Aplicando Query Services (e quebrando a cara)

26/09/2012 10:00

Olá, zumbizada faminta por código.

No ano passado, o Fábio do ABAP101 fez uma série de posts explicando o que são e como funcionam os ***Object Services*** (OS). Eu gostei da ideia, e como fui recentemente envolvido numa demanda de construção de relatórios, achei que seria bacana aplicar o serviço de *queries* para ver como – e se – funciona. Infelizmente, apanhei e não gostei do resultado.

"Ele vai apanhar, vai apanhar muito. Não vai ter conversinha dessa vez" declarou Query Services em entrevista polêmica

O *help* da SAP sobre este assunto é fundamental para entender como funcionam as classes e os métodos relacionados a OS. Usando esse conhecimento e os exemplos que estão nos posts do ABAP101, é possível começar a usar persistência no código até com certa facilidade.

Só que com essa mesma facilidade, os problemas começam a ficar evidentes. O Fábio já apontou algumas das desvantagens de se usar OS, mas algumas coisas só se encontra quando se tenta implementar, não é mesmo?

A persistência não é persistente

Uma das primeiras características dos OS mencionadas no *help* é que **o serviço de persistência não é realmente persistente, porque não existe bloqueio de objetos transientes**. Isso significa que não existe uma administração global de objetos dentro do serviço de persistência; se, por exemplo, dois programas diferentes buscarem o mesmo objeto persistente ao mesmo tempo, cada programa terá a sua própria versão transiente em memória até que um deles execute um COMMIT WORK.

Isso é, sem falsa modéstia, inaceitável. Fica a pergunta a ser feita para a SAP: qual é a razão por trás disso? Por quê liberar uma API pela metade, só para que os desenvolvedores tenham o "gostinho" de usar um conceito amplamente difundido há muito tempo em outras linguagens que usam OO? As "melhorias" estão em planejamento, desenvolvimento ou foram abandonadas?

Atrito desnecessário

Há alguns pequenos problemas espalhados nas classes de OS. Exemplos:

- Os parâmetros dos métodos da interface IF_OS_QUERY_EXPR_FACTORY (responsável por criar filtros para as *queries*) só aceitam os valores como STRING;
- Os filtros de *queries* não são compatíveis com OpenSQL, então ao criá-los não é possível usar diretamente comparadores como IS INITIAL, IN, BETWEEN, etc.;
- Ainda na interface IF_OS_QUERY_EXPR_FACTORY, os métodos que criam filtros com base em outros filtros (CREATE_AND_EXPR, CREATE_OR_EXPR ou CREATE_NOT_EXPR) retornam a exceção CX_SY_REF_IS_INITIAL se um dos objetos estiver vazio. Isso é um problema quando se está montando um filtro composto, como por exemplo quando se concatena os valores dos *ranges* de uma tela de seleção e nem todos estão preenchidos;
- O método GET_PERSISTENT da classe agente retorna a exceção CX_OS_OBJECT_NOT_FOUND se o registro não for encontrado. Isso é uma solução muito mais drástica quando comparada ao OpenSQL, que nesse caso retornaria somente SY-SUBRC = 0 e a estrutura vazia, ou mesmo comparando com o método IF_OS_CA_PERSISTENCY~GET_PERSISTENT_BY_QUERY, que retornaria apenas uma tabela vazia;
- Não é possível usar diretamente a tabela de retorno da *query* (tipo OSREFTAB) para uma outra seleção, como num FOR ALL ENTRIES, sem antes montar um filtro para a nova *query*.

Esses são apenas exemplos, mas provavelmente a classe agente deve ter mais alguns problemas parecidos. É claro que tudo isso pode ser contornado pelo desenvolvedor, que pode por exemplo criar um classe *helper* para encapsular a chamada desses métodos. Mas esses comportamentos poderiam muito bem ser mais suaves e amigáveis, sem jogar exceções pra todo lado e mais bem adaptados aos elementos de Dynpro (principalmente para o uso de *ranges*) que já estão em uso há muitos anos.

Falta de integração

Isso é algo que o Fábio já cita em seus *posts*, mas ao implementar é que se percebe como a integração com outros elementos do SAP faz falta. Por exemplo, uma integração com o *Business Object Repository* (BOR) seria extremamente bem vinda para trazer/manipular dados de tabelas *standard*; em vez disso, somos obrigados a criar classes persistentes para cada tabela/visão, ou até mesmo para cada utilização diferente dos dados da tabela.

