Page 1 July 2006 Volume II Issue 3 ORAtipsJournal

The Life of an Email

By Steve Callan

Editor's Note: Having just returned from vacation, I had over 600 emails in my inbox. 523 of them were marked SPAM and rightly so. If you can relate, please keep reading. Have you ever wondered why and how you get email you don't want, and more importantly what your options are to get rid of it once and for all. ORAtips Associate Editor Steve Callan explores the life of electronic mail, and while it is not Oracle specific, it's definitely an ORAtip worth sharing.

Introduction

Chances are you are reading this article in ORAtips because you clicked on a link in an email. As far as complexity is concerned, the email informing you about the July issue of ORAtips falls into a category known as a service mailing. Other types of emails have transactional components, that is, clicking on a link takes you to an account management or customer service landing page. In any event, when you open an email, a sender can learn a great many things about you. Facts reported back to the sender include the fact you opened the email in the first place (or did not receive it at all), but did you know that the sender can determine how many times you opened an email, track which links you clicked, measure how long the mail was open, and even measure how long your mouse hovered over a link?

Let's start off with an overview about the lifecycle of an email, and then go into database and application design strategies.

Background Information

Databases play an integral part in the lifecycle. It's not just the fact there are databases involved, but also the types of databases involved, namely production and warehouse. On the production side, use of the database can be divided into two types of schemas. What differentiates the types of schemas is personalization. When you open an email and see "Dear John," (no, not that type of "Dear John"), your email has been personalized, and that personalization information is stored in a production database. The other type is impersonal and is easily identified by greetings such as "Dear Valued Customer" or even no greeting at all.

Unfortunately, not all ISPs are created equal, and those willing to take money from "blast houses" are part of the reason why we have junk mail folders at work and at home.

If the email vendor or provider maintains personalization information, then it is very common for the vendor's client to provide personalization details (where you originated from, sign up date, first and last name, address, and preferences, to name a few personal items). The email vendor receives updates from the client (plus information about new customers), and typically never removes the customer from the database. It would seem obvious that if a client is no longer a customer or does not want to be contacted again that the record should be deleted. However, there are several reasons why such a record will not be removed. Two of the reasons are based on customer service and the CAN-SPAM act of 2003.

The Customer Service Side

To track non-mailable email messages, one approach involves keeping the email address in the database and assigning a suppression flag. To maintain good customer relations, originating senders (i.e., the email provider's customer) do not want to alienate customers. Due to email's more impersonal nature, a customer is likely to be more, shall we say, "vocal" concerning his or her irritation about being contacted or spammed. If the sender is not a brick and mortar store, there is virtually no face-to-face contact and the sender can do very little to rescue or rehabilitate an alienated customer.

Another aspect of the customer service perspective is the relationship between an email vendor and its Internet Service Providers (ISP). Email vendors generally want to maintain a good relationship with their servicing ISPs, and that includes not being labeled or identified as a spammer. Unfortunately, not all ISPs are created equal, and those willing to take money from "blast houses" are part of the reason why we have junk mail folders at work and at home.

The Legal Side

One of the CAN-SPAM act's provisions is stated below.

It requires that your email give recipients an opt-out method. You must provide a return email address or another Internet-based response mechanism that allows a recipient to ask you not to send future email messages to that email address, and you must honor the requests. You may create a "menu" of choices to allow a recipient to opt out of certain types of messages, but you must include the option to end any commercial messages from the sender.

The CAN-SPAM act allows for financial penalties to be levied against offending senders, so there are compelling reasons to maintain a clean list of addresses. Part of the cleaning process includes good data hygiene, and this is one area where roles and responsibilities between the vendor and sender need to be clearly defined. Which party is responsible for suppressing known opt-outs? Does the vendor have permission to alter badly formed addresses, and if so, what are the rules? Did a customer submit an address of john@yazoo.com? It would seem obvious that the domain should be yahoo.com, but how do you really know that? Is yazoo.com a valid domain? In this case the answer is yes. So, in your noble effort to perform hygiene on a list, you just potentially sent spam.

As an email vendor, there are two direct ways to allow a recipient to opt out of future mailings. Down at the bottom of the email (from "good" senders), you'll typically see a legal disclaimer which includes not only a physical address, but also a link to unsubscribe. The link-based unsubscribe method may be one of three types: one, two and three-click.

