

Re: Maths Formula Question

Source: <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2004-09/4026.html>

From: Ian (*ian_at_NoWhere.com*)

Date: 09/29/04

Date: Wed, 29 Sep 2004 10:49:42 +0100

Hi Hugo.

Ok here is the DDL of the main tables triggers and the stored procedure for doing the copy and past off records.

I have not included all the tables.

I would have liked to do what you are recommending about remove the triggers but then how would I cater for when a user then reorders an individual instruction.

At the moment if a user wants to swap 9 and 10 around then they just click a button but behind the scenes 9 gets updated to 10.5 and the trigger then kicks in and reorders the instructions.

So my database layout is this

One Scenario to Many Instructions

The point of the copy and past procedure is to make duplicates of Instructions that can then later be customised by the user, witch ends up being faster than the user creating these Instructions from scratch. If the user had to create them from scratch then they would have to figure out the order of the instructions again. That is what takes up all the time. and that is why this is the most important part of the entire application. If the order of the instructions is lost when inserting them into another scenario then they many aswell just built the instructions from scratch.

As for the max number of records to be copied and pasted the likely hood of there being more that 1000 at any one time is not very high.

Just so you know. This structure that I am showing you works perfectly every time up to the thousands.

My problem is that I have redesigned the triggers.(Also attached bellow the DDL)

And some times when copy and past happens Duplicate InstructionOrder numbers are generated with in the scenario. And that is not allowed. They must always be unique with in the scenario to keep the order.

The tbl_BTP_Instructions Table has allot of fields but there is really never more than say 6 fields with data in each instruction. What ever the ActionID is depends on what fields have data in them.

Thanks

Ian

```
/**DDL START***/
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_Instructions_tbl_BTP_Actions]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_Instructions] DROP CONSTRAINT  
FK_tbl_BTP_Instructions_tbl_BTP_Actions  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_BaseLine_tbl_BaseLineTypes]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_BaseLine] DROP CONSTRAINT  
FK_tbl_BTP_BaseLine_tbl_BaseLineTypes  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_Instructions_tbl_BTP_BaseLineTypes]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_Instructions] DROP CONSTRAINT  
FK_tbl_BTP_Instructions_tbl_BTP_BaseLineTypes  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_Scenario_tbl_Modules]') and OBJECTPROPERTY(id,  
N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_Scenario] DROP CONSTRAINT  
FK_tbl_BTP_Scenario_tbl_Modules  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Main_Project_Table_tbl_Product_table]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Main_Project_Table] DROP CONSTRAINT  
FK_tbl_Main_Project_Table_tbl_Product_table  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Main_Project_Table_tbl_Releases]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Main_Project_Table] DROP CONSTRAINT  
FK_tbl_Main_Project_Table_tbl_Releases  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_tbl_Users]') and OBJECTPROPERTY(id,  
N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP] DROP CONSTRAINT FK_tbl_BTP_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_ApplicationErrorLog_tbl_Users]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_ApplicationErrorLog] DROP CONSTRAINT  
FK_tbl_BTP_ApplicationErrorLog_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_RunInfo_tbl_Users]') and OBJECTPROPERTY(id,  
N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_RunInfo] DROP CONSTRAINT  
FK_tbl_BTP_RunInfo_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_Scenario_tbl_Users]') and OBJECTPROPERTY(id,  
N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_Scenario] DROP CONSTRAINT  
FK_tbl_BTP_Scenario_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_Scenario_tbl_Users1]') and OBJECTPROPERTY(id,  
N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_Scenario] DROP CONSTRAINT  
FK_tbl_BTP_Scenario_tbl_Users1  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_ImpactAnalysisData_tbl_Users]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_ImpactAnalysisData] DROP CONSTRAINT  
FK_tbl_ImpactAnalysisData_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Impact_Analysis_Home_tbl_Users]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Impact_Analysis_Home] DROP CONSTRAINT  
FK_tbl_Impact_Analysis_Home_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Impact_Analysis_Home_tbl_Users1]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
```

```
ALTER TABLE [dbo].[tbl_Impact_Analysis_Home] DROP CONSTRAINT  
FK_tbl_Impact_Analysis_Home_tbl_Users1  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Main_Project_Table_tbl_Users]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Main_Project_Table] DROP CONSTRAINT  
FK_tbl_Main_Project_Table_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Main_Project_Table_tbl_Users1]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Main_Project_Table] DROP CONSTRAINT  
FK_tbl_Main_Project_Table_tbl_Users1  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Project_Milestones_Users_tbl_Users]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Project_Milestones_Users] DROP CONSTRAINT  
FK_tbl_Project_Milestones_Users_tbl_Users  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_tbl_Main_Project_Table]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP] DROP CONSTRAINT  
FK_tbl_BTP_tbl_Main_Project_Table  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_BTP_BaseLine_tbl_Main_Project_Table]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_BTP_BaseLine] DROP CONSTRAINT  
FK_tbl_BTP_BaseLine_tbl_Main_Project_Table  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Impact_Analysis_Home_tbl_Main_Project_Table]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Impact_Analysis_Home] DROP CONSTRAINT  
FK_tbl_Impact_Analysis_Home_tbl_Main_Project_Table  
GO
```

```
if exists (select * from dbo.sysobjects where id =  
object_id(N'[dbo].[FK_tbl_Project_Detail_Form_tbl_Main_Project_Table]') and  
OBJECTPROPERTY(id, N'IsForeignKey') = 1)  
ALTER TABLE [dbo].[tbl_Project_Detail_Form] DROP CONSTRAINT  
FK_tbl_Project_Detail_Form_tbl_Main_Project_Table
```

GO

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_tbl_Project_Documentation_Main_Window_tbl_Main_Project
_Table]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[tbl_Project_Documentation_Main_Window] DROP CONSTRAINT
FK_tbl_Project_Documentation_Main_Window_tbl_Main_Project_Table
GO
```

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_tbl_Project_Milestones_tbl_Main_Project_Table]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[tbl_Project_Milestones] DROP CONSTRAINT
FK_tbl_Project_Milestones_tbl_Main_Project_Table
GO
```

