe one minute later. Does Connection 1 find that the FOOBAR column has the old value of 56 or the updated value of 15? And is this behavior required by the SQL standard?

Answer from My Friend Who Works at Oracle

"OK: When you execute a query, the first thing the kernel does is take note of the time you started the query. As the query progresses through the table, it will look at the SCN ('system change number') of each row, which indicates when it has last been updated. If it finds that the row was updated after the query began, the kernel goes to the rollback buffer to fetch the last value the row held *before* the query began. So in your example, Connection 1 will not see the update made by Connection 2 if the update is done after the query begins.

"In fact, even after the query finishes and you start another, it *still* won't see the change . . . not until you explicitly commit the update. (I shouldn't say 'explicitly' since closing the connection cleanly will also perform a commit.) But even if you commit while the query is running, it still won't get the new value. However, the next time you query, it will catch the change.

(Continued)

Oracle Exhibit 1011
Page 121 of 171

I was so impressed with Unix and Usenet that I went to talk to the sysadmin, a paunchy black-T-shirted guy who sat in his cube all day reading science fiction novels. He'd been saying for years that he would upgrade the DEC-20's operating system to Digital's latest release, but instead the stack of novels just grew larger. Even top managers didn't dare challenge his terrifying power. He knew PDP-10 assembler and they didn't. It took all of my courage to even approach the man, but I decided that conquering my terror would be worth it. "Would you please delete my Unix account?" I asked.

About a decade later, my 18th year on the Internet, my officemate at MIT showed me the alt.sex hierarchy. I had no idea . . .

(Continued from previous page)

"But wait, there's more. Let's say you don't do a 'commit' in Connection 2, but you enter the same query as before, this time in Connection 2. While that's running, you also run the query in Connection 1. The results will differ. Why? Because the query running in Connection 2 *knows* you've made an update, and sees those SCNs rather than the ones stored in the permanent table structure. (Technically, before a commit, the results of a write operation are stored in the ITL, or Interested Transaction Layer. A session has access to the portion of those results performed in that session *only;* these operations are 'presumably committed' for the purposes of whatever you do in that session. It cannot, however, see anything in other sessions, which is why Connection 1 has no clue what you've done until you truly commit it.) This illustrates one of the challenges of concurrent programming with databases. Lesson 1: Always commit your changes as soon as you know they're permanent. (Corollary: Always have frequent backups for those plentiful occasions when they weren't and aren't reversible.)

"Regarding locks (a subject which I know for a fact varies among database vendors), Connection 1 will not lock the table, unless the person executing the query has explicitly done so beforehand by doing 'select * from table_name for update'. If so, it would be released after an explicit commit is typed, and Connection 2 would wait and/or time out while the lock is held.

"Most likely, you would not want to lock an entire table for a query. The only reasons I can dream of doing this are if

- You need to run a very long query (many hours).

(Continued)

Oracle Exhibit 1011
Page 122 of 171

With this deep affinity for and experience with bboard systems, then, who could have predicted that one day I would write my own?

CAN 30 MILLION USENET USERS REALLY BE WRONG?

Given the high average quality of information on Usenet, the obvious intelligence of the people posting, the high performance and reliability of news servers, and the quality user interface of such programs as rn (the incredibly primitive original Unix news reader), it is tough to believe that anyone would dare try improving on Usenet. Nonetheless, there are a few areas in which a database-backed bboard can be an improvement over Usenet:

- Have an optional "mail me when a response is posted" field.

(Continued from previous page)

- There are tons of transactions going on simultaneously.
- Those transactions are less important than the query results.
- You have a *very* small amount of rollback space.

"Remember that point where it finds a newer SCN and fetches the old value from the rollback? If the rollback doesn't go back far enough, the query errors out with an 'ORA-1555 : snapshot too old' error. So as long as the entire table isn't locked, and no other connections are locking the rows you need to update, Connection 2 will do its update right away.

"When Connection 2 does update, it will either lock the entire table or just the row it is trying to update, depending on the database's system and session parameters, such as whether something like 'ROW_LOCKING=ALWAYS' is in your oracle.ini file. I think the default is to lock the table, but don't quote me on that (it would be silly if we did, since row locking is a big thing for Oracle).

"The question of yours regarding the SQL standard: That I don't know. I've never read the ANSI standards for SQL or SQL92, but I do not believe there are any rules regarding the behavior of concurrent sessions. If there are, though, I'm fairly confident we're doing what it says. (If we weren't, we'd be fixing it, and I've heard of no intentions to do so.)"

My Conclusion

Thank God I don't have to write my own RDBMS.

Oracle Exhibit 1011
Page 123 of 171

- Provide full-text indexing (assuming your RDBMS supports it).
- Control access via all the usual HTTP mechanisms,
- Do secure transmission of data to and from the bboard via SSL.
- Use the admin pages to delete stale/ugly/whatever messages.

If you have enough money to buy and maintain an RDBMS, it is a fairly safe bet that you have enough money to buy Netscape News and Collabra Servers. These are reasonably full-featured products. However, they don't come with source code or much customization ability. If you want to learn how to design and build a collaboration environment that suits your users, keep reading . . .

STEP 1: THE DATA MODEL

```
create table bboard (

        msg_id          char(6) not null primary key,
        refers_to       char(6),
        email           text,
        name            text,
        one_line        text,
        message         text,
        notify          boolean default 'f',
```

Where's the Beef?

The only thing hard about building a threaded bboard system is sorting into threads. If all you have is a binary "response to" relation (that is, that Message N is a response to Message J), then you need to calculate the transitive closure of this relation to find the thread. I like using the phrase "transitive closure" because it makes me feel that computer science graduate school *really was* the best ten years of my life. Anyway, the transitive closure of a relation X is just the set of things that are related under relation X.

So what?

