Os (meus) 3 maiores problemas do ABAP

10/05/2016 10:00

Eu estava fazendo as contas aqui e percebi que ano que vem eu completo DEZ ANOS trabalhando com SAP (nove como ABAP). Esse tempo não é muita coisa quando comparado com muita gente do mercado aí (incluindo a galera que a gente entrevistou recentemente no *podcast*), mas ainda assim eu diria que é uma experiência pelo menos relevante. Nesse tempo, eu consegui aprender bastante sobre a linguagem ABAP, suas características e suas limitações. Por isso, dessa vez eu quero fazer um *post* menos técnico e mais estilão jornalismo-de-listinha-*buzzfeed* pra desabafar sobre 3 aspectos do trabalho com ABAP que eu particularmente considero muito ruins. Eu restringi a três grandes problemas porque sempre dá pra ficar listando MILHARES de coisas como esse já clássico *post* sobre o PHP, e eu não quero fazer isso aqui porque eu não sou nenhum "guru do ABAP". Muito menos vim pra oferecer soluções. Eu só vim pra bater no cérbero. Então vamos lá:

---

1. O programador ABAP tem atribuições de DBA

É claro que a maior parte das atribuições de DBA no SAP cabe ao Basis, por causa da própria natureza e volume do trabalho. O Basis instala, configura, faz *backup*, restaura *backup*, gerencia usuários e autorizações e monitora e ajusta a performance do banco de dados ligado ao SAP. Inclusive, eu diria que é bem raro ou talvez quase impossível encontrar no mundo SAP a figura do *devops*, que é muito mais comum quando se trabalha com outras plataformas/linguagens, porque dentro do SAP isso representaria um conjunto de conhecimentos e responsabilidades muito grande para apenas um cargo.

PORÉM, algumas das atribuições mais importantes do DBA acabam ficando na mão do programador, que são justamente as tarefas de planejar e implementar tabelas, índices e demais objetos do banco. O desenvolvedor ABAP fica responsável por isso porque:

1. é necessária uma chave de desenvolvedor para realizar estas tarefas, o que o programador ABAP já tem e portanto seria dispendioso alocar outro recurso com outra chave somente para tal;
2. a integração clássica do banco de dados com o SAP faz com que muitas das ferramentas de desenvolvimento do banco de dados sejam abandonadas, porque:
    - as funções dessas ferramentas estão nas transações que manipulam o dicionário de dados (por exemplo, a SE11), ou
    - os dados são tratados diretamente nas aplicações, seja por causa da própria arquitetura e do OpenSQL ou por problemas de performance (isso mudou com o HANA, mas o problema persiste).

Isso geralmente configura um problema porque **nem todo programador é um bom DBA.** Quantas vezes vocês já se depararam com os seguintes problemas, sejam em objetos Z ou *standard*?

- Elemento de dados sem descrição nem texto nenhum, obrigando o ABAP a criar textos no programa.
- Campos de tabela com nome genérico sem descrição (por causa também do problema acima), que ninguém sabe para que servem.
- Campos de tabela com dados errados porque o domínio foi criado sem nenhuma restrição de entrada ou com o tipo errado, e a aplicação não valida as entradas.
- Tabelas desnormalizadas (dados de cabeçalho no item, campos faltando na chave, campos sobrando na chave, etc.).
- Registros em tabelas de item sem referência na tabela de cabeçalho, porque não há chave externa.

Quando existem erros na definição de tabelas, índices, relacionamentos, entidades e demais objetos do banco, **a(s) aplicação(ões) é que tem que contornar tudo.** E isso significa muito mais trabalho, que poderia ser evitado. Eu acredito que esses erros não surgem necessariamente porque as pessoas nunca tem um bom entendimento sobre banco de dados, mas com certeza porque elas são OBRIGADAS a realizar essas atividades. Esse é o problema.

---

2. APIs em ABAP que não existem ou funcionam pela metade

É fato que uma grande parte da funcionalidade existente em ABAP não está disponível através de APIs. Isso não é uma constatação minha, mas sim do próprio diretor executivo da SAP para produtos e inovação, Björn Goerke, em seu blog (sim, é aquele mesmo *post* mitológico que eu menciono no segundo episódio do nosso *podcast*).

