Analise de Console.log e Trace

Outubro 2017

OBJETIVOS

- O objetivo deste treinamento é capacitar analistas internos na analise do Console Log e trace gerado pelo DBAccess.

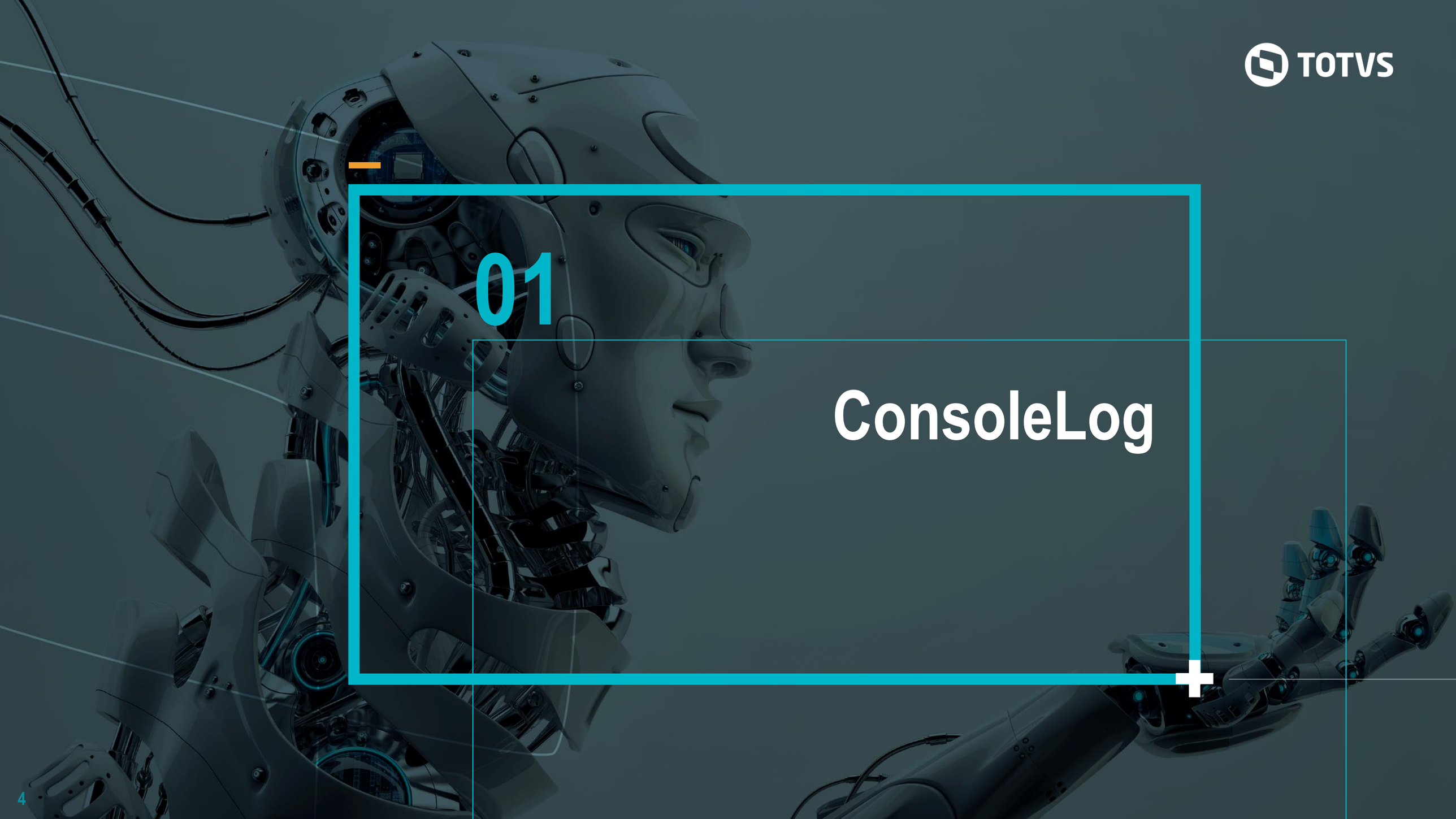
HOJE VAMOS FALAR SOBRE

1. Console.log

- Conceito
- Configuração e Geração do Console.Log
- Estrutura da Thread
- Analise do Log
- Fw LogProfiler

2. Trace

- Conceito
- DBAccess Monitor
- Trace
- Analise



01

ConsoleLog

Conceito

ConsoleLog

ConsoleLog parte do princípio de ser uma configuração da seção [General] do TOTVS | Application Server onde sua função é gravar o log das mensagens de execução.

Quando a chave é ativada, as informações, erros, mensagens de conout, etc. de execuções dos programas e serviços são gravadas em um arquivo de log chamado console.log no diretório do disco onde encontra-se os executáveis do TOTVS | Application Server, exceto quando configuradas com o arquivo personalizado com a chave ConsoleFile

LogProfiler

Trata-se de uma chave que quando habilitada faz a gravação detalhada da execução de processos no log de console do Totvs | Application Server

Configuração e Geração do Console.Log

TOTVS | Application Server (appserver.ini)

A geração do arquivo Console.log com as informações detalhadas do recurso Logprofiler depende da inclusão de determinadas chaves no arquivo appserver.ini

– **ConsoleLog**=1 na seção [GENERAL]

- Essa configuração surtirá efeito quando o serviço for reiniciado.

