

## Re: Classic Nest SP with Transaction Question

*Source:*

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2005-02/4962.html>

---

**From:** Ron Strong (*rstrong\_at\_DoNotSpamerols.com*)

**Date:** 02/21/05

Date: Sun, 20 Feb 2005 21:19:07 -0500

I didn't come up with a clean approach, but I did implement the kludge I mentioned earlier and it seems to work. Here's a modified version of my InnerSP.

```
-- INNER SP
create procedure InnerSP as begin
declare @ErrNo int
declare @InitTranccount int

set @InitTranccount = @@TRANCOUNT

/* Do something here */

select @ErrNo = @@ERROR

if @ErrNo <> 0 begin
    if @@TRANCOUNT > 0 rollback transaction

    -- Do logging here -- since we're not inside the rolled back
    -- logged data will not be lost

    /* increment the transaction count if not same as on entry to avoid
    error 266 */

    if @InitTranccount > @@TRANCOUNT begin transaction

    return @ErrNo -- on return, OuterSP will rollback the new
transaction
end

return 0
end
```

This takes care of the issues I had (immediate rollback and logging) but isn't pretty. My actual implementation also includes a flag passed to ChildSP telling it whether it should consider its actions part of an enclosing transaction or should begin and commit/rollback its own

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

transaction.

```
--
Ron Strong
"Chad" <chad.dokmanovich@unisys.com> wrote in message
news:cv8upg$qdc$1@trsvr.tr.unisys.com...
> I echo Ron's points and also have been using an approach very similar to
> Tom's method of using a SAVE PT instead of starting a new TRANSACTION if
the
> ChildSP is called from the ParentSP. Howber, I am still frustrated by the
> how I'vebeen handling logging of db errors.
>
> As Ron mentioned, He would like to add db logging immediatley after the
> error occurs. If the error occurs in the ChildSP, it is logged by the
> ChildSP (remember, the ChildSP may be called directly). However, if the
> childSP is called by the ParentSP and an error occurs in the ChildSP, the
> ChildSP rolls back to the SAVEPT and returns an error code to the
ParentSP.
>
> The ParentSp now has to ROLLBACK everything up to that point. As Ron
> mentioned, the ParentSP may have called many child SPs, and performed many
> updates up to this point, and all of these need to be rolled back.
>
> The frustrating point is that the ParentSP must, before rolling back the
> whole Transaction, must look up the logged error info and save it to
memory
> variables because the ROLLBACK that the Parent is about to perform will
roll
> it out erasing all record of the error in the Child. After the Parent
> performs the ROLLBACK, the Parent must then RELOG the error back to the
Log
> table using the informationit saved in the memory variables. This is the
> part that especially is inconvenient.
>
> Ron,
>
> It sounds to me like we are experiencing the same issues. If you nail down
a
> clean approach , I would be indebted if you could share it with me. While
> this exchange has been useful, it has so far helped more to confirm that I
> do understand what is going on, which was not my original assertion.
>
>
> "Tom Moreau" <tom@dont.spam.me.cips.ca> wrote in message
> news:eH%23ZJrtFFHA.3368@TK2MSFTNGP10.phx.gbl...
> > Well, you could nevertheless have the child manage itself, since it
never
> > knows how it will be called. When it throws its error, it returns to
the
> > calling proc, which then decides to rollback and exit, without calling
any
> > more procs.
> >
> > --
> > Tom
> >
> > -----
> > Thomas A. Moreau, BSc, PhD, MCSE, MCDBA
> > SQL Server MVP
> > Columnist, SQL Server Professional
> > Toronto, ON Canada
> > www.pinnaclepublishing.com