O motivo disso ser um problema é óbvio, mas vamos fazer um experimento mental: você já imaginou ter que recriar um relatório com a cara de um ALV porém SEM USAR AS FUNÇÕES OU CLASSES DE ALV? Eu nem sei dizer quantos MESES eu estimaria pra fazer uma coisa desse tipo. Isso mostra a funcionalidade prática das APIs: **elas servem para que se escreva código mais conciso, de maneira muito mais rápida.**

É claro que existem APIs para muitas coisas em ABAP, e eu nem saberia listar todas. ALV, XML, BAPI, CL_GUI_FRONTEND_SERVICES, são só algumas que eu consigo citar de cabeça. O problema que eu quero desenhar aqui é outro: faltam APIs para certas tarefas básicas em ABAP, e muitas das APIs que existem não funcionam como é esperado.

"Mas como assim, tarefas básicas?" Vou tentar dar um exemplo. Algum tempo atrás uns amigos meus comentaram sobre uma questão de um teste de programação, onde havia um método que, dadas duas palavras, respondia se são anagramas entre si. Então, cada um começou a reimplementar esse método em sua linguagem preferida ou do dia-a-dia. Um deles implementou em Ruby:

```
def anagram? (word1, word2)
  word1.chars.sort.join == word2.chars.sort.join
  end
```

Uma linha de método. Depois, veio o .NET:

```
static bool isAnagram (string a, string b)
{
    return a.ToCharArray().OrderBy(c => c).Sum(c => c) == b.ToCharArray().OrderBy(c => c).Sum(c =>
}
```

Só uma linha também. Java?

```
public static Boolean isAnagram(String s1, String s2){
    return Arrays.equals(s1.toLowerCase().chars().sorted().toArray(), s2.toLowerCase().chars().sor
}
```

Que surpresa, UMA LINHA. Além de tudo, todos aqui tem mais ou menos a mesma idéia: pegue a *string*, separe por caracteres, ordene e jogue tudo num *array*, depois retorne o resultado da comparação dos *arrays*.

Vamos ver o ABAP. Primeiro, versão <= 7.31:

```
CLASS-METHODS: is_anagram
                  IMPORTING
                    s1 TYPE string
                    s2 TYPE string
                  RETURNING VALUE(result) TYPE abap_bool.
```

```
***

  METHOD is_anagram.

    DATA: _s1    TYPE string,
          _s2    TYPE string,
          offset TYPE i.

    result = abap_true.

    IF strlen( s1 ) <> strlen( s2 ).
      result = abap_false.
    ELSE.
      _s1 = to_lower( s1 ).
      _s2 = to_lower( s2 ).
      DO strlen( s1 ) TIMES.
        offset = sy-index - 1.
        IF count( val = _s1 sub = _s1+offset(1) ) <> count( val = _s2 sub = _s1+offset(1) ).
          result = abap_false.
          EXIT.
        ENDIF.
      ENDDO.
    ENDIF.

  ENDMETHOD.
```

Vamos tentar ~~ofuscar~~ simplificar um pouco as coisas com a 7.4? Vamos:

```
CLASS-METHODS: is_anagram_740
                  IMPORTING
                    s1 TYPE string
                    s2 TYPE string
                  RETURNING VALUE(result) TYPE abap_bool.

***

  METHOD is_anagram_740.

    result = boolc(
      strlen( s1 ) = strlen( s2 ) AND
      REDUCE i(
        LET _s1 = to_lower( s1 )
            _s2 = to_lower( s2 ) IN
        INIT diff = 0
        FOR off = 0 UNTIL off > strlen( s1 ) - 1
        NEXT diff = diff + COND i( WHEN count( val = _s1 sub = _s1+off(1) ) = count( val = _s2 sub
      ) = 0
    ).

  ENDMETHOD.
```

Percebam bem que nessas implementações eu:

- não usei tabelas, somente *strings* ou inteiros ou "booleanos ABAP";
- usei as funções de manipulação de *string* mais adequadas que eu pude;
- sem tabelas nem ordenação, acabei usando outra lógica: comparar comprimentos e as contagens das letras entre as duas palavras;
- me esforcei bastante pra fazer tudo em um único comando na implementação 7.4.

É claro que a ENORME diferença entre as minhas implementações em ABAP e as implementações das outras linguagens se devem a um grande número de fatores estranhos do ABAP, entre eles a inexistência de *arrays* clássicos, a verbosidade (e olha que o Java é bem verboso, hem), a implementação esquisita de OO, etc. Porém, notem que

apesar dessa questão vir de um teste de programação, ela representa uma série de problemas cotidianos que todos nós enfrentamos corriqueiramente ao manipular *strings* em ABAP.