– **LogProfiler**=1 na seção [Environment]

- Essa configuração será ativada ao iniciar o SmartClient

– **ConsoleFIle**=<ex: C:\console\server_01.log> na seção [GENERAL]

- Essa configuração possíbilita definer um local e um nome para o arquivo de Log que no padrão é gerado na pasta AppServer.

Configuração e Geração do Console.Log

Para gerar um arquivo Console.Log consistente para ser analisado é necessário seguir os seguintes passos:

- Ter um ambiente de separado de testes com acesso exclusivo

- Limpar o arquivo console.log existente da pasta AppServer

- Efetuar o teste com a rotina que esta com problema de performance

Depois da rotina concluir o teste é necessário sair do Protheus para o Console.Log ser gravado com todas as informações que esperamos analisar.

Observação: Caso o arquivo (console.log) ultrapasse o tamanho de 5MB, o arquivo será renomeado, automaticamente, para a extensão .BAK e um novo será criado.

Configuração e Geração do Console.Log

Novo procedimento para geração do Log

- A partir da última LIB disponibilizada no portal é possível gerar o arquivo com o recurso LogProfiler direto da rotina que esta apresentando lentidão.

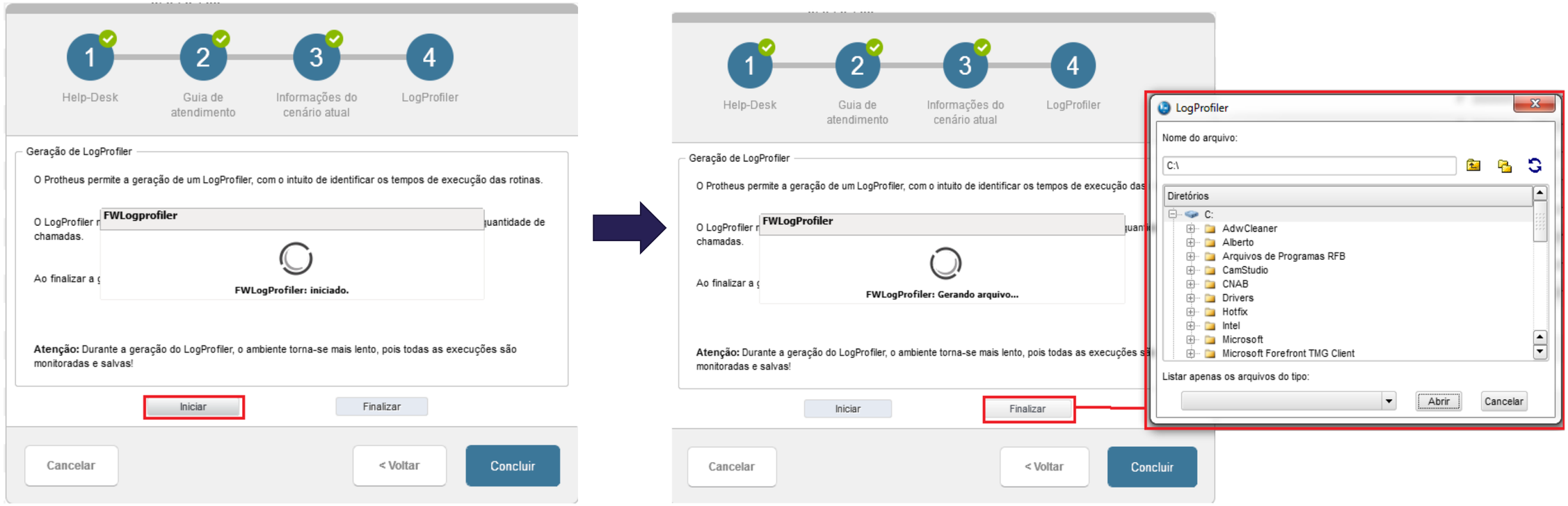
- O novo Shitf + F6 apresentara a data dos fontes com um inspetor de objetos do TOTVS DevStudio na tela 2 e na tela 4 será possível iniciar e finalizar o processamento.

- A diferença entre o arquivo gerado pelo novo Shift+F6 com o arquivo gerado pela configuração no APPSERVER.INI esta na estrutura da Thread, pois no novo método o arquivo vai conter apenas informações detalhadas das chamadas das funções enquanto o configurado no INI vai gerar a Thread completa contendo informações e sobre a máquina que gerou o arquivo, informações detalhadas das funções e a data dos fontes

Configuração e Geração do Console.Log

Novo procedimento para geração do Log

- O sistema começa a rastrear o processamento a partir do momento que o usuário clica no botão Iniciar e após a realização do teste ao voltar no Wizard do Shif+F6 e clicar no botão finalizar será solicitado um diretório para gravação do arquivo



