

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/2004-04/0081.html>

---

**From:** Hugo Kornelis (*hugo\_at\_pe\_NO\_rFact.in\_SPAM\_fo*)

**Date:** 04/21/04

Date: Thu, 22 Apr 2004 01:12:45 +0200

Hi Rhonda,

Allow me to start with a bit of criticism. Not with the intention to bash you, but as an attempt to learn you something.

Earlier in this discussion, you mentioned that you attempt to tackle this complex task in small bites (forgive me for not going back to your previous posts to find the exact words used). I think that is not the best way to tackle such situations. Imagine ordering a pizza and trying to imitate it in this way: you take one bite and taste dough, tomatoes and cheese, baked, so you heat the oven, knead the dough, slap on some tomatoes and cheese and put it in the oven. Then, you take the second bite and taste mushrooms, so you take your pizza-to-be out of the oven, add mushrooms and put it back in. Not yet knowing that on your next bites, you'll taste pineapple, ham, olives and salmon as well....

If you start with a partial solution and keep adapting your design to cater for more exceptions, you will either find yourself stuck in a design that doesn't fit the total problem, yet has influenced your thinking, became part of your mind-set, making it hard to back out of that design. Or you make a design that's laden with special cases and exceptions. Or even both. In all cases, the end result will be much more complicated than necessary.

I think the best way to model a complex situation is to completely forget about modeling first. First, you have to come to grips with the subject you're dealing with. Make sure that you have data (real examples from the real world of your company work best) that is representative for your companies' business. Try to find the underlying structure, the way that some data interacts with other data. In short, and without the incrowd words: make sure you get a thorough understanding of the process you're trying to design and database for and of the information that will be stored in that DB.

That's exactly what I've been trying to do in the previous messages. I used the DDL you posted as a starting point. But you may have noted

Re: => Trigger to split Trailer Loads

that many questions I asked were of the type: "why did you design it this way – what actual process or actual piece of information in the world of Turners prompted you to make this choice?" From your answers, I think I now have a reasonable understanding of what happens in the planning department where your system will be used. Enough to propose a database design that will probably work much better, as it closely reflects the bit of reality it describes.

You should, of course, check my suggestions carefully against the real situation at Turners. My information stems only from your posts in this newsgroups. I may have misunderstood parts of your messages and my knowledge of Turners is almost certainly still very incomplete.

First: the basics.

Turners takes orders to transport goods; Each order is to transport a specific number of loads from a specific place to a specific place.

The goods to be transported come from various suppliers. The goods have to be picked up at a collection point. Suppliers may use more than one collection point and a collection point may be used by several suppliers.

The goods have to be delivered to various supermarkets. The goods are delivered at a depot. Supermarkets may have more than one depot and a depot may be used by several supermarkets.

If an order is for more than 26 loads ("large order"), it can't be transported in one trailer. The order will be split over two or more partial orders. All but one of the partial orders will be for 26 loads. The final partial order will be for the remaining loads. From now on, I'll use the term refer to each partial order as well as each "small order" (26 loads or less) as a "transport order".

Transport orders may be consolidated (combined). Restrictions to consolidation are not in the scope of the project, though some information (like temperature etc.) has to be stored in order for the responsible people to check the restrictions. The total number of loads in the consolidated order may not exceed 26. (This rules out a consolidation of a transport order for 26 loads with any other transport order and it imposes a theoretic maximum of 26 transport orders for any consolidation).

Ideally, all transport orders that are consolidated go from the same collection point to the same depot. Other possibilities are:

- 1) Two transport orders with a collection point near each other have to go to one depot. These transport orders may first be transported (individually) to one of Turners' own locations (a "hub") and are then transported (consolidated) from the hub to the depot.
- 2) Two transport orders from the same collection point have to go to two depots near each other. These transport orders may first be

transported (consolidated) to a hub and are then transported (individually) to the respective depots.

3) Two transport orders with a collection point near each other have to go to two depots near each other. These transport orders may first be transported (individually) to a hub, next (consolidated) to another hub and finally (individually) to the respective depots.

NOTE 1: Possibility #1 is described clearly in your posts. I could not find any specific details on possibilities #2 and #3, but it seems logical to me that all these possibilities are considered by the planning department.

