### Forma de Migração

Com base no nosso diagrama (Tabela de Log única), decidimos implementar o processo de exportação com apenas um ficheiro CSV, que vai ser criado periodicamente através de um evento que vai chamar o *Store Procedure* ExportarCSV(). Com base no nosso sistema, chegamos à conclusão que a periodicidade a que nós acordamos seria de hora a hora.

**CSV vs XML vs JSON: O que escolher?**

Nós decidimos escolher CSV comparando com XML e JSON por duas razões. XML e JSON, sendo os dois bastantes semelhantes, guardam ambos bastante informação adicional como autor, data, nome dos campos, tipo de dados. Normalmente podem ser ambos vistos como formatos que guardam objectos. Ao contrário destes CSV é relativamente mais simples, apenas guarda os dados em cada campo separados por virgulas e não tem nenhuma estrutura orientada para objectos. Isto torna os ficheiros incrivelmente mais pequenos mas com muito menos informação. Após analisarmos o projecto apercebemo-nos que a exportação de logs (com tabela única) seria uma operação repetitiva e exaustiva em caso de escalamento da base de dados. Sendo assim a informação extra seria desnecessária e a eficácia importante. Resumindo, escolhemos CSV porque os ficheiros são mais pequenos (facilitando assim também a escrita do ficheiro) e porque contêm menos informação desnecessária.

**EXPORTAÇÃO DE FICHEIRO (ORIGEM):**

Como já mencionamos, a nossa implementação necessita de apenas um ficheiro por cada exportação. Esse ficheiro vai ter um nome fixo de *logs*.csv . Uma amostra de um dos *Stored Procedures* que utilizamos na exportação é o ExportarCSV():

**DECLARE d timestamp;**

**SELECT current\_timestamp into d;**

**SELECT id, Data, Utilizador, Operação, Tabela, Coluna, idObjecto, Valor**

**FROM logs**

**WHERE Enviado = 0 AND Data < d**

**INTO OUTFILE 'C:/Users/…/SID/logs.csv'**

**FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'**

**LINES TERMINATED BY '\n';**

**UPDATE logs SET Enviado = 1 WHERE Data < d;**

**Atenção:** Os caminhos de escrita dos ficheiros (*paths*) do Windows no MySQL têm de ser escritos com “/” e não com a tradicional “\”.

**Criação de Eventos:**

Para a exportação periódica, usamos Eventos. Os eventos são semelhantes aos triggers, no entanto, não activam com o desenrolar de acções, mas sim periodicamente. Uma amostra do evento de exportação do *Main*:

**CREATE definer='SistemsGestaoDados'@ 'localhost' EVENT Exportar**

**ON SCHEDULE EVERY 1 hour starts current\_timestamp()**

**ON COMPLETION PRESERVE**

**DO call ExportCSV();**

Como podemos verificar, o Evento invoca o *Stored Procedure* ExportCSV() de hora a hora através do Utilizador ‘SistemsGestaoDados’ que vai ter as permissões sobre todos os *Stored Procedures* relacionados com a exportação e importação de dados.

**Sistema de prevenção de erros**

Após ponderar várias soluções possíveis para este problema, como comunicação entre servidores através de ficheiros, decidimos que a solução mais simples (reduzindo a complexidade do sistema) para implementar seria meter um “downtime” máximo, ou seja, o sistema faz exportação e importação em tempos diferentes permitindo assim, em caso de queda de um dos sistemas, a recuperação e resolução de problemas, prevenindo assim uma importação falhada e possíveis erros como duplicação ou omissão dos logs.

**IMPORTAÇÃO DO FICHEIRO (DESTINO):**

Ao contrário da exportação, encontramos dificuldades na importação devido ao MySQL não permitir que os *Stored Procedures* executem o comando LoadIntoFile por razões de segurança. Por essa mesma razão, optamos por criar uma tarefa no Windows para fazer a importação, logo não houve necessidade de criação de *Stored Procedures* na Base de Dados destino. Essa tarefa por vez irá executar um ficheiro **.bat** que por si irá invocar um outro ficheiro **.sql** (query de importação).