E ainda que eu tenha usado 3 funções diferentes de manipulação de *string* (que eu tenho certeza que muita gente nem sabe que EXISTEM), não é uma solução nada concisa. E dá pra pensar em diversos outros casos de problemas triviais assim que também não tem soluções simples. O que dizer, por exemplo, da manipulação de datas e horários? Do *batch input*? E de outras coisas não tão triviais, como o paralelismo?

O outro viés desse problema é quando as APIs existem, mas não funcionam completamente. Eu já falei aqui no *blog* sobre *query services* e seus diversos problemas. Não é difícil encontrar casos de BAPIs no ECC que, segundo a documentação, não estejam liberadas completamente para uso. Ou mesmo produtos COMPLETAMENTE sem APIs. Eu trabalhei recentemente customizando uma aplicação chamada SUS (*Supplier Self-Services*) para um cliente. Apesar dessa solução ter seus próprios dados e documentos, diferentes dos que estão no ECC, não existe NENHUM tipo de BAPI para interagir com esses registros. Você fica relegado a tentar decifrar as funções *standard* da própria aplicação para poder ampliá-la, gastando um tempo enorme e às vezes chegando à conclusão que não dá para utilizar nenhuma delas e é melhor fazer tudo Z.

---

3. Não existe documentação para a maioria dos objetos construídos em ABAP

Quando eu ainda era estagiário de Basis, eu desenvolvi bastante em C# por causa da faculdade. Aprendi tudo por conta própria na *internet*, com várias videoaulas e materiais diversos de apoio. Se eu não tivesse seguido o caminho do ABAP, hoje em dia eu provavelmente estaria trabalhando com .NET. </historinha>

E durante o desenvolvimento, é comum que você se depare com alguns problemas que você talvez não saiba imediatamente como resolver. Digamos, por exemplo, que surja a seguinte dúvida:

Dada uma data, como eu descubro o dia da semana relacionado?

Um problema simples, assim como o dos anagramas já mencionado. Pode ser até pergunta de teste. Se você já programou pelo menos uma semana na sua vida é quase certo que você já tenha se deparado com ele. Portanto, é óbvio que você pode pesquisar no Google, e você encontrará centenas de milhares de resultados em *blogs*, listas de discussão, sites de pergunta/resposta, etc., afinal é um problema trivial que já foi resolvido muitas vezes.

Porém, ao buscar no Google por "C# weekday", o primeiro resultado vai ser de um site sensacional que é a biblioteca do MSDN. Ali você encontra um repositório de documentações para QUAISQUER TIPOS de *framework* que a Microsoft já tenha desenvolvido. Inclusive dá pra fazer a mesma pesquisa diretamente dentro da biblioteca, e logo você encontrará o método DateAndTime.Weekday (um nome claro e objetivo), junto com uma descrição COMPLETA contendo as dependências, os parâmetros, os tipos de *exception* que o método retorna, e exemplos de uso. E se por algum acaso você estiver sem internet, o Visual Studio tem uma CÓPIA LOCAL dessa biblioteca para a versão do *framework* .NET que você estiver usando. Problema trivial, solução simples e bem definida, documentação e exemplos abundantes.

E no ABAP?

Continua óbvio que você pode pesquisar no Google. Mas em vez de centenas de milhares de resultados sobre a mesma solução, você vai encontrar centenas de milhares de resultados com soluções diferentes: uma infinidade de *posts* no SCN com A MESMA PERGUNTA, *blogs* e fóruns de qualidade diversa (em todo tipo de idioma), aqueles *sites* de catálogo de documentação SAP como o SE80, aqueles sites de tutoriais como o SAPTechnical... você acaba pensando: "putz, qualquer um serve, vai".

E quando você decide escolher o "qualquer um", você acaba com funções como DATE_COMPUTE_DAY (mas como assim, computar o quê?), DATE_TO_DAY (cadê o *week*?), FTR_DAY_GET_TEXT (FTR? "texto"?),

RH_GET_DATE_DAYNAME (RH? mas eu não quero nome!), etc. Além disso, dessas aí a ÚNICA que tem documentação na SE37 é a DATE_COMPUTE_DAY. (Alías, segue aqui o problema anterior: essa função tem 8 linhas de código. Às vezes é melhor copiar e implementar localmente.)