Well, transitive closure in the worst case (where all the messages are one thread) provably requires computing resources that grow as the cube of the number of messages ("order *n*-cubed" or $O[n^3]$). That's bad. It could be very expensive for the server and slow for the user if one had to calculate the threads every time the bboard was surfed.

My program cheats by calculating a sort key every time a message is added to the bboard. Then the thread sort is done with a simple SQL "order by" clause. This kind of sorting is "$O[n \log n]$" or maybe 100 times faster for a moderate-sized bboard.

```
        posting_time    timestamp(0),
        sort_key        text

);

create index bboard_index on bboard using btree ( msg_id );

create table msg_id_generator (

        last_msg_id     char(6)

);

insert into msg_id_generator values ('000000');
```

If you are using Oracle or Sybase, you'll need to use VARCHAR(2000) instead of TEXT for some fields and will have to decide if you want your message column to be LONG or VARCHAR (limited to 2,000 bytes in Oracle).

STEP 2: LEGAL TRANSACTIONS

There is only one legal user transaction:

```
begin transaction;
select unique last_msg_id from msg_id_generator
using ( lock = table, exclusive);
update msg_id_generator set last_msg_id = '00001n';
insert into bboard
(msg_id,refers_to,email,
name,one_line,
message,
notify,sort_key,posting_time)
values
('00001n',NULL::char(6),'philg@mit.edu',
'Philip Greenspun','generating some queries for my book',
'I''m posting this so that I can look in the server.log
and harvest the queries for my book...',
't','00001n',current_timestamp(0));
end transaction;
```

The first portion of the transaction is a bit odd. For message IDs, I'm using six-character strings. This bboard has been in use for awhile so the latest message had an ID of 00001m. The next ID after that is 00001n, which is what our new message will bear. Each character cycles through the integers, then the capital letters, then the lowercase letters. So after 00001z the next message is 000020, nine messages later, 000029, then 00002A, and 00002Z will be followed by 00002a.

This kind of sequence is rather tough to generate in standard SQL so I do it all in Tcl. However, that means that I have to read from the msg_id_generator table, do some computation, then write to the msg_id_generator table. That's the classic recipe for deadlock. We have to tell the database to lock the msg_id_generator table immediately and not start any other transactions using this table. That's why I added the "using (lock = table, exclusive)" clause to the first SELECT. (Note: in Oracle, we'd just use "SELECT .. FOR UPDATE" to accomplish the same goal.)

Sort keys deserve some discussion. The sort key for a top-level message like this is just the message ID, 00001n. For messages that respond to the top level, the sort keys are 00001n.00, 00001n.01, . . . , 00001n.09, 00001n.0A, . . . , 00001n.0Z, 00001n.0a, . . . , 00001n.0z, 00001n.10, . . . There are 62 possibilities for each character and two characters per thread level. So any one message can have 3,844 immediate responses. A response to a response will have a sort key with two extra characters, for example, 00001n.0900 would be the first response to the ninth response to top-level message 00001n.

This sort key enables me to get the messages out in thread-sorted order with

```
select msg_id, one_line, sort_key from bboard order by sort_key
```

and the level of a message can be calculated simply as a function of the length of the sort key.

The administrator is able to delete messages subject to the constraint that threads or subthreads must be removed together. In other words, it is not possible to remove a message if it has dependent messages. The user interface offers an option "Remove this message and its four dependents?" That way we make sure that no message ever has a REFER_TO column pointing to a nonexistent message.

Here's the kind of deletion that works:

```
delete from bboard b1
where msg_id in
 (select b2.msg_id from bboard b2 where b2.sort_key like '0000e6.%');
```

Just to be on the safe side, we've used correlation names for the two different uses of the table bboard. Because of the way the sort keys are constructed, this query will remove the message with ID 0000e6 and the entire thread that it supports.

STEP 3: MAPPING TRANSACTIONS ONTO WEB FORMS

I initially conceived my bboard system as a kind of runt Usenet. I went to a lot of trouble to make fully threaded discussions work and was going to deliver all of that power to users. So I dragged out the big Web user interface iron: frames. I'd use the top frame to show subject lines and the bottom

frame to show the full message, just like all the Usenet clients (including the one built into Netscape Navigator itself). See Figures 13.9 and 13.10 for examples of Usenet and my bboard software.