Estrutura da Thread

Estrutura da Thread com a informação do programa executado

Início da Thread

--- BEGIN APP PROFILER (THREAD [X]) ----------

Informações sobre ambiente e computador

---------- ADITIONAL MEMORY INFO - RUNNING ENV ----------

Informações sobre consumo de memória e tempo total gasto

--- CALLS DETAILED INFO (SORT BY NAME) ---

Informações detalhadas das chamadas das funções

--- SOURCE DETAILED INFO ---

Data dos fontes executados na Thread

--- END APP PROFILER ---

Fim da Thread

Estrutura da Thread

Estrutura da Thread com a informação do programa executado

--- BEGIN APP PROFILER (THREAD [X]) ----------

[user: usuário de rede]

[computer: hostname]

[environment: nome do ambiente]

[rpodb: tipo do RPO]

[localfiles: dicionário de dados]

[apo: diretório do rpo]

[begin: data e hora de criação da thread]

[spent: 00:03:28]

[build: binário]

[thread: 9808]

[remark: empresa e filial logado | usuário | módulo | rotina]

Estrutura da Thread

Estrutura da Thread com a informação do programa executado

---------- ADITIONAL MEMORY INFO - RUNNING ENV ----------

Total de chamadas internas [X] tempo [X] em segundos

Total de Chamadas CIMethod [X]tempo [X] em segundos

Chamadas APO do usuário total [X]tempo [X] em segundos

--- CALLS DETAILED INFO (SORT BY NAME) ---

CALL <FUNNAME> (<RECURSO>) C <CALLTOT> T <TOMERTOT> M <SLOWEST>--

<FUNNAME> - Função executada

<RECURSO> - Para funções básicas da linguagem AdvPL, o recurso é Internal e para funções de RPO o recurso é o nome do código-fonte

<CALLTOT> - Numero total de chamadas da função

<TIMERTOT> - Tempo total utilizado por todas as chamadas

<SLOWEST> - Chamada desta função que demorou mais tempo

Estrutura da Thread

Estrutura da Thread com a informação do programa executado

FROM ORIGEM LN <LINE> C <THISCALL> T <THISTIME> M <SLOWEST>--

<ORIGEM> - Função e respectivo código-fonte

<LINE> - A linha do código fonte onde a chamada foi feita

<THISCALL> - Quantas chamadas para a função <FUNNAME>

<THISTIME> - Quandotempo gasto com as chamadas

<SLOWEST> - Qual a chamada mais lenta

--- SOURCE DETAILED INFO ---

[Thread X] <NOME, DATA e HORA do fonte>

------ END APP PROFILER ---

Analise do Log

Como interpretar as informações geradas no Log

Uma vez obtido um log de profiler com as informações, a análise das informações geradas para a tomada de decisão e identificação de gargalos ou pontos críticos deve seguir um roteiro pré-determinado.

Identificar onde é gasto aproximadamente 80% do tempo de processamento, sempre começando pelas funções básicas da linguagem (funções identificadas no profiler como “Internal”).

Verificar dentro das funções identificadas começando pela que consome maior tempo, se existem diferenças de tempo significativas nas chamadas de acordo com a origem e/ou se existem pontos que chamam muitas vezes a mesma função.

Nas análises iniciais feitas com este recurso em diversos fontes, algumas ocorrências comuns foram identificadas

Analise do Log

Como interpretar as informações geradas pelo Profiler

Identificaronde é gasto a maior parte do tempo de processamento sempre começando peças funções básicas da linguagem (funções identificadas no profiler como “Internal”), começando pela que consome maior tempo, se existem diferenças de tempo significativas nas chamadas de acordo com a origem e/ou se existem pontos que chamam muitas vezes a mesma função.

Função DBSeek() em tabela acessada via TOTVS | DBAccess

Esta função realiza uma busca a partir de uma chave e posiciona o registro da tabela que atende a condição de busca.

Por exemplo, verificando um determinado profiler, em uma operação de 50 segundos, 23,77 segundos foram gastos com instruções DBSeek(). Vide exemplo abaixo, onde foram isoladas apenas as maiores chamadas e origens:

CALL	dbseek(Internal)		C	3237	T	23.770	M 0.078
FROM TSTCLASS:GETCPOINFO	(TSTX0010.PRW) (123)		C	458	T	14.039	M 0.078

Neste caso, o código-fonte foi analisado e foi possível fazer uma *query* leve no banco de dados que retorna em uma requisição todos os registros necessários para processamento, sem precisar fazer as 458 chamadas de DBSeek().

Analise do Log

Como interpretar as informações geradas pelo Log

Uso da função AScan() em arrays grandes

Função que percorre um array procurando por um valor especificado. Pode ser especificado um valor a ser buscado, ou pode ser informada uma condição de busca através de um bloco de código.

Se pensarmos em um array de 1000 elementos onde o elemento procurado foi encontrado na posição 750 do array, a função Recno() foi chamada 750 vezes, seria mais vantajoso neste caso armazenar o Recno() em uma variável de memória e fazer a busca com ele.

Tempo alto em determinadas operações DBSkip()

Função que desloca para outro registro na tabela corrente.