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> > .
> > "Ron Strong" <rstrong@DoNotSpamerols.com> wrote in message
> > news:O2r1EmtFFHA.3648@TK2MSFTNGP09.phx.gbl...
> > This will take care of what is done in the child, but what I want to do
is
> > rollback the entire outer transaction - the one initiated in the outer
SP.
> >
> > My example may have been too brief -- the outer SP may be making calls
to
> > several child SPs. What I would like is that any error, whether
> > encountered in the outer SP or its child SPs, results in an immediate
> > rollback of all the work performed within the transaction initiated in
the
> > outer SP.
> >
> > The rule, enforced by the raising of error 266, that entry Tranccount =
> > exit
> > Tranccount, seems to preclude doing this.
> >
> >
> > Ron Strong
> >
> >
> > "Tom Moreau" <tom@dont.spam.me.cips.ca> wrote in message
> > news:#2UHhStFFHA.3972@TK2MSFTNGP15.phx.gbl...
> >> PMFJI, but if your child proc is using an explicit tran, then it can be
> >> coded as follows:
> >>
> >> create proc dbo.ChildProc
> >> as
> >> set nocount on
> >>
> >> declare @trancount int
> >>
> >> set @trancount = @@TRANCOUNT
> >>
> >> if @trancount > 0
> >>     begin tran ChildProcTran
> >> else
> >>     save tran ChildProcTran
> >>
> >> /*
> >>     Do some stuff
> >> */
> >>
> >> if @@ERROR > 0
> >> begin
> >>     raiserror ('We have a problem.', 16, 1)
> >>     rollback ChildProcTran
> >>     return
> >> end
> >> else if @trancount = 0      -- began our own
> >>     commit tran
> >> go
> >>
> >> This way, only the child proc's txn will be rolled back by the child
> >> proc.
> >> The parent proc will be unaffected.
> >>
> >> --
> >> Tom
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >>
> >> -----
> >> Thomas A. Moreau, BSc, PhD, MCSE, MCDBA
> >> SQL Server MVP
> >> Columnist, SQL Server Professional
> >> Toronto, ON Canada
> >> www.pinnaclepublishing.com
> >> .
> >> "Ron Strong" <rstrong@DoNotSpamerols.com> wrote in message
> >> news:ulKZzJtFFHA.1528@TK2MSFTNGP09.phx.gbl...
> >> Brian & Chad
> >>
> >> I believe I'm having the same issue as Chad with nested stored
> >> procedures
> >> inside a transaction.
> >>
> >> What I'd like to do is begin a transaction in an outer SP. If all goes
> >> well, the transaction will be committed in the outer stored procedure -
> >> no
> >> problem there. However, if an error or other unexpected condition is
> >> encountered, I would like to rollback as close to the error as possible
> >> (in
> >> the statement following detection, if possible).
> >>
> >> Problem is this might involve a transaction begun in an outer SP being
> >> rolled back in an inner SP. Due to the fact that on entry to inner SP
> >> @@Trancount == 1 but on exit @@Trancount == 0, a new error, Error 266,
> >> gets
> >> generated.
> >>
> >> Following illustrates the problem:
> >>
> >> -- INNER SP
> >> create procedure InnerSP as begin
> >> declare @ErrNo int
> >>
> >> /* Do something here */
> >>
> >> select @ErrNo = @@ERROR
> >>
> >> if @ErrNo <> 0 begin
> >>     if @@TRANCOUNT > 0 rollback transaction
> >>     return @ErrNo -- on return new error (266) generated
> >> end
> >>
> >> return 0
> >> end
> >>
> >> -- OUTER SP
> >> create procedure OuterSP as begin
> >> declare @ErrNo int
> >>
> >> Begin Transaction
> >>
> >> exec @ErrNo = InnerSP
> >>
> >> -- if InnerSP failed, @@ERROR = 266 here
> >>
> >> if @ErrNo <> 0 begin
> >>     if @@TRANCOUNT > 0 rollback transaction
> >>     return @ErrNo
> >> end
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >>
> >> commit transaction
> >> return 0
> >> end
> >>
> >>
> >> I could hold off on performing the rollback until OuterSP examines the
> >> return value from InnerSP, but this is not ideal:
> >> (1) Immediately after the error I may want to do some logging or
> >> other fixup. If these are done before the rollback, they will be wiped
> >> out
> >> by the rollback.
> >> (2) In code subsequent to ther error there is always the
> >> possibility
> >> that execution will be terminated due to a severe error, preventing the
> >> enclosing SP from ever executing the rollback. While the lack of a
> >> subsequent COMMIT will ultimately lead to the transaction being rolled
> >> back,
> >> I would have no control over when the rollback occurs.
> >>
> >> I can avoid this problem via the kludge of a new "Begin Transaction"
> >> statement just before returning the error code from InnerSP to OuterSP.
> >> Is
> >> there a cleaner way to resolve this problem (beyond waiting for SQL
> >> Server
> >> 2005 try...catch blocks)?
> >>
> >>
> >> Ron Strong
> >>
> >> "Chad" <chad.dokmanovich@unisys.com> wrote in message
> >> news:cv7rc2$h6a$1@trsvr.tr.unisys.com...
> >> > Brian,
> >> >
> >> > Thank you again for your feedback. I appreciate the tip, in
particular
> >> on
> >> > handling concurrency problems using RowVersion, and I believe
> >> > understand
> >> > the
> >> > thrust of your points.
> >> >
> >> > However, I would like to place a spot light on a point I originally
> >> > made
> >> > that I feel may not have been addressed:
> >> >
> >> > *** If @@TRANCOUNT is = X in ParentSP when ChildSP is called, it must
> >> > be
> >> > =
> >> > X
> >> > immediately after returning from the CHILD call. , else an error
> >> results***
> >> >
> >> > If feel that this is the situation in the example you proposed.
> >> >
> >> > If the ParentSP BEGINS a TRANSACTON (Transaction count is now 1),
then
> >> calls
> >> > ChildSP, which does a ROLLBACK within Child, TranCount will be 0 when
> >> > control is returned to the Parent. Since TranCount was 1 just prior
to
> >> > calling the Child and it is zero immeditely after returning, this
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > result
> >> in
> >> > an ERROR:
> >> >
> >> > > Server: Msg 50000, Level 16, State 1, Procedure ChildSP, Line 10
> >> > > an error was raised
> >> > > Server: Msg 266, Level 16, State 2, Procedure ChildSP, Line 26
> >> > > Transaction count after EXECUTE indicates that a COMMIT or ROLLBACK
> >> > > TRANSACTION statement is missing. Previous count = 1, current count
> >> > =
> >> > 0.
> >> >
> >> >
> >> > > This is the part that I am missing. It seems to me that the Child
> >> > > cannot
> >> > > do
> >> > > the rollback if the Parent already began a Transaction.
> >> > >
> >> > > I hope I am not trying your patience. I would really like to get this
> >> > > point
> >> > > down.
> >> > >
> >> > > Thanks,
> >> > > Chad
> >> > >
> >> > >
> >> > > "Brian Selzer" <BrianSelzer@discussions.microsoft.com> wrote in
message
> >> > > news:9E615D99-D61A-4AF3-BEFB-09C2C12281D3@microsoft.com...
> >> > > The code I provided will work when called directly or from another
> >> > > stored
> >> > > > procedure. Use it as a template for both the parent and the child
> >> > > > procedure--in fact use this mechanism in all of your procedures.
> >> > > >
> >> > > > You should declare an additional variable, @RC, in the parent
> >> > > > procedure
> >> > > > to
> >> > > > > receive the return code from the stored procedure call. Otherwise
you
> >> > > > will
> >> > > > > lose the error code that originally caused the failure, for
example:
> >> > > >
> >> > > > > DECLARE @RC INT, @_ERROR INT
> >> > > > > EXEC @RC = ChildProc
> >> > > > > SET @_ERROR = @@ERROR
> >> > > > > IF @RC != 0 OR @_ERROR != 0 GOTO ERROR
> >> > > > >
> >> > > > > The key to this approach is that any error, regardless of the
reason