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_tbl_BTP_History_tbl_BTP]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[tbl_BTP_History] DROP CONSTRAINT
FK_tbl_BTP_History_tbl_BTP
GO
```

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_tbl_BTP_Scenario_tbl_BTP]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[tbl_BTP_Scenario] DROP CONSTRAINT
FK_tbl_BTP_Scenario_tbl_BTP
GO
```

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_tbl_BTP_Instructions_tbl_BTP_Scenario]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[tbl_BTP_Instructions] DROP CONSTRAINT
FK_tbl_BTP_Instructions_tbl_BTP_Scenario
GO
```

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_tbl_BTP_RunInfo_tbl_BTP_Scenario]') and
OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[tbl_BTP_RunInfo] DROP CONSTRAINT
FK_tbl_BTP_RunInfo_tbl_BTP_Scenario
GO
```

```
/****** Object: Trigger dbo.CascadDelete_tbl_Main_Project_Table Script
Date: 29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[CascadDelete_tbl_Main_Project_Table]') and
OBJECTPROPERTY(id, N'IsTrigger') = 1)
drop trigger [dbo].[CascadDelete_tbl_Main_Project_Table]
GO
```

```
/****** Object: Trigger dbo.InsertReOrder_BTP_Instructions Script Date:
29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[InsertReOrder_BTP_Instructions]') and OBJECTPROPERTY(id,
N'IsTrigger') = 1)
drop trigger [dbo].[InsertReOrder_BTP_Instructions]
GO

/****** Object: Trigger dbo.UpdateReOrder_BTP_Instructions Script Date:
29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[UpdateReOrder_BTP_Instructions]') and OBJECTPROPERTY(id,
N'IsTrigger') = 1)
drop trigger [dbo].[UpdateReOrder_BTP_Instructions]
GO

/****** Object: Trigger dbo.DeleteReOrder_BTP_Instructions Script Date:
29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[DeleteReOrder_BTP_Instructions]') and OBJECTPROPERTY(id,
N'IsTrigger') = 1)
drop trigger [dbo].[DeleteReOrder_BTP_Instructions]
GO

/****** Object: Stored Procedure dbo.sp_CopyPastInstructions Script
Date: 29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[sp_CopyPastInstructions]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[sp_CopyPastInstructions]
GO

/****** Object: Table [dbo].[tbl_BTP_Instructions] Script Date:
29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_BTP_Instructions]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tbl_BTP_Instructions]
GO

/****** Object: Table [dbo].[tbl_BTP_Scenario] Script Date: 29/09/2004
10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_BTP_Scenario]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tbl_BTP_Scenario]
GO

/****** Object: Table [dbo].[tbl_BTP] Script Date: 29/09/2004 10:16:17
*****/
if exists (select * from dbo.sysobjects where id =
```

```
object_id(N'[dbo].[tbl_BTP]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[tbl_BTP]
GO
```

```
/****** Object: Table [dbo].[tbl_Main_Project_Table] Script Date:
29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_Main_Project_Table]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tbl_Main_Project_Table]
GO
```

```
/****** Object: Table [dbo].[tbl_BTP_Actions] Script Date: 29/09/2004
10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_BTP_Actions]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[tbl_BTP_Actions]
GO
```

```
/****** Object: Table [dbo].[tbl_BTP_KeyPress] Script Date: 29/09/2004
10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_BTP_KeyPress]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tbl_BTP_KeyPress]
GO
```

```
/****** Object: Table [dbo].[tbl_BTP_BaseLineTypes] Script Date:
29/09/2004 10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_BTP_BaseLineTypes]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tbl_BTP_BaseLineTypes]
GO
```

```
/****** Object: Table [dbo].[tbl_Modules] Script Date: 29/09/2004
10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_Modules]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[tbl_Modules]
GO
```

```
/****** Object: Table [dbo].[tbl_Product_table] Script Date: 29/09/2004
10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_Product_table]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[tbl_Product_table]
GO
```

```
/****** Object: Table [dbo].[tbl_Releases] Script Date: 29/09/2004
10:16:17 *****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_Releases]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[tbl_Releases]
GO
```

```
/****** Object: Table [dbo].[tbl_Users] Script Date: 29/09/2004 10:16:17
*****/
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[tbl_Users]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[tbl_Users]
GO
```