Observe bem aqui, caro zumbi, que eu não estou criticando a documentação da LINGUAGEM – essa é abundante e bem útil. Por exemplo, quando eu precisei implementar expressões regulares pra fazer algumas validações, eu descobri que o tópico sobre esse assunto na documentação ABAP é o suficiente para que você aprenda tudo que precisa. E além disso, onde existem APIs normalmente há documentação (vejam como esse problema está muito ligado ao anterior). Porém, pise fora do cercadinho e você estará perdido.

É óbvio que não adianta querer aqui dar uma de paladino da moral e dos bons costumes e dizer em letras douradas e garrafais "Ó NOBRES COLEGAS, EU VOS SUPLICO, NÃO DEFINAMOS MAL NOSSOS DICIONÁRIOS DE DADOS, ORGANIZEMOS NOSSOS DESENVOLVIMENTOS COM APIs, PENSEMOS NÓS EM ESCALABILIDADE, CRIEMOS NÓS DOCUMENTAÇÃO PRA TUDO", etc., porque eu sei que nem sempre é possível controlar tudo isso, e também são problemas que estão no *standard*. De novo: só vim aqui pra reclamar.

E aí? Muita groselha? Ou tá certo? Comentem aí o que vocês acham de ruim (ou bom?) no nosso querido COBOLscript.

Abraços e até a próxima.

Comentários

Luis — *06/05/2017 22:29*

Boa Noite! Leo... estou começando a estudar ABAP, e estou tentando montar um relatório..
a minha dificuldade e descobrir a estrutura das tabelas usadas pelo sistema original em CO no material ledger..por exemplo..a tabela que guarda a sequencia de processamento dos materiais no custo?

Se você poder me dar umas dicas neste sentido eu te agradeço..

Paulo Silva — *17/03/2017 15:19*

Li o seu texto agora a pouco, coisa que gosto mas que eu não fazia já faz um tempo.

Sobre "anagramas entre si em ABAP".
O comando abaixo não resolveria o problema?
————————————————
IF _s1 CO _s2 AND NOT _s1 CN _s2.
result = 'X'.
ENDIF.
————————————————

s

Leo Schmidt — *17/03/2017 15:34*

Olá Paulo, obrigado por acompanhar o blog.

> Pela própria documentação vemos que NOT CN é a mesma coisa que CO, e CO não é suficiente pra fazer esse teste. Portanto, continuamos com o problema.
>
> (Admito que tive que escrever isso pra testar 🙂)
>
> Abraços!

> **Paulo Silva** — *17/03/2017 15:56*
>
> kkkkkk
> Fui presunçoso! Da próxima vez testo o código antes de enviar.
> Pelo que você disse o código melhoraria para:
> ———————————————————
> IF _s1 CO _s2 AND _s2 CO _s1.
> result = ‘X’.
> ENDIF.
> ———————————————————
>
> O problema seria “somente” quando se repetissem caracteres, ex: “ABCCC” para “CCBBA”.
> Vou resolver (kkkk) assim que eu tiver um sistema na minha frente, então retorno.
> Abraços.
> Value.

Evandro — *13/02/2017 14:23*

Bom Leo tudo bem?

Estou pesquisando a um bom tempo e até agora não encontrei nada.

Você sabe se da para colocar a Descrição ao invés do código de Um search help ou de um Dominio, em uma ALV?

Tem algum tutorial ou dica, explicando isso?

Obrigado.

Custodio de Oliveira — *20/06/2016 03:30*

Belo post, Leo, mas me permita dar uma troladinha:

```
CLASS lcl_string DEFINITION.
PUBLIC SECTION.
CLASS-METHODS: is_anagram
IMPORTING s1 TYPE string
s2 TYPE string
RETURNING VALUE(result) TYPE abap_bool.
ENDCLASS.

CLASS lcl_string IMPLEMENTATION.
METHOD is_anagram.
result = xsdbool( s1 = reverse( s2 ) ).
ENDMETHOD.
ENDCLASS.
```

Obrigado. De nada.

> **Shai** — *19/08/2016 21:26*
>
> I'm afraid that your code doesn't check for anagram, but only if one string is the reverse of another string (Which is only one option of anagram).

