The Elusive Conditional WHERE Clause

From time to time, I have a need to utilize a conditional WHERE clause; in other words, I need to create a stored procedure which filters for or against different parameters, depending upon varying criteria. In the past, I tried creating sprocs using an IF statement to select one query over another. I’ve also used dynamic T-SQL (yes, ‘tis true), again, with an IF statement. The problem is that both solutions really aren’t. Solutions, that is. Now I believe that I have a solution, one which is sargable. We know that the SQL Optimizer discards tautologies immediately, so a “1=1” in the WHERE clause spends about a nanosecond of CPU time for it for the entire query.

But, what if we use a “1=@MyCondition”?

Since the @MyCondition is a variable, the Optimizer doesn’t recognize the statement as a tautology. This allows us the opportunity to use any number of different sargable WHERE conditions, depending on our need.

Before we “take a deep dive” into this, allow me to explain the sproc, a bit. The basic design was put to me as a challenge, not long ago, to create a list of university school weeks by WeekNumber and DayOfTheWeekNumber, i.e., Monday of the first week of school would be W1D1, Tuesday would be W1D2, etc. This was to be done using a Common Table Expression (CTE). This is probably the only way to perform this (here’s another challenge for you: can you do this *WITHOUT* using a CTE?). I took the challenge a bit further by extending the date range beyond what a recursive CTE allows (100 iterations), using a table variable, and by showing/not showing holidays (the holidays selected are not necessarily represented as the actual holidays, but merely as examples) as such.

if @MidDate<@pSchoolEndDate begin

;with cte ([Date],Wk,DOW)

as (

select @StartDate,@Wk,datepart(dw,@StartDate)

union all

select [Date]+1,case when DOW=1 then Wk+1 else Wk end,datepart(dw,[Date]+1)

from cte

where [Date]<@MidDate

)

insert into @Dates

select 'W'+cast(Wk as varchar)+'D'+cast(DOW-1 as varchar)

,convert(varchar(10),[Date],101)

,Wk

,DOW

from cte

where 1=1

and datepart(dw,[Date]) not in (1,7)

end

I chose a MERGE JOIN to handle the holidays; whether to show them, or not.

merge @Dates as d

using (select [Date]

from @Holidays) h on d.[Date]=h.[Date]

when matched and 0=@pShowHolidays then

delete

when matched and 1=@pShowHolidays then

update set d.WeekAndDay='Holiday'

when not matched then

insert([Date])

values('19000101');

Since this sproc has us pass in the holidays, we need to use Jeff Moden’s Tally, or, Integer table to convert the Comma-Delimited list of holidays. We check the number of actual characters passed into the sproc for holidays to determine how many iterations are needed with @CharCount.

declare @CrLf char(2),@Delim char(1),@CharCount int

declare @Holidays table([Date] datetime)

declare @integers table(n int identity(1,1) not null primary key clustered,x char(1))

select @Delim=','

set @pHolidayList=@Delim+@pHolidayList+@Delim

set @pHolidayList=replace(@pHolidayList,@Delim+@Delim,@Delim)

set @CharCount=len(@pHolidayList)

set rowcount @CharCount

insert into @integers

select 'x'

from dbo.syscolumns sc1

cross join dbo.syscolumns sc2

set rowcount 0

insert into @Holidays

select cast(substring(@pHolidayList,n+1,charindex(@Delim,@pHolidayList,n+1)-(n+1)) as datetime)

from @integers

where n<len(@pHolidayList)

and substring(@pHolidayList,n,1)=@Delim

We also need to populate The Days Of The Week , so, since they are static, we can just insert them directly into a table variable.

insert into @Days select 1,'Sun'

union all select 2,'Mon'

union all select 3,'Tue'

union all select 4,'Wed'

union all select 5,'Thu'

union all select 6,'Fri'

union all select 7,'Sat'

Now let speculate, not wildly, that the university schedules classes by the following criteria:

College Algebra: Mon-Fri

Calculus: Mon-Wed-Fri

Linear Algebra: Tue, Thu

Differential Equations: Mon-Wed-Fri

Advanced Calculus (Elementary Topology): Mon, Wed

Modern Algebra: Wed

The condition [array] uses the following:

where (0=@pUseCondition

or (1=@pUseCondition and d1.[WeekDay]='mon')

or (2=@pUseCondition and d1.[WeekDay]='tue')

or (3=@pUseCondition and d1.[WeekDay]='wed')

or (4=@pUseCondition and d1.[WeekDay]='thu')

or (5=@pUseCondition and d1.[WeekDay]='fri')

or (6=@pUseCondition and d1.[WeekDay] in ('mon','wed','fri'))

or (7=@pUseCondition and d1.[WeekDay] in ('tue','thu'))

or (8=@pUseCondition and d.[Date]>=@pEarliestDate)

or (9=@pUseCondition and d1.[WeekDay] in ('mon','fri'))

or (10=@pUseCondition and patindex(@HideDay,d.WeekAndDay)=0)

This is the flexibility of this technique; you may set whatever conditional CONDITIONS you require.

Here is the sproc:

use TempDB

go

if exists (select \* from dbo.sysobjects where id=object\_id(N'dbo.tp\_TestConditional\_WHERE\_Clause') and objectproperty(id,N'IsProcedure')=1)

drop procedure dbo.tp\_TestConditional\_WHERE\_Clause

go

set ansi\_nulls on

go

set quoted\_identifier on

go

/\*--------------------------------------------------------------------------------------

Procedure: tp\_TestConditional\_WHERE\_Clause

Called by:

declare @pSchoolStartDate datetime,@pSchoolEndDate datetime,@pHolidayList varchar(max),@pUseCondition int

,@pShowHolidays bit,@pHideDay int,@pEarliestDate datetime

exec TempDB.dbo.tp\_TestConditional\_WHERE\_Clause @pSchoolStartDate='20120826'

,@pSchoolEndDate='20130531'

,@pHolidayList='20121129,20121130,20121225,20130101,20130107'

,@pUseCondition=6 -- 0=No conditions; 1=Mon,2=Tue,3=Wed,4=Thu,5=Fri; 6=Mon,Wed,Fri; 7=Tue,Thu; 8=Earliest date to return; 9=Mon,Fri; 10=Exclude @pHideDay

,@pShowHolidays=1 -- 0=Hide Hols, 1=Show Hols

,@pHideDay=0 -- Use DOW number (1=Mon,2=Tue,3=Wed,4=Thu,5=Fri) Use with @pUseCondition=10

,@pEarliestDate='20130228'

History: 20130316, jhick

Created procedure.

--------------------------------------------------------------------------------------\*/

create procedure [dbo].[tp\_TestConditional\_WHERE\_Clause] (

@pSchoolStartDate datetime

,@pSchoolEndDate datetime

,@pHolidayList varchar(max)='19000101'

,@pUseCondition int=0

,@pShowHolidays bit=1

,@pHideDay int=0

,@pEarliestDate datetime=0

)

as

begin

set nocount on;

--####################################################### Main Code #######################################################

declare @CrLf char(2),@Delim char(1),@CharCount int

declare @Holidays table([Date] datetime)

declare @integers table(n int identity(1,1) not null primary key clustered,x char(1))

select @Delim=','

set @pHolidayList=@Delim+@pHolidayList+@Delim

set @pHolidayList=replace(@pHolidayList,@Delim+@Delim,@Delim)

set @CharCount=len(@pHolidayList)

set rowcount @CharCount

insert into @integers

select 'x'

from dbo.syscolumns sc1

cross join dbo.syscolumns sc2

set rowcount 0

insert into @Holidays

select cast(substring(@pHolidayList,n+1,charindex(@Delim,@pHolidayList,n+1)-(n+1)) as datetime)

from @integers

where n<len(@pHolidayList)

and substring(@pHolidayList,n,1)=@Delim

----========================================================================================================================

declare @StartDate datetime,@MidDate datetime,@Wk int,@HideDay char(3)

declare @Dates table(WeekAndDay varchar(10) null,[Date] datetime null,Wk int null,DOW int)

declare @Days table(DOW int,[WeekDay] char(3))

set @HideDay='%D'+cast(@pHideDay as varchar)

insert into @Days select 1,'Sun'

union all select 2,'Mon'

union all select 3,'Tue'

union all select 4,'Wed'

union all select 5,'Thu'

union all select 6,'Fri'

union all select 7,'Sat'

select @StartDate=@pSchoolStartDate,@MidDate=dateadd(dd,100,@StartDate)

select @Wk=case when datepart(dw,@StartDate)!=2 then 0 else 1 end

if @MidDate<@pSchoolEndDate begin

;with cte ([Date],Wk,DOW)

as (

select @StartDate,@Wk,datepart(dw,@StartDate)

union all

select [Date]+1,case when DOW=1 then Wk+1 else Wk end,datepart(dw,[Date]+1)

from cte

where [Date]<@MidDate

)

insert into @Dates

select 'W'+cast(Wk as varchar)+'D'+cast(DOW-1 as varchar)