> >> > > > > (Constraint violation, out of memory, Deadlock victim, etc.) is
> >> > > > > > detected
> >> > > > > > and
> >> > > > > > > handled immediately after it occurs, and the error handling code
> >> > > > > > > rolls
> >> > > > > > > back
> >> > > > > > > the transaction. When an error occurs in the child procedure, it
> >> > > > > > > rolls
> >> > > > > > > back
> >> > > > > > > any pending transaction and returns the error code to the caller.
> >> > > > > > > The
> >> > > > > > > parent
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > > procedure detects that an error occurred by examining the return
> >> > > code,
> >> and
> >> > > transferrs control to its own error handler. Since the transaction
> > had
> >> > > already been rolled back in the child procedure, @@TRANCOUNT is
zero
> > and
> >> > > thus
> >> > > a rollback in the parent's error handler would cause an additional
> >> error.
> >> > > The condition IF @@TRANCOUNT > 0 prevents this. (It also prevents
> >> > > an
> >> > > additional error in the event the procedure is chosen as a deadlock
> >> > > victim.)
> >> > >
> >> > > I often extend this mechanism to detect concurrency problems. For
> >> > > example:
> >> > >
> >> > > DECLARE @_ERROR INT, @_ROWCOUNT INT
> >> > >
> >> > > BEGIN TRANSACTION
> >> > >
> >> > > UPDATE t1 SET coll = @coll where key1 = @Key and ver1 = @version
> >> > > SELECT @_ERROR = @@ERROR, @_ROWCOUNT = @@ROWCOUNT
> >> > > IF @_ERROR != 0 OR @_ROWCOUNT = 0 GOTO ERROR
> >> > >
> >> > > COMMIT TRANSACTION
> >> > > RETURN 0
> >> > >
> >> > > ERROR:
> >> > > IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
> >> > > IF @_ERROR = 0 AND @_ROWCOUNT = 0
> >> > > RETURN -1 -- indicate that a record was changed by
> >> another
> >> > > user
> >> > > ELSE
> >> > > RETURN @_ERROR
> >> > > END
> >> > >
> >> > > ver1 is a rowversion (timestamp) column, which is changed any time
a
> >> > > record
> >> > > is changed. If another user changes the record after the time it
was
> >> > > read,
> >> > > then ver1 will be different than @version, the update statement
will
> > not
> >> > > affect any rows, and consequently @@ROWCOUNT will be zero.
> >> > >
> >> > >
> >> > > "Chad" wrote:
> >> > >
> >> > >> Thank you for the response. I don't fully understand.
> >> > >>
> >> > >> In my example, I wanted to be able to call a ChildSP directly, or
> > call
> >> a
> >> > >> ParentSP which calls the ChildSP, and if an error occurs in the
> > child,
> >> > >> everything gets rolled back. Your exaple only included one stored
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> > proc,
> >> > >> so I
> >> > >> was a little unclear.
> >> > >>
> >> > >> I tried to create a 2 SP example using your style. In your error
> >> handler,
> >> > >> you check to see if @@TranCount > 0. If so, you know that there
was
> > an
> >> > >> error
> >> > >> above. However, if we take this approach in the ChildSP, performing
a
> >> > >> Rollback would cause the @@TranCount to be set to zero, and when
you
> >> > >> return
> >> > >> to the ParentSP, we find that @@TranCount is now 0, but it was 1
> >> perform
> >> > >> we
> >> > >> called ChildSP, and so we get the error:
> >> > >>
> >> > >> Server: Msg 50000, Level 16, State 1, Procedure ChildSP, Line 10
> >> > >> an error was raised
> >> > >> Server: Msg 266, Level 16, State 2, Procedure ChildSP, Line 26
> >> > >> Transaction count after EXECUTE indicates that a COMMIT or
ROLLBACK
> >> > >> TRANSACTION statement is missing. Previous count = 1, current
count
> >> > >> =
> >> 0.
> >> > >>
> >> > >> Try running the code below.
> >> > >>
> >> > >> I would be very much indebted if you could take the 2 SP example
and
> >> > >> modify
> >> > >> it to a approach that works and is sane.
