**Audit Schema Change Report**

by [Jason Brimhall](http://jasonbrimhall.info/author/admin/)

FROM: <http://jasonbrimhall.info/2015/03/11/audit-schema-change-report/>

Categories: [News](http://jasonbrimhall.info/category/news/), [Professional](http://jasonbrimhall.info/), [SQLServerPedia Syndication](http://jasonbrimhall.info/category/sqlserverpedia-syndication/), [SSC](http://jasonbrimhall.info/category/ssc/)

Tags: [Audit](http://jasonbrimhall.info/tag/audit/), [Extended Events](http://jasonbrimhall.info/tag/extended-events/), [SQL Internals](http://jasonbrimhall.info/tag/sql-internals/), [SQL Script](http://jasonbrimhall.info/tag/sql-script/)

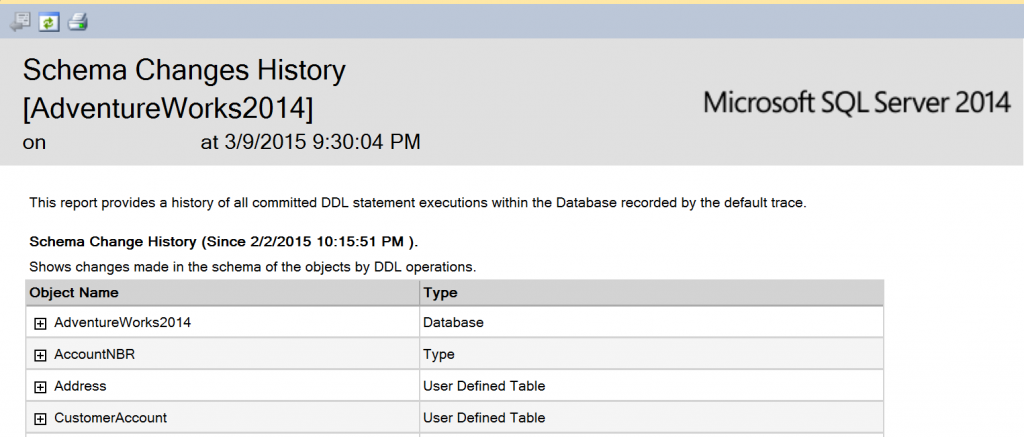
Comments: [2 Comments](http://jasonbrimhall.info/2015/03/11/audit-schema-change-report/#comments)

Published on: **March 11, 2015**

In a recent article on SSG, I discussed how to use Extended Events to function in a Profiler like fashion.  You can read about that [here](http://bit.ly/XEProfilerSSG).  I recommend reading that article first because it helps to lay some of the groundwork for this article.

Within Management Studio, from a righ-click context menu, there is an option to run a report called “Schema Changes History”.  The name of this report might indicate that there is some means to find when a change occurred in the schema and potentially what the change was as well as who made the change.  It does say “Schema Changes History” after-all.

If you run the report, you would be able to see a report similar to the following.

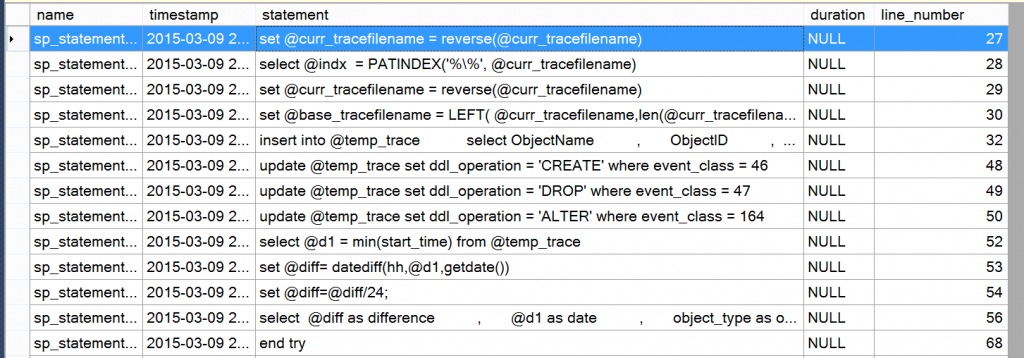
[](http://jasonbrimhall.info/wp-content/uploads/2015/03/Change_Report.png)

This looks like it could be a pretty useful report.  If I drill into the collapsed items, I can get better information.  But, how exactly is this report producing this kind of data?  I don’t have a database audit running to trap this information.  I also don’t have SSDT-BI installed, so I can’t export the report and check the report definition.