Opting Out

In the one-click method, clicking the link typically takes you to a landing page, and the page has immediate feedback ("Your email address of **you@domain.com** has been unsubscribed."). If you view the source behind the page, you may see HTML name-value pairs like addr=you@domain.com&unsub=yes or "address_id=124xyz&unsub=1." If using a one-click unsub method, the sender probably has some knowledge about your account.

In a two-click opt-out process, you are likely to have a more detailed relationship with the sender. You still go to a landing page, but this page may try to "rescue" you. Having failed to entice you to stay or abort the unsub process, the next click will finalize the process.

Fact two is the sender knows whether or not you opened the email (and how many times).

The three-click method also entails a relationship, but not quite as strong as found in the two-click method. The page containing the last click (typically containing a Submit button) will also have a field where you enter your email address. This method sends an email back to a customer service entity, and the entity may or may not know specific details about your unsub request. The basic amount of information may be nothing more than a request to be unsubscribed from Acme's mailing list. Where things get confusing is if you receive separate product mailings from Acme. Did you mean to unsubscribe from all of Acme's mailings (a global opt-out), or just this particular product line-based mailing (a product level opt-out)? Whoever designs the landing page should include detailed behind-the-scene information in the body of the opt-out email.

Report Metrics

Email is cheap (and effective), but not free. Acme just sent out 7 million emails and probably paid a good bit of money to do so. How did this particular campaign perform? This stage in the life of an email is where a data warehouse comes into play. Let's divide up the states or facts of an email after it has been sent.

Fact one is the sender knows whether or not the send to an address was successful. The email made it into your mailbox or it bounced (more on bounces in a moment).

Fact two is the sender knows whether or not you opened the email (and how many times). How is this accomplished? One practice is to embed a one-by-one pixel image in the body of the email. Downloading that image triggers a message back to the sender that the email was opened. Outlook, Hotmail, Yahoo and many other email providers allow for varying degrees of privacy or protection, one of which is being able to block images. If you want to reduce the amount of junk mail, do not open, as an example, "$1500 Emergency Payday C-A-S-H" emails. If you do open suspect emails, do not allow images to automatically appear.

July 2006 Volume II Issue 3

ORAtipsJournal

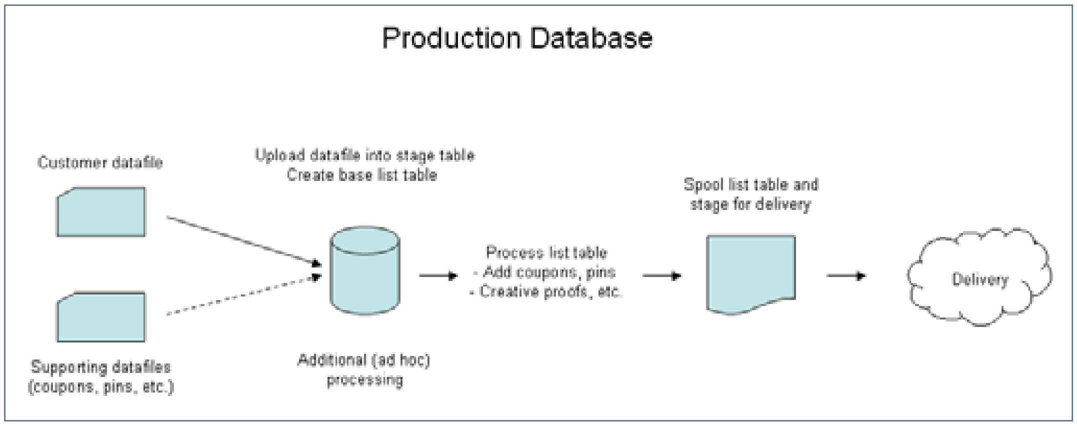
Production Database
Customer datafile
Upload datafile into stage table
Create base list table
Spool list table and stage for delivery
Process list table
- Add coupons, pins
- Creative proofs, etc.
Delivery
Supporting datafiles (coupons, pins, etc.)
Additional (ad hoc) processing

Figure 1: Production Database Workflow

Fact three is that you clicked on a link within the email. What happens when you click a link is what separates the good guys from the bad guys. The good guys will redirect you to an intermediate page which page records the hit (and where from), and then send you on to the "real" page. The bad guys (the phishers) don't redirect you, but instead, use "real looking" text for the link, and the link takes you to bad guy land where you enter your account number and password to help "protect" your account from some purported threat. Nothing new about all of this, but it does lead up to how a data warehouse comes into play.