> **Custodio de Oliveira** — *21/08/2016 22:01*
>
> Hi Shai,
>
> You are absolutely right, and I'm a bit embarrassed actually 😕
>
> I really don't how I could read it so wrong. I'll blame lack of clarity in the functional spec 😀
>
> Anyway, I read the problem again, and Leo's solution too, and I think the problem is (slightly) better/easier solved with the help of internal tables (the ruby/.net/java implementations convert the words to arrays, right?). Maybe Leo could elaborate why he chose not to use internal tables?
>
> CLASS lcl_string DEFINITION.
> PUBLIC SECTION.
> CLASS-METHODS: is_anagram IMPORTING s1 TYPE string
> s2 TYPE string
> RETURNING VALUE(result) TYPE abap_bool.
> ENDCLASS.
>
> CLASS lcl_string IMPLEMENTATION.
> METHOD is_anagram.
>
> DATA first_char_table TYPE SORTED TABLE OF c WITH NON-UNIQUE KEY table_line.
> DATA second_char_table TYPE SORTED TABLE OF c WITH NON-UNIQUE KEY table_line.
> DATA offset TYPE i.
>
> DO strlen(s1) TIMES.
> offset = sy-index – 1.
> INSERT CONV #(to_lower(s1+offset(1))) INTO TABLE first_char_table.
> ENDDO.
>
> DO strlen(s2) TIMES.
> offset = sy-index – 1.
> INSERT CONV #(to_lower(s2+offset(1))) INTO TABLE second_char_table.
> ENDDO.
>
> result = xsdbool(first_char_table = second_char_table).
>
> ENDMETHOD.
> ENDCLASS.
>
> I'll try to come up with a better/shorter/ version for 7.40+ later.
>
> Cheers

> **Leo Schmidt** — *22/08/2016 10:24*

> Wow, there's a lot that went unsaid in this post/thread.
>
> I figured Custódio said he was trolling in the original/classical sense of it, because he provided a partial solution as if it was general, thus baiting other unwary readers to reply saying he's wrong. Maybe I misinterpreted it? 🙂
>
> To clarify, my intents with this anagram example were:
>
> 1) to try and stay as close as possible to the other languages' implementations, so I didn't want to declare any local variables (which I didn't manage to do; even with the use of REDUCE I had to declare iterators) and use as many built-in string functions as possible;
> 2) to showcase the lack of (and also how cumbersome are some) APIs in ABAP.
>
> Anyhow, my solutions are in fact a little obfuscating. I welcome all to contribute with their code to this challenge.
>
> Cheers!

Leo Schmidt — *20/06/2016 09:28*

Olha Custódio, eu até ficaria bravo, mas como vc usou o termo "trolar" no contexto **mais correto possível**, eu vou é aplaudir seu comentário.

Raju Shrestha — *07/06/2016 10:42*

Very well written pain points in ABAP. Enjoyed the post.

Regards,
Raju.

> **Leo Schmidt** — *07/06/2016 10:49*
>
> Thanks Raju! I'm glad for people reaching out, even if the content is in portuguese.

Renato Piovesano Bartolamei — *11/05/2016 17:11*

Cada linguagem tem suas peculiaridades e isso faz parte do trabalho.
O que mais me incomoda com o ABAP não a parte técnica, mas o mercado.
Há um "socialismo" no mercado onde quase a totalidade dos consultores são seniores e a taxa deve ser "igual" para todos, sendo valorizados apenas aqueles com habilidade de negociação. Já passei por muitos projetos em 15 anos de ABAP e sempre vi aqueles colegas que não teriam condição de serem assistentes de vice candidato a

estagiário de aspirante a ABAP, mas ganham como senior. E também vi muitos ABAPs TOP de linha que fazem realmente a diferença ganhar a mesma coisa que os outros “porque é isso que o mercado paga!”. O mercado trata o ABAP como comodite e com isso todos são considerados como tendo a mesma capacidade técnica. Esta falta de meritocracia (não sei como é no primeiro mundo) aqui no Brasil só desestimula completamente o profissional. Mas as consultorias sabem cobrar por um consultor diferenciado, não refletindo isso no pagamento.

Sonikro — *10/05/2016 23:12*

Concordo com todos os pontos.
Uma das coisas que me incomoda mais, além da syntax bizarra de OO, é a falta de flexibilidade em relação a Threads e Objetos. Nada como implementar um Runnable na sua classe, e sair disparando e controlando as threads.
No ABAP (até onde meu limitado conhecimento vai), ficamos presos a funções starting new task, Nada elegantes.

> **Leo Schmidt** — *11/05/2016 13:58*
>
> As *threads* do SAP são os *work processes*, mas nós como reles ABAPs não podemos controlá-las. O STARTING NEW TASK na verdade cria uma chamada RFC assíncrona pra permitir o paralelismo.
>
> Mas ainda assim, como vc disse, nada elegante.