```
/****** Object: Table [dbo].[tbl_BTP_Actions] Script Date: 29/09/2004
10:16:19 *****/
CREATE TABLE [dbo].[tbl_BTP_Actions] (
[ActionID] [int] IDENTITY (1, 1) NOT NULL ,
[ActionDescription] [varchar] (50) NULL ,
[ActionListOrder] [int] NULL ,
[ShowInList] [tinyint] NULL ,
[ActionDescriptionDefaults] [varchar] (50) NULL
) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[tbl_BTP_KeyPress] Script Date: 29/09/2004
10:16:21 *****/
CREATE TABLE [dbo].[tbl_BTP_KeyPress] (
[KeyPressID] [int] IDENTITY (1, 1) NOT NULL ,
[KeyNumber] [int] NOT NULL ,
[KeyName] [varchar] (50) NOT NULL ,
[ListOrder] [int] NULL ,
[ShowInList] [tinyint] NOT NULL
) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[tbl_BTP_BaseLineTypes] Script Date:
29/09/2004 10:16:21 *****/
CREATE TABLE [dbo].[tbl_BTP_BaseLineTypes] (
[BaseLineTypeID] [int] IDENTITY (1, 1) NOT NULL ,
[BaseLineType] [varchar] (50) NULL ,
[ShowInList] [tinyint] NULL
) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[tbl_Modules] Script Date: 29/09/2004
10:16:21 *****/
CREATE TABLE [dbo].[tbl_Modules] (
[ModuleId] [int] IDENTITY (1, 1) NOT NULL ,
[Code] [varchar] (15) NULL ,
```

```
[Name] [varchar] (50) NULL ,  
[FRName] [varchar] (50) NULL ,  
[DEName] [varchar] (50) NULL ,  
[NLName] [varchar] (50) NULL ,  
[ITName] [varchar] (50) NULL ,  
[ESName] [varchar] (50) NULL ,  
[Group] [int] NULL ,  
[hlpActive] [bit] NULL ,  
[frmActive] [bit] NULL  
) ON [PRIMARY]  
GO
```

```
/****** Object: Table [dbo].[tbl_Product_table] Script Date: 29/09/2004  
10:16:22 *****/  
CREATE TABLE [dbo].[tbl_Product_table] (  
[Product_ID] [int] IDENTITY (1, 1) NOT NULL ,  
[Sage_Product] [varchar] (50) NULL  
) ON [PRIMARY]  
GO
```

```
/****** Object: Table [dbo].[tbl_Releases] Script Date: 29/09/2004  
10:16:22 *****/  
CREATE TABLE [dbo].[tbl_Releases] (  
[Release_ID] [int] IDENTITY (1, 1) NOT NULL ,  
[Release_Name] [float] NULL ,  
[Release_Description] [varchar] (50) NULL ,  
[Release_Date] [varchar] (50) NULL ,  
[Product_ID] [int] NULL  
) ON [PRIMARY]  
GO
```

```
/****** Object: Table [dbo].[tbl_Users] Script Date: 29/09/2004 10:16:22  
*****/  
CREATE TABLE [dbo].[tbl_Users] (  
[UserID] [int] IDENTITY (1, 1) NOT NULL ,  
[UserName] [varchar] (50) NULL ,  
[Password] [varchar] (50) NULL ,  
[Initials] [varchar] (50) NULL ,  
[FullName] [varchar] (50) NULL ,  
[NoOfHoursPerWeek] [float] NULL ,  
[WorkPercentage] [float] NULL ,  
[Contractor] [bit] NOT NULL ,  
[CurrentEmployee] [bit] NOT NULL ,  
[Auther] [bit] NOT NULL ,  
[BTPWriter] [bit] NOT NULL ,  
[BTPAutomata] [bit] NOT NULL ,  
[FlowWriter] [bit] NOT NULL ,  
[Manager] [bit] NOT NULL ,  
[Admin] [bit] NOT NULL  
) ON [PRIMARY]  
GO
```

```
/****** Object: Table [dbo].[tbl_Main_Project_Table] Script Date:  
29/09/2004 10:16:22 *****/
```

```
CREATE TABLE [dbo].[tbl_Main_Project_Table] (  
  [Project_Number] [varchar] (50) NOT NULL ,  
  [Project_Name] [varchar] (50) NULL ,  
  [IsBaseLine] [bit] NULL ,  
  [Project_Type_ID] [int] NULL ,  
  [Required_Project_Number] [varchar] (50) NULL ,  
  [Required_Project_Number_2] [varchar] (50) NULL ,  
  [Original_Project_Number] [varchar] (50) NULL ,  
  [Notes] [varchar] (2000) NULL ,  
  [Licensed_As] [varchar] (50) NULL ,  
  [Also_See_Project] [varchar] (50) NULL ,  
  [Walkthrough_ID] [int] NULL ,  
  [Project_Doc_ID] [int] NULL ,  
  [FormHelp_ID] [int] NULL ,  
  [Release_ID] [int] NULL ,  
  [Product_ID] [int] NULL ,  
  [Project_Last_Updated] [smalldatetime] NULL ,  
  [Help_reviewed] [bit] NOT NULL ,  
  [Project_AuthorID] [int] NULL ,  
  [Project_TesterID] [int] NULL  
) ON [PRIMARY]  
GO
```

```
/****** Object: Table [dbo].[tbl_BTP] Script Date: 29/09/2004 10:16:23  
*****/
```

```
CREATE TABLE [dbo].[tbl_BTP] (  
  [BTP_ID] [int] IDENTITY (1, 1) NOT NULL ,  
  [BTP_Project_Number] [varchar] (50) NOT NULL ,  
  [BTP_Project_Name] [varchar] (50) NULL ,  
  [BTP_Machine] [varchar] (50) NULL ,  
  [BTP_Port] [int] NULL ,  
  [BTP_Company] [varchar] (50) NULL ,  
  [BTP_User] [varchar] (50) NULL ,  
  [BTP_Password] [varchar] (50) NULL ,  
  [BTP_Formserver] [varchar] (50) NULL ,  
  [BTP_Formserver_Port] [int] NULL ,  
  [BTP_OS] [varchar] (50) NULL ,  
  [BTP_Release] [varchar] (50) NULL ,  
  [BTP_BaseLineTypeID] [int] NULL ,  
  [BTP_DateStarted] [char] (10) NULL ,  
  [BTP_Comments] [varchar] (4000) NULL ,  
  [BTP_DocumentOne] [varchar] (200) NULL ,  
  [BTP_DocumentTwo] [varchar] (200) NULL ,  
  [BTP_DocumentThree] [varchar] (200) NULL ,  
  [BTP_WriterID] [int] NULL ,  
  [BTP_OS_User] [varchar] (50) NULL ,  
  [BTP_OS_Password] [varchar] (50) NULL  
) ON [PRIMARY]  
GO
```