,convert(varchar(10),[Date],101)

,Wk

,DOW

from cte

where 1=1

and datepart(dw,[Date]) not in (1,7)

end

select @StartDate=max([Date]),@Wk=max(@Wk) from @Dates

select @Wk=Wk from @Dates where [Date]=@StartDate

set @StartDate=dateadd(dd,1,@StartDate)

set @MidDate=dateadd(dd,100,@StartDate)

if @MidDate<@pSchoolEndDate begin

;with cte ([Date],Wk,DOW)

as (

select @StartDate,@Wk,datepart(dw,@StartDate)

union all

select [Date]+1,case when DOW=1 then Wk+1 else Wk end,datepart(dw,[Date]+1)

from cte

where [Date]<@MidDate

)

insert into @Dates

select 'W'+cast(Wk as varchar)+'D'+cast(DOW-1 as varchar)

,convert(varchar(10),[Date],101)

,Wk

,DOW

from cte

where 1=1

and datepart(dw,[Date]) not in (1,7)

end

select @StartDate=max([Date]),@Wk=max(@Wk) from @Dates

select @Wk=Wk from @Dates where [Date]=@StartDate

set @StartDate=dateadd(dd,1,@StartDate)

set @MidDate=dateadd(dd,100,@StartDate)

if @MidDate<@pSchoolEndDate begin

;with cte ([Date],Wk,DOW)

as (

select @StartDate,@Wk,datepart(dw,@StartDate)

union all

select [Date]+1,case when DOW=1 then Wk+1 else Wk end,datepart(dw,[Date]+1)

from cte

where [Date]<@MidDate

)

insert into @Dates

select 'W'+cast(Wk as varchar)+'D'+cast(DOW-1 as varchar)

,convert(varchar(10),[Date],101)

,Wk

,DOW

from cte

where 1=1

and datepart(dw,[Date]) not in (1,7)

end

select @StartDate=max([Date]),@Wk=max(Wk) from @Dates

if @StartDate is null begin

set @StartDate=@pSchoolStartDate

set @Wk=1

end

else begin

select @Wk=Wk from @Dates where [Date]=@StartDate

end

set @StartDate=dateadd(dd,1,@StartDate)

set @MidDate=dateadd(dd,100,@StartDate)

;with cte ([Date],Wk,DOW)

as (

select @StartDate,@Wk,datepart(dw,@StartDate)

union all

select [Date]+1,case when DOW=1 then Wk+1 else Wk end,datepart(dw,[Date]+1)

from cte

where [Date]<@pSchoolEndDate

)

insert into @Dates

select 'W'+cast(Wk as varchar)+'D'+cast(DOW-1 as varchar)

,convert(varchar(10),[Date],101)

,Wk

,DOW

from cte

where 1=1

and datepart(dw,[Date]) not in (1,7)

----##############################################

merge @Dates as d

using (select [Date]

from @Holidays) h on d.[Date]=h.[Date]

when matched and 0=@pShowHolidays then

delete

when matched and 1=@pShowHolidays then

update set d.WeekAndDay='Holiday'

when not matched then

insert([Date])

values('19000101');

----##############################################

select d.WeekAndDay,d1.[WeekDay],convert(varchar(10),d.[Date],101) as [Date]

from @Dates d

join @Days d1 on d1.DOW=d.DOW

where (0=@pUseCondition

or (1=@pUseCondition and d1.[WeekDay]='mon')

or (2=@pUseCondition and d1.[WeekDay]='tue')

or (3=@pUseCondition and d1.[WeekDay]='wed')

or (4=@pUseCondition and d1.[WeekDay]='thu')

or (5=@pUseCondition and d1.[WeekDay]='fri')

or (6=@pUseCondition and d1.[WeekDay] in ('mon','wed','fri'))

or (7=@pUseCondition and d1.[WeekDay] in ('tue','thu'))

or (8=@pUseCondition and d.[Date]>=@pEarliestDate)

or (9=@pUseCondition and d1.[WeekDay] in ('mon','fri'))

or (10=@pUseCondition and patindex(@HideDay,d.WeekAndDay)=0)

