Dias atras escutei nos corredores que a ferramenta de Auditoria do SQL era pesada e não funcionava corretamente pois causava um Overhead no ambiente. Causando concorrência ou em alguns casos podendo até derrubar o ambiente.

Diante disso, achei interessante analisar a questão, já que havia implantado o SQL Audit em outros ambientes e não sofri da mesma situação.

Passando a entender o cenário, verifiquei que o Audit criado não continha filtros, ou seja, auditava tudo que passava pelo SQL. Desse modo, realmente o Audit vai causar um overhead no ambiente, gerando um stress na corporação.

A auditoria é uma ferramenta que deve ser objetiva e ter seus filtros claros, para auditar somente aquilo que procura. Caso contrário, você criará uma arapuca para si mesmo.

Abaixo, irei mostrar como configurei o meu Audit com assertividade, leveza e o principal objetividade. Encontrando o que procurávamos sem precisar derrubar o ambiente :D.

No cenário em questão precisávamos capturar todos os objetos que sofreram alteração no ambiente. Para isso utilizamos o Audit Action Type “SCHEMA\_OBJECT\_CHANGE\_GROUP e SERVER\_OBJECT\_CHANGE\_GROUP”.

Segue link de todos os tipos de auditoria: <https://docs.microsoft.com/pt-br/sql/relational-databases/security/auditing/sql-server-audit-action-groups-and-actions?view=sql-server-ver16>

Definido o que queríamos auditar, verificamos se teríamos espaço em disco pois o **Audit salva a sua coleta fisicamente. Estimar a volumetria também é importante**.

Em seguida iniciamos a criação do AUDIT com o script abaixo:

[1-Create\_ServerAudit](https://github.com/jcdba/SqlServerAudit/blob/main/1-Create_ServerAudit)

Percebam que no script, além do **ACTION TYPE**, inclui um **WHERE**. Isso mesmo, conseguimos incluir outros tipos de filtros no Audit Server, isso com certeza afunila a busca e nos leva com mais assertividade ao alvo. Sem contar que não causa overhead no ambiente.

No nosso caso, tirei as alterações dos Objetos provenientes da **UPDATE STATISTICS.**

|  |
| --- |
| WHERE (NOT [statement] like '%UPDATE STATISTICS%') –FILTRO |

Bom, após criar o AUDIT SERVER, precisamos criar o **AUDIT SPECIFICATIONS**, que será especificamente onde vamos informar o AUDIT TYPE, o script abaixo já está pronto, com os TYPES que informei.

[2\_Create\_AuditSpecifications.sql](https://github.com/jcdba/SqlServerAudit/blob/main/2_Create_AuditSpecifications.sql)

|  |
| --- |
| ADD (SCHEMA\_OBJECT\_CHANGE\_GROUP), --AUDIT TYPE 1  ADD (SERVER\_OBJECT\_CHANGE\_GROUP) -- AUDIT TYPE 2 |

**SERVER\_OBJECT\_CHANGE\_GROUP -** *Esse evento é gerado para operações CREATE, ALTER ou DROP em objetos de servidor.*

**SCHEMA\_OBJECT\_CHANGE\_GROUP** *- Esse evento é gerado sempre que algum esquema de qualquer banco de dados é alterado.*

Após criar nossa Auditoria, será criado o arquivo no caminho que informamos no primeiro Script, já iniciando sua gravação.

Arquivo criado:

|  |
| --- |
|  |

Certo, mas com o arquivo criado, o que fazer?

Pois bem, conseguimos fazer a leitura do arquivo e o que ele auditou por dentro do SQL, utilizando em resumo a query abaixo, na qual eu inseri filtros:

[3\_Select\_Audit.sql](https://github.com/jcdba/SqlServerAudit/blob/main/3_Select_Audit.sql)

|  |
| --- |
| --SELECT INFORMANDO O ARQUIVO ESPECÍFICO  SELECT \*  FROM Sys.fn\_get\_audit\_file('F:\AUDIT\_SQL\AUDIT\_OBJECT\_DEV\_C3B993FF-AC7C-443F-B0AB-CF9582893A57\_0\_133041166209690000.sqlaudit',default,default)    --SELECT DE TODOS OS ARQUIVOS  SELECT event\_time, session\_id, action\_id, server\_principal\_name, object\_name, database\_name, statement,  transaction\_id  FROM Sys.fn\_get\_audit\_file('F:\AUDIT\_SQL\\*.sqlaudit',default,default) |

Segue result da query com as colunas selecionadas, e que eu particularmente prefiro executar.

Além disso, já está com as informações para análise, como: Database, Objeto alterado, usuário, comando executado, SPID e Transação ID.

|  |
| --- |
|  |

Veja que nesse exemplo já temos registros consideráveis e na coluna **ACTION\_ID**, informa o tipo da Alteração.

**AL = ALTER**

**CR – CREATE**

**DR – DROP**

Por fim, essa auditoria é um case implantado em ambiente DEV, onde sofre constantes alterações, e mesmo após um mês o arquivo principal ainda está com 116 MB e sem causar qualquer overhead no ambiente.

Isso ocorre por conta de as definições estarem bem atribuídas, ou seja, Audit Type e filtros estarem totalmente voltado par ao objetivo da auditoria.

Assim, sendo a coleta mínima e assertiva.

Por fim, acredito que seja valido um laboratório com Audit Server para entender as funcionalidades, antes de implementar em produção.

Por fim, espero ajudar com esse conteúdo.