

microsoft.public.sqlserver.mseq: Re: => Trigger to split Trailer Loads

Re: => Trigger to split Trailer Loads

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.mseq/2004-04/0037.html>

From: Hugo Kornelis (*hugo_at_pe_NO_rFact.in_SPAM_fo*)

Date: 04/08/04

Date: Thu, 08 Apr 2004 22:28:05 +0200

On Thu, 8 Apr 2004 08:41:03 -0700, Rhonda Fischer wrote:

>Hello,

>

>I am trying to design a trigger that upon record INSERT checks the number of pallets ordered
>and ensures that no record has more than 26 pallets, as this is the maximum that
>will fit onto a trailer. Some how dividing 26 into the number of pallets copying all lines with less
>than 26 pallets into a new table and copying records with over 26 pallets in quantities of 26 and
>then the final last record of less than 26.

>

>Before Trigger:

>tblOrder

>

> DeliveryID CollectionID TempID OrderDate NumOfLoads

> 2 3 3 30/03/2004 45

> 2 4 3 30/03/2004 27

> 2 5 3 30/03/2004 13

>

>

>After Trigger: copied split entries of max 26 pallets to a new table

>tblSplitOrder

>

> Id DeliveryID CollectionID TempID OrderDate NumOfLoads

> 1 2 3 3 30/03/2004 26

> 2 2 3 3 30/03/2004 19

> 3 2 4 3 30/03/2004 26

> 4 2 4 3 30/03/2004 1

> 5 2 5 3 30/03/2004 13

>

>I tried the follow trigger but have not got very far – struggling. Not sure how to code trigger to achieve end.

>I'm not sure I explained my problem effectively in my last post. Hope you can help.

>

>Thank you very much for any ideas you might have.

>

>Best Regards

>Rhonda

Re: => Trigger to split Trailer Loads

Rhonda,

It's not very clear how you try to achieve your goal. You mention a trigger that will split the order upon insert, yet you code the trigger "for insert, update, delete". And both the body of the trigger and the examples above give me the impression that you want to recalculate all split orders for all orders in your table upon each change (either insert, update, or delete) to one or more orders.

I would not recommend recalculating all SplitOrders upon each and every change to any Order. That leaves you with two options:

1. Use a trigger on Orders for insert that will insert SplitOrders for only the new Orders. But if you do this, you'd also need either a trigger or a referential integrity action to make sure SplitOrders are deleted again when an Order is deleted. And the hardest part of this approach would be the trigger on Orders for update – 'cause a change in the NumOfLoads of an Order may result in either inserting SplitOrders or deleting SplitOrders, plus possibly changing one of the existing SplitOrders.
2. Get rid of the SplitOrders table but define a SplitOrders view instead. This will always be up to date. Changes to the Orders table will be faster, as no trigger needs to be executed. The price for this will be that getting data from SplitOrders will take longer, as the view has to be rebuilt every time. Or you could choose to use a materialized view; in that case, the view's result set will be permanently stored, queries on the view will be fast again and changes to Orders will slow down a bit (as SQL Server will have to recompute the view each time an Order changes).

I think I can help you with both approaches. Please select which option looks most promising to you. Also, please post DDL that I can copy and paste into Query Analyzer to recreate the relevant part of your table structure (including primary keys, indexes, foreign keys and relevant other constraints) to test my solutions.

Best, Hugo

--

(Remove `_NO_` and `_SPAM_` to get my e-mail address)