<https://www.littlekendra.com/2016/01/26/partitioned-tables-missing-boundary-points/>

# How to Fix Lopsided Partitioned Tables

By Kendra Little, January 26, 2016 [**1**](https://www.littlekendra.com/2016/01/26/partitioned-tables-missing-boundary-points/#comments)

Over the years I’ve gotten lots of emails and questions from students that start like this:

Help! My partitioned table has the wrong data in a partition! It’s lopsided. I started trying to fix it, but…

The next sentence is one of a few things. Sometimes it’s all of them.

* It was taking forever, so I stopped it.
* It was blocking users, so I stopped it.
* My transaction log blew up and ran out of space and it failed and rolled back.

The story usually then goes on:

* It took forever, and now I’m afraid to touch it again.

And then the question: **What do I do now?**

### The Problem: Missing Boundary Points in the Partitioned Table

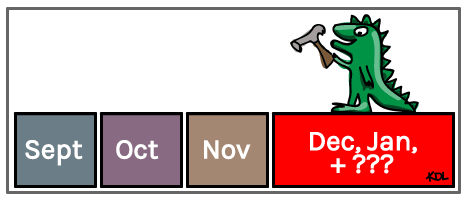
First, let’s explain a bit about what went wrong. When you set up partitioned tables in SQL Server, you define “boundary points” as part of your Partition Function. This explains logically how the data is laid out.

For a simple example, let’s say we have a table named dbo.SalesHistory. It’s partitioned on the SalesDate column, and we’ve defined these boundary points using “Range Right” so that the boundary point itself stays with the data following it:

* Sept 1, 2015
* Oct 1, 2015
* Nov 1, 2015
* Dec 1, 2015

Everything was fine, but we forgot to put a boundary point in for Jan 1, 2016… and now it’s well into January. All our data for January 2016 is in the partition with the December data.

The more time passes, the more data is going to go into that partition, and it’ll get bigger and more lopsided. If we’re switching out old partitions by month, eventually that’s not going to work. And partition elimination won’t work for anything after Dec 1, 2015, either!

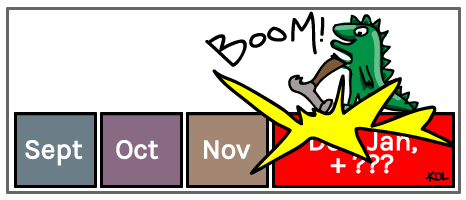
[](http://littlekendra.com/2016/01/26/partitioned-tables-missing-boundary-points/partitioning-lopsided-missing-boundary-point/)

### You Can Split a Partition, But that Has Problems

The first fix that most people try is simply adding in the boundary point for Jan 1, 2016 by altering the partition function and using [SPLIT RANGE](https://msdn.microsoft.com/en-us/library/ms186307.aspx). For small partitions, that can work well.

But usually partitioned tables are quite large. And usually nobody notices this for a while. The “lumpy” partition typically has millions and millions of rows and may be hundreds of gigabytes of data. Just running SPLIT can be a disaster.

[This post by Kalyan Yella and Denzil Ribeiro explains why](http://blogs.msdn.com/b/sql_pfe_blog/archive/2013/08/13/oops-i-forgot-to-leave-an-empty-sql-table-partition-how-can-i-split-it-with-minimal-io-impact.aspx): The SPLIT operation is actually doing two fully logged operations behind the scenes in one giant transaction: a DELETE and an INSERT. This is why it’s prone to be time consuming and use vast amounts of transaction log space.

[](http://littlekendra.com/2016/01/26/partitioned-tables-missing-boundary-points/partitioning-lopsided-missing-boundary-point-smash-split/)

### My Favorite Workarounds: Switch Out, Fix the Boundary Points, then Reload

This workaround is not perfect: your data will be offline while you do it.

This isn’t something I came up with on my own, either– it’s a modified version of the recommendation in the post I linked above, and it’s also been recommended [to limit log growth when splitting partitions by Microsoft Program Manager Sunil Agarwal](https://connect.microsoft.com/SQLServer/feedback/details/379763/split-partition-is-causing-an-abnormal-log-growth-with-the-simple-recovery-model).

But if you can take a downtime, I find this to be simple to code, test, and optimize.

#### Workaround Variation One (More Downtime)

* Expand storage so you have enough space in your data and logfiles to complete the procedure below. This requires restoring the database and testing, but you want to do that anyway.
* Start your maintenance window and lock out users
* Switch the data in the “lumpy” partition out to a temporary table in the same filegroup (dbo.SalesHistoryTemp). This leaves the partition in the active table empty
* Split the partition in the active table and add multiple boundary points (you don’t want to get into this situation again). This will be immediate, because you switched out all the data.
* Run INSERT commands to put data back into the correct filegroups, using syntax to make sure you get minimal logging and make it as fast as possible. (This may require changing the recovery model before and after the INSERT commands)
* Run queries to verify that you inserted all the data properly before truncating and dropping dbo.SalesHistoryTemp and ending your maintenance window
* Validate that backups are healthy, and potentially take a full backup following the change. Make sure you retain backups prior to the change as well.

#### Workaround Variation One (Less Downtime, but More Complex to  Pull Off)

You can potentially minimize the downtime by optimizing the order like this, as long as data isn’t changing:

* Select the data by month out into separate tables in the filegroup (dbo.SalesHist\_DecTemp, and dbo.SalesHist\_JanTemp)
* Validate you have all the data in those staging tables
* Switching data out from the production partition to a new table named dbo.SalesHist\_OldData
* Splitting boundary points
* Switching data from dbo.SalesHist\_DecTemp and dbo.SalesHist\_JanTemp back in

If data is changing, you’ll have to get fancier, using triggers or some other method to make sure you don’t miss rows.

Pick the method that’s right for you depending on your comfort scripting and tolerance for downtime.

### What was that About Minimal Logging?

Minimal logging is just what it sounds like: a way you can insert a lot of data without writing all of it to the transaction log. It can really speed up operations like this, particularly if you’ve got a lot of data and aren’t on the world’s fastest storage. It only works in the SIMPLE and BULK LOGGED recovery models, but doing the workaround requires an outage, anyway.

For information on how to get minimal logging, read the [Microsoft Data Loading Performance Guide](https://technet.microsoft.com/en-us/library/dd425070(v=sql.100).aspx) by Thomas Kejser, Peter Carlin and Stuart Ozer.

### Partitioning Best Practice: Always Have Empty Partitions on the Left and Right of Your Table

Long term, you just don’t want to get into this situation. Anywhere you use partitioning (and are NOT using [the rotating log pattern](http://kejser.org/table-pattern-rotating-log-ring-buffer/)), you should have a job set up that checks and makes sure you have at least two empty partitions ahead of your highest boundary point. The job should fail and send an alert to someone who knows how to fix it if you get into the danger zone.

Why two partitions? That person can’t always drop whatever they’re doing to fix it right away. If you work in a large company, three partitions might even be better for you.