Em determinadas circunstâncias a tabela em questão estava utilizando uma expressão de filtro que não é bem elaborada havendo internamente na aplicação um tráfego adicional de registros para validação no TOTVS | Application Server onde muitos registros são lidos e trafegados, mas poucos registros atendem a condição de filtro.

Analise do Log

Como interpretar as informações geradas no Log

```
CALL                dbskip (       Internal) C  1458865 T   134.200 M     0.019 D 0
[Thread 9808] -- FROM      DATAVALIDA (DATAVALI.PRG) LN     69 C       12 T     0.002 M     0.001 D 0
[Thread 9808] -- FROM       FA190IMPR4 (FINR190.PRX) LN   2852 C        1 T     0.000 M     0.000 D 0
[Thread 9808] -- FROM       FA190IMPR4 (FINR190.PRX) LN   2916 C       34 T     0.000 M     0.000 D 0
[Thread 9808] -- FROM       FA190IMPR4 (FINR190.PRX) LN   3335 C      465 T     0.012 M     0.002 D 0
[Thread 9808] -- FROM       FA190IMPR4 (FINR190.PRX) LN   3381 C  1445360 T   134.152 M     0.019 D 0
[Thread 9808] -- FROM       FA190IMPR4 (FINR190.PRX) LN   3712 C        3 T     0.000 M     0.000 D 0
[Thread 9808] -- FROM       FWLOADSM0 (FWFILIAL.PRW) LN   1263 C        3 T     0.000 M     0.000 D 0
[Thread 9808] -- FROM FWPERGUNTEREPORT (PROTHEUSFUNCTIONREPORT.PRW) LN    459 C       82 T     0.001 M     0.001 D 0
[Thread 9808] -- FROM          PERGUNTE (MSLIB.PRW) LN   1789 C       41 T     0.000 M     0.000 D 0
[Thread 9808] -- FROM        SENDSX2 (APLIB100.PRW) LN   3684 C     6377 T     0.017 M     0.012 D 0
[Thread 9808] -- FROM        SENDSX2 (APLIB100.PRW) LN   3700 C     6377 T     0.016 M     0.001 D 0
[Thread 9808] -- FROM      TABLEINDEX (APLIB200.PRW) LN   1775 C      110 T     0.000 M     0.000 D 0
[Thread 9808]
```

A informação na linha CALL mostra que a função DBSkip foi chamada 1.458.865 vezes, consumiu um tempo total de 134,200 segundos e a chama mais demorada levou 0,019 segundos.

Analisando a informação FROM a chamada mais demorada ocorre na função FA190IMPR4 linha 3381 do fonte FINR190.PRX.

Fw Log Profiler

Conceito

Tem como objetivo a analise de Log Profiler gerado pelo TOTVS | Application Server sendo capaz de ler e organizar o Profiler de diversas threads e ler também o Profiler do webservice

A ferramenta possibilita a ordenação das informações por chamadas da função, tempo total da chamada e a chamada desta função que demorou mais tempo

Por se tratar de um facilitador para a analise do console.log é indispensável a analise do console.log em um editor de texto pois nele conseguiremos identificar a thread a ser analise e data dos fontes envolvidos no processo

Fw Log Profiler

Instalação do Plugin Fw Log Profiler

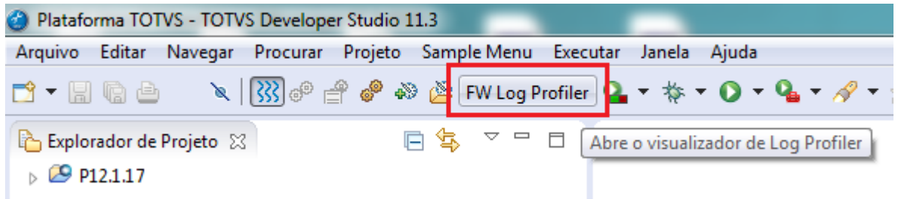
Para instalar o plugin será necessário executar os seguintes passos:

Abrir o TDS - Totvs Developer Studio e clicar no item de Menu “**Ajuda > Instalar Novo Programa**“

Clique no botão **"Adicionar“**, defina um nome e informe o caminho para download do Plugin - http://10.171.66.27/update_site_toolkit/