NOTE 2: The description above mentions two transport orders. Actual consolidations may consist of more than two transport orders. This can even be so complex as: "transport orders 1, 2 and 3 will be taken from collection #1 to hub#1, where they will be consolidated with transport orders 4 and 5 (that are from collection #2); these 5 transport orders are then taken to hub#2, where they are split: transport order 3 and 5 are for depot #1 and transport orders 1, 2 and 4 for depot #2"

A transport is the actual process of moving one individual transport order or one consolidation of two or more transport orders from one place to another place. To carry out a transport, we need a trailer (to put the goods in), a vehicle (to carry the trailer) and a driver (to drive the vehicle). Each trailer, vehicle and driver can be used in many transports, though not necessarily in the same combination.

Some trailers have two loading zones (front and rear), that can store goods at different temperatures. For transports using these trailers, it is important to know which transport orders to load in the front and which in the rear. The maximum capacity of these twin trailers is still 26 loads. The capacity of each compartment can be adapted to fit the requirement (but there are some restrictions? this is still not clear!)

If I'm not mistaken, the above is roughly the way the planning department of Turners assigns orders to trailers.

Based on the above description, my suggestion for your database design would be: (Note – I changed a lot of the table names from your design to names that made more sense to me. Feel free to change them back for what works for you, but I find I make mistakes if I use table names that don't fit my grasp of what the data in it means).

```
CREATE TABLE Orders -- original: tblAddOrder
  (OrderID int NOT NULL IDENTITY,
   FromLocation varchar(50) NOT NULL,
   FromSupplier varchar(50) NOT NULL,
   ToLocation varchar(50) NOT NULL,
   ToSupermarket varchar(50) NOT NULL,
   NumOfLoads int NOT NULL CHECK(NumOfLoads > 0),
   other columns(min/max temperature;
```

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

```
    first/last date of delivery/pickup;  
    etc),  
    PRIMARY KEY (OrderID)  
    FOREIGN KEY (FromLocation, FromSupplier)  
    REFERENCES CollectionPoints (Location, Supplier),  
    FOREIGN KEY (ToLocation, ToSupermarket)  
    REFERENCES DeliveryPoints (Location, Supermarket)  
)
```

Note: if you can use the same order ID as the sales dept or the billing dept, that would be a lot better than an artificial key.

```
CREATE TABLE CollectionPoints -- original: tblCollection  
    (Location varchar(50) NOT NULL,  
    Supplier varchar(50) NOT NULL,  
    PRIMARY KEY (Location, Supplier),  
    FOREIGN KEY (Location)  
    REFERENCES Locations (Location)  
)
```

There is no foreign key for suppliers. There is no suppliers table either. Since you store nothing but the suppliers' name, you might as well store that name just in this table.

Note: If you're concerned about misspelled supplier names, you can still choose to add a suppliers table, with just one column (Suppliername); then add a foreign key to this table. I'd recommend against using an artificial ID – it just adds extra joins to your queries.

```
CREATE TABLE DeliveryPoints -- original: tblDelivery  
    (Location varchar(50) NOT NULL,  
    Supermarket varchar(50) NOT NULL,  
    PRIMARY KEY (Location, Supermarket),  
    FOREIGN KEY (Location)  
    REFERENCES Locations (Location)  
)
```

No supermarket table either – see above.

In your original design, you had a bookingtime and a journeytime in this table as well. I think they should be in the Transports table, as there is not just one booking time or journey time for a delivery point.

```
CREATE TABLE Locations -- original three tables: tblDepot,  
    -- tblLocation and tblCompanyDepot.  
    (Location varchar(50) NOT NULL,  
    LocationType char(10) NOT NULL CHECK(LocationType IN  
        ('Delivery', 'Collection', 'Hub'))  
    Address1 varchar(30) NULL,  
    Address2 varchar(30) NULL,  
    Address3 varchar(30) NULL,
```

Re: => Trigger to split Trailer Loads

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

```
Address4 varchar(30) NULL,  
Phone varchar(15) NULL,  
Fax varchar(15) NULL,  
PRIMARY KEY (Location)  
)
```

I combined depots, locations and companydepots (hubs) into one table. Otherwise, separate tables would have been needed for transport from collection to delivery, collection to hub, hub to hub and hub to delivery.