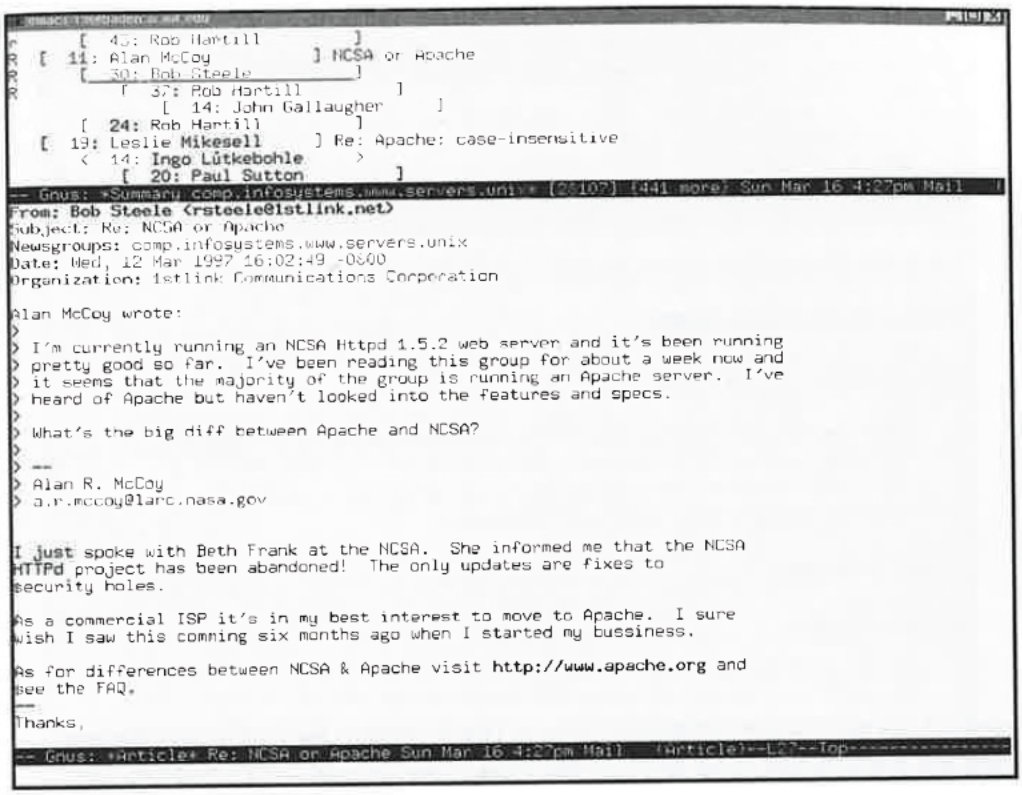


Figure 13.9: A view of the Usenet group comp.infosystems.www.servers.unix, presented by GNUS, the news reader built into GNU Emacs (a good programmer never leaves Emacs for any reason!). Subject lines for about 450 messages are available in the top window, the full text of one posting in the bottom window. Here Alan McCoy of NASA's Langley Research Center has asked about the difference between NCSA 1.5 and Apache. Bob Steele, who apparently runs a commercial ISP, tells him that NCSA has shut down their HTTP server project and points McCoy to the FAQ at www.apache.org. Despite what you might hear from politicians, this kind of exchange constitutes the overwhelming majority of Network News Transfer Protocol (NNTP) traffic.

After observing the forums in operation for a while, I realized that actually threads were annoying and overly complicated. A bboard anchored to some Web content was best presented as a question and answer forum. The questions would mostly be directed at the author but other readers might wish to answer as well. It would be a real user-interface convenience to be able to see the question and all the answers on one page. See Figure 13.11 for an example.

Oracle Exhibit 1011
Page 127 of 171

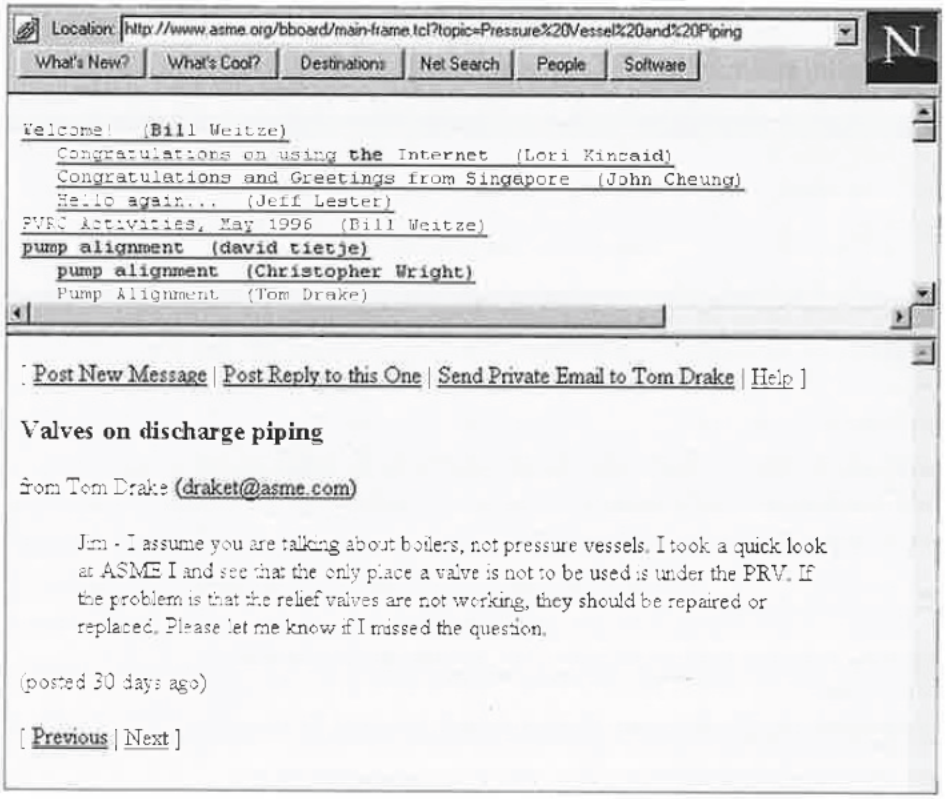


Figure 13.10: My bboard software in use at The American Society of Mechanical Engineers (http://www.asme.org/). ASME runs more than 40 public and private forums, mostly using the full threads interface. Casual inspection of the topics, though, reveals that most discussions have only two levels (question, answer, answer, answer, question, answer, answer, answer, . . .).

For a few minutes, I thought about starting from scratch to write a Q&A forum. Then I realized that I could drive the same data model from both threaded and Q&A user interfaces. Furthermore, I could make them interoperate. A posting from the threaded interface would be visible in the Q&A interface and vice versa!

How did I do it? An answer added from the Q&A interface has a subject line of "Response to $original_question_subject_line" automatically inserted. This is never displayed to users of the Q&A interface but threads users will see it in the subject frame. An answer added from the threads interface will have a custom subject line. This is displayed in boldface between answers in the Q&A interface.