/****** Object: Table [dbo].[tbl_BTP_Scenario] Script Date: 29/09/2004

10:16:24 *****/

```
CREATE TABLE [dbo].[tbl_BTP_Scenario] (  
  [ScenarioID] [int] IDENTITY (1, 1) NOT NULL ,  
  [ScenarioOrder] [float] NULL ,  
  [ScenarioName] [varchar] (50) NULL ,  
  [ScenarioDescription] [varchar] (1000) NULL ,  
  [BTP_ID] [int] NOT NULL ,  
  [ScenarioWriterID] [int] NULL ,  
  [CheckedOut] [tinyint] NOT NULL ,  
  [LastCheckedOutBy] [int] NULL ,  
  [CreatedDate] [datetime] NULL ,  
  [LastEditDate] [datetime] NULL ,  
  [IsSavedSet] [tinyint] NULL ,  
  [TestRun] [tinyint] NULL ,  
  [LogAll] [tinyint] NULL ,  
  [MakeUnique] [tinyint] NULL ,  
  [UniqueValue] [char] (2) NULL ,  
  [ScenarioLinkID] [int] NULL ,  
  [ModuleId] [int] NULL  
) ON [PRIMARY]  
GO
```

/****** Object: Table [dbo].[tbl_BTP_Instructions] Script Date:

29/09/2004 10:16:24 *****/

```
CREATE TABLE [dbo].[tbl_BTP_Instructions] (  
  [InstructionID] [int] IDENTITY (1, 1) NOT NULL ,  
  [ScenarioID] [int] NULL ,  
  [InstructionOrder] [float] NULL ,  
  [ActionID] [int] NULL ,  
  [ActionDescription] [varchar] (50) NULL ,  
  [KeyNumber] [int] NULL ,  
  [OptionName] [varchar] (200) NULL ,  
  [PlainText] [varchar] (200) NULL ,  
  [PlainText2] [varchar] (200) NULL ,  
  [FormName] [varchar] (100) NULL ,  
  [FieldName] [varchar] (100) NULL ,  
  [FieldIndex] [int] NULL ,  
  [FieldName2] [varchar] (100) NULL ,  
  [FieldIndex2] [int] NULL ,  
  [VariableOneFieldName] [varchar] (50) NULL ,  
  [VariableOneCustomName] [varchar] (50) NULL ,  
  [VariableOneIndex] [int] NULL ,  
  [VariableOne] [varchar] (50) NULL ,  
  [VariableTwoFieldName] [varchar] (50) NULL ,  
  [VariableTwoCustomName] [varchar] (50) NULL ,  
  [VariableTwoIndex] [int] NULL ,  
  [VariableTwo] [varchar] (50) NULL ,  
  [Comments] [varchar] (2000) NULL ,  
  [BreakOnLine] [tinyint] NULL ,  
  [StoppedOnLine] [tinyint] NULL ,
```

```
[RequiredInstruction] [tinyint] NULL ,  
[CriticalAction] [tinyint] NULL ,  
[ExpectedResults] [varchar] (4000) NULL ,  
[BaseLineTypeID] [int] NULL ,  
[MakeUnique] [tinyint] NULL ,  
[ColIndex] [int] NULL ,  
[SkipInstruction] [tinyint] NULL ,  
[Operator] [varchar] (2) NULL ,  
[Sign] [varchar] (2) NULL  
) ON [PRIMARY]  
GO
```

```
CREATE CLUSTERED INDEX [IX_ScenarioID] ON  
[dbo].[tbl_BTP_Instructions]([ScenarioID]) ON [PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_Actions] ADD  
CONSTRAINT [DF_tbl_Actions_Exclude] DEFAULT (1) FOR [ShowInList],  
CONSTRAINT [PK_tbl_Actions] PRIMARY KEY NONCLUSTERED  
(  
[ActionID]  
) ON [PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_KeyPress] ADD  
CONSTRAINT [DF_tbl_BTP_KeyPress_Exclude] DEFAULT (1) FOR [ShowInList],  
CONSTRAINT [PK_tbl_BTP_KeyPress] PRIMARY KEY NONCLUSTERED  
(  
[KeyPressID]  
) ON [PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_BaseLineTypes] ADD  
CONSTRAINT [DF_tbl_BaseLineTypes_ShowInList] DEFAULT (0) FOR [ShowInList],  
CONSTRAINT [PK_tbl_BaseLineTypes] PRIMARY KEY NONCLUSTERED  
(  
[BaseLineTypeID]  
) ON [PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_Modules] ADD  
CONSTRAINT [PK_tbl_Modules] PRIMARY KEY NONCLUSTERED  
(  
[ModuleId]  
) ON [PRIMARY]  
GO
```

```
CREATE INDEX [Code_tbl_Modules] ON [dbo].[tbl_Modules]([Code]) ON  
[PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_Product_table] ADD
CONSTRAINT [PK_tbl_Product_table] PRIMARY KEY NONCLUSTERED
(
  [Product_ID]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[tbl_Releases] ADD
CONSTRAINT [PK_tbl_Releases] PRIMARY KEY NONCLUSTERED
(
  [Release_ID]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[tbl_Users] ADD
CONSTRAINT [PK_tbl_Users] PRIMARY KEY NONCLUSTERED
(
  [UserID]
) ON [PRIMARY] ,
CONSTRAINT [UserName_tbl_Users] UNIQUE NONCLUSTERED
(
  [UserName]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[tbl_Main_Project_Table] ADD
CONSTRAINT [DF_tbl_Main_Project_Table_IsBaseLine] DEFAULT (0) FOR
[IsBaseLine],
CONSTRAINT [DF_tbl_Main_Project_Table_Help_reviewd] DEFAULT (0) FOR
[Help_reviewed],
CONSTRAINT [PK_tbl_Main_Project_Table] PRIMARY KEY NONCLUSTERED
(
  [Project_Number]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[tbl_BTP] ADD
CONSTRAINT [DF_tbl_BTP_BTP_Port] DEFAULT (38113) FOR [BTP_Port],
CONSTRAINT [DF_tbl_BTP_BTP_Formserver_Port] DEFAULT (20005) FOR
[BTP_Formserver_Port],
CONSTRAINT [PK_tbl_BTP] PRIMARY KEY NONCLUSTERED
(
  [BTP_ID]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_Scenario] ADD
CONSTRAINT [DF_tbl_BTP_Scenario_CheckedOut] DEFAULT (0) FOR [CheckedOut],
CONSTRAINT [DF_tbl_BTP_Scenario_IsSavedSet] DEFAULT (0) FOR [IsSavedSet],
CONSTRAINT [DF_tbl_BTP_Scenario_TestRun] DEFAULT (0) FOR [TestRun],
CONSTRAINT [PK_tbl_BTP_Scenario] PRIMARY KEY NONCLUSTERED
```

```
(  
  [ScenarioID]  
) ON [PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_Instructions] ADD  
  CONSTRAINT [DF_tbl_BTP_Instructions_BreakOnLine] DEFAULT (0) FOR  
  [BreakOnLine],  
  CONSTRAINT [DF_tbl_BTP_Instructions_StoppedOnLine] DEFAULT (0) FOR  
  [StoppedOnLine],  
  CONSTRAINT [DF_tbl_BTP_Instructions_RequiredInstruction] DEFAULT (0) FOR  
  [RequiredInstruction],  
  CONSTRAINT [DF_tbl_BTP_Instructions_CriticalAction] DEFAULT (0) FOR  
  [CriticalAction],  
  CONSTRAINT [DF_tbl_BTP_Instructions_SkipInstruction] DEFAULT (0) FOR  
  [SkipInstruction],  
  CONSTRAINT [PK_tbl_BTP_Instructions] PRIMARY KEY NONCLUSTERED  
  (  
    [InstructionID]  
  ) ON [PRIMARY]  
GO
```

```
CREATE INDEX [InstructionOrderIndex] ON  
[dbo].[tbl_BTP_Instructions]([InstructionOrder]) ON [PRIMARY]  
GO
```

```
ALTER TABLE [dbo].[tbl_Main_Project_Table] ADD  
  CONSTRAINT [FK_tbl_Main_Project_Table_tbl_Product_table] FOREIGN KEY  
  (  
    [Product_ID]  
  ) REFERENCES [dbo].[tbl_Product_table] (  
    [Product_ID]  
  ),  
  CONSTRAINT [FK_tbl_Main_Project_Table_tbl_Releases] FOREIGN KEY  
  (  
    [Release_ID]  
  ) REFERENCES [dbo].[tbl_Releases] (  
    [Release_ID]  
  ),  
  CONSTRAINT [FK_tbl_Main_Project_Table_tbl_Users] FOREIGN KEY  
  (  
    [Project_AuthorID]  
  ) REFERENCES [dbo].[tbl_Users] (  
    [UserID]  
  ),  
  CONSTRAINT [FK_tbl_Main_Project_Table_tbl_Users1] FOREIGN KEY  
  (  
    [Project_TesterID]  
  ) REFERENCES [dbo].[tbl_Users] (  
    [UserID]  
  )  
)
```