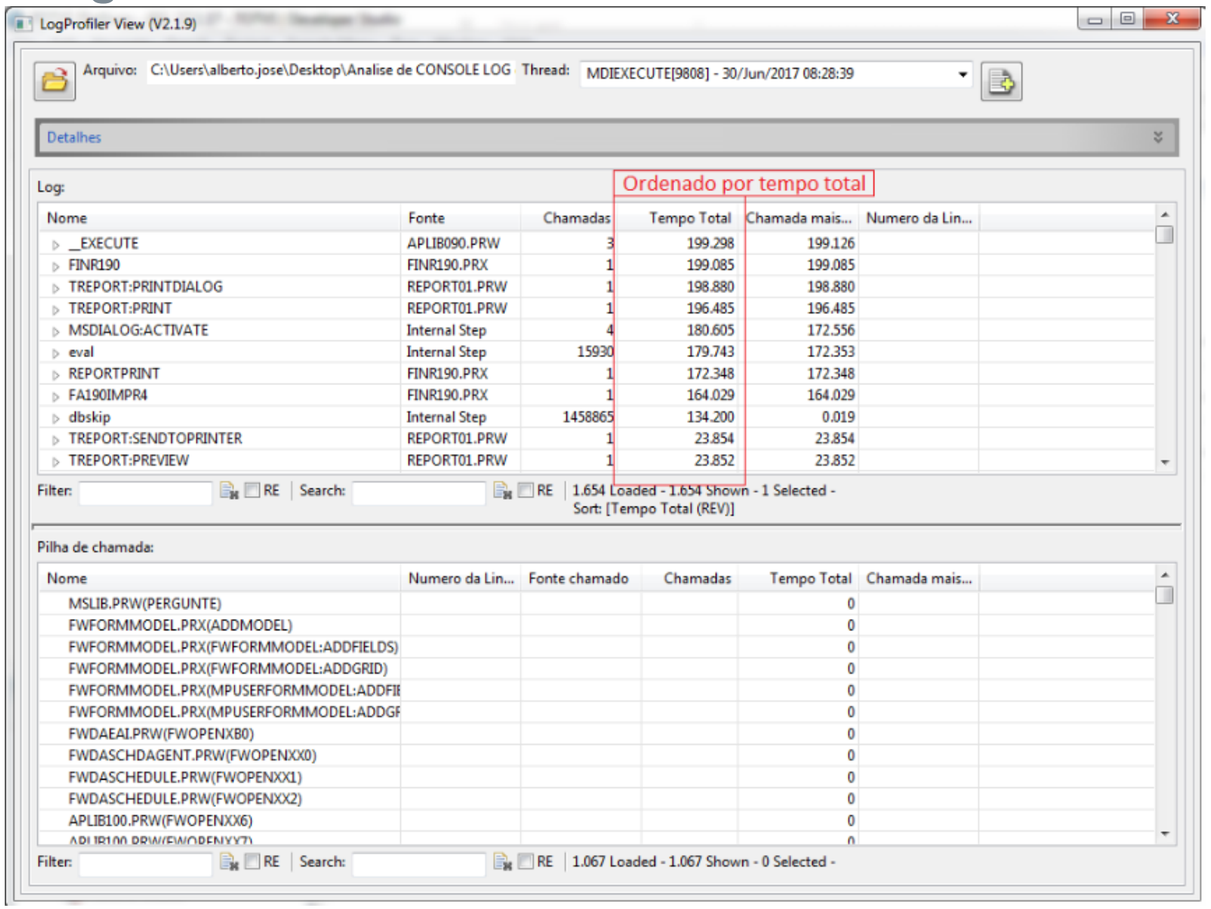
Ao concluir a instalação do plugin o TDS deve ser reiniciado

Depois de reiniciar o TDS, um botão ‘Fw Log Profiler’ vai aparecer na parte superior