Oracle Exhibit 1011
Page 128 of 171

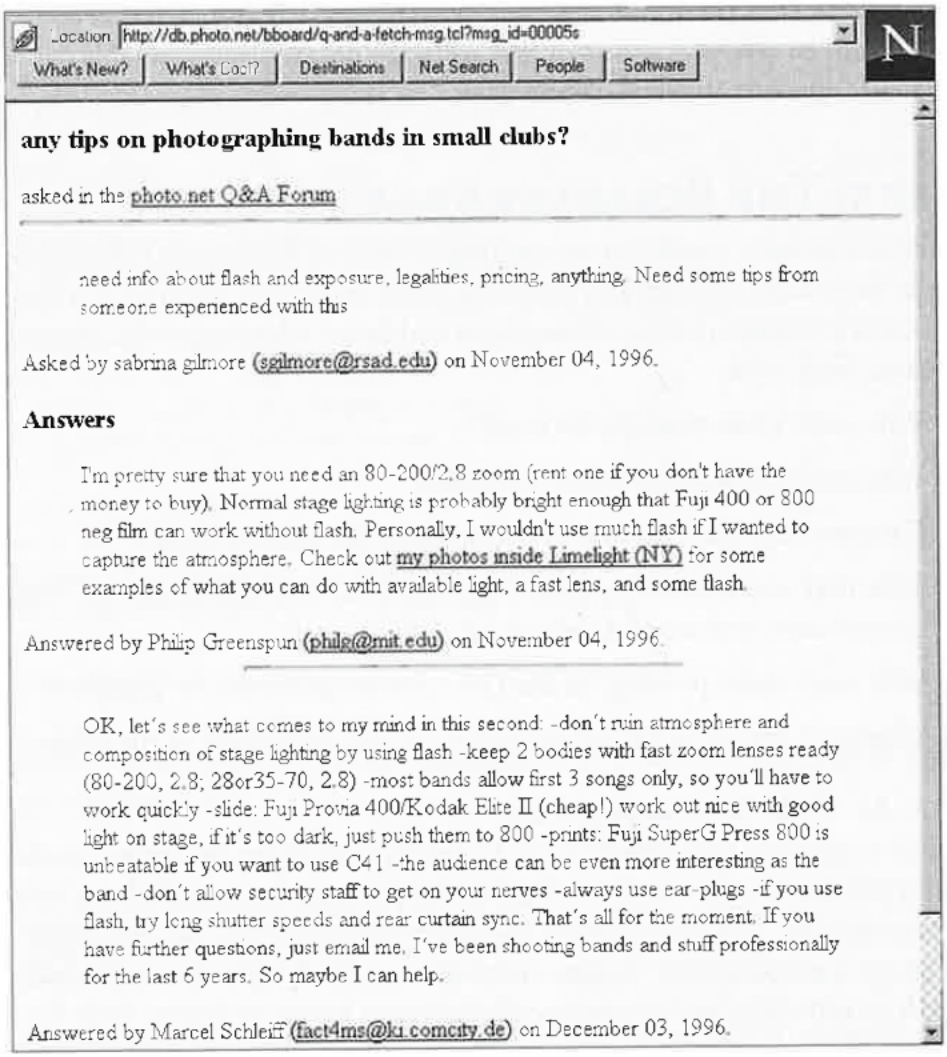


Figure 13.11: A typical question and answer in the http://photo.net/photo/ Q&A forum. You can see the question and all the responses on the same page. Note that I personally answered the question on November 4, 1996. One month later, a much more experienced photographer from Germany contributed a much better answer. Both answers were sent by e-mail to the woman who asked the question. This interoperates with the threaded discussion user interface against the same data model. That is, a posting in the threaded interface is visible in the Q&A interface and vice versa.

Oracle Exhibit 1011
Page 129 of 171

STEP 4: WRITING CODE TO PROCESS THE FORMS

I'm reluctant to give you a tour of this software because it is available for download from http://demo.webho.com.

CASE 5: THE BULLETIN BOARD (AGAIN)

The bboard project taught me something profound: When you build something in three days, you tend to leave out some important features. I released the bboard software to what I thought would be an adoring public. Instead they came back with

- Why can't I run multiple forums?
- Why can't I password-protect a forum?
- Can you send me e-mail every time there is a new message?
- Now that you've added password protection, why can't I use the Web server's user/password database for authentication?
- Why can't older postings in the Q&A forum be sorted by category?
- Why can't you show the poster's name or e-mail address in the subject line?

STEP 1: THE DATA MODEL

In order to support multiple forums, I need to store some information about each forum. No longer can I assume that the person who installed the software is the moderator. I need to support distributed moderation. So we at least need to store a maintainer_name, maintainer_e-mail, and admin_password for each group. Also, each forum needs to have a name and some links back to the static content with which it is associated. Finally there are some columns associated with Q&A presentation. Here's a new table in which information about bboard_forums is represented:

```
create table bboard_topics (
  topic          text not null primary key,
  backlink       text, -- a URL pointing back to the relevant page
  backlink_title text,  -- what to say for the link back
  blather        text, -- arbitrary HTML text for top of page
  admin_password    text,
  user_password     text,  -- for restricted bboards
  ns_perm_group     text,  -- for using AOLserver's permissions system
  -- non-null if we added a custom group when creating this topic
  ns_perm_group_added_for_this_forumtext,
  maintainer_name            text,
```

Oracle Exhibit 1011
Page 130 of 171

```
  maintainer_email      text,
  subject_line_suffix   text,   -- e.g., 'name'
  -- send email to the maintainer when a message is added?
  notify_of_new_postings_p    boolean default 't',
-- HTML encouraging user to search elsewhere before posting
  pre_post_caveat             text,
  -- present the Q&A or threads from the top level list?
  q_and_a_primary_p     boolean default 'f'
  -- stuff just for Q&A use
  -- note that the sort_order has an integrity constraint that
  -- prevents it from containing values other than 'asc' and 'desc'
  q_and_a_sort_order    text default 'asc' not null
        check (q_and_a_sort_order in ('asc','desc')),
  q_and_a_categorized_p boolean default 'f',
  -- defines what is considered "new" and therefore displayed above
  -- that older, categorized, questions
  q_and_a_new_days      integer default 7,
  -- at posting time, do we ask the user to suggest a category?
  q_and_a_solicit_category_from_user boolean default 't',
  -- do we allow users to type in new categories?
  q_and_a_categorization_user_extensible_p boolean default 'f'
);
```

Now that bboard messages can be categorized, we need to keep track of what the legal categories are:

```
create table bboard_q_and_a_categories (
   topic         text not null references bboard_topics,
   category      text not null,
   primary key (topic, category)
);
```

Note the integrity constraint that rows here must have a TOPIC column value that is present in bboard_topics. Also note the primary key constraint that prevents two rows from having the same TOPIC and CATEGORY.