GO

```
ALTER TABLE [dbo].[tbl_BTP] ADD
CONSTRAINT [FK_tbl_BTP_tbl_Main_Project_Table] FOREIGN KEY
(
  [BTP_Project_Number]
) REFERENCES [dbo].[tbl_Main_Project_Table] (
  [Project_Number]
),
CONSTRAINT [FK_tbl_BTP_tbl_Users] FOREIGN KEY
(
  [BTP_WriterID]
) REFERENCES [dbo].[tbl_Users] (
  [UserID]
)
GO
```

```
alter table [dbo].[tbl_BTP] nocheck constraint
[FK_tbl_BTP_tbl_Main_Project_Table]
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_Scenario] ADD
CONSTRAINT [FK_tbl_BTP_Scenario_tbl_BTP] FOREIGN KEY
(
  [BTP_ID]
) REFERENCES [dbo].[tbl_BTP] (
  [BTP_ID]
),
CONSTRAINT [FK_tbl_BTP_Scenario_tbl_Modules] FOREIGN KEY
(
  [ModuleId]
) REFERENCES [dbo].[tbl_Modules] (
  [ModuleId]
),
CONSTRAINT [FK_tbl_BTP_Scenario_tbl_Users] FOREIGN KEY
(
  [ScenarioWriterID]
) REFERENCES [dbo].[tbl_Users] (
  [UserID]
),
CONSTRAINT [FK_tbl_BTP_Scenario_tbl_Users1] FOREIGN KEY
(
  [LastCheckedOutBy]
) REFERENCES [dbo].[tbl_Users] (
  [UserID]
)
GO
```

```
ALTER TABLE [dbo].[tbl_BTP_Instructions] ADD
CONSTRAINT [FK_tbl_BTP_Instructions_tbl_BTP_Actions] FOREIGN KEY
(
```

```
[ActionID]
) REFERENCES [dbo].[tbl_BTP_Actions] (
  [ActionID]
),
CONSTRAINT [FK_tbl_BTP_Instructions_tbl_BTP_BaseLineTypes] FOREIGN KEY
(
  [BaseLineTypeID]
) REFERENCES [dbo].[tbl_BTP_BaseLineTypes] (
  [BaseLineTypeID]
),
CONSTRAINT [FK_tbl_BTP_Instructions_tbl_BTP_Scenario] FOREIGN KEY
(
  [ScenarioID]
) REFERENCES [dbo].[tbl_BTP_Scenario] (
  [ScenarioID]
) ON DELETE CASCADE
GO
```

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO
```

```
/****** Object: Stored Procedure dbo.sp_CopyPastInstructions Script
Date: 29/09/2004 10:16:25 *****/
```

```
-----
-- Name: sp_CopyPastInstructions
```

```
--
-- Summary:      This will Copy a Selection of instructions Change there
Instruction Order Number to keep them
-- In Order and allow them to go between 2 instructions and then inserts
them back into the table.
```

```
--
-- Example calls : EXEC sp_CopyPastInstructions ScenarioID, From, To,
InsertAfter, NoOfTimes
-- EXEC sp_CopyPastInstructions 11, 50, 100, 201, 3
```