> >> > >>
> >> > >>
> >> > >> CREATE TABLE [dbo].[Table1] (
> >> > >> [col1] [int] NULL ,
> >> > >> [col2] [int] NULL
> >> > >> ) ON [PRIMARY]
> >> > >>
> >> > >>
> >> > >> CREATE procedure ParentSP
> >> > >>
> >> > >> as
> >> > >>
> >> > >> begin
> >> > >>
> >> > >> DECLARE @_ERROR INT
> >> > >>
> >> > >> BEGIN TRANSACTION
> >> > >>
> >> > >> exec @_ERROR = ChildSP 1
> >> > >>
> >> > >> SELECT @_ERROR = @@ERROR
> >> > >>
> >> > >> IF @_ERROR != 0 GOTO ERROR
> >> > >>
> >> > >> COMMIT TRANSACTION
> >> > >>
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > >> RETURN 0
> >> > >>
> >> > >> ERROR:
> >> > >>
> >> > >> IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
> >> > >>
> >> > >> RETURN @_ERROR
> >> > >>
> >> > >> end
> >> > >>
> >> > >>
> >> > >>
> >> > >> CREATE procedure ChildSP
> >> > >>
> >> > >> (@RaiseError bit)
> >> > >>
> >> > >> as
> >> > >>
> >> > >> begin
> >> > >>
> >> > >> DECLARE @_ERROR INT
> >> > >>
> >> > >> BEGIN TRANSACTION
> >> > >>
> >> > >> if (@RaiseError = 1)
> >> > >>
> >> > >> RAISERROR ('an error was raised', 16, 1)
> >> > >>
> >> > >> ELSE
> >> > >>
> >> > >> UPDATE table1 set col1 = 1
> >> > >>
> >> > >>
> >> > >> SELECT @_ERROR = @@ERROR
> >> > >>
> >> > >> IF @_ERROR != 0 GOTO ERROR
> >> > >>
> >> > >> UPDATE table1 set col2 = 2
> >> > >>
> >> > >> SELECT @_ERROR = @@ERROR
> >> > >>
> >> > >> IF @_ERROR != 0 GOTO ERROR
> >> > >>
> >> > >> COMMIT TRANSACTION
> >> > >>
> >> > >> RETURN 0
> >> > >>
> >> > >> ERROR:
> >> > >>
> >> > >> IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
> >> > >>
> >> > >> RETURN @_ERROR
> >> > >>
> >> > >> end
> >> > >>
> >> > >>
> >> > >>
> >> > >> "Brian Selzer" <BrianSelzer@discussions.microsoft.com> wrote in
> > message
> >> > >> news:97330484-71EA-4142-9B25-DAE902EA8836@microsoft.com...
> >> > >> > There's a few things you should know:
> >> > >> > First: always check for errors after each DML statement or SP
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
call
> >> > >> > within
> >> > >> > a
> >> > >> > transaction, because it is possible for an early DML statement
to
> >> fail,
> >> > >> > and
> >> > >> > later ones to pass which causes an insidious data consistency
bug
> >> that
> >> > >> > is
> >> > >> > extremely difficult to find. Here's what I do:
> >> > >> >
> >> > >> > DECLARE @_ERROR INT
> >> > >> >
> >> > >> > BEGIN TRANSACTION
> >> > >> >
> >> > >> > UPDATE t1 set col1 = @col1
> >> > >> > SELECT @_ERROR = @@ERROR
> >> > >> > IF @_ERROR != 0 GOTO ERROR
> >> > >> >
> >> > >> > UPDATE t2 set col2 = @col2
> >> > >> > SELECT @_ERROR = @@ERROR
> >> > >> > IF @_ERROR != 0 GOTO ERROR
> >> > >> >
> >> > >> > COMMIT TRANSACTION
> >> > >> > RETURN 0
> >> > >> >
> >> > >> > ERROR:
> >> > >> > IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
> >> > >> > RETURN @_ERROR
> >> > >> >
> >> > >> > This approach should make your inquiry moot, since @@ERROR is
set
> > on
> >> > >> > exit
> >> > >> > from a procedure if @@TRANCOUNT is less than what it was upon
> > entry.
> >> > >> >
> >> > >> >
> >> > >> > I only use save points if I want to roll back only part of a
> >> > >> > transaction,
> >> > >> > here's what I do:
> >> > >> >
> >> > >> > DECLARE @_TRANCOUNT INT SET @_TRANCOUNT = @@TRANCOUNT
> >> > >> > DECLARE @_ERROR INT
> >> > >> >
> >> > >> > IF @_TRANCOUNT = 0
> >> > >> >     BEGIN TRANSACTION savePoint
> >> > >> > ELSE
> >> > >> >     SAVE TRANSACTION savePoint
> >> > >> >
> >> > >> > UPDATE t1 set col1 = @col1
> >> > >> > SELECT @_ERROR = @@ERROR
> >> > >> > IF @_ERROR != 0 GOTO ERROR
> >> > >> >
> >> > >> > UPDATE t2 set col2 = @col2
> >> > >> > SELECT @_ERROR = @@ERROR
> >> > >> > IF @_ERROR != 0 GOTO ERROR
> >> > >> >
> >> > >> > IF @_TRANCOUNT = 0