We'll have to add two columns, TOPIC and CATEGORY, to the BBOARD table:

```
create table bboard (
  msg_id        char(6) not null primary key,
  refers_to     char(6),
  topic         text not null references bboard_topics,
  category      text,  -- only used for categorized Q&A forums
  email         text,
  name          text,
  one_line      text,
  message       text,
```

Oracle Exhibit 1011
Page 131 of 171

```
  notify        boolean default 'f',
  posting_time  timestamp(0),
  sort_key      text
);
```

Note that the CATEGORY column is only relevant for top-level posts in Q&A forums.

It was always my intent that the bboard system support casual collaboration by sending posters e-mail copies of responses to their messages. Also, the forum maintainer can opt to receive immediate e-mail notification of new postings. However, a lot of users asked for periodic summaries of postings. Here's the table that registers their interest:

```
create table bboard_email_alerts (
       email   text not null,
       topic   text not null references bboard_topics,
       -- we set this to 'f' if we get bounces
valid_p  boolean default 't',
       frequency text,
       keywords  text
);
```

Fundamentally this registers EMAIL's interest in TOPIC. Frequency is one of instant, daily, Monday/Thursday, or weekly. If I sign up for a weekly alert for the photo.net topic then I'll get a summary prepared early Monday morning. If I add a string to the KEYWORDS column, I'll get only those new messages that also match in the full-text index (see below). For example, I can restrict any frequency alert to only return messages related to "nikon." Instant alerts are e-mailed individually as new messages are posted.

We need a housekeeping table to implement the alerts. This contains the last time that alerts were mailed, used when querying for new messages, plus a count of all the messages that have been sent.

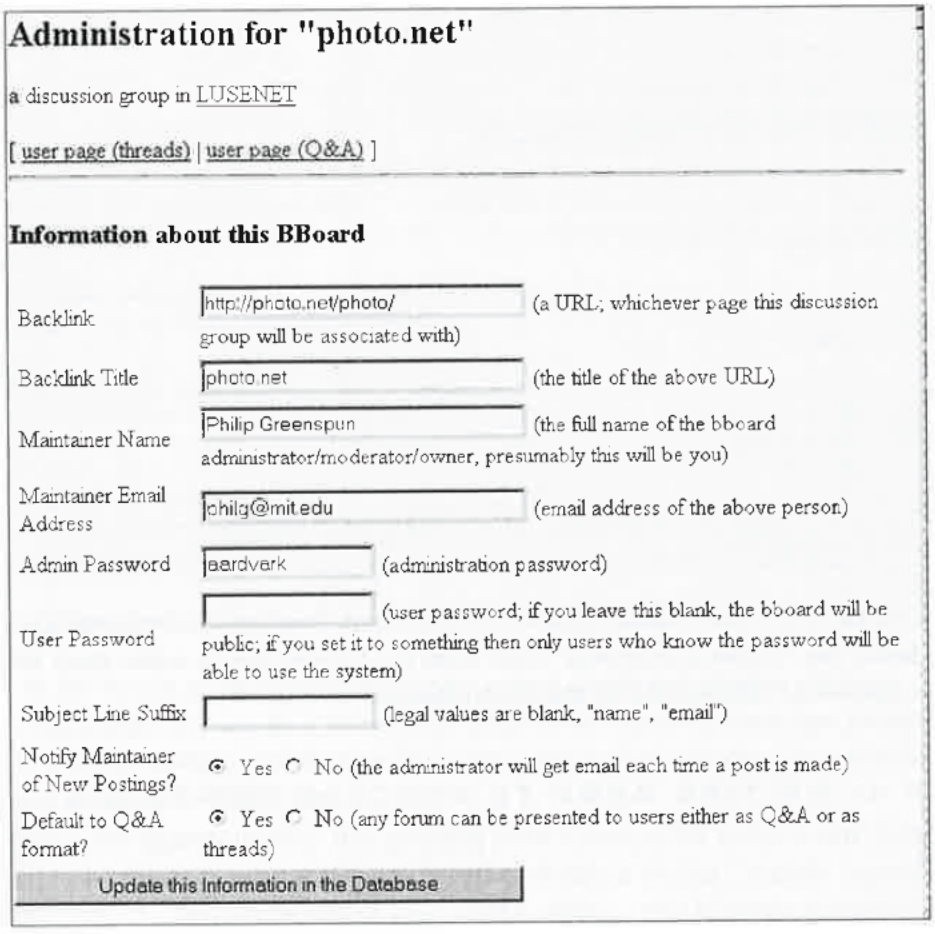
```
create table bboard_email_alerts_last_updates (
       weekly timestamp(0),
       weekly_total integer,
       daily timestamp(0),
       daily_total integer,
       monthu timestamp(0),
       monthu_total integer
);
```

STEP 2: LEGAL TRANSACTIONS

Not much has changed. If the user provides a category when posting a new question from the Q&A interface, we insert that into the bboard table. There are a bunch of new admin functions to manage categorization. There are a bunch of obvious transactions on the alerts table when users add and disable alerts.