```
--
-- Version:      001
-- History:      Created on 06/07/2004
```

```
--
-- Author:       Ian Killoran
```

```
--
-- Version      Date          Modifier          Description
```

```
CREATE PROCEDURE sp_CopyPastInstructions(
  @lScenarioID int,
  @lCopyFrom int,
  @lCopyTo int,
  @lInsertAfter int,
```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
)
@lNoOfTimes int
)
AS
BEGIN
/*
--These are the inputs For Testing and development
DECLARE @lScenarioID int
DECLARE @lCopyFrom int
DECLARE @lCopyTo int
DECLARE @lInsertAfter int
DECLARE @lNoOfTimes int
SET @lScenarioID = 11
SET @lCopyFrom = 40
SET @lCopyTo = 60
SET @lInsertAfter = 80
SET @lNoOfTimes = 3
-- End Inputs
*/
--Strings
DECLARE @sSQL varchar(8000)
DECLARE @sFieldListSQL varchar(8000)
DECLARE @sSelectSQL varchar(8000)
DECLARE @sCreateTableSQL varchar(8000)
DECLARE @sFieldName varchar(200)
DECLARE @sDataType varchar(200)
DECLARE @AddField varchar(100)
--Int
DECLARE @lNewRowCount int
DECLARE @lLength int
DECLARE @lRowCountHolder int
DECLARE @lRowCountCurrent int
DECLARE @iLoopControl int
DECLARE @lFirstComma int
--Decimal
DECLARE @dOrderIncrement decimal(28,20)
DECLARE @fNewOrderIncrement float(30)
--Prep Variables
SET @sSQL = ''
SET @sSelectSQL = ''
SET @sCreateTableSQL = ''
SET @sFieldName = ''
SET @sDataType = ''
SET @iLoopControl = 1
SET @lLength = 0
SET @lNewRowCount = 0
SET @fNewOrderIncrement = @lInsertAfter
SET @lFirstComma = 0
-- WORK OUT THE INCREMENT
SELECT @dOrderIncrement = 1/((cast(@lCopyTo as float) - cast(@lCopyFrom
as float) + @lNoOfTimes) * @lNoOfTimes)
--This is a fix for if they only want to copy one record
IF @dOrderIncrement = 1
    BEGIN
        SELECT @dOrderIncrement = 0.5
    END
-- CREATE THE DYNAMIC FIELD LIST
STRINGS -----
-- #FieldData to hold the data for later use
Create Table #FieldData (FieldName varchar(200), DataType varchar(200),
Length int, lRowCount int)
--Fill #FieldData with data
SET @sSQL = 'INSERT INTO #FieldData SELECT syscolumns.name AS FieldName,
```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
systypes.name AS DataType, syscolumns.length, 0' +
' FROM sysobjects INNER JOIN syscolumns ON sysobjects.id = syscolumns.id
INNER JOIN' +
' systypes ON syscolumns.xusertype = systypes.xusertype' +
' WHERE (sysobjects.name = N' + char(39) + 'tbl_BTP_Instructions' +
char(39) + ')' +
' ORDER BY syscolumns.colid'
EXEC(@sSQL)
--Fill up the count field for us in the loop
UPDATE #FieldData
SET @lNewRowCount = lRowCount = (@lNewRowCount + 1)
--get the first recorde
SELECT @lRowCountHolder = MIN(lRowCount)
FROM #FieldData
--Fill the variables with the first reccords data
SELECT @sFieldName=FieldName, @sDataType=DataType, @lLength=Length,
@lRowCountCurrent = lRowCount
FROM #FieldData
WHERE lRowCount = @lRowCountHolder
WHILE @iLoopControl = 1
--Now loop round creating the strings for later use in Dynamic SQL statment
BEGIN
IF @sFieldName <> 'InstructionID' --No need for this field it is and
IDENTITY
BEGIN
SET @sSelectSQL = @sSelectSQL + @sFieldName + ', '
END
IF @sDataType = 'int' OR @sDataType = 'tinyint' --No need to specify size
for these
IF @sFieldName = 'ScenarioID' OR @sFieldName =
'InstructionID' --InstructionID is IDENTITY and ScenarioID is created
manually
BEGIN
-- This is really just doing nothing so that it misses out the ScenarioID
and Instruction ID
SET @sCreateTableSQL = @sCreateTableSQL
END
ELSE
BEGIN
SET @sCreateTableSQL = @sCreateTableSQL + ' ADD ' + @sFieldName + '
' + @sDataType + ', '
END
ELSE
IF @sFieldName = 'InstructionOrder' --This one needs to be bigger
than the actual table
BEGIN
SET @sCreateTableSQL = @sCreateTableSQL + ' ADD ' + @sFieldName +
' ' + @sDataType + '(30), '
END
ELSE
BEGIN
SET @sCreateTableSQL = @sCreateTableSQL + ' ADD ' + @sFieldName + ' ' +
@sDataType + '(' + cast(@lLength as varchar(100)) + ')', '
END
-- Reset looping variables.
SELECT @lRowCountHolder = NULL
-- get the next @lRowCountHolder
SELECT @lRowCountHolder = MIN(lRowCount)
FROM #FieldData
WHERE lRowCount > @lRowCountCurrent
-- did we get a valid next @lRowCountHolder?
IF ISNULL(@lRowCountHolder,0) = 0
```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
BEGIN
--This will happen at table EOF
    BREAK
END
--Else get the next row.
SELECT @sFieldName=FieldName, @sDataType=DataType, @lLength=Length,
@lRowCountCurrent = lRowCount
FROM #FieldData
WHERE lRowCount = @lRowCountHolder
END
--Remove the last Comma ,
Select @sSelectSQL = LEFT(@sSelectSQL,Len(@sSelectSQL)-1)
--Print @sSelectSQL
--Print @sCreateTableSQL
DROP TABLE #FieldData
SET @sSQL = ''
-- END SECTION -----
-- CREATE THE TEMP TABLE WITH DATA AND REORDER -----
--Create the temp table so that it may be Altered
CREATE TABLE #ReOrder(ScenarioID int)
--Get the first alter
SELECT @lFirstComma = charindex(',', @sCreateTableSQL)
WHILE @lFirstComma > 0 --Loop round Adding fields to the temp table
    BEGIN
        SET @AddField = LTRIM(RTRIM(substring(@sCreateTableSQL,1,
charindex(',',@sCreateTableSQL) -1)))
        --print 'ALTER TABLE #ReOrder ' + @AddField
        EXEC ('ALTER TABLE #ReOrder ' + @AddField)
        --Truncate @sCreateTableSQL and get the next add
        SET @sCreateTableSQL = substring(@sCreateTableSQL,@lFirstComma + 1,
Datalength(@sCreateTableSQL))
        SELECT @lFirstComma = charindex(',', @sCreateTableSQL)
    END
-- This is how many times that insert will occur
SET @iLoopControl = @lNoOfTimes
-- Create the @sSQL for inserting
SET @sSQL = 'INSERT INTO #ReOrder SELECT ' + @sSelectSQL +
' FROM tbl_BTP_Instructions WHERE (ScenarioID = ' + cast(@lScenarioID as
varchar(500)) +
') AND (InstructionOrder >= ' + cast(@lCopyFrom as varchar(500)) +
' AND InstructionOrder <= ' + cast(@lCopyTo as varchar(500)) + ')' +
' ORDER BY InstructionOrder'
--print @sSQL
WHILE @iLoopControl <> 0 --Loop round inserting
    BEGIN
        EXEC(@sSQL)
        SELECT @iLoopControl = @iLoopControl - 1
    END
SET @sSQL = ''
--Select * From #ReOrder
-- Do The ReOrdering
UPDATE #ReOrder
SET @fNewOrderIncrement = InstructionOrder = @fNewOrderIncrement +
@dOrderIncrement
-- END
SECTION -----
-----
-- Insert The new Copied updated Records
INSERT INTO tbl_BTP_Instructions SELECT * FROM #ReOrder
--Select * from #ReOrder
--Clear the temp table
DROP TABLE #ReOrder
```

```

END
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
/***** Object: Trigger dbo.CascadDelete_tbl_Main_Project_Table      Script
Date: 29/09/2004 10:16:25 *****/
CREATE TRIGGER [CascadDelete_tbl_Main_Project_Table] ON
[tbl_Main_Project_Table]
FOR DELETE
AS
BEGIN
print 'Hello World'
END
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/***** Object: Trigger dbo.InsertReOrder_BTP_Instructions      Script Date:
29/09/2004 10:16:25 *****/
CREATE TRIGGER InsertReOrder_BTP_Instructions ON
dbo.tbl_BTP_Instructions
FOR INSERT
AS
SET NOCOUNT ON
DECLARE @lScenarioID int
DECLARE @lCounter Float(8)
SET @lCounter = 0
SELECT @lScenarioID = ScenarioID FROM INSERTED
--Line For Testing
--PRINT @lScenarioID
CREATE TABLE #ReOrderInstructions( InstructionID int, InstructionOrder real)
INSERT INTO #ReOrderInstructions
SELECT InstructionID, InstructionOrder FROM tbl_BTP_Instructions
WHERE ScenarioID = @lScenarioID
ORDER BY InstructionOrder
--Line For Testing
--SELECT * FROM #ReOrderInstructions
UPDATE #ReOrderInstructions
SET @lCounter = InstructionOrder = @lCounter + 1
UPDATE tbl_BTP_Instructions
SET tbl_BTP_Instructions.InstructionOrder = t.InstructionOrder
FROM #ReOrderInstructions t
WHERE t.InstructionID = tbl_BTP_Instructions.InstructionID
DROP TABLE #ReOrderInstructions
SET NOCOUNT OFF
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON

```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
GO
SET ANSI_NULLS ON
GO
/***** Object: Trigger dbo.UpdateReOrder_BTP_Instructions    Script Date:
29/09/2004 10:16:25 *****/
CREATE TRIGGER [UpdateReOrder_BTP_Instructions] ON
dbo.tbl_BTP_Instructions
FOR UPDATE
AS
IF UPDATE(InstructionOrder)
BEGIN
SET NOCOUNT ON
DECLARE @lScenarioID int
DECLARE @lCounter Float(8)
SET @lCounter = 0
SELECT @lScenarioID = ScenarioID FROM INSERTED
--Line For Testing
--PRINT @lScenarioID
CREATE TABLE #ReOrderInstructions( InstructionID int, InstructionOrder
real)
INSERT INTO #ReOrderInstructions
SELECT InstructionID, InstructionOrder FROM tbl_BTP_Instructions
WHERE ScenarioID = @lScenarioID
ORDER BY InstructionOrder
--Line For Testing
--SELECT * FROM #ReOrderInstructions
UPDATE #ReOrderInstructions
SET @lCounter = InstructionOrder = @lCounter + 1
UPDATE tbl_BTP_Instructions
SET tbl_BTP_Instructions.InstructionOrder = t.InstructionOrder
FROM #ReOrderInstructions t
WHERE t.InstructionID = tbl_BTP_Instructions.InstructionID
DROP TABLE #ReOrderInstructions
SET NOCOUNT OFF
END
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/***** Object: Trigger dbo.DeleteReOrder_BTP_Instructions    Script Date:
29/09/2004 10:16:25 *****/
CREATE TRIGGER [DeleteReOrder_BTP_Instructions] ON
dbo.tbl_BTP_Instructions
FOR DELETE
AS
SET NOCOUNT ON
DECLARE @lScenarioID int
DECLARE @lCounter Float(8)
SET @lCounter = 0
SELECT @lScenarioID = ScenarioID FROM DELETED
--Line for testing
--PRINT @lScenarioID
CREATE TABLE #ReOrderInstructions( InstructionID int, InstructionOrder real)
INSERT INTO #ReOrderInstructions
SELECT InstructionID, InstructionOrder FROM tbl_BTP_Instructions
WHERE ScenarioID = @lScenarioID
ORDER BY InstructionOrder
```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
--Line for testing
--SELECT * FROM #ReOrderInstructions
UPDATE #ReOrderInstructions
SET @lCounter = InstructionOrder = @lCounter + 1
UPDATE tbl_BTP_Instructions
SET tbl_BTP_Instructions.InstructionOrder = t.InstructionOrder
FROM #ReOrderInstructions t
WHERE t.InstructionID = tbl_BTP_Instructions.InstructionID
DROP TABLE #ReOrderInstructions
SET NOCOUNT OFF
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
/**DDL END**/
/**NEW TRIGGER START**/
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
ALTER          TRIGGER [DeleteReOrder_BTP_Instructions] ON
dbo.tbl_BTP_Instructions
FOR DELETE
AS
SET NOCOUNT ON
DECLARE @lScenarioID int
--DECLARE @InstructionOrder Float(8)
SELECT @lScenarioID = ScenarioID
FROM DELETED
UPDATE tbl_BTP_Instructions
    SET InstructionOrder = (SELECT COUNT(*)
        FROM tbl_BTP_Instructions T1
        WHERE T1.InstructionOrder <= tbl_BTP_Instructions.InstructionOrder
        AND T1.ScenarioID=tbl_BTP_Instructions.ScenarioID)
    WHERE ScenarioID=@lScenarioID
/*
This will only work if all the rows in the scenario are unique
It determines the number of rows which are less than or equal to the current
row... so if you had the following:
100
200
300
There are 1 rows <= 100, 2 <=200, and 3 <= 300. Even when they're out of
order, it still works. There are still always 2 rows less than or equal to
200.
*/
SET NOCOUNT OFF
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
ALTER          TRIGGER InsertReOrder_BTP_Instructions ON
dbo.tbl_BTP_Instructions
FOR INSERT
AS
SET NOCOUNT ON
```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
DECLARE @lScenarioID int
SELECT @lScenarioID = ScenarioID
FROM INSERTED
UPDATE tbl_BTP_Instructions
  SET InstructionOrder = (SELECT COUNT(*)
                        FROM tbl_BTP_Instructions T1
                        WHERE T1.InstructionOrder <= tbl_BTP_Instructions.InstructionOrder
                        AND T1.ScenarioID=tbl_BTP_Instructions.ScenarioID)
  WHERE ScenarioID=@lScenarioID
/*
This will only work if all the rows in the scenario are unique
It determines the number of rows which are less than or equal to the current
row... so if you had the following:
100
200
300
There are 1 rows <= 100, 2 <=200, and 3 <= 300. Even when they're out of
order, it still works. There are still always 2 rows less than or equal to
200.
*/
SET NOCOUNT OFF
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
ALTER      TRIGGER [UpdateReOrder_BTP_Instructions] ON
dbo.tbl_BTP_Instructions
FOR UPDATE
AS
IF UPDATE(InstructionOrder)
BEGIN
  SET NOCOUNT ON
  DECLARE @lScenarioID int
  SELECT @lScenarioID = ScenarioID
  FROM INSERTED
  UPDATE tbl_BTP_Instructions
  SET InstructionOrder = (SELECT COUNT(*)
                        FROM tbl_BTP_Instructions T1
                        WHERE T1.InstructionOrder <= tbl_BTP_Instructions.InstructionOrder
                        AND T1.ScenarioID=tbl_BTP_Instructions.ScenarioID)
  WHERE ScenarioID=@lScenarioID
/*
This will only work if all the rows in the scenario are unique
It determines the number of rows which are less than or equal to the current
row... so if you had the following:
100
200
300
There are 1 rows <= 100, 2 <=200, and 3 <= 300. Even when they're out of
order, it still works. There are still always 2 rows less than or equal to
200.
*/
  SET NOCOUNT OFF
  END
GO
SET QUOTED_IDENTIFIER OFF
GO
```

microsoft.public.vb.general.discussion: Re: Maths Formula Question