Fw Log Profiler

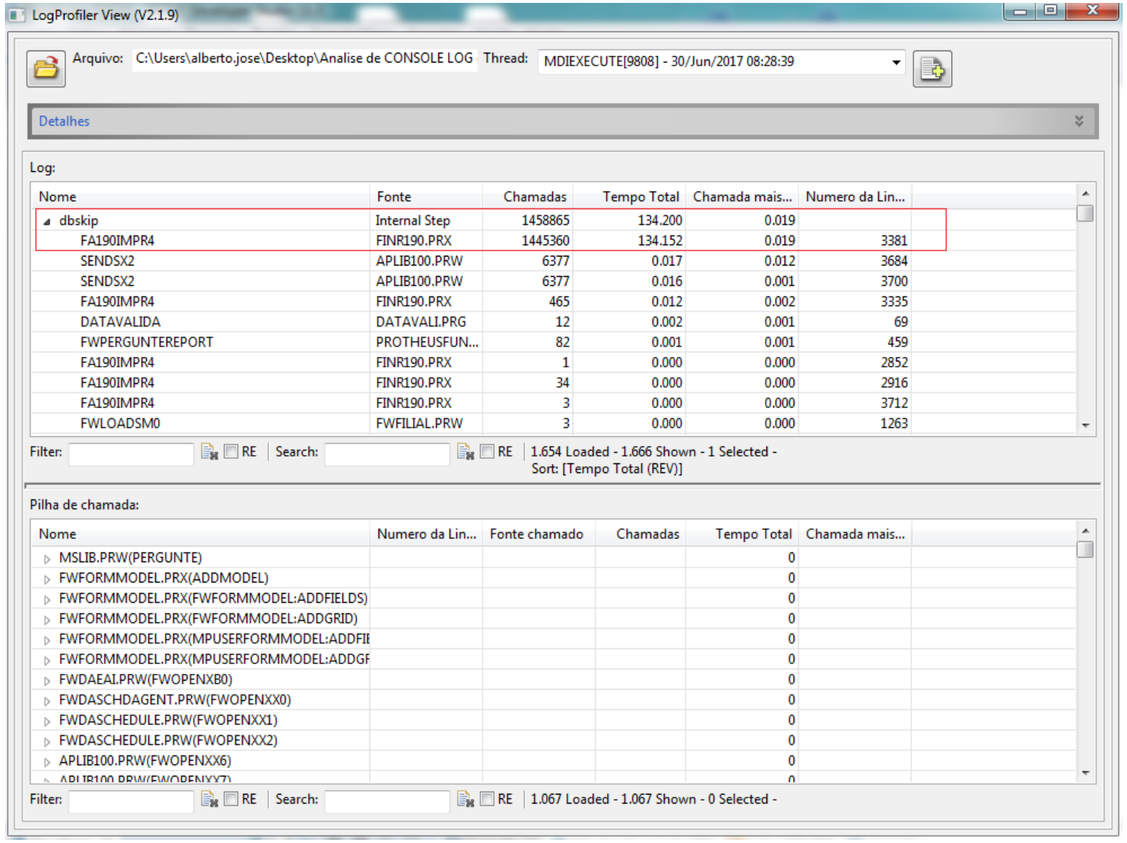
Analise do Log



Nesta imagem é possível ter um norte para analisar a causa do problema performance pois ordenando as funções por tempo total é possível notar que a função DBSkip representa cerca de 70% do processamento com uma quantidade significativa de chamadas.

Fw Log Profiler

Analise do Log



Se expandirmos a chamada DBSkip é possível identificar que praticamente todo tempo gasto com a execução da função foi dentro da função FA190IMPR4 e esse DBSkip esta sendo chamado na linha 3381 do fonte FINR190.PRX. A data do fonte encontramos no arquivo console.log pelo editor de texto e a para identificar a causa da lentidão é necessário possuir um conhecimento intermediário na linguagem AdvPL

02

Trace

Conceito

Trace

É o nome dado ao arquivo gerado por um recurso da guia Usuários do TOTVS | DBAccess Monitor que possibilita rastrear a atividade de um usuário

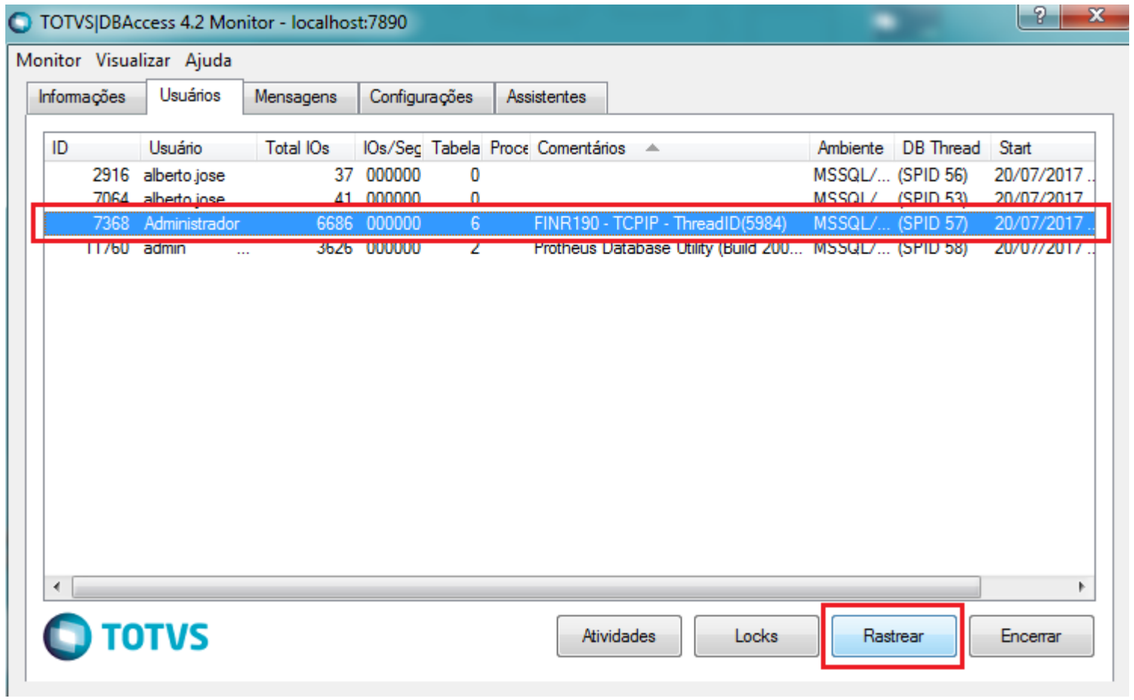
O arquivo é normalmente usado para identificar o ponto por exemplo que o sistema está em looping além de fornecer informações sobre todas as queries enviadas ao servidor pelo usuário selecionado

Este tipo de detalhamento é útil ao desenvolvedor, e traz informações técnicas detalhadas do processo de acesso aos dados geradas no Trace

DBAccess Monitor

Para gerar um arquivo Trace consistente para ser analisado é necessário seguir os seguintes passos