STEP 3: MAPPING TRANSACTIONS ONTO WEB FORMS

Nothing too surprising here. I'll let the figures (13.12, 13.13, and 13.14) do the talking.



Administration for "photo.net"

a discussion group in LUSENET

[user page (threads) | user page (Q&A)]

Information about this BBoard

Backlink	http://photo.net/photo/	(a URL; whichever page this discussion group will be associated with)
Backlink Title	photo.net	(the title of the above URL)
Maintainer Name	Philip Greenspun	(the full name of the bboard administrator/moderator/owner, presumably this will be you)
Maintainer Email Address	philg@mit.edu	(email address of the above person)
Admin Password	aardvark	(administration password)
User Password		(user password; if you leave this blank, the bboard will be public; if you set it to something then only users who know the password will be able to use the system)
Subject Line Suffix		(legal values are blank, "name", "email")
Notify Maintainer of New Postings?	(•) Yes ( ) No	(the administrator will get email each time a post is made)
Default to Q&A format?	(•) Yes ( ) No	(any forum can be presented to users either as Q&A or as threads)

Update this Information in the Database

Figure 13.12: Top of the main administration page for the photo.net forum. Note that the administration can choose whether or not the Q&A interface is primary. There is a separate administration section for items that are only relevant to the Q&A code (see Figure 13.13).

Oracle Exhibit 1011
Page 133 of 171

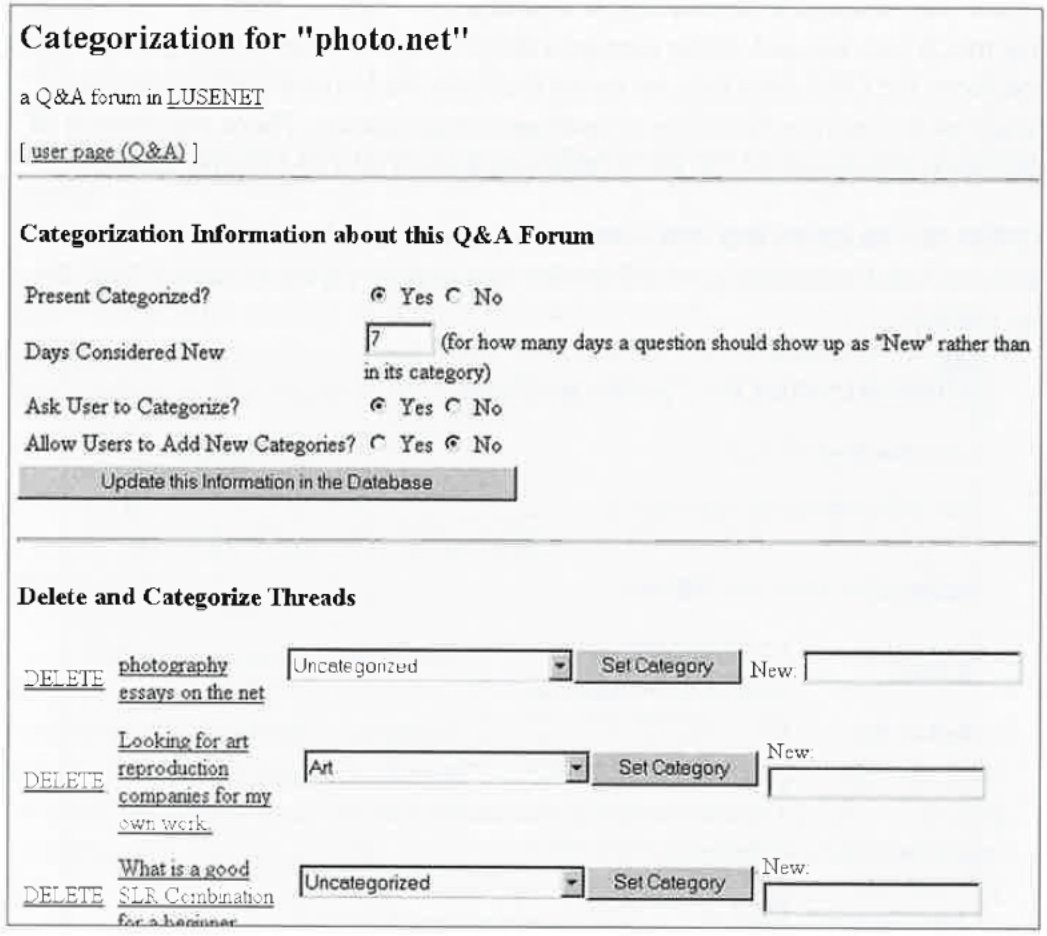


Figure 13.13: Q&A forum administration for photo.net. The top-level threads are listed below the "forum personality" slots from the bboard_topics table. Each thread can be deleted or categorized by the administrator.

STEP 4: WRITING CODE TO PROCESS THE FORMS

I noticed that a lot of naïve users were posting the same message two or even three times. Hence I added a check to insert-msg.tcl to see if there was already a row with the same TOPIC, ONE_LINE, and MESSAGE:

```
if [catch { set n_previous \
                [database_to_tcl_string $db \
                                    "select count(*) from bboard
where topic = '$QQtopic'
and one_line = '$QQone_line'
and message = '[bboard_convert_plaintext_to_html $QQmessage]'"]] \
    errmsg] {
    ns_log Notice "failed trying to look up prev posting: $errmsg"
```