```
SET ANSI_NULLS ON
GO
/**NEW TRIGGER END**/
"Hugo Kornelis" <hugo@pe_NO_rFact.in_SPAM_fo> wrote in message
news:jqpjl09i0s2qmp17vhj51q3rekv7jptmjc@4ax.com...
> On Tue, 28 Sep 2004 11:12:00 +0100, Ian wrote:
>
> >Hi Hugo
> >
> >This is a really great idea. but there is triggers on the table that keep
> >all the instructions in and incremental order so should I do that then
the
> >data will just change back before I finish.
> >
> >The way I am currently doing it is holding the data in a temp table that
I
> >want duplicated then I do an insert into table and the trigger then only
> >fires once.
>
> Hi Ian,
>
> Did you consider getting rid of the trigger and doing all updates to the
> table through stored procedures (that incorporate the part of the trigger
> logic that's relevant to the operation carried out in the proc).
>
> The reason I suggest this is this: you are basically storing integer data.
> But the way you are now planning to perform your inserts forces you to
> declare the column as a numeric datatype, as you now need to store
> fractions - that will immediately afterwards be replaced by integer data!
> Not very efficient.
>
> You'll also have to find out the number of digits needed to the right of
> the decimal point to hold separate values for the maximum number of rows
> you'll ever have to insert (and once you defined that maximum, you can
> expect a user to try it with just one or two more - Murphy's law always
> kicks in in cases like this).
>
> If against all advise you do decide to take this path, just forget about
> the complicated formula and use fixed increments (the minimum that the
> declared datatype for the column can handle). So if your datatype is
> numeric(15,4), your rows "between" 12 and 13 would be numbered 12.0001,
> 12.0002, 12.0003, ....., and finally 12.0007.
>
> Best, Hugo
> --
>
> (Remove _NO_ and _SPAM_ to get my e-mail address)
```