Note: I did not include anything to prevent a location that's defined as a delivery or hub being used in the CollectionPoints table. I do suggest you add such a check.

```
CREATE TABLE TransportOrders -- original: tblOrder  
(TranspOrderID int NOT NULL IDENTITY,  
OrderID int NOT NULL,  
NumOfLoads int NOT NULL CHECK(NumOfLoads BETWEEN 1  
AND 26),  
PRIMARY KEY (TranspOrderID),  
FOREIGN KEY (OrderID)  
REFERENCES Orders (OrderID)  
)
```

I'm usually quite opposed to artificial ID's, but I'll make an exception this time. The "natural" key would be orderID + serial number (starting from 1 within each order), but that is also somewhat artificial and it involves more complicated SQL without giving much benefit.

Note: no collection or delivery in this table, since the transport of a transport order might involve one, two or even three stages.

Note: you can use the triggers I provided earlier in this discussion to split each order into one or more transport orders.

```
CREATE TABLE Transports -- original: tblConsolidate (loosely)  
(TransportID int NOT NULL IDENTITY,  
FromLocation varchar(50) NOT NULL,  
ToLocation varchar(50) NOT NULL,  
DriverID int NOT NULL,  
VehicleID int NOT NULL,  
TrailerID int NOT NULL,  
other columns(date of delivery/pickup;  
temperature setting for front/back;  
etc),  
PRIMARY KEY (TransportID),  
FOREIGN KEY (FromLocation)  
REFERENCES Locations (Location),  
FOREIGN KEY (ToLocation)
```

Re: => Trigger to split Trailer Loads

```
REFERENCES Locations (Location),
FOREIGN KEY (DriverID)
REFERENCES Drivers (DriverID),
FOREIGN KEY (VehicleID)
REFERENCES Vehicles (VehicleID),
FOREIGN KEY (TrailerID)
REFERENCES Trailers (TrailerID)
)
```

This table can be done without the identity column. Many combinations of two columns are candidate key: either delivery date/time, start date/time or early pickup date/time, combined with either driver, vehicle or trailer will yield a unique combination. In fact, you will probably find yourself defining indexes for these columns, as I expect them to be heavy searched ("where is Williams tomorrow?" "when will the 2002 Ford Artic be in Exning again?" etc.) I decided to stick with the identity to keep the Cargo table a bit more compact.

I suggest you stick two temperature settings (front/rear) in this table. Proper normalization calls for a separate table with TransportID, FrontOrRear and TempSetting, but in this case you're better off with the denormalized version.

```
CREATE TABLE Cargo -- Original tblTrlRearLoads,
-- tblTrlFrontLoad and a bit of
-- tblConsolidate (again)
(TransportID int NOT NULL,
TranspOrderID int NOT NULL,
FrontOrRear char(5) NULL CHECK(FrontOrRear IN
('Front', 'Rear'))
PRIMARY KEY (TransportID, TranspOrderID),
FOREIGN KEY (TransportID)
REFERENCES Transports (TransportID),
FOREIGN KEY (TranspOrderID)
REFERENCES TransportOrders (TranspOrderID)
)
```

The FrontOrRear column is nullable. It should be NULL if the transport uses a single trailer and it should be either Front or Rear if a twin trailer is used.

I've not looked into the Vehicles, Trailers, and Drivers tables, as they are related only very loosely to the consolidation problem. I do suggest that you try to get rid of the identify columns. For drivers, you could use a payroll number or social security number. For vehicles, the license plate or the chassis number are candidates. And trailers might be identified by a serial number (or maybe by the trlNo column I saw in your design?)

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

I will probably have missed something or made some errors (I really should not be typing this kind of messages at 1 AM <grin>), but I do hope that you'll find this a useful framework to start off with. It may hurt a little to throw away what you've done so far, but I think that you'll end up with a much cleaner, simpler database that's easy to use, easy to understand and easy to maintain.

Best, Hugo

--

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