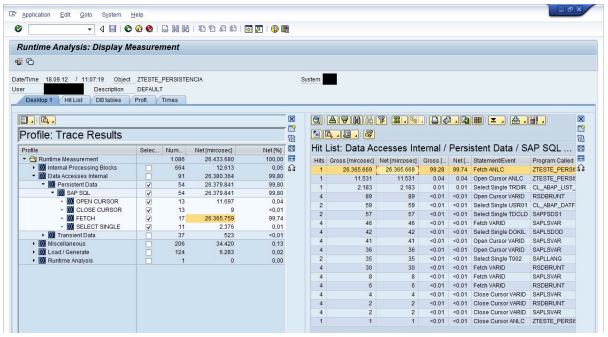
Performance

Tem gente que arrepia só de ouvir essa palavra, e com razão. É talvez o aspecto mais difícil de se controlar num programa ABAP. E é triste informar que OS não resolve esse problema.

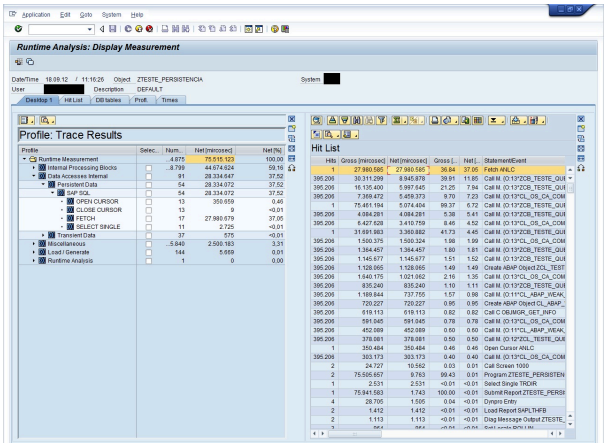
Eu fiz um teste de performance com uma classe persistente Z para a tabela ANLC (Campos de valor do imobilizado). No ambiente em que eu fiz o teste, essa tabela tinha ~7.500.000 entradas, e eu informei como parâmetros os campos BUKRS e GJAHR, de modo a restringir a seleção a ~400.000 resultados. Não é um volume muito grande, mas é uma aproximação do que um relatório real teria que fazer.

O programa apenas seleciona os dados de todos os campos da tabela usando OpenSQL ou OS, dependendo da escolha na tela de seleção. Com este programa, eu fiz dois testes: um rodando em *background* duas vezes, uma vez usando OpenSQL e outra usando OS, e depois executei a análise *on-line* da transação SAT, também com uma opção de cada vez. Eis os resultados:

Resultado dos jobs do programa teste (clique para ver maior)



Resultados da análise SAT usando OpenSQL (clique para ver maior)



Resultados da análise SAT usando Query Services (clique para ver maior)

Analisando os resultados:

- O programa, ao usar OS, demora o dobro do tempo para trazer a mesma quantidade de dados comparado com o uso de OpenSQL (a análise SAT mostra que é quase o triplo, porém o tempo da SAT pode ser maior por causa do servidor usado para o teste);
- O tempo de seleção dos dados é quase o mesmo nos dois casos (ligeiramente maior usando OS), porque no fundo ambas as situações estão usando SELECT, sendo que a diferença é que o SELECT usado na classe agente é dinâmico;
- O tempo extra se deve às chamadas da classe base e das classes auxiliares de OS, que são feitas em 18 pontos distintos e executadas uma vez para cada registro (lembrando que o método chamado aqui é o IF_OS_CA_PERSISTENCY~GET_PERSISTENT_BY_QUERY).

Eu repeti os mesmos testes apenas omitindo o valor do parâmetro GJAHR, o que resultaria em ~2.000.000 resultados. Executando o programa com essa seleção e usando OS, a execução resultou em *dump* por falta de memória tanto rodando em *background* como *on-line*, enquanto a execução com OpenSQL respondeu normalmente nos dois modos.

Conclusão

OS é pra se jogar fora? Ainda não. A implementação pode ser meio ríspida, mas é uma técnica que funcionaria bem para controlar, por exemplo, um cadastro com um volume de pequeno a médio (~500.000 registros, talvez) baseado em tabelas Z.