I could run a trace and refresh the report data and see what pops there.  I would really hate to use Profiler though, and I really don’t want to go through the hassle of creating a server side trace to figure out the source data definitions.  Then the lightbulb moment occurs.  I already have an XE session that can trace the SQL queries just like Profiler.  Why not use a better tool to do the job and satisfy my curiosity at the same time?

So, go read that [article,](http://bit.ly/XEProfilerSSG)I wrote for SSG, to get the XE session definition in order to follow along a little better.  With that XEvent session running, I can refresh the report data and start to evaluate what is happening in order to produce the Audit report.

By this point, you have probably thought to yourself that the source of the data must be coming from the default trace.  If you thought that, then you are correct.  It is a good presumption based on a knowledge of what kind of data is being tracked in the default trace.  But I really wanted to know for certain what the source of the data was.  As I perused the XEvent session data, I noticed several queries similar to the following in the results.

[](http://jasonbrimhall.info/wp-content/uploads/2015/03/XE_Capture.png)

Yes, I am using the GUI (\*shrug\*) in this image to explore the data from the session.  That is just for display friendliness.  But, as you look through those queries you start to see a few patterns and some obvious signs that the source of the data is the default trace.  Here is a cleaner look at the code being used to get the data-set.

schema\_history

Transact-SQL



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | Declare @curr\_tracefilename varchar(256)  ,@indx varchar(256)  ,@base\_tracefilename varchar(256)    select @curr\_tracefilename = path from sys.traces where is\_default = 1    set @curr\_tracefilename = reverse(@curr\_tracefilename)  select @indx  = PATINDEX('%\%', @curr\_tracefilename)  set @curr\_tracefilename = reverse(@curr\_tracefilename)    set @base\_tracefilename = LEFT( @curr\_tracefilename,len(@curr\_tracefilename) - @indx) + '\log.trc';    select ObjectName as obj\_name          ,       ObjectID          ,       DatabaseName          ,       StartTime as start\_time          ,       EventClass as event\_class          ,       EventSubClass          ,       ObjectType as object\_type          ,       ServerName          ,       LoginName          ,       NTUserName          ,       ApplicationName          ,       convert(varchar(128),'temp') as DDL\_Operation  INTO #temp\_trace          FROM ::fn\_trace\_gettable( @base\_tracefilename, default )          where EventClass in (46,47,164) and EventSubclass = 0  and DatabaseID = db\_id()    update #temp\_trace set DDL\_Operation = 'CREATE' where event\_class = 46    update #temp\_trace set DDL\_Operation = 'DROP' where event\_class = 47  update #temp\_trace set DDL\_Operation = 'ALTER' where event\_class = 164    DECLARE @d1 DATETIME  ,@diff DATETIME  select @d1 = min(start\_time) from #temp\_trace  set @diff= datediff(hh,@d1,getdate())      select  @diff as difference  ,       @d1 as date  ,       object\_type as obj\_type\_desc  ,       (dense\_rank() over (order by obj\_name,object\_type ) )%2 as l1  ,       (dense\_rank() over (order by obj\_name,object\_type,start\_time ))%2 as l2  ,       \*  from #temp\_trace  where object\_type not in (21587) -- don't bother with auto-statistics as it generates too much noise  order by start\_time desc    drop table #temp\_trace |

Now, I must confess that I altered it a bit to make sure it was doing what I thought.  Just a little thing like swapping a missing definition for the table variable for a quick insert into a temp table.  But the reality is, it is pretty much the source code of the report.  It is not the prettiest of code for the source, but it works.

Due to the lack of prettiness to the code and the seemingly overly complex means to get to the end result, I decided I wanted to “tune” it up a little bit.

Better Query

Transact-SQL



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | DECLARE @DBName sysname = 'AdventureWorks2014'  ,@d1 DATETIME  ,@diff INT;    SELECT ObjectName    , ObjectID    , DatabaseName    , StartTime    , EventClass    , EventSubClass    , ObjectType    , ServerName    , LoginName    , NTUserName    , ApplicationName    , CASE EventClass  WHEN 46  THEN 'CREATE'  WHEN 47  THEN 'DROP'  WHEN 164  THEN 'ALTER'  END AS DDLOperation  INTO #temp\_trace  FROM sys.fn\_trace\_gettable(CONVERT(VARCHAR(150),  ( SELECT REVERSE(SUBSTRING(REVERSE(path),  CHARINDEX('\',REVERSE(path)),256)) + 'log.trc'  FROM    sys.traces  WHERE   is\_default = 1)), DEFAULT) T    WHERE EventClass in (46,47,164)  AND EventSubclass = 0  AND ObjectType <> 21587-- don't bother with auto-statistics as it generates too much noise  AND DatabaseName = @DBName;      SELECT @d1 = MIN(StartTime)  FROM #temp\_trace;    SET @diff= DATEDIFF(hh,@d1,GETDATE());      SELECT @diff AS HrsSinceFirstChange  , @d1 AS FirstChangeDate  , sv.name AS obj\_type\_desc  , tt.ObjectType  , tt.DDLOperation  , tt.DatabaseName,tt.ObjectName,tt.StartTime  , tt.EventClass,tt.EventSubClass  , tt.ServerName,tt.LoginName, tt.NTUserName  , tt.ApplicationName  , (dense\_rank() OVER (ORDER BY ObjectName,ObjectType ) )%2 AS l1  , (dense\_rank() OVER (ORDER BY ObjectName,ObjectType,StartTime ))%2 AS l2  FROM #temp\_trace tt  INNER JOIN sys.trace\_events AS te  ON tt.EventClass = te.trace\_event\_id  INNER JOIN sys.trace\_subclass\_values tsv  ON tt.EventClass = tsv.trace\_event\_id  AND tt.ObjectType = tsv.subclass\_value  INNER JOIN master.dbo.spt\_values sv  ON tsv.subclass\_value = sv.number  AND sv.type = 'EOD'  ORDER BY StartTime DESC;    DROP TABLE #temp\_trace; |

Now, I have something that is easier to read and maintain (my opinion) and works well.  It can also be used easily enough in an RDL should you wish to create an additional report different from the canned report in SSMS.  Or just continue to use it from within Management Studio and check the results without the overhead of the report presentation.

**Related Posts:**

* [Audit Configuration Changes](http://jasonbrimhall.info/2013/01/24/audit-configuration-changes/)
* [Audit Database File Size Changes](http://jasonbrimhall.info/2014/11/26/audit-database-file-size-changes/)
* [T-SQL Tuesday #45 Follow the Yellow Brick Road](http://jasonbrimhall.info/2013/08/12/t-sql-tuesday-45-follow-the-yellow-brick-road/)
* [Last Known Good CheckDB](http://jasonbrimhall.info/2012/11/20/last-known-good-checkdb/)
* [Auditing and Event SubClasses](http://jasonbrimhall.info/2014/05/28/auditing-and-event-subclasses/)

2 Comments - [Leave a comment](http://jasonbrimhall.info/2015/03/11/audit-schema-change-report/#respond)

1. http://0.gravatar.com/avatar/a317d552a57d2d23efcd2048629934c9?s=32&d=&r=G[*Troy Ketsdever*](http://www.42and5.com) says:

[March 11, 2015 at 15:15](http://jasonbrimhall.info/2015/03/11/audit-schema-change-report/comment-page-1/#comment-94994)

Nice timing on this, Jason – just what I needed. This is an awesome lightweight alternative to the DDL triggers we were using back in the day to track changes on all of those customer db instances.

[Reply](http://jasonbrimhall.info/2015/03/11/audit-schema-change-report/?replytocom=94994#respond)

* + http://1.gravatar.com/avatar/7338921071ab1beaea281ca0528ad192?s=32&d=&r=G*Jason Brimhall* says:

[March 11, 2015 at 15:36](http://jasonbrimhall.info/2015/03/11/audit-schema-change-report/comment-page-1/#comment-94997)

Thanks Troy. I’m glad the timing worked out well for you. And yes – I agree on the trigger method.