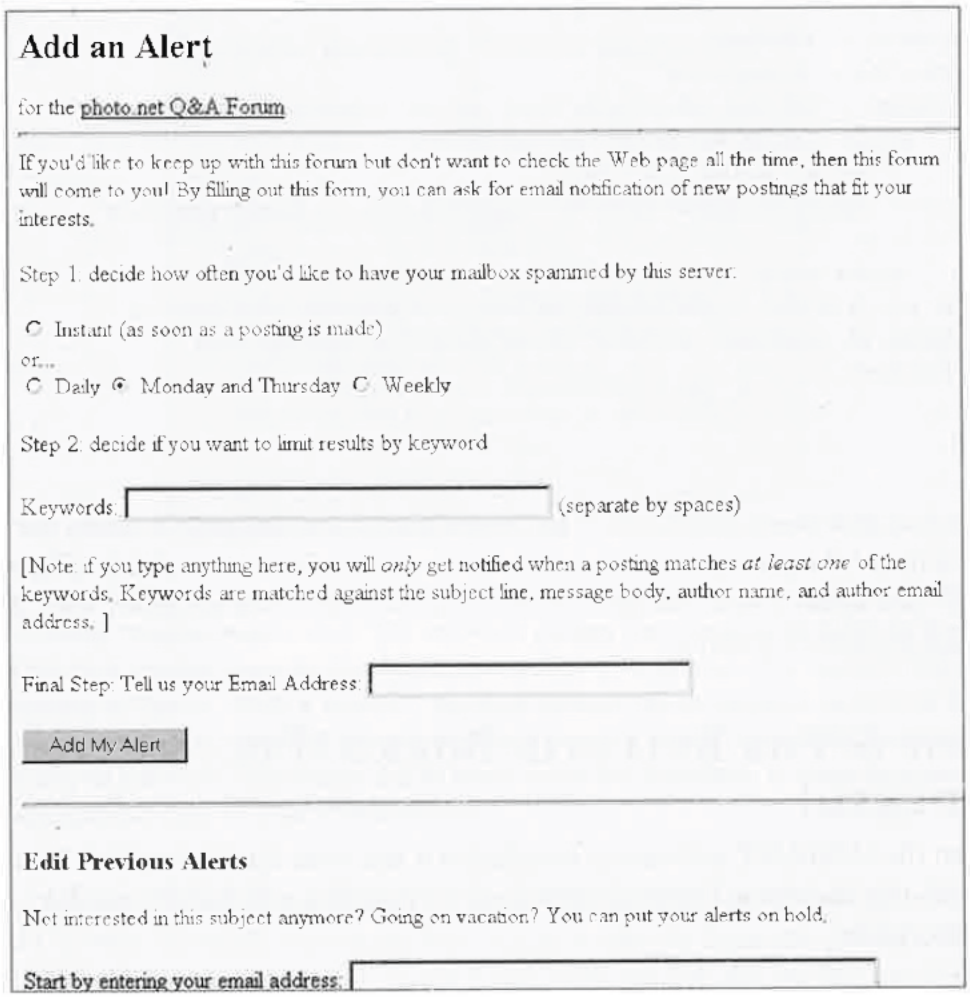


Figure 13.14: Forms to let users specify new e-mail alerts and disable old ones. Instant notification is done by a thread that stays alive following each new posting to the forum. Daily, weekly, and Monday/Thursday notifications are done by a procedure scheduled inside the AOLserver.

```
    } else {
    # lookup succeeded
    if { $n_previous > 0 } {
       incr exception_count
       append exception_text "<li>There are already $n_previous
messages in the database with the same subject line and body.
Perhaps you already posted this?  Here are the messages:
<ul>
"
        set selection [ns_db select $db \
                           "select name, email, posting_time
```

Oracle Exhibit 1011
Page 135 of 171

```
from bboard
where topic = '$QQtopic'
and one_line = '$QQone_line'
and message = '[bboard_convert_plaintext_to_html $QQmessage]'"]
        while {[ns_db getrow $db $selection]} {
            set_variables_after_query
            append exception_text "<li>$posting_time by $name ($email)\n"
        }
        append exception_text "</ul>
If you are sure that you also want to post this message, then back up
and change at least one character in the subject or message area,
then resubmit."

    }
}
```

Note that this check doesn't have too much confidence in itself. It offers the user a list of the previous messages and ultimately relies on human intelligence. Input validation is nice but there is always a risk that it will outsmart itself and squash legitimate postings.

CASE 6: THE BULLETIN BOARD (FULL-TEXT INDEXED)

When the LUSENET software is installed at a site with the Illustra PLS Blade for full-text indexing, I take advantage of it by adding a PLS index on the bboard table:

```
create index bboard_pls_index on bboard using pls
( one_line, message, email, name );
```

This means that all the words in all four of the named columns will be indexed. How do you query using the index? Here's an example:

```
select msg_id, one_line
from PlsQueryOrdered('bboard_pls_index',
                     'Canon macro lenses')::setof("Table bboard")
where topic='photo.net';
```

This user is looking for a discussion of "Canon macro lenses" in the http://photo.net/photo/ bboard (see Figure 13.15). The function PlsQueryOrdered returns a set of rows from the bboard_pls_index that match this query, ordered by relevance. For obscure reasons known only to Illustra, these must be explicitly typecast to "set of rows from the bboard table" with

```
setof("Table bboard")
```

Oracle Exhibit 1011
Page 136 of 171

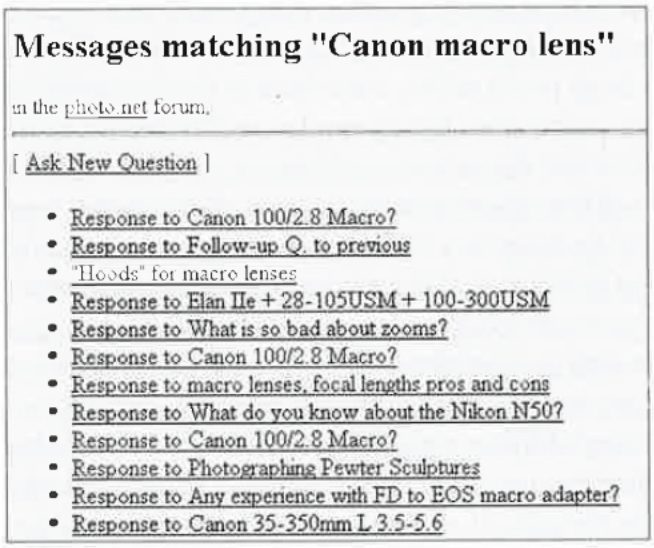


Figure 13.15: The brilliant Illustra/PLS Blade finds all the messages relevant to the query string "Canon macro lens." My software makes each search hit a link to the full question and answer page so that users see answers in context. The subject line "Response to Canon 100/2.8 Macro?" appears several times because several of the six responses to this message are relevant to the query string. I could reformulate the query to eliminate duplicates but in some ways the repetition is good because it encourages the user to look at that thread.