Mesmo assim, eu confesso que fiquei desapontado com esses resultados. É ruim constatar que OS tem todos esses problemas, principalmente porque esse conceito seria essencial para alavancar o uso de OO dentro do ABAP. É mais fácil enxergar OO quando cada registro ou conjunto de registros é representado por uma classe que tem a sua própria lógica, em vez de apenas serem um conjunto de dados em algumas tabelas. E embora OpenSQL seja algo bom, e que tende a melhorar com a adoção de tecnologias como HANA, ainda se trata de código procedural.

E você, concorda ou discorda? Acha que minhas conclusões estão furadas ou ficou decepcionado(a) também? Esse assunto é interessante e pode render muito, então sinta-se à vontade para comentar.

[]s e até a próxima.

Comentários

Flávio Furlan — *02/11/2012 09:45*

No SITSP eu até comentei com o Maurício sobre usar o ECC ter algo parecido com o BOL do CRM. Será que o OS não seria um ensaio de BOL dentro do ECC?

Eu já tentei usar OS para relatórios, mas ele acabou não agregando muito. Eu acho que o OS foi pensando para os casos onde o programa precisa trabalhar com um objeto de negócio por vez, a exemplo das transações standards (VAs, VT, VLs etc.). Se for isso mesmo, o uso do OS fica muito limitado mesmo, pois é raro esse caso de uso comparado com relatórios, ou seja, geralmente a cada 13.0E23 relatórios, você precisa fazer um programa on-line.

Está certo a minha comparação entre OS e BOL?

Nuno Godinho — *28/09/2012 13:45*

Olá!

Embora tenha muita curiosidade para experimentar OS, continuava céptico em relação à performance. Há uns tempos li o livro da SAP Press chamado Object Services e gostei, mas fiquei cheio de dúvidas e pouco convencido com algumas coisas. O teu artigo conclui o que eu já desconfiava: tem muitas falhas! É esperar que a SAP corrija isto. A usar, só mesmo para a gestão de pouca quantidade de dados. E mesmo assim, a questão dos bloqueios não serem tratados pela API, porra!, é uma vergonha.

Outra coisa que falha: há 2 anos atrás considerei usar OS mas precisava que as classes OS fossem abstractas pois queria ter várias sub-classes ligeiramente diferentes. Desisti porque não dá para fazer classes OS abstractas. Pena. Se calhar faz sentido não dar, mas precisava mesmo. Se calhar ainda bem que não escolhi OS porque nesse caso a performance era muito importante!

Abraço,
Nuno

> **ABAP da Depressão** — *28/09/2012 20:58*
>
> Grande Nuno!
>
> Eu até tive curiosidade sobre esse livro de OS, mas agora tenho certeza que não vou aplicar essa técnica em mais nenhum desenvolvimento antes da SAP anunciar correções.
>
> A questão da abstração seria interessante, só que eu achei o help nesse assunto muito convoluto e, partindo dos resultados da implementação simples, não tenho vontade de tentar usar OS com herança ou polimorfismo, ou mesmo sem 😕

> > **Nuno Godinho** — *30/09/2012 20:40*
> >
> > Embora o livro defenda a utilização daquilo, e apresenta (oferecendo a implementação) uma solução para a questão da falta de bloqueios. Não vi em detalhe, mas fiquei com a sensação de é uma solução fita-cola.

> **Mauricio Cruz** — *27/09/2012 15:24*
>
> “O serviço de persistência não é realmente persistente, porque não existe bloqueio de objetos transientes”
>
> Zuado. Sem mais...

> **Fawcs** — *26/09/2012 11:13*
>
> acho que deveria a amostra de testes deveria ter N >1000 e calcular o desvio padrão e média.

> > **ABAP da Depressão** — *26/09/2012 18:56*
> >
> > Nossa equipe de consultores off-shore altamente especializados está analisando o seu requerimento e responderá em breve ao seu chamado.
> >
> > Atenciosamente,
> > Equipe ABAPZombie Consultoria SAP Ltda.

> > > **Mauricio Cruz** — *27/09/2012 14:12*
> > >
> > > Senhor Fawcs,
> > >
> > > Analisamos o seu requerimento e descobrimos que ele pertence a outra área. Estamos redirecionando o chamado para o site ABAP101.com .
> > >
> > > Atenciosamente,
> > > Mauricio Cruz
> > > Membro da Equipe ABAPZombie Consultoria SAP Ltda.

> **Lauffer** — *27/09/2012 17:56*
>
> Manda o .nugg pra ele fazer os cálculos todos e gerar gráficos no excel, pq user gosta é de ver tudo no excel mesmo =P