> >> > >> >     COMMIT TRANSACTION savePoint
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > >> >
> >> > >> > RETURN 0
> >> > >> >
> >> > >> > ERROR:
> >> > >> > IF @@TRANCOUNT > @_TRANCOUNT
> >> > >> >     ROLLBACK TRANSACTION savePoint
> >> > >> >
> >> > >> > RETURN @_ERROR
> >> > >> >
> >> > >> >
> >> > >> >
> >> > >> >
> >> > >> > "Chad" wrote:
> >> > >> >
> >> > >> >>> I have 2 Stored Procedures, "Parent" and "Child". The Parent
SP
> >> >>> calls
> >> > >> >>> the
> >> > >> >>> Child SP, but also the Child may be called directly.
> >> > >> >>>
> >> > >> >>> If the Child returns an error (which occurs whenever it is
passed
> >> > >> >>> a
> >> > >> >>> value
> >> > >> >>> of
> >> > >> >>> 2), I want all updates to be rolled out.
> >> > >> >>>
> >> > >> >>> I have a couple of working version of these 2 SPs, but what I
am
> >> > >> >>> looking
> >> > >> >>> for
> >> > >> >>> is "What is a the best (or a good way) of doing this?"
> >> > >> >>>
> >> > >> >>> This is what I understand about transactions:
> >> > >> >>> 1) Performing a BEGIN TRAN increments @@TRANCOUNT
> >> > >> >>> 2) Performing an END TTRAN decrements @@TRANCOUNT
> >> > >> >>> 3) A ROLLBACK TRAN returns @@TRANCOUNT to 0
> >> > >> >>> 4) If @@TRANCOUNT is = X in Parent when CHILD is called, it
must
> >> > >> >>> be
> >> > >> >>> =
> >> > >> >>> X
> >> > >> >>> immediately after returning from the CHILD call.
> >> > >> >>>
> >> > >> >>> I've played around with SAVE POINTs within a transaction, but I
> >> > >> >>> do
> >> > >> >>> not
> >> > >> >>> have
> >> > >> >>> a sample here.
> >> > >> >>>
> >> > >> >>> I am looking for the simpilist, most intuitive sane and
> >> > >> >>> hopefully
> >> > >> >>> common
> >> > >> >>> approach to take here, I'm not sure I like what I've done-It's
> >> > >> >>> seems
> >> > >> >>> counter
> >> > >> >>> intuitive.
> >> > >> >>>
> >> > >> >>> Please alter my example as you would code it. Many thanks.
> >> > >> >>>
> >> > >> >>> --*****EXAMPLE 1
> >> > >> >>> --Setup: Create a table
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > >> >>
> >> > >> >> CREATE TABLE [dbo].[Table1] (
> >> > >> >> [col1] [int] NULL ,
> >> > >> >> [col2] [int] NULL
> >> > >> >> ) ON [PRIMARY]
> >> > >> >> GO
> >> > >> >>
> >> > >> >> --Throw a rec into it
> >> > >> >> INSERT INTO table1 (col1,col2) values (1,1)
> >> > >> >>
> >> > >> >> --Create the Parent SP
> >> > >> >> CREATE procedure dbo.parent
> >> > >> >> as
> >> > >> >> begin
> >> > >> >>
> >> > >> >> declare @res int
> >> > >> >>
> >> > >> >> begin transaction
> >> > >> >>
> >> > >> >> update table1 set col1 = 1
> >> > >> >> update table1 set col1 = 2
> >> > >> >>
> >> > >> >> exec @res = child 2
> >> > >> >>
> >> > >> >> if @Res = -1
> >> > >> >>     begin
> >> > >> >>     rollback transaction
> >> > >> >>     return @res
> >> > >> >>     end
> >> > >> >>
> >> > >> >> commit transaction
> >> > >> >>
> >> > >> >> return 0
> >> > >> >>
> >> > >> >> end
> >> > >> >>
> >> > >> >> --CREATE CHILD SP -
> >> > >> >>
> >> > >> >> CREATE procedure Child
> >> > >> >> @col2 int
> >> > >> >> as
> >> > >> >> begin
> >> > >> >>
> >> > >> >> begin transaction
> >> > >> >>
> >> > >> >> update table1 set col1 = @col2
> >> > >> >> update table1 set col2 = @col2
> >> > >> >>
> >> > >> >> if @col2 = 2 --DONT PASS 2!! It's an error!
> >> > >> >>     begin
> >> > >> >>         IF @@Trancount = 1
> >> > >> >>         Rollback TRANSACTION
> >> > >> >>     else
> >> > >> >>         Commit transaction --needs to be the same value as when we
> >> > >> >> entered
> >> > >> >> this
> >> > >> >> SP
> >> > >> >>     PRINT 'TRANCOUNT IN CHILD ' + CAST(@@TRANCOUNT AS
VARCHAR(10))
> >> > >> >>     return -1
> >> > >> >>     end
> >> > >> >>
```

## microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > >> >>
> >> > >> >> commit transaction
> >> > >> >>
> >> > >> >> return 0
> >> > >> >>
> >> > >> >> end
> >> > >> >>
> >> > >>
> >>
>
>>> -----
-
> >> -----
> >> > >> >> --*****EXAMPLE 2
> >> > >> >>
> >> > >> >> ALTER procedure dbo.parent
> >> > >> >> as
> >> > >> >> begin
> >> > >> >>
> >> > >> >> declare @res int
> >> > >> >>
> >> > >> >> begin transaction
> >> > >> >>
> >> > >> >> update table1 set col1 = 1
> >> > >> >> update table1 set col1 = 2
> >> > >> >>
> >> > >> >> exec @res = child 2
> >> > >> >>
> >> > >> >> if @Res = -1
> >> > >> >>     begin
> >> > >> >>         rollback transaction
> >> > >> >>     return @res
> >> > >> >>     end
> >> > >> >>
> >> > >> >> commit transaction
> >> > >> >>
> >> > >> >> return 0
> >> > >> >>
> >> > >> >> end
> >> > >> >>
> >> > >> >>
> >> > >> >> ALTER procedure Child
> >> > >> >> @col2 int
> >> > >> >> as
> >> > >> >> begin
> >> > >> >>
> >> > >> >> begin transaction
> >> > >> >>
> >> > >> >> update table1 set col1 = @col2
> >> > >> >> update table1 set col2 = @col2
> >> > >> >>
> >> > >> >> if @col2 = 2 --DONT PASS 2!! It's an error!
> >> > >> >>     begin
> >> > >> >>
> >> > >> >>         if @@trancount > 1 --outer SP started the transaction
> >> > >> >>             commit transaction --leave it to the parent to
rollback
> >> outer
> >> > >> >> trans
> >> > >> >>     else
> >> > >> >>         rollback transaction
> >> > >> >>
```

microsoft.public.sqlserver.programming: Re: Classic Nest SP with Transaction Question

```
> >> > >> >>   return -1
> >> > >> >>   end
> >> > >> >>
> >> > >> >>
> >> > >> >> commit transaction
> >> > >> >>
> >> > >> >> return 0
> >> > >> >>
> >> > >> >> end
> >> > >> >>
> >> > >> >>
> >> > >> >>
> >> > >>
> >> > >>
> >> > >>
> >> >
> >> >
> >>
> >>
> >
> >
>
>
```