Print Article Close Window

From: www.cio.com

Raiders of the Lost Archive: SaaS and Disaster Recovery

– David Taber, CIO

September 09, 2009

The starting point for SaaS applications: everything related to the apps is in the cloud, so data maintenance, redundancy, and recovery is the responsibility of the SaaS vendor. But the reality is more complex, so interesting subtleties have developed in the largest SaaS deployments. While this article is focused on Salesforce.com's applications, the lessons learned can be applied to nearly any SaaS vendor.

The Database Basics

Let's start with the basics: the database underlying the application. For service continuity, nearly any SaaS vendor must have clustering or replication strategies for the customer data. Salesforce.com has a continuous replication of user data and a significant amount of redundancy in their data centers. As they have a clustered multi-tenant architecture, the backup and redundancy services are pretty sophisticated—and they have a lot of work to do. A main production cluster may have to handle 10,000 customers and the transactions of 100,000 users, and SFDC has an excellent record of uptime.

[For timely data center news and expert advice on data center strategy, see CIO.com's Data Center Drilldown section.]

While data backup is included for free in SaaS applications, data recovery is free only if it's needed to recover from a *vendor's* error. If a customer needs to recover data to some historical point in time because of a user error, getting this data out is a chargeable extra. Given the number of simultaneous backup threads in flight at all times, you can imagine the complexity of unraveling the historical state of just your records from three weeks ago.

So the first lesson learned is, do a regular backup of your own data. If your SaaS vendor has an automatic export or archive function that pushes the data to local file storage, use it. If not, use a high-speed data loader. For a CRM system, a complete weekly snapshot taken early Saturday morning (US timezone) works best, and we typically recommend keeping 6 months worth of backup files. Don't forget to develop a strategy to back up attached files (as well as the pointers to them in the object model).

The Plot Complication

But it's not quite that simple, because there is inevitably data that you'll need which is omitted from the standard export tool. You'll really want every scintilla of data from every table, including administrative logs. For example, a client of ours is dealing with the discovery phase of a lawsuit from a disgruntled employee, and they need to show that the employee was not logging into the system as often as they were supposed to do. Two years ago. The cost of recovering that data from the SaaS vendor involves fees that would make even lawyers blush.

Further, the SaaS backup systems will not do a snapshot of the system's object model, metadata, customizations, report definitions, or your code. These don't need to be backed up every week, but it doesn't hurt for configuration control purposes. Backing up these data may involve some outboard utilities to extract data through the application's APIs, but these utilities are usually open source and without charge.

Go Into the Archives

The next thing to consider is archival: removing inactive or obsolete records from the online system. This may be required because of your company's information retention policy, performance issues (particularly with big reports), or a desire to reduce storage charges. I have yet to find a situation where data that's been untouched for 7 years needs to stay in a CRM system, and in certain businesses data that's more than 2 years old may never need to be seen again.

But CRM data is never a simple database, and removing records from the system can have complex repercussions. Depending on the vendor, CRM databases comprise between 10 and 200 tables, and user-level objects may create some really amusing pointer chains across tables. For this reason, some CRM objects can never be removed from a system. We further recommend that the Account and Opportunity objects never be removed, as they are at the center of a large number of pointers.

The easiest things to archive and remove from the system are objects at the leaf nodes of the pointer tree. For example, archiving old attached documents, emails, notes, and leads is fairly straightforward. However, the whole point of making an archive is to be able to get to the data if needed so make sure that each archive includes a "readme" file that includes the checklist of how the archive was made. Six months down the road, no one will remember how to unpack or interpret the data in the archive.

For objects that are more central to the CRM system, creating an archive can be quite complex. Properly archiving "Contacts" in Salesforce.com, for example, involves 10 extracts and a sequence of deletion passes that must be done in order.

The alternative? In many cases, it's easier to hide unwanted data than to actually remove it from the system. Hiding the data typically involves setting special record type values to indicate inactive data. The key to this strategy is making sure that all views, reports, workflows, trigger thresholds, and external interfaces are modified to exclude the marked records. This may sound complicated, but with proper configuration management this approach can be more straightforward than archiving the most deeply embedded of CRM data.

David Taber is the author of the new Prentice Hall book, "Salesforce.com Secrets of Success" and is the CEO of SalesLogistix, a certified Salesforce.com consultancy focused on business process improvement through use of CRM systems. SalesLogistix clients are in North America, Europe, Israel, and India, and David has over 25 years experience in high tech, including 10 years at the VP level or above.

Do you Tweet? Follow everything from CIO.com on Twitter @CIOonline.