O processo começa pela a criação de um ficheiro **.bat** e **.sql**.

O ficheiro .sql irá ter a *query* de importação de dados. Exemplo:

**LOAD DATA infile 'C:/Users/…/SID/logs.csv'**

**INTO table logsimport**

**FIELDS TERMINATED BY ',' ENCLOSED BY '"'**

**LINES TERMINATED BY '\n';**

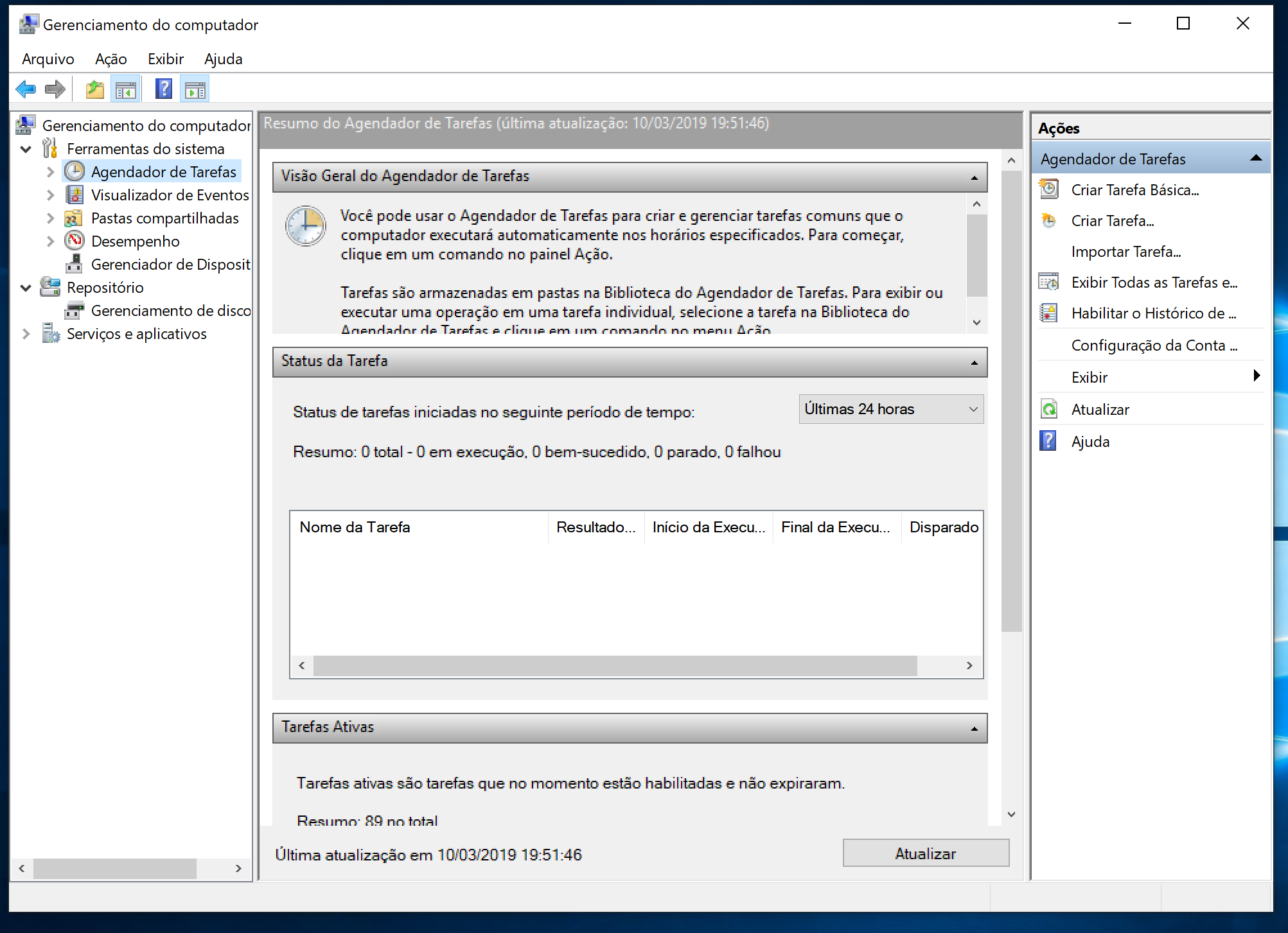
De seguida, iremos criar o ficheiro .bat. Exemplo:

**echo off**

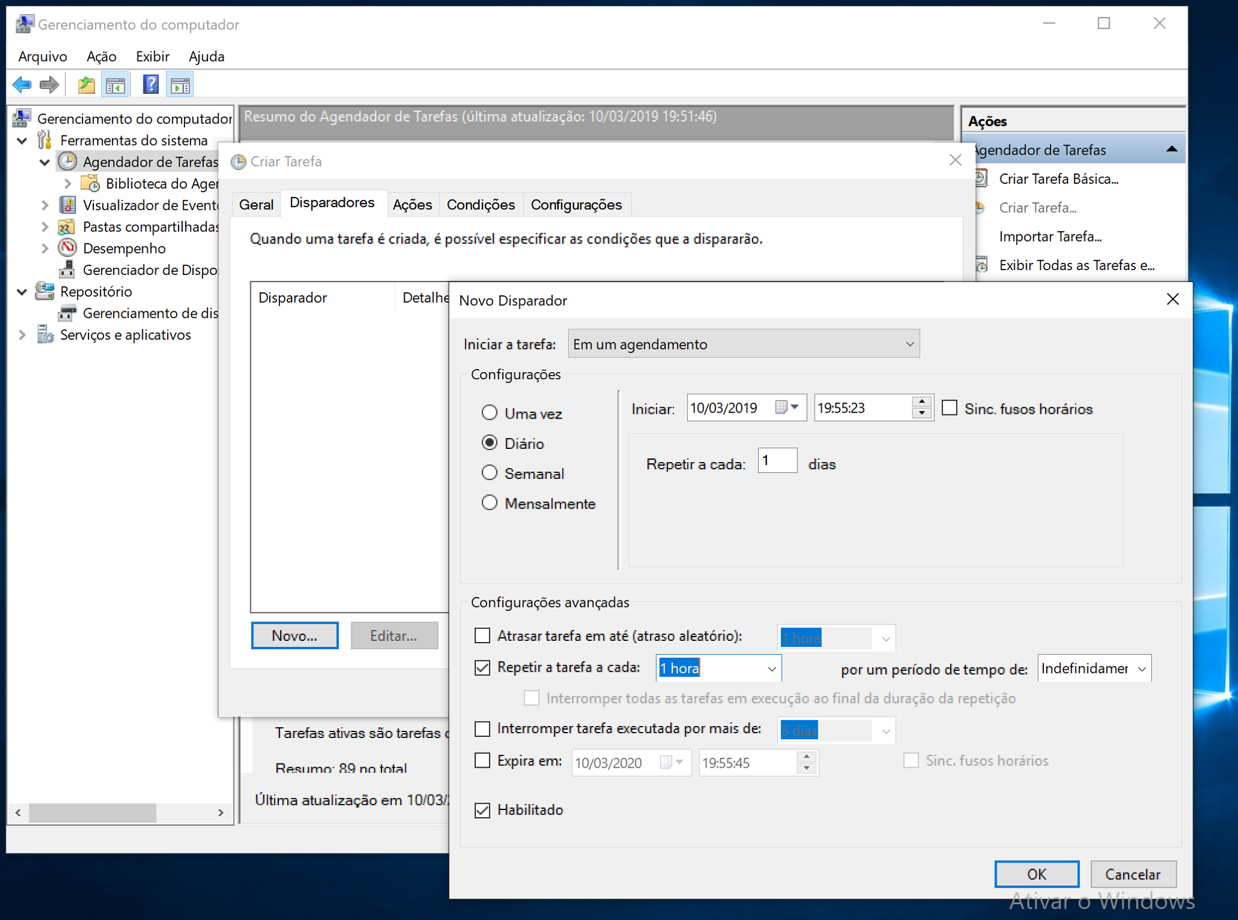
**sqlcmd -S <ComputerName>\<InstanceName> -i <sql\_file\_Path>**

