Foreign key constraint may cause cycles or multiple cascade paths?

Asked 11 years, 1 month ago Active 9 months ago Viewed 129k times

176

I have a problem when I try to add constraints to my tables. I get the error:

> Introducing FOREIGN KEY constraint 'FK74988DB24B3C886' on table 'Employee' may cause cycles or multiple cascade paths. Specify ON DELETE NO ACTION or ON UPDATE NO ACTION, or modify other FOREIGN KEY constraints.

20

My constraint is between a `Code` table and an `employee` table. The `Code` table contains `Id`, `Name`, `FriendlyName`, `Type` and a `Value`. The `employee` has a number of fields that reference codes, so that there can be a reference for each type of code.

I need for the fields to be set to null if the code that is referenced is deleted.

Any ideas how I can do this?

sql sql-server constraints

edited Jul 9 '12 at 20:41
Ricardo Altamirano
11.5k ● 18 ● 65 ● 103

asked May 12 '09 at 7:47
Christian Nielsen

One of the solution is here – IsmailS May 7 '12 at 16:40

9 Answers

Active Oldest Votes

180

SQL Server does simple counting of cascade paths and, rather than trying to work out whether any cycles actually exist, it assumes the worst and refuses to create the referential actions (CASCADE): you can and should still create the constraints without the referential actions. If you can't alter your design (or doing so would compromise things) then you should consider using triggers as a last resort.

FWIW resolving cascade paths is a complex problem. Other SQL products will simply ignore the problem and allow you to create cycles, in which case it will be a race to see which will overwrite the value last, probably to the ignorance of the designer (e.g. ACE/Jet

does this). I understand some SQL products will attempt to resolve simple cases. Fact remains, SQL Server doesn't even try, plays it ultra safe by disallowing more than one path and at least it tells you so.

Microsoft themselves advises the use of triggers instead of FK constraints.

edited Sep 16 '19 at 11:13
Álvaro González
122k ● 33 ● 214 ● 307

answered May 12 '09 at 10:03
onedaywhen
48.4k ● 12 ● 85 ● 127

2 one thing that i still cannot understand is that, if this "problem" can be solved by using a trigger, then how come that a trigger will not "cause cycles or multiple cascade paths ..." ? – armen Jul 24 '14 at 10:17

5 @armen: because your trigger will explicitly supply the logic that the system couldn't implicitly figure out on it's own eg if there are multiple paths for a delete referential action then your trigger code will define which tables are deleted and in which order. – onedaywhen Aug 5 '14 at 15:22

6 And also the trigger executes after the first operation completes so there is no race going on. – Bon Aug 4 '15 at 16:50

2 @dumbledad: I mean, only use triggers when constraints (maybe on combination) can't get the job done. Constraints are declarative and their implementations are the responsibility of the system. Triggers are procedural code and you must code (and debug) the implementation and endure their disadvantages (worse performance etc). – onedaywhen Feb 22 '16 at 15:14

1 The problem with this is that the trigger only works as long as you remove the foreign key constraint, which means you then have no referential integrity checking on database inserts and so you need even more triggers to handle that. The trigger solution is a rabbit hole leading to a degenerate database design. – Neutrino Feb 20 at 16:29

99

A typical situation with multiple cascasing paths will be this: A master table with two details, let's say "Master" and "Detail1" and "Detail2". Both details are cascade delete. So far no problems. But what if both details have a one-to-many-relation with some other table (say "SomeOtherTable"). SomeOtherTable has a Detail1ID-column AND a Detail2ID-column.

```
Master { ID, masterfields }

Detail1 { ID, MasterID, detail1fields }

Detail2 { ID, MasterID, detail2fields }

SomeOtherTable {ID, Detail1ID, Detail2ID, someothertablefields }
```

In other words: some of the records in SomeOtherTable are linked with Detail1-records and some of the records in SomeOtherTable are linked with Detail2 records. Even if it is guaranteed that SomeOtherTable-records never belong to both Details, it is now impossible to make SomeOhterTable's records cascade delete for both details, because there are multiple cascading paths from

Master to SomeOtherTable (one via Detail1 and one via Detail2). Now you may already have understood this. Here is a possible solution:

```
Master { ID, masterfields }

DetailMain { ID, MasterID }

Detail1 { DetailMainID, detail1fields }

Detail2 { DetailMainID, detail2fields }

SomeOtherTable {ID, DetailMainID, someothertablefields }
```

All ID fields are key-fields and auto-increment. The crux lies in the DetailMainId fields of the Detail tables. These fields are both key and referential contraint. It is now possible to cascade delete everything by only deleting master-records. The downside is that for each detail1-record AND for each detail2 record, there must also be a DetailMain-record (which is actually created first to get the correct and unique id).

