ABAPZombie Guide to ABAP – Parte 28 – COLLECT

07/12/2011 14:16

E cá estou eu, continuando essa saga que não tem um post novo desde Agosto. Eu não desisti, e não desistirei **jamais**! 😈

Enfim, hoje vou explicar um comando que deixa ABAPers Jrs maravilhados pelo mundo todo, o **COLLECT!**

Eu resumo o funcionamento desse comando da seguinte forma: *se é valor, soma essa parada.* Fim do post de hoje galera!

🙄

Mas a idéia é essa mesma: se você utilizar um COLLECT na hora de adicionar valores de uma work area para a sua tabela interna, o sistema vai verificar se existe alguma outra entrada com a mesma chave primária na tabela. **Se tiver,** ele vai somar todos os campos com valores, ao invés de duplicar a linha. **Se não tiver,** ele cria uma nova entrada.

Vamos ver isso funcionando na prática:

```
REPORT  zombie_collect.

TYPES: BEGIN OF ty_val,
         nome TYPE char10,
         sal  TYPE i,
       END OF ty_val.

DATA: t_val  TYPE SORTED TABLE OF ty_val WITH UNIQUE KEY nome,
      wa_val LIKE LINE OF t_val.

wa_val-nome = 'Zombie1'.
wa_val-sal  = 123.
COLLECT wa_val INTO t_val.

wa_val-nome = 'Zombie2'.
wa_val-sal  = 321.
COLLECT wa_val INTO t_val.

wa_val-nome = 'Zombie1'.
wa_val-sal  = 321.
COLLECT wa_val INTO t_val.
```

Esse é o funcionamento básico. Mas como toda boa operação com tabelas internas no ABAP, o COLLECT funciona de maneira diferente para cada tipo de tabela. No caso acima, se tentarmos adicionar um campo não numérico **fora** da chave primária, o código não compila por conta do COLLECT. Vejam:

```
REPORT  zombie_collect.

TYPES: BEGIN OF ty_val,
         nome  TYPE char10,
         cargo TYPE char10,
         sal   TYPE i,
       END OF ty_val.

DATA: t_val  TYPE SORTED TABLE OF ty_val WITH UNIQUE KEY nome,
      wa_val LIKE LINE OF t_val.

wa_val-nome  = 'Zombie1'.
```

```
wa_val-cargo = 'Gerente'.
wa_val-sal   = 123.
COLLECT wa_val INTO t_val. "Não Compila!!!
```

A busca pela linha feita pelo COLLECT para tabelas SORTED é sempre feita com um algoritmo de busca binária. Ah, e o COLLECT também não destrói a ordenação da tabela, caso o registro não exista e ele tenha que criar uma nova entrada. Divertido, não? 🙂

O funcionamento do COLLECT para SORTED e HASHED tables é praticamente o mesmo, a única diferença é que o sistema utiliza o algoritmo hash para fazer a busca na tabela interna, ao invés de fazer a busca binária.

Maaaas, e se não for tabela SORTED nem HASHED?

```
REPORT  zombie_collect.

TYPES: BEGIN OF ty_val,
         nome  TYPE char10,
         cargo TYPE char10,
         sal   TYPE i,
       END OF ty_val.

* Lembrando que quando existe só o "TABLE", o sistema
* assume que a tabela é uma STANDARD TABLE automaticamente
DATA: t_val  TYPE TABLE OF ty_val,
      wa_val LIKE LINE OF t_val.

wa_val-nome  = 'Zombie1'.
wa_val-cargo = 'Gerente'.
wa_val-sal   = 123.
COLLECT wa_val INTO t_val.

wa_val-nome = 'Zombie2'.
wa_val-cargo = 'Diretor'.
wa_val-sal  = 321.
COLLECT wa_val INTO t_val.

wa_val-nome = 'Zombie1'.
wa_val-cargo = 'Func'.
wa_val-sal  = 321.
COLLECT wa_val INTO t_val.
```

Notem que esse é o mesmo exemplo do COLLECT com tabela SORTED que não compila. Mas no caso acima, o sistema vai criar 3 linhas! Sim, porque para tabelas STANDARDs sem chave declarada, o COLLECT vai comparar TODOS os campos não numéricos para saber se pode ou não somar. No exemplo acima, troque todos os cargos para "Diretor" e teste novamente 🙂

Se a entrada não existir, o COLLECT vai ter o mesmo efeito do APPEND em STANDARD TABLES.

Uma curiosidade legal sobre o COLLECT em STANDARD TABLES: o sistema monta um "cache" da tabela em memória organizada por um algoritmo hash, sempre que ela é preenchida com o COLLECT. Isso faz o acesso posterior pelo COLLECT ficar bem rápido. Porém, se qualquer outro comando como INSERT ou APPEND alterarem a tabela, esse "cache" vai para o espaço, e o sistema faz a busca de forma linear caso algum COLLECT seja novamente utilizado.

Meio enrolado né? 😛 Eu vou fazer um post explicando somente Hash Tables em breve!

Divirta-se utilizando o COLLECT, e sofra quando você descobrir uma diferença no valor em um campo de uma tabela com 34 campos de valores ocasionada aparentemente pelo COLLECT. Lembre-se que a culpa nunca é do comando 😀

Abraços!

Comentários

Gustavo Büchle — *23/11/2016 19:25*

Valeu pessoal!!!!

Eduardo Pereira — *01/03/2016 15:31*

Excelente post!!!
Não sabia dessas restrições do Collect.
Ajudou bastante,
parabéns!

Leonardo — *09/12/2011 12:37*

Sempre muito bom os posts! Parabéns. Este comando auxilia muito para não fazer as somas na mão. Abs