**set /p delExit=Press the ENTER key to exit...:**

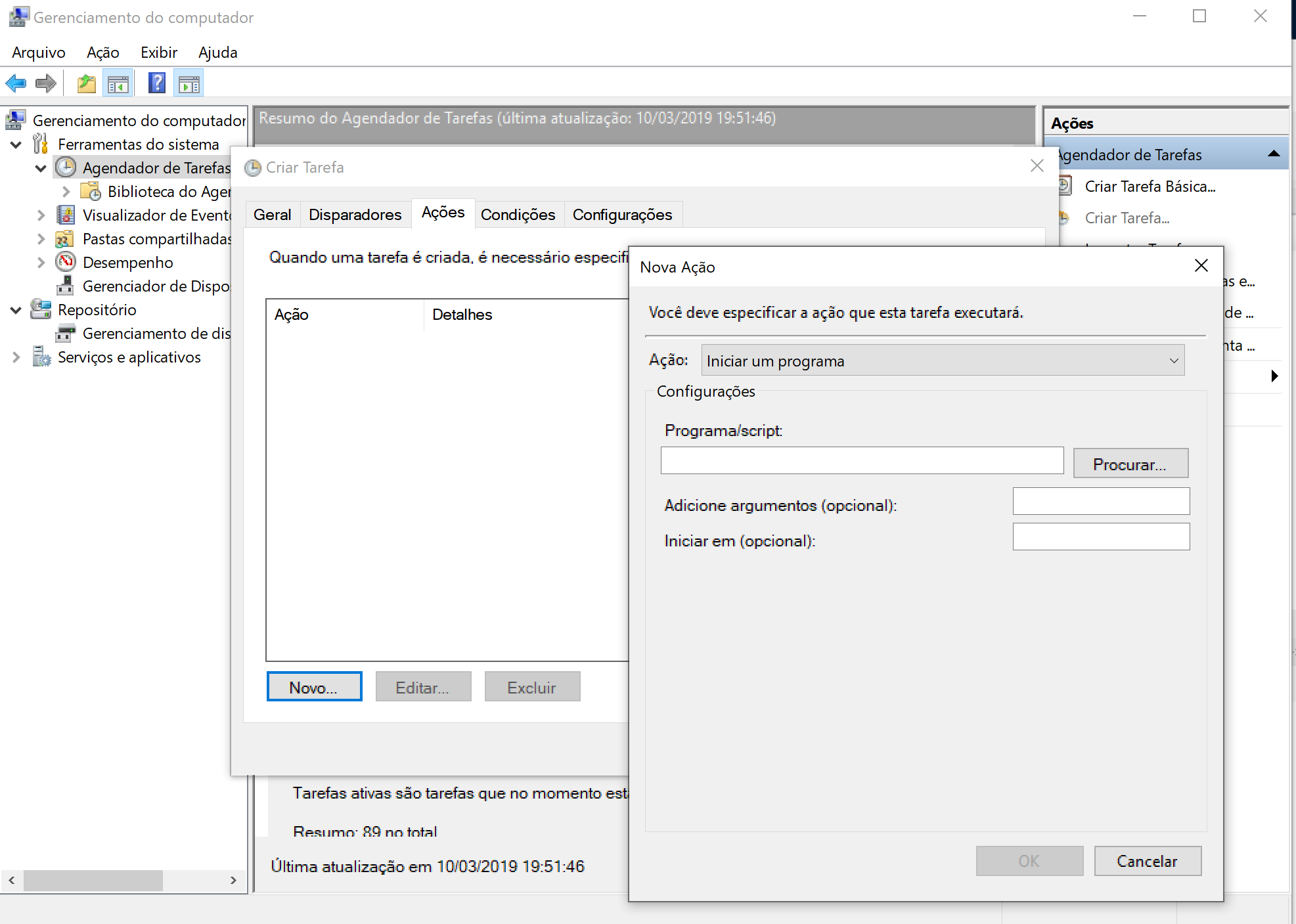
A seguir, iremos abrir o painel de Gestão de computadores para criar a tarefa do Windows. Selecionamos o Programador de Tarefas no menu à esquerda.