After the cast, we can do anything we want with these rows. We can restrict the SELECT with WHERE clauses, in this case to messages from the photo.net topic. We can restrict the columns retrieved with the FROM list. We can JOIN with another table.

Although in general Illustra/PLS is clever enough that I don't have to even try to be clever, I did do one intelligent thing in my implementation of full-text search for my bboard system. In the Q&A interface, when the user's query matches a response to a message, clicking on the search hit takes the user to the standard page with the full question plus all the responses. In the threads interface, the user gets the option of seeing the full thread or just the message in question.

CASE 7: CLASSIFIED ADS

Once you have an RDBMS that can index the strings, it would be a crime if you didn't build a good classified ad system. I've built about five classified systems and think I've learned one or two useful things.

First is a publishing philosophy point. People who come from the dead-trees world always want to charge money for online classifieds. Their dead-trees publication makes a huge profit selling classifieds to readers and they think that they can cup their palms then lap up similar profits online. In meetings with such folks, I'd point out the existence of Usenet. Anyone who is online has access to hundreds of free classified services, including Usenet newsgroups. Why should they pay to be listed in a Web classifieds system that surely gets less traffic than ba.market.vehicles (1,000 messages a month; *ba* stands for "Bay Area"—San Francisco, that is) or rec.photo.marketplace (5,000 messages a month)?

It then occurred to me that there was a deeper point to be made. Given that Web publishers pay nothing for paper and printing, someone placing a classified ad is doing the publisher a big favor. The ad placer is contributing content. The publisher can use that content to build traffic, sell ads, and lure readers into other services. If anything, the publisher should be paying advertisers to post classifieds.

That's Useful Thing 1: Don't think about charging money unless you have some kind of monopoly.

My second epiphany is that full-text search engines work well enough that you don't need much structure in your data model. Typical RDBMS-backed Web sites appear to have been developed by 15-year-olds who unpacked their Oracle boxes and said, "Cool, I can have lots of different columns and then limit my searches based on values in those columns." The casual user of such a site is very quickly confronted with a 15-input search form. Fifteen inputs to fill out? All the user wanted to do was browse around and see what was in the system. After making those fifteen choices, the complicated query is submitted to a relational database management system that works heroically to produce...zero matches.

This design might have been great if there were 275,000 ads in the database, but not if there are only 300. For some kinds of classifieds systems, personals, for example, part of the fun is browsing ads that aren't 100-percent relevant.

Oracle ConText Instead

With Oracle ConText, the syntax is a bit hairier for creating the index:

```
execute ctx_ddl.create_policy('BBOARD_POLICY', BBOARD.MESSAGE);
execute ctx_ddl.create_index('BBOARD_POLICY');
```

but then somewhat simpler for queries:

```
select * from BBOARD where contains(MESSAGE, 'Canon macro lenses')>0;
```

Forms that encourage overconstrained searches rob users of their fun. Furthermore, the heavy structure of the data model imposes a burden on users when they are entering ads.

Useful Thing 2: Don't add columns to your data model if you won't be allowing users to constrain searches based on those values.

At this point, you're probably nodding your head and wondering how someone who belabors the obvious to the extent that I do can get published on real dead trees. Yet in February 1997 I was asked by a publisher, call them Naif News, to evaluate a bunch of commercial classified ad systems. Naif News publishes more than 100 small newspapers so they were being wooed heavily. Millions of dollars were about to change hands. The vendors had already sold systems to dozens of newspapers and were online with graphics-heavy user interfaces.

What did I find? All of these vendors had fallen into the "15-year-old with Oracle" trap. It was very difficult to construct a query that would return any ads at all, even when they had thousands of ads in their database. Furthermore, their structured systems weren't able to use the structure to do simple things such as return Chevrolet and Chevy ads to someone querying for "Chevy" in the manufacturer column (any full-text search engine would have figured out for itself that Chevy and Chevrolet are related words after seeing a few hundred ads).

Enough about how stupid everyone else is . . . let's talk about how stupid I am.

CLASSIFIEDS IDEA: CATEGORIZE

My generic classified ads system depends first upon categorization of ads. User interface research back in the 1960s concluded that humans are very bad at formulating boolean queries, for example "Nikon AND NOT microscope AND NOT 'integrated circuit mask stepper'". The Holy Grail of information retrieval was supposed to be the full-text search engine that is so common today. Just type in a string that reflects your curiosity, look at some results, then refine by picking a good result and asking for "more like this."

Why then categorize? Why bother to make users of my http://photo.net/photo/ classifieds say whether they are placing an ad for Underwater Equipment or Darkroom? Why can't the search engine direct people to appropriate ads?

It turns out that users have trouble coming up with that initial query string. It is a lot easier to browse than to think. It is also a lot more efficient for the RDBMS to grab all ads in the Underwater category rather than send PLS off looking for "flash for my Nikonos V." There are only about ten ads in the Underwater section most of the time so why not let the user see all of them?

Oracle Exhibit 1011
Page 139 of 171