edited Mar 27 '12 at 12:23
Chuck Norris
14.7k ● 11 ● 79 ● 118

answered Aug 23 '10 at 13:55
hans riesebos
999 ● 6 ● 2

1 Your comment helped me a lot to understand the problem that I am facing. Thank you! I would prefer turning off the cascade delete for one of the path, then handle delete of other records some other ways(stored procedures; triggers; by code etc). But I keep your solution(grouping in one path) in mind for possible different applications of same problem... – freewill Feb 14 '14 at 21:39

1 One up for use of the word crux (and also for explaining) – masterwok Dec 11 '14 at 4:35

Is this better than writing triggers? It seems odd to add an additional table just to get the cascade working. – dumbledad Feb 2 '16 at 10:04

Anything is better than writing triggers. Their logic is opaque and they are inefficient compared to anything else. Breaking large tables into smaller ones for finer control is just a natural consequence of a better normalized database and not in itself something to be concerned by. – Neutrino Feb 20 at 16:38

12

I would point out that (functionally) there's a BIG difference between cycles and/or multiple paths in the SCHEMA and the DATA. While cycles and perhaps multipaths in the DATA could certainly complicated processing and cause performance problems (cost of "properly" handling), the cost of these characteristics in the schema should be close to zero.

Since most apparent cycles in RDBs occur in hierarchical structures (org chart, part, subpart, etc.) it is unfortunate that SQL Server assumes the worst; i.e., schema cycle == data cycle. In fact, if you're using RI constraints you can't actually build a cycle in the data!

I suspect the multipath problem is similar; i.e., multiple paths in the schema don't necessarily imply multiple paths in the data, but I have less experience with the multipath problem.

Of course if SQL Server *did* allow cycles it'd still be subject to a depth of 32, but that's probably adequate for most cases. (Too bad that's not a database setting however!)

"Instead of Delete" triggers don't work either. The second time a table is visited, the trigger is ignored. So, if you really want to simulate a cascade you'll have to use stored procedures in the presence of cycles. The Instead-of-Delete-Trigger would work for multipath cases however.

Celko suggests a "better" way to represent hierarchies that doesn't introduce cycles, but there are tradeoffs.

answered May 4 '10 at 19:29

Bill Cohagan

339 ● 3 ● 9

"if you're using RI constraints you can't actually build a cycle in the data!" -- good point! – onedaywhen Mar 23 '16 at 17:11

Sure you can build data circularity, but with MSSQL only using UPDATE. Other RDBMs support deferred constraints (integrity ensured at time of commit, not at time of insert/update/delete). – Carl Krig Jun 13 '19 at 15:51

7

There is an article available in which explains how to perform multiple deletion paths using triggers. Maybe this is useful for complex scenarios.

http://www.mssqltips.com/sqlservertip/2733/solving-the-sql-server-multiple-cascade-path-issue-with-a-trigger/

answered May 28 '15 at 14:38

Javier

1,565 ● 23 ● 39

3

By the sounds of it you have an OnDelete/OnUpdate action on one of your existing Foreign Keys, that will modify your codes table.

So by creating this Foreign Key, you'd be creating a cyclic problem,

E.g. Updating Employees, causes Codes to changed by an On Update Action, causes Employees to be changed by an On Update Action... etc...

If you post your Table Definitions for both tables, & your Foreign Key/constraint definitions we should be able to tell you where the problem is...

edited Oct 2 '18 at 8:50

Mihai Alexandru-Ionut
36.7k ● 7 ● 60 ● 93

answered May 12 '09 at 7:52

Eoin Campbell
39.2k ● 17 ● 92 ● 149

1 They are fairly long, so I dont think I can post them here, but I would much appreciate your help - dont know if there is some way i can send them to you? Ill try and describe it: The only constraints that exist are from 3 tables that all have fields that reference codes by a simple INT Id key. The problem seems to be that Employee has several fields that reference the code table and that i want them all to cascade to SET NULL. All I need is that when codes are deleted, the references to them should be set to null everywhere. – Christian Nielsen May 12 '09 at 8:13

post them anyway... I don't think anyone here will mind, and the code window will format them properly in a scrolling block :) – Eoin Campbell May 12 '09 at 8:30

2

This is because Emplyee might have Collection of other entity say Qualifications and Qualification might have some other collection Universities e.g.

```
public class Employee{
public virtual ICollection<Qualification> Qualifications {get;set;}
```

}

```
public class Qualification{

public Employee Employee {get;set;}

public virtual ICollection<University> Universities {get;set;}
```

}

```
public class University{

public Qualification Qualification {get;set;}
```

}

On DataContext it could be like below

```
protected override void OnModelCreating(DbModelBuilder modelBuilder){

modelBuilder.Entity<Qualification>().HasRequired(x=> x.Employee).WithMany(e =>
e.Qualifications);
modelBuilder.Entity<University>.HasRequired(x => x.Qualification).WithMany(e =>
e.Universities);
```

}

in this case there is chain from Employee to Qualification and From Qualification to Universities. So it was throwing same exception to me.

It worked for me when I changed

```
    modelBuilder.Entity<Qualification>().**HasRequired**(x=> x.Employee).WithMany(e =>
e.Qualifications);
```

To

```
    modelBuilder.Entity<Qualification>().**HasOptional**(x=> x.Employee).WithMany(e =>
e.Qualifications);
```

edited Dec 16 '13 at 17:11

answered Dec 16 '13 at 16:58

Rajnikant

1,231 ● 11 ● 16

Trigger is solution for this problem:

1

```
IF OBJECT_ID('dbo.fktest2', 'U') IS NOT NULL
    drop table fktest2
IF OBJECT_ID('dbo.fktest1', 'U') IS NOT NULL
    drop table fktest1
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'fkTest1Trigger' AND type = 'TR')
    DROP TRIGGER dbo.fkTest1Trigger
go
create table fktest1 (id int primary key, anQId int identity)
go
    create table fktest2 (id1 int, id2 int, anQId int identity,
        FOREIGN KEY (id1) REFERENCES fktest1 (id)
            ON DELETE CASCADE
            ON UPDATE CASCADE/*,
        FOREIGN KEY (id2) REFERENCES fktest1 (id) this causes compile error so we have
to use triggers
            ON DELETE CASCADE
            ON UPDATE CASCADE*/
            )
go

CREATE TRIGGER fkTest1Trigger
ON fkTest1
AFTER INSERT, UPDATE, DELETE
AS
    if @@ROWCOUNT = 0
        return
    set nocount on

    -- This code is replacement for foreign key cascade (auto update of field in
destination table when its referenced primary key in source table changes.
    -- Compiler complains only when you use multiple cascased. It throws this compile
error:
    -- Rrigger Introducing FOREIGN KEY constraint on table may cause cycles or multiple
cascade paths. Specify ON DELETE NO ACTION or ON UPDATE NO ACTION,
    -- or modify other FOREIGN KEY constraints.
    IF ((UPDATE (id) and exists(select 1 from fktest1 A join deleted B on B.anqid =
A.anqid where B.id <> A.id)))
    begin
        update fktest2 set id2 = i.id
            from deleted d
            join fktest2 on d.id = fktest2.id2
            join inserted i on i.anqid = d.anqid
    end
    if exists (select 1 from deleted)
        DELETE one FROM fktest2 one LEFT JOIN fktest1 two ON two.id = one.id2 where
two.id is null -- drop all from dest table which are not in source table
GO
```

```
insert into fktest1 (id) values (1)
insert into fktest1 (id) values (2)
insert into fktest1 (id) values (3)

insert into fktest2 (id1, id2) values (1,1)
insert into fktest2 (id1, id2) values (2,2)
insert into fktest2 (id1, id2) values (1,3)

select * from fktest1
select * from fktest2

update fktest1 set id=11 where id=1
update fktest1 set id=22 where id=2
update fktest1 set id=33 where id=3
delete from fktest1 where id > 22

select * from fktest1
select * from fktest2
```

answered Dec 3 '15 at 14:18

Tone Škoda
1,117 ● 13 ● 19

0

This is an error of type database trigger policies. **A trigger is code and can add some intelligences or conditions to a Cascade relation like Cascade Deletion.** You may need to specialize the related tables options around this like *Turning off CascadeOnDelete*:

```
protected override void OnModelCreating( DbModelBuilder modelBuilder )
{
    modelBuilder.Entity<TableName>().HasMany(i =>
i.Member).WithRequired().WillCascadeOnDelete(false);
}
```

Or Turn off this feature completely:

```
modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();
```

answered Oct 10 '14 at 11:35

Amirhossein Mehrvarzi
7,974 ● 4 ● 33 ● 57

-2

My solution to this problem encountered using ASP.NET Core 2.0 and EF Core 2.0 was to perform the following in order:

1. Run `update-database` command in Package Management Console (PMC) to create the database (this results in the "Introducing FOREIGN KEY constraint ... may cause cycles or multiple cascade paths." error)
2. Run `script-migration -Idempotent` command in PMC to create a script that can be run regardless of the existing tables/constraints
3. Take the resulting script and find `ON DELETE CASCADE` and replace with `ON DELETE NO ACTION`
4. Execute the modified SQL against the database