Agora clicamos no menu à direita “criar tarefa”. Selecionamos a configuração de diário, onde especificamos que a tarefa é realizada de hora em hora.

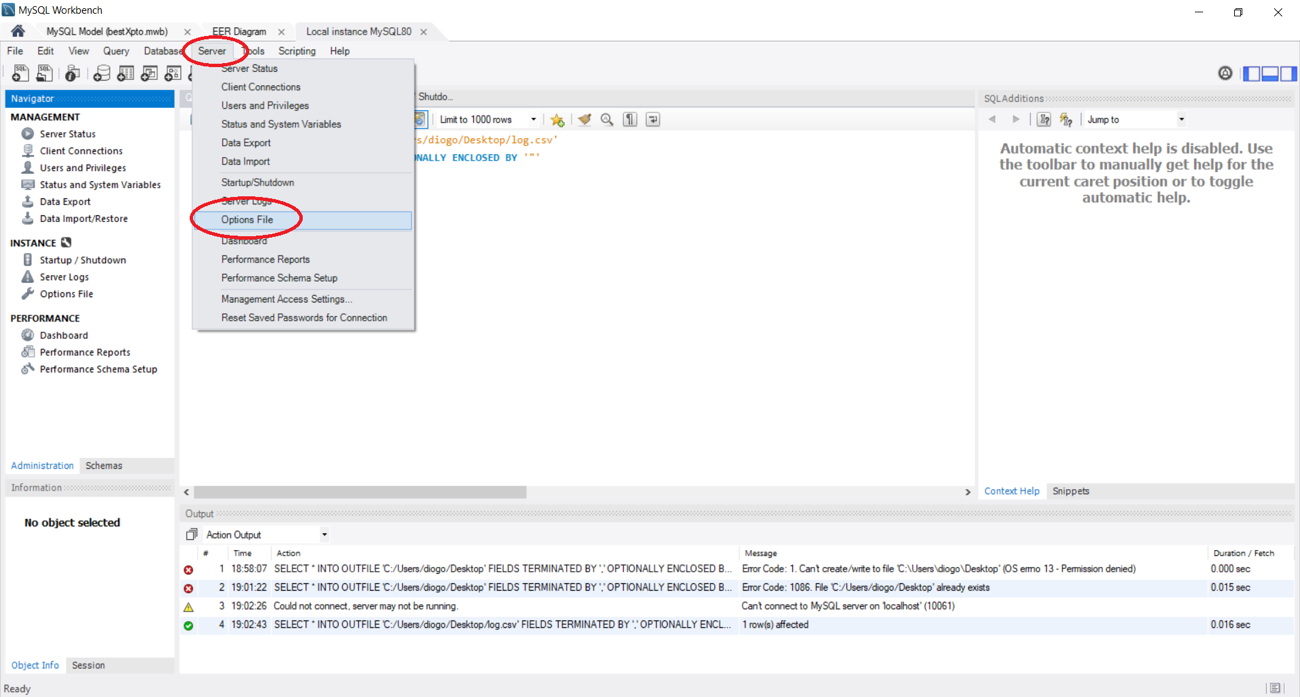


Para terminar, selecionamos o separador Ações e clicamos em “Novo”. Uma janela “Nova Ação” irá abrir onde poderemos inserir o path do ficheiro .bat no campo “Programa/script”.

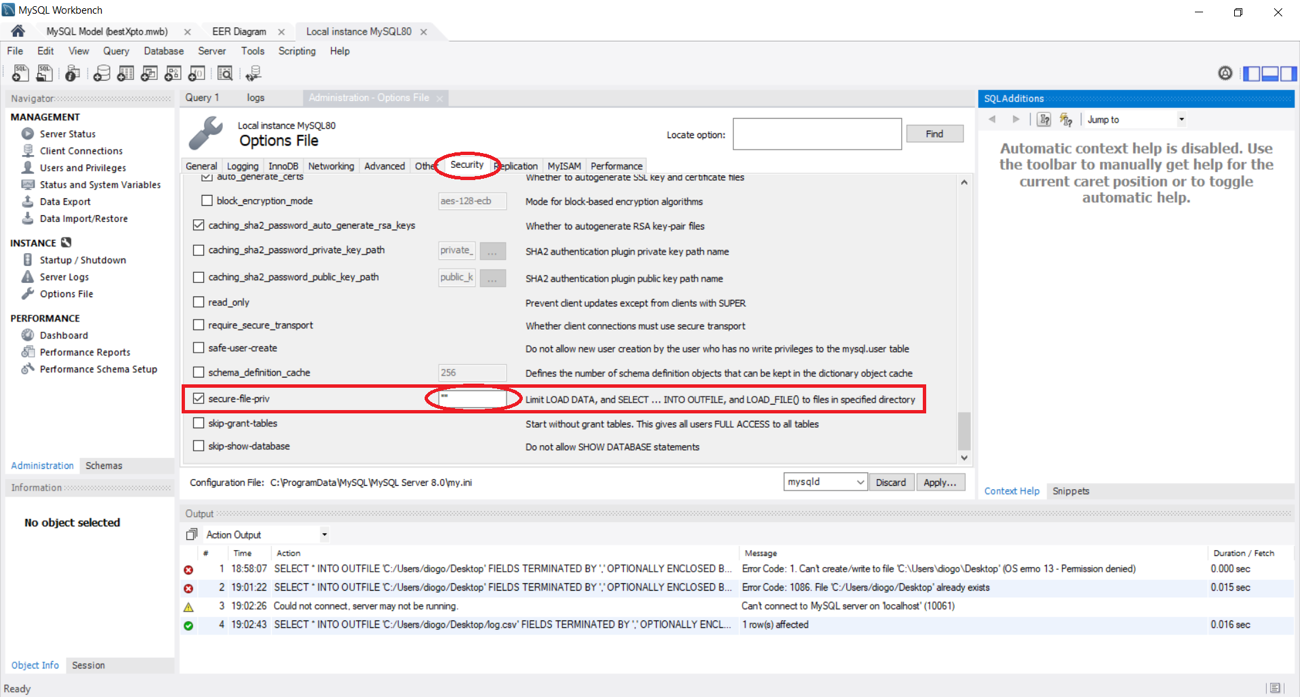


**ERROS POSSÍVEIS:**

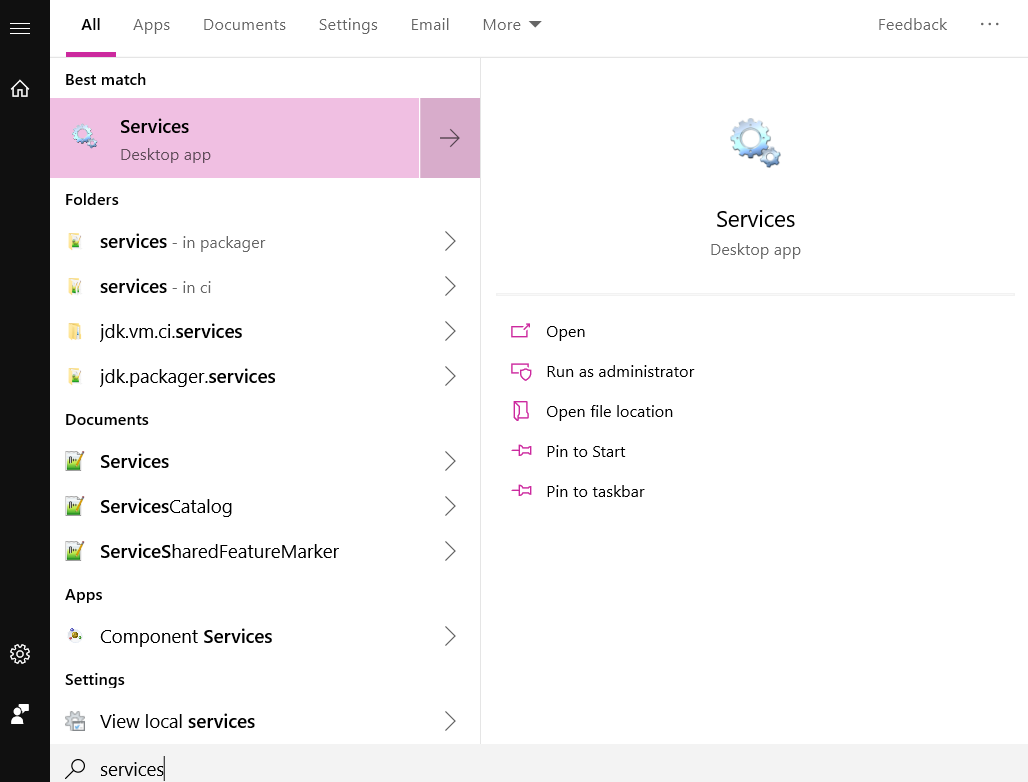
**Caso tenha o erro “Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement”,** clique na janela server, onde abrindo o menu, carregamos em options file.



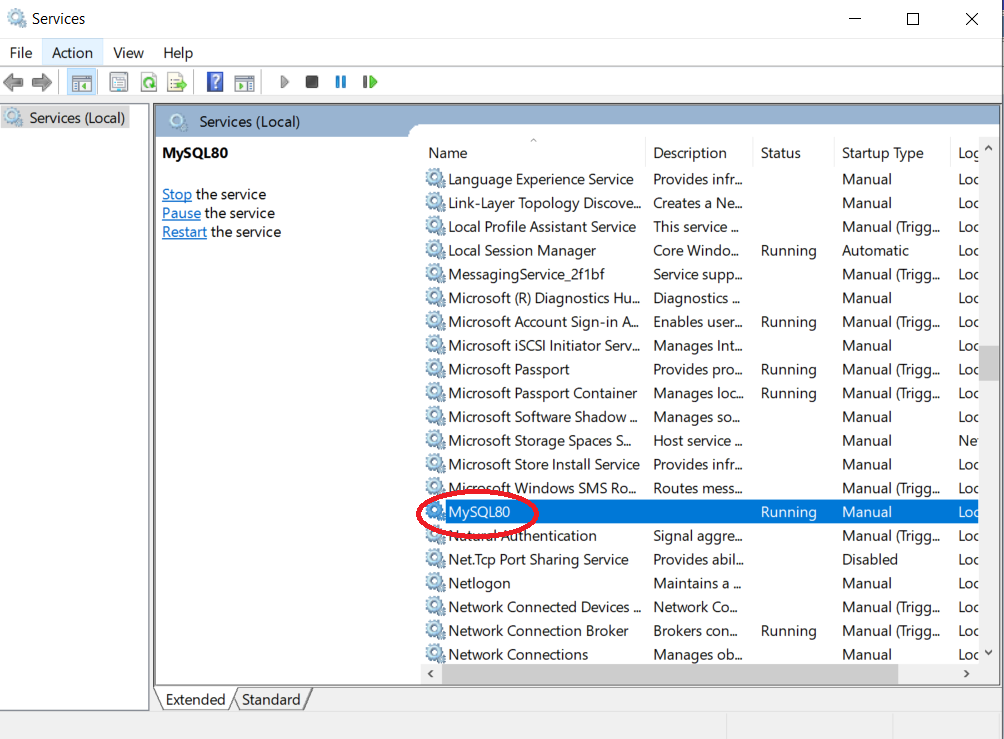
Prossegue-se clicando na opção Security, procurar pela opção *secure-file-priv*, (caso não esteja selecionado, selecionar) e subsitituir o *path* na caixa de texto por “”.



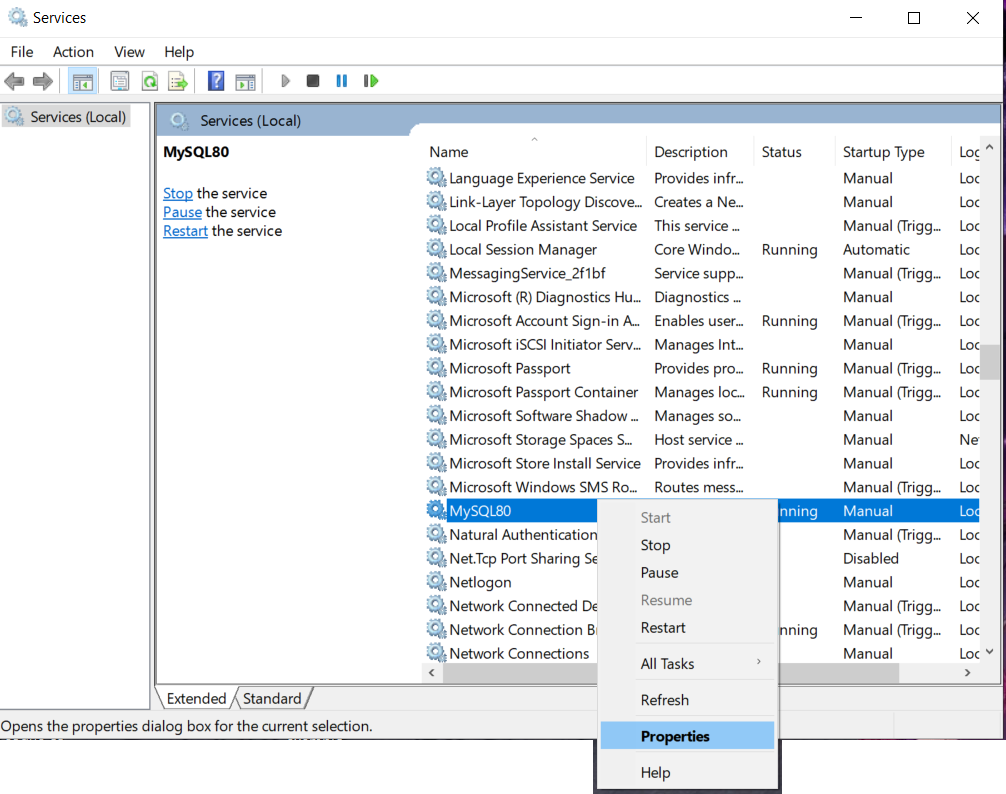
**Caso aconteça o erro “ERROR 1 (HY000): Can't create/write to file *'path'* (Errcode: 13 "Permission denied")”,** abrir o painel de serviços (assumindo que o OS é Windows).



Procurar pelo o serviço MySQL80.



Clicar com o botão direito do rato, e prosseguir indo às propriedades.



Ir à janela do Log On, e trocar de *This account* para *Local System account,* e caso pretenda usar o desktop, também selecionar a opção abaixo.