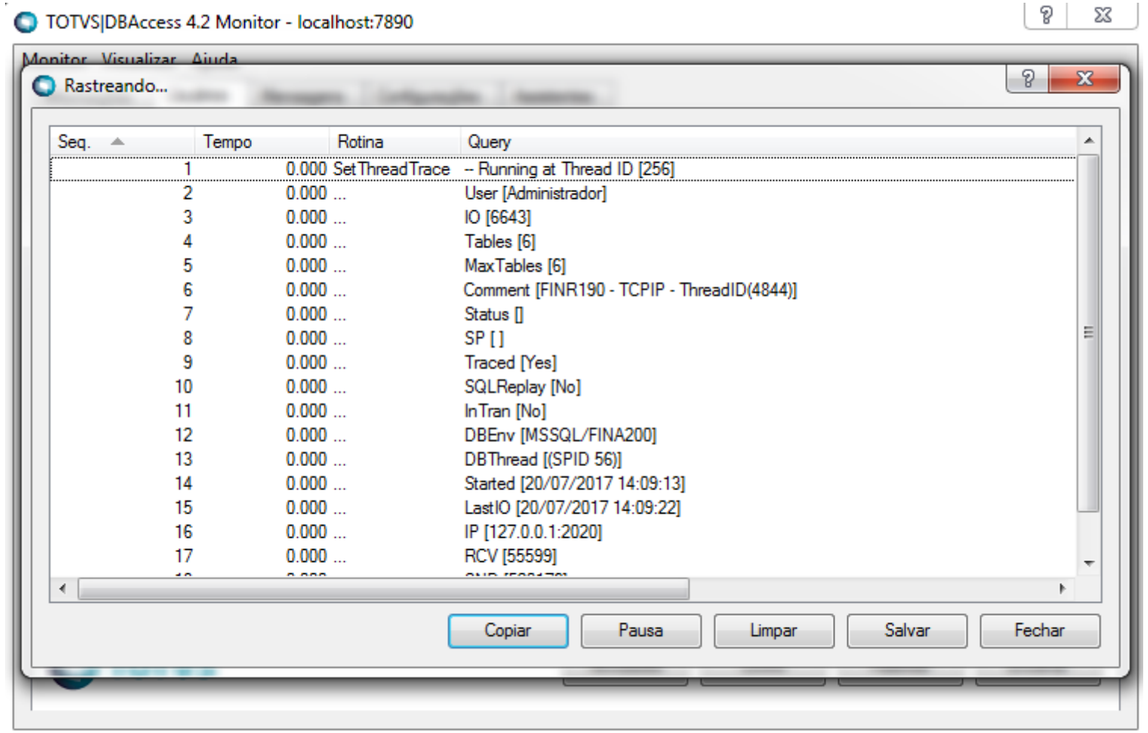
Acessar a guia Usuários do DBAccess Monitor, selecionar o usuário com a rotina expecífica e clicar para rastrear para dar inicío a geração do trace e somente depois execute a rotina do sistema



DBAccess Monitor

Ao rastrear a atividade de um usuário:

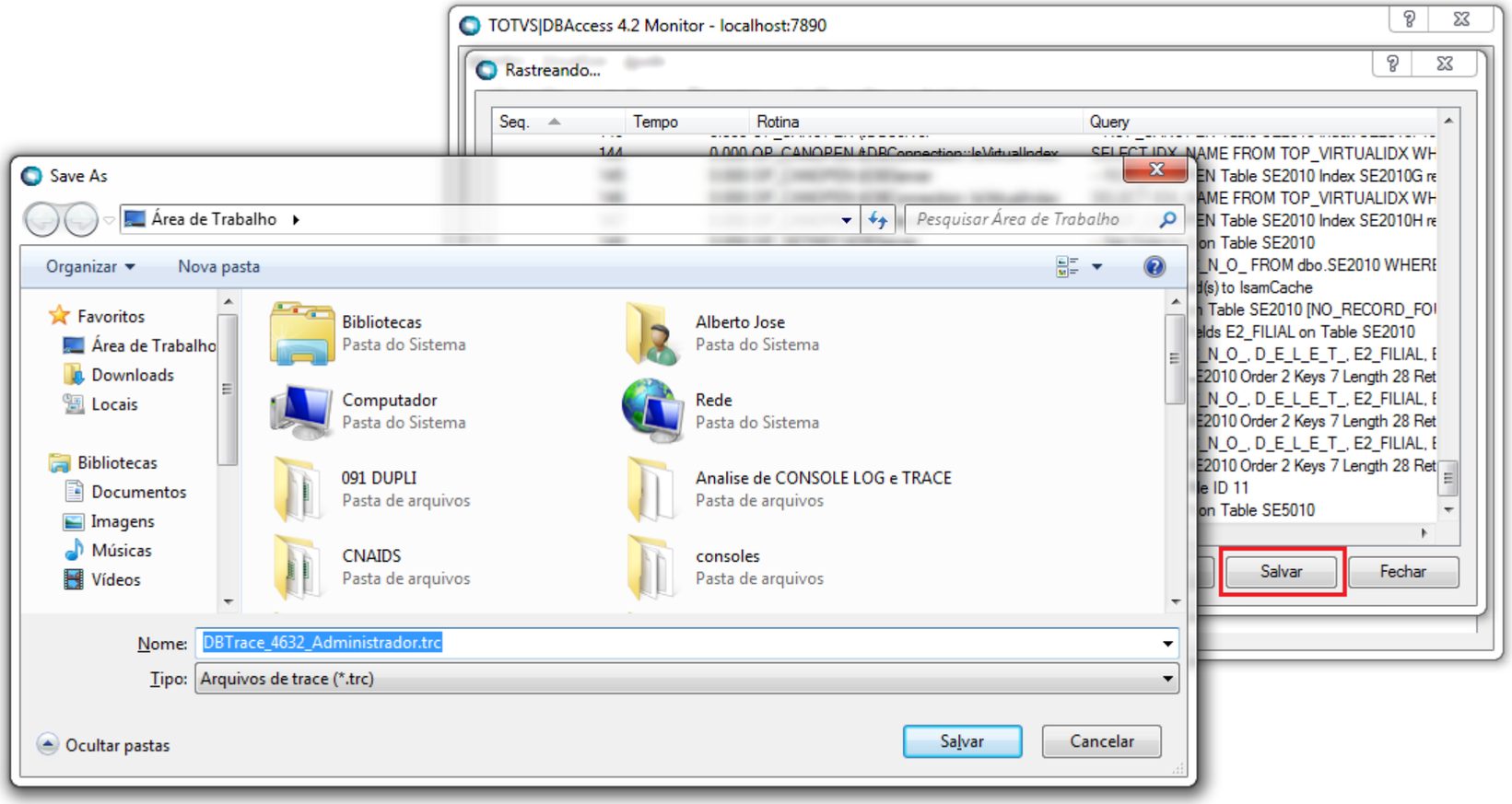
O sistema apresenta uma nova tela, relacionando as informações de acesso do usuário, divididas em duas colunas:



Seq.	Informa a sequência das colunas realizadas.
Tempo	Informa o tempo utilizado para a consulta.
Rotina	Informa através de qual rotina interna do DBAccess que a instrução está sendo executada.
Query	Descreve as operações realizadas.

DBAccess Monitor

Ao concluir o processamento da rotina clique em salvar e defina o diretório para salvar o arquivo



Detalhamento de Operações

As operações mostradas em uma conexão rastreada constituem em informações enviados ao banco de dados para execução, atualização ou consulta, e também podem informar operações solicitadas pelo Protheus ao DBAccess, que não necessariamente envolveram um acesso ao Banco de Dados.

Todas as instruções onde a coluna "Query" iniciar com -- (dois hífens), são detalhamentos de operações solicitadas ao DBAccess que não implicaram na execução de um statement diretamente no Banco de Dados.

Exemplo:

-- Set Order to <N> on Table <X>

Informa que a aplicação Advpl solicitou a troca da ordem de navegação de indice da tabela <X>, para a ordem do índice <N>, através da instrução DbSetOrder()

Trace

-- Load [<N>] record(s) to IsamCache

Informa que <N> registros foram armazenados no cache de navegação ISAM de tabela.

-- First Recno on Table <X> [FLAGS]

Informa que a aplicação Advpl executou uma instrução de navegação para o top de uma tabela ou Query (DBGoTop). No caso de uma Query, a tabela <x> virá com o nome em branco.

-- Begin Skip File <X> ID <N> Count <S> From Record <F>

Informa que a aplicação Advpl solicitou um pulo de registro (SKIP) ao DBAccess, na tabela <X> , para pular <S> registro(s), a partir do registro atual <F> .

-- Opened Query ID <N>

Informa quem uma Query solicitada pela aplicação Advpl foi aberta sob o ID <N>.

AS queries feitas por uma rotina Advpl são indicadas no trace com a rotina [X::GetQueryFile]

Trace

-- Close Query File ID <N>

Informa que uma Query aberta pela aplicação Advpl foi fechada.

-- Begin Seek Table <X> Order <I> Keys <S> Length <T>

Informa que uma operação de SEEK foi iniciada na tabela X, usando a ordem de índice I , onde foram informado(s) <S> campo(s) chave, com uma chave de busca com tamanho total <T>

-- End Seek : Return <R> RECNO <N> [FLAGS]

Informa o status da operação de busca -- SEEK -- executada. Caso a chave de busca exata não tenha sido encontrada, é retornado o código -25.

Como interpretar as informações do arquivo trace

A necessidade de analisar um arquivo trace depende muito do problema que o sistema apresentando e é usada para situações de looping

Looping por exemplo é mais fácil analisar o arquivo em um editor de texto e procurar as operações de SELECT na coluna Query para ver se o Recno é repetido constantemente e assim pegar esta operação e procurar no código fonte aonde é montada para se ter uma ideia do motivo do problema

Para outros tipos de problema é recomendado o uso do Excel pois o mesmo possibilita filtrar informações das colunas Rotina e Query afim de visualizar apenas um tipo de informação

Na maior parte dos casos que é necessário a analise do trace a causa do problema efetivamente é apenas encontrada através de um debug

OBRIGADO

Alberto Teixeira

Help Desk Financeiro – Suporte Investigativo

Alberto.jose@totvs.com.br

TOTVS

Tecnologia + Conhecimento são nosso DNA

O sucesso do cliente é o nosso sucesso

Valorizamos gente boa que é boa gente

totvs.com

company/totvs

blog.totvs.com

@totvs

fluig.com

#SOMOS**TOTVERS**