)

----========================================================================================================================

----##################################################### End Main Code #####################################################

end ---- tp\_TestConditional\_WHERE\_Clause

go

set quoted\_identifier off

go

set ansi\_nulls on

go

--grant execute on dbo.tp\_TestConditional\_WHERE\_Clause to UserList

--go

print @@servername

Looking at the WHERE condition, we see this:

where (0=@pUseCondition

or (1=@pUseCondition and d1.[WeekDay]='mon')

or (2=@pUseCondition and d1.[WeekDay]='tue')

or (3=@pUseCondition and d1.[WeekDay]='wed')

or (4=@pUseCondition and d1.[WeekDay]='thu')

or (5=@pUseCondition and d1.[WeekDay]='fri')

or (6=@pUseCondition and d1.[WeekDay] in ('mon','wed','fri'))

or (7=@pUseCondition and d1.[WeekDay] in ('tue','thu'))

or (8=@pUseCondition and d.[Date]>=@pEarliestDate)

or (9=@pUseCondition and d1.[WeekDay] in ('mon','fri'))

or (10=@pUseCondition and patindex(@HideDay,d.WeekAndDay)=0)

The “0=@pCondition” tells our sproc to ignore all subsequent @pCondition filters. The “or (1=@pCondition=1...)” tells the query engine to focus on this filter, wherein we are seeking Mondays, only. The @pShowHolidays filter resides in the MERGE JOIN code, above, and is not affected by the conditional WHERE, except where the @pCondition forces or rejects certain days of the week.

Using the calling code embedded within the sproc (instructions for use are included as comments),

declare @pSchoolStartDate datetime,@pSchoolEndDate datetime,@pHolidayList varchar(max),@pUseCondition int

,@pShowHolidays bit,@pHideDay int,@pEarliestDate datetime

exec TempDB.dbo.tp\_TestConditional\_WHERE\_Clause @pSchoolStartDate='20120826'

,@pSchoolEndDate='20130531'

,@pHolidayList='20121129,20121130,20121225,20130101,20130107'

,@pUseCondition=0 -- 0=No conditions; 1=Mon,2=Tue,3=Wed,4=Thu,5=Fri; 6=Mon,Wed,Fri; 7=Tue,Thu; 8=Earliest date to return; 9=Mon,Fri; 10=Exclude @pHideDay

,@pShowHolidays=1 -- 0=Hide Hols, 1=Show Hols

,@pHideDay=0 -- Use DOW number (1=Mon,2=Tue,3=Wed,4=Thu,5=Fri) Use with @pUseCondition=10

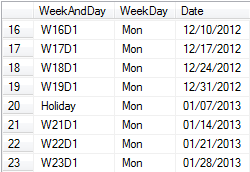
,@pEarliestDate='20130228'

let’s test this out.

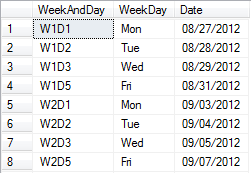
If we change just the @pCondition, our results will differ according to the @pCondition that we choose. For example, if we call the sproc using @pUseCondition=0 and @pShowHolidays=1, we get everything back, including the holidays, marked as such.



If, however, we call the sproc using @pUseCondition=1, we are returned only Mondays, no matter what else we selected.



Let’s take this a step further and set @pUseCondition=10. Let us also set @pHideDay=4. This should hide ALL Thursdays, holiday, or not.



I sincerely hope that this has been of some value to you. With this having been a problem for so long, I was skeptical about introducing any code utilizing such a technique into production, so I tested this into oblivion. If I am way off my rocker on this, please let me know in the discussion (but, back it up with facts, not vitriol). Also, please bear in mind that my company doesn’t allow me to participate in forums or blogs on company machines, so I will be able to respond only after hours in US Mountain Time. I genuinely hope that is actually a solution and not another problem.