What's in store for the Rnext database

by
Laura
Rutherford

Level: All
Works with: Domino Rnext
Updated: 11/01/2001

The database plays an integral role in Notes and Domino—from storing your mail messages to storing information needed for system configuration. Read on to learn what Kimilee Gile, co-project leader of the Database team, has to say about the new and enhanced database features in Rnext, and what those features mean to you and your Domino environment.

Can you explain the role of the database in a Notes/Domino environment?
The database subsystem is the document store for Domino and Notes. When you think about that, you think “Well, I have a mail box, all of my messages go into that; and I have views and folders where I can organize my stuff.” But the database system actually has a much larger role in Domino than most people know. The Domino Directory, for instance, which stores server configuration information along with all of the user addresses, is stored as a database.

In fact, just about all the information that Domino needs to run is stored in one database or another. We have log databases, administration request databases, and template databases, which are used in turn to create new databases. Domino cannot do anything without the database subsystem.

Have there been major database changes over the years, from release to release?
There certainly have been. I have only been here four years, so I can’t tell you about the history before the R5 release. But in the R5 release, we made a major change to the database and how it works by including Transaction Logging information. The information is used to roll forward changes for restart and media recovery. This provides the opportunity for more efficient back-up recovery solutions. And when you restart a crashed server, you won’t lose any data. Any information that hadn’t been written to disk will automatically be applied to that database when the server is restarted. So that was a huge change in how the database subsystem works and the kind of structural information we have to store in the database to make recovery happen.

What were your overall goals for the database when planning for Rnext? What made you focus on these goals?
When we plan for a new release in the database subsystem, the goals come from a lot of different areas. Certainly customer feedback is a big thing that we use to determine what to do next. Disk space reduction is a big area that we look at because, as I said in the beginning, for everything that Domino does, data is stored in a database somewhere. If we can store it in a way that takes less space, that will help customers out a lot. They won’t have to keep buying storage capacity.

Another big area is network transfer performance. A lot of Domino configurations need to be fast. People want the information and they want it yesterday. One place where things can slow down is when you are sending data over a network. So we want to look at any way that we can reduce network latency.

Customers often tell us they like a certain feature but wish it were a little easier to use. We try to look at usability issues and make things easier to use when we plan.

Also, with Rnext, we wanted to take a couple of existing features to the next level. For example, we took the Transaction Logging features that we put in R5 and extended them to cover view logging.

What are the major areas of database improvements for Rnext? What drove those improvements?
In the disk space reduction area we tried to think of ways to reduce some of the redundant information. One of the ways we found to do that was with a Single Copy Template. With a Single Copy Template we can reference design information in the template, instead of having to store the information with each database. For example, the information that describes what a mail memo form looks like used to be stored in every single mail database. Now we can keep this type of information in the template itself and reference it instead of storing it a hundred times for a hundred users.

For network transfer performance boost we are working on a streaming mechanism that doesn’t require as many acknowledgements to transfer data from one server to another or from a client to a server.

Our attachments were already stored in a compression algorithm to reduce empty space—we decided to add an optional, newer algorithm for Rnext. For Rnext format databases, you can use the LZ1 compression algorithm. The goal of this feature is to reduce disk usage and network latency when attachments are transferred.

The View Logging was, of course, a major area of improvement. Currently with R5, if you crash on a busy server, when the server restarts, chances are the views in your Domino Directory are going to be corrupt and need to be rebuilt. It's going to take some amount of time for that to happen before your server can complete a restart. So implementing Transaction Logging for these views will reduce the amount of time it takes for the server to come back up, since the views will be consistent after restart recovery and will not require rebuilds.

So the big new things are Single Copy Template (SCT), LZ1 compression for attachments, the streaming mechanism, and View Logging. Those are the four major new features.

What about enhanced features?
The major one is SCOS (Single Copy Object Store) or shared mail. We made some major enhancements to this to reduce contention and make it easier to use and manage. We made some minor changes to our tools—compact, updall, and fixup—to make it easier for people to use them.

Let's take a closer look at some of the features, starting with the ODS. What is the ODS and why did you change it?
ODS stands for "on disk structure." It is the format that we store the data in on disk and all the metadata needed to manage the information in the database. We only change the ODS version when we absolutely have to because it takes time to compact databases to change them to the new version. We generally change the ODS to allow new features or to disallow something. For example, the code needs to know that it can't replicate an attachment stored as LZ1 to an R5 format database without first recompressing it to the R5 format. We've made several changes in Rnext that require us to go to a new ODS.

View Logging is one of those changes. You can only do View Logging on a new ODS database because there are new structures and new methods for managing the data in the views. Single Copy Template is another feature that requires the new ODS. There are new ways of handling how to get the design information from the new reference pointers. As I mentioned, LZ1 is another feature—you cannot have LZ1 in an old format ODS database because the structures to handle this new compression format do not exist.

What does the new ODS mean for R5 applications and R5 administrators?
It should mean nothing. We take great pains to make sure all ODS databases are backwards compatible.

What can you tell us about View Logging? Is it basically Transaction Logging for views?
That's exactly what it is. The Transaction Logging mechanism that we included in R5 is now applied to views. We did not log views for R5. There are two major reasons we added this feature: I've already talked about the restart time. Servers can be down for 20 minutes—30 minutes if you've got a large directory. With View Logging, the time will be significantly reduced. Server restart recovery just runs through the same transaction logs that you know and love from R5 and your view will be up-to-date.

View Logging is also used in media recovery. In R5, if you did media recovery using the transaction log, your views would not be up-to-date and they would have to be rebuilt. In Rnext, you do backup of a database, recover it, and assuming it is an Rnext ODS database, after recovery, the views will all be up-to-date. You won't have to wait for that rebuild time.

What else is new for Transaction Logging?

There are a few things that have been added or improved for Transaction Logging since R5. We've added a new style. There used to be two styles—circular and archive. Circular was a set of logs of up to 4 gig size. When the logger was at the end of the log, it would start over at the beginning again. It would just keep going in a circular manner. Archiving did not write over logs; it would create new log extents as needed, which would be occasionally archived off to your archive media. Archive logging was the style you would use if you were doing backups just to make sure you didn't lose any information needed to roll forward your databases.

We've added a new style called linear or looping linear. It is similar to circular logging, only you can have as much space as you want. It is no longer limited to 4 gigs. This allows you to do backup and media recovery using a circular concept, so you do not have to have archive media. If you have enough disk space to support a log big enough to contain all the information recorded between backups, this new style will be good for your situation.

We added a new API for backup vendors called NSF_RECOVER_DATABASES_WITH_CALLBACK. We have some instances where, for example, you are communicating with countries that are restricted. Maybe the government needs to occasionally go through and list all the mail you sent to a certain country. This function allows you to run through your log with this callback to check every note updated.

There is also a flag to allow you to have alternate retrieve paths for recovered log extents. Archived log extents are restored to your log directory to be rolled forward during media recovery. Using this flag, you can restore the extents to a different drive to avoid contention and drive-head seek time on your log drive.

There is also a new API called IS_NEW_BACKUP_NEEDED. You can call it in a circular log instance to find out if you have written over the top of your log, in which case, you have to do new database backup because the old backup is no longer valid.

Another new feature is Single Copy Template (SCT). What prompted its development? Is saving disk space its only benefit?
Saving disk space is the major benefit. What SCT means is that you don't have to store the design elements in every single database. You just have them in the template and you reference them there. For example, when you open your Rnext mail database, the template that it is now based on is a Single Copy Template—but you won't see anything different at all. The code is automatically in the background saying "I'm going to open a note but I don't know what a note looks like so I am going to talk to the template." The template sends back information saying "This is what a note memo is supposed to look like." Then the mail database just fills in the actual information.

In an R5 database, the this-is-what-a-note-looks-like information is stored in each mail database. With SCT, there's a significant savings—about 6 meg for the mail SCT template. So if you create a new mail SCT database in Rnext, it will end up being about 680 K in size. If you created it without SCT, it is a bit over 6 meg. You can see that if you are creating a thousand mail databases, you are going to save a significant amount of disk space.

What do administrators have to do to implement SCT?
It's not automatic. It's a property of the template itself. On the Design tab of the Database properties box for the template, there is an area with information that announces "I am a template, this is my name." There is a checkbox for Rnext ODS databases that says "I am a Single Copy Template." When this box is checked, the Designer removes the design elements from all the databases derived from the template and puts in the references to the template. This happens the next time the database is opened or when the

Designer is scheduled to run on all databases overnight. The next time Compact runs, that space will be reclaimed. What the administrators need to do is figure out which templates they think should be SCT.

The Single Copy Object Store (SCOS) has been around since R4. Can you explain briefly what it is and tell us what's new in Rnext?
Single Copy Object Store, or shared mail, is also another space saving feature. What SCOS does is to save a single copy of a message that is being sent to multiple users, instead of saving that message in everybody's mail database. It saves it in one spot—the shared mail database. The users don't see this at all. The users open their database and they think the message is in their database.

In R5, there was generally one shared mail database that every mail database used, which you can imagine, caused some contention. What we have done in Rnext is to allow the automatic configuration of multiple SCOS databases to reduce contention. You can have multiple directories as well to scatter your shared mail databases over several disk drives if you want to.

There are two other things that we did to try to ease the administration load. You used to have to run a task periodically that would delete any messages that were no longer referenced in user mail databases. Now the messages automatically get deleted when the last user deletes them. The other big improvement is that we've put the configuration for this in the Server Configuration document of the Domino Directory. There is a new tab in the document called Shared Mail where you specify which directories to use, how many databases are in each of those directories, and how big a directory is allowed to get before it stops accepting new shared mail. If all of your directories are full, you will stop using shared mail and the mail will go fully into user mail databases. The idea behind the UI enhancement is to make management less person-intensive

Both SCT and SCOS help reduce database size. What else are you doing to reduce database size?
LZ1 compression is a big one. It is a newer and better compression algorithm for attachments. We are strongly recommending that you use this only in Rnext configurations. When an R5 client is reading an LZ1 attachment from a server, the server has to recompress the attachment to Huffman because the client would not know how to interpret the LZ1 algorithm. This whole process will take extra time, so we recommend using LZ1 in configurations that are fully Rnext.

The amount of savings you get over Huffman with LZ1 differs for each type of attachment. Some attachments are really dense; for instance, an executable file is going to be denser than a text file. So when I say how much you are going to save, it really depends on what type of attachment you are attaching. With the executable, it averages 15 to 20 percent smaller with an LZ1 than with a Huffman attachment. You can see that if you have a situation with a lot of attachments, it can save you a considerable amount of disk space.

How does LZ1 compression affect network performance?
It affects network performance in an Rnext-only scenario because the attachment is smaller so there is less data going across the wire.

The new support for streaming attachments and replication also must improve network performance. What can you tells us about these features?
Streaming is an exciting feature. Without streaming, when you send data over the network, a client or server requests data from another server, and the server sends a chunk of data over. Then the requestor says "OK. I got that, now send more." The sending server waits for that acknowledgement before it sends the next chunk, and they go back and forth. With streaming, the client or server requests the information and the server sends the entire thing all at

once. The data is subdivided into network buffer packets, but the receiving client or server does not acknowledge every single packet. This eliminates the acknowledgement wait time.

Streaming is used in replication, in sending attachments over the wire, and in database copy for some of the administration tasks, such as moving a mail file. Streaming is used in replication pull-only, which means that in server-to-server replication you might want to change from a push-pull to a pull-pull.

Anything else about the new database features and enhancements that you want to mention?
There are a couple of things we've done with fixup, updall, and compact. It used to be that you could run these tasks on a database or all of your databases. One or two of them allowed you to specify a directory. What we have done with all three of these tools for Rnext is to allow you to specify a database (like before) and a directory on all of them, but we have also created an indirect file—it has an IND extension. This file can have a list of databases and/or directories, so you can easily customize when you run these tools on subsets of your data. We think this is going to be an appreciated feature. I've used it myself, and it is amazing we got along without it. The IND file is a text file, so you just list the databases or directories you want.

Also, we have put in a feature to allow you to run fixup while the server is running. Fixup can't run on a file that is open, as is likely to happen while the server is running. So what we have added is a way that fixup can take the database off-line exclusively.

We are also working on a mechanism to specify a number of retries and a wait time between retries for renaming temp files during copy style compaction.

Out of all the new features and enhancements, which one is your favorite? Which is the biggest win?
My favorite would be View Logging, because that is what I have worked on. Administrators are going to like View Logging because it will speed up their restart time. And people with huge mail configurations may be thrilled with the SCOS enhancements. I would say View Logging, Single Copy Templates, and Single Copy Object Store are probably the three big ones.

Where do you see the Notes and Domino database headed post-Rnext?
You can always improve space usage, so we will be working on further ways to do that. Ease of use with some of the tools is something we always look at too. Anything we can do to make things faster or smaller will be looked at.

About Kimilee Gile
Kimilee started working at Iris in 1997 as a Quality Engineer on the Database team. Her past career lives included time at Digital Equipment Corporation and at a small startup named Mango. She currently divides her time at work between project lead, development, and QE tasks. Kimilee is in training as a figure skater and has a doomed hope to be on the U.S. team for the 2002 Olympics.

About Laura Rutherford
Laura worked as a user assistance writer for Lotus until she had her daughter, Kate, in January, 1999. Since then, she had another daughter (!) Maggie, born in September, 2000. Now Laura spends most of her time taking care of her two daughters, two dogs, and one husband (basically in that order). In her free time, she loves to read, run, and, believe it or not, write articles for *Iris Today*.