Article
Tani Frankel · Aug 29, 2016 5m read

My growing journals - how do I minimize this?

Following my previous post, some urged me to get to the point – ok, so I found my "star" journaling globals, the ones that take up the most space – but how do I avoid this? How do I minimize the journal's size?

[DISCLAIMER: Some might still be disappointed after this post as well but wait till the next one...]

So unfortunately I'll need to disappoint you a little longer and postpone the discussion (it's coming soon…) by first asking two questions that I believe are important to cover before we arrive at how we avoid what we don't what journaled to get journaled, and these are:

1. What actually gets journaled?
2. And why?

The reason I think these questions are important to answer before getting into not journaling, is because fundamentally journaling is good, that's why your data was in the journal in the first place, and you'd have to have a very good reason why you'd decide not to journal.

So let's try and answer those questions -

Generally every SET and KILL command (as well as TSTART and TCOMMIT/TROLLBACK) is recorded in the journal file.

For example if I run this command

```
ENSEMBLE>set ^myGlobal=1
```

I see in the journal:

168624	2013-06-16 13:28:34	8852	SET	No	^myGlobal	c:\intersystems\ensemble201310\mgr\ensemble\
Offset	Time	Process	Type	InTransaction	GlobalNode	Database

The journal file has a few purposes:

a. To protect from data loss in system startup recovery after a crash – when the system comes up it rolls forward the actions in the Journal if they haven't made it into the database yet (assuming they are not part of an uncommitted transaction, in which case they'd be rolled back).
b. To complement the disaster recovery plan - by allowing restoring data changed since the last available

backup. For example if a backup was run during the night, and for some reason the database (i.e. the CACHE.DAT) file has been compromised (the disk was damaged or whatever) at noon, one could restore from the backup getting the situation back to the way it was in the morning, and then restore from the journals right back to the point until noon when the system failed, minimizing data loss.

c. Transaction rollback support – to be able to rollback a transaction for whatever reason.

More info here.

(A separate post could cover in more detail each of these purposes and illustrate their usage etc. This could also cover what a rollback looks like in the Journal etc. This is not a post I am currently planning, but someone else is welcome to tackle this topic.

So you see there is good reason data is journaled and it's very useful and even critical (so much so that we even have a switch saying that if journaling fails for any reason we should freeze the whole system).

(I hope at this stage I'm not making you feel guilty for considering not to journal... )

Every database in Caché has a flag whether the data in it is jounraled or not. The default and recommendation is to journal. Even if you decide not to though, you should be aware, that any SET or KILL within a transaction will still get journaled, even if the database the data resides in, is marked as not to be journaled (for the purposes of allowing a rollback as described above). [An important exeception are globals residing in the CACHETEMP database, and we'll use that later on]

For example the SAMPLES database is not journaled, so a simple SET does not get journaled but one in a transaction does:

```
SAMPLES>set ^myGlobal=1

SAMPLES>tstart

TL1:SAMPLES>set ^myGlobal=1

TL1:SAMPLES>tcommit

SAMPLES>
```

The journal does not show the first SET (outside the transaction) just the second one:

171584	2013-06-16 13:28:34	8852	BeginTrans	Yes		
171600	2013-06-16 13:28:34	8852	SET	Yes	^myGlobal	c:\intersystems\ensemble201310\mgr\samples\
171652	2013-06-16 13:28:34	8852	CommitTrans	Yes		

The same goes for objects, which are automatically saved within transactions.

For example, again in SAMPLES:

```
SAMPLES>set person = ##Class(Sample.Person).%New()

SAMPLES>Set person.Name="Doe, John"

SAMPLES>Set person.SSN="123-45-6789"

SAMPLES>Write person.%Save()
1
```

We get this (though we didn't explicitly issue a TSTART command, this happens behind the scenes by the object filer):

173672	2013-06-16 13:28:34	8852	BeginTrans	Yes		
173688	2013-06-16 13:28:34	8852	SET	No	^Sample.PersonD	c:\intersystems\ensemble201310\mgr\samples\
173740	2013-06-16 13:28:34	8852	SET	Yes	^Sample.PersonD(201)	c:\intersystems\ensemble201310\mgr\samples\
173828	2013-06-16 13:28:34	8852	BitSET	Yes	^Sample.PersonI("$Person",1)	c:\intersystems\ensemble201310\mgr\samples\
173896	2013-06-16 13:28:34	8852	SET	Yes	^Sample.PersonI("NameIDX"," DOE, JOHN",201)	c:\intersystems\ensemble201310\mgr\samples\
173984	2013-06-16 13:28:34	8852	SET	Yes	^Sample.PersonI("SSNKey"," 123-45-6789",201)	c:\intersystems\ensemble201310\mgr\samples\
174060	2013-06-16 13:28:34	8852	BitSET	Yes	^Sample.PersonI("ZipCode","",1)	c:\intersystems\ensemble201310\mgr\samples\
174128	2013-06-16 13:28:34	8852	CommitTrans	Yes		

So we answered, albeit briefly, the 2 questions we had – what gets journaled and why.

This should lead us to the next step – how to avoid certain data from getting journaled.

As you realize the first issue is to decide if you can truly "afford" not to journal the data you want to avoid filling up your journals.

Though let me address first a very simple case – "Wow! I was not aware we are even setting this global?!".

E.g. – looking at the journal revealed to us that perhaps somewhere in our code we had some temporary switch we might have set some time, or some log we needed to write to for debugging purposes or some mechanism with another investigative temporary nature, and we don't need it at all and we can just remove this code (or comment it out, or not call it, or turn off the switch or whatever).

This would be a relatively easy case and we would not need to get into the next discussions of can we not journal and how we do not journal.

Now, about this case, you might be thinking – you needed to look at the Journal in order to find out that you had this "runaway" global setting, didn't you see the actual database grow?

Well, maybe indeed yes, the database did also grow and you missed that, or did not give it too much attention and/or thought it was just natural growth. But what could have also happened, and this is the point I want to stress, is that journal growth does not have to be tied to database growth.

To take an extreme example:

You can set one global's value, alternate it between two values, say 0 and 1, many times every second. The database would not grow at all, because the data's size did not change, but the journal would have thousands of entries for every SET.

Or you could be setting a temporary large global tree structure, and after seconds Kill it, so database size-wise the tree size occupied might be more or less constant, but the journal would fill up from all of the sets and kills.

So, assuming this was not the case, you were not surprised you are setting this data, indeed you are well aware of this, and your application needs to set this data. In that case we come back to our previous question – can you truly afford not to journal this data?

To answer this question let's try and break it up into a little smaller questions:

1. Can you afford the data entered (or deleted) before a crash not to be there (or reappear) after?
2. Can you afford the data entered (or deleted) not to be recovered (or reappear) in a system back recovery process?
3. Don't you need that data to be rolled back as part of a transaction?

If you answered "yes" to all 3 questions then you can indeed consider not journaling the data in question.

But if you do need this data to be recovered after a crash, or recovered as part of a backup, or it needs to be rolledback as part of a transaction. You'll need to keep it journaled.

In this case you'll simply need to calculate and determine the true journal growth rate and make sure you have enough disk space [You can see another Post of mine, with some related code on Git, that could assist in estimating journal space required for Ensemble interfaces. You could obviously adapt this code for other use-cases as well].

And just to give you a preview, these are the options I'll try and cover in the next post:

- Turning off journaling system wide
- Mapping globals to CACHETEMP
- Mapping globals to non-journaled databases
- Turning off journaling for a process
- Turning off transactions for object filing

See you all next time... (I promise to get to the actual point then... )

Here is the next post.

#Journaling #System Administration #Caché