Fact four, and one that advertisers really like, is the capture of which links your mouse hovered over. More sophisticated vendors embed timers in the message body, and how long you hovered on a link in an ad banner is somewhat indicative of the amount of time you spent at least considering whether or not to click the link. Which ads are the most appealing, and therefore, which ads are the ones you are likely to read and mouse over, which in turn leads to the question of what to charge for advertisement placement (those near the top cost more) are questions and answers of relevance here.

Finally, the deliverability status notification (DSN), particularly those for bounces, is also recorded. Senders and ISPs obviously prefer high deliverability/low bounce rates, and a bounce rate of around 5 per cent is indicative of a hygienic list. Vendors referred to as blast houses, as you can imagine, tend to have much higher bounce rates. The bounce rate for this article is likely to be quite low, because after all, if you're paying for a subscription type newsletter or journal, what you enter for an email address is very likely to be quite accurate.

More sophisticated vendors embed timers in the message body, and how long you hovered on a link in an ad banner is somewhat indicative of the amount of time you spent at least considering whether or not to click the link

Data/Workflow on the]\ Production Database

At this point, a picture is useful in presenting an overview of the front-end part of the lifecycle. Figure 1 illustrates a high-level representation of the process.

In an email-vendor-maintained database, the datafile from the customer will contain the agreed upon primary key/ID value and other data elements (promotional codes, welcome message codes, coupon recipients, preferences, and so on). Email address and personalization information is then derived from the subscriber database, which entails at least one join (a customer or subscriber table).

Once a list table is created (useful for reporting and historical purposes, as opposed to pulling the list off of a live table every time), it is populated and values/flags are assigned via campaign rules (or a campaign work statement). The list table is then spooled out to a disposition file and handed off to media production/delivery personnel. From there, email buckets (files) are created and fed into an MTA (mail transfer agent) and out the door the email goes.

How would you design a customer table for this part of the lifecycle?

There is the plain vanilla heap-organized table, but given how customer data is organized, perhaps using an index-organized table (IOT) is the way to go. If there is an initial bulk load of customer data, one of the limitations of IOT tables is that direct loading is not an option. That may be a small price to pay for the initial setup work, but in the long run, assuming the number of inserts down the road is relatively small, IOTs present many advantages.

How do you account for global versus product-level suppressions? One pass thru each record (in a cursor) would be to check for a "Y" flag at the product level ("Y" meaning suppressed in this case) and a numeric non-zero value for global suppression. Instead of doing two comparisons (and assuming a send is based on a single product at a time), perhaps it would be better to make the flags binary and then check if global_flag plus product_flag is greater than zero. Either one, or both, being set to one will cause the record to be suppressed in the send/list table.

Data/Workflow in the Data Warehouse

Log files are used to collect actions, and a log loader process parses log file contents and uploads data elements into schema stage tables (based on each client having its own report portal). After the raw data is uploaded, jobs are run to summarize/aggregate the data and update summary tables. The summary tables can then be referenced by external reporting tools (Microsoft Reporting Services, Oracle tools, etc.). Figure 2 illustrates the overall concept of how your opening of an email eventually ends up as a report item.

The design of the data warehouse is no small effort, and several issues factor into its design. Is there a service level agreement for data retention, and if so, for how long? If data is offloaded from the warehouse, is there a maximum turnaround time to have it restored? Which reporting tool (or tools) is going to be used? Is it a tool that can directly query the warehouse, or is it a tool that uses an intermediary (like what is illustrated in Figure 2)?

For stage tables, are partitions appropriate, and if so, what is the pruning policy? What is the nature of after-the-fact ad hoc reports requested by clients? If the ad hocs typically involve actual email addresses, where and how are those stored? Temporarily in the stage tables or permanently in a flight history (and then referenced via an email address dimension table)?

Log files are used to collect actions, and a log loader process parses log file contents and uploads data elements into schema stage tables.

"We want our reports and we want them now!" How do you meet that requirement? Real-time data is difficult to come by, so periodic refreshes (more than once per day) of summary tables may be necessary. Can your server handle that type of load? And speaking of servers, how do you maintain high availability? Data Guard may be of interest to you if not using RAC (although those are far from being mutually exclusive).

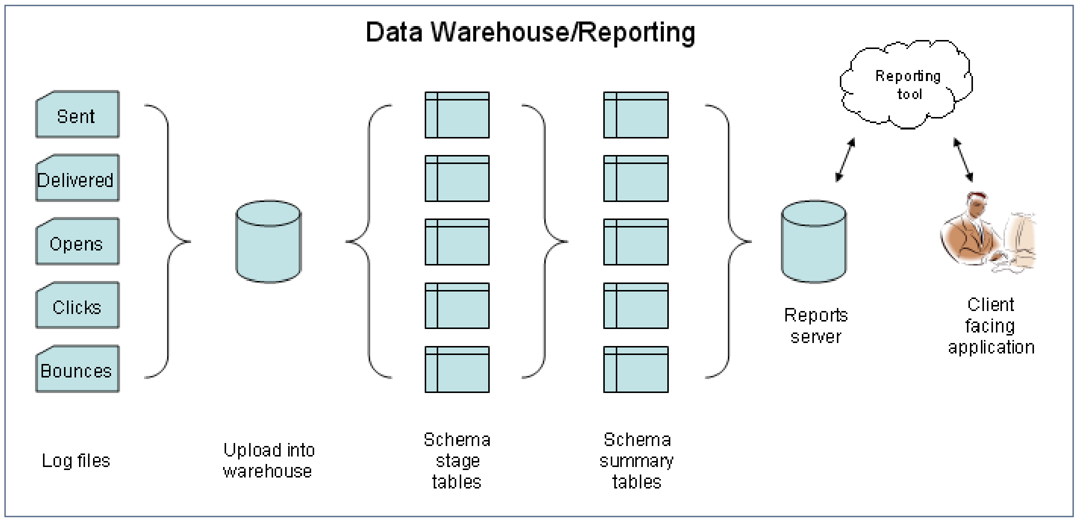

Figure 2: Data Warehouse/Reporting Workflow

Knowing an email address (or its associated primary key) is a potential gateway into hacking a database. Take a close look at the source behind HTML-based email and see what you can see about yourself within the code. Ideally, your email address should appear as translated string (look for something along the lines of "email_message_id").

People generally unfamiliar with how email is produced are almost always amazed about how much information a sender can collect by the simple act of opening an email.

Many Oracle applications have become Web-based, so what applies in the world of emails with respect to privacy applies in the Oracle Applications world as well. "Security through obscurity" is an invitation for trouble. Are you curious as to what users (and from where) are accessing your applications? The Web server logs will tell you exactly that – all for free and by default. Granted, your Apps audience consists of willing captives, but it doesn't mean users are prevented from committing malicious acts.

Finally, another carry over analogy from email to Apps usage is the recording of what is being accessed on a page. Are you spending hours maintaining pages (including forms and reports) for rarely visited links? Capturing/reporting link/URL usage can pay huge dividends down the road when you need to justify the elimination of links, pages, forms or reports. Happy emailing, and remember, look before you click!

Conclusion

People generally unfamiliar with how email is produced are almost always amazed about how much information a sender can collect by the simple act of opening an email. Knowing some of the mechanics about this process can help you design a better database, and that includes taking privacy into consideration. Most people do not want senders (or their clients) sharing lists, and just as importantly, to honor your request to not be contacted. If your email address or account information is being passed back as an unencrypted name-value pair, that should raise some concern.

Steve Callan – Steve is an Oracle DBA and developer. His Oracle experience includes several versions of the RDBMS, Forms & Reports, and Application Server. In addition to working with Oracle, Steve also spends time on researching other database systems such as SQL Server and DB2 and would someday like to start his own software company. Steve may be contacted at **Steve.Callan@ERPtips.com.**

July 2006 Volume II Issue 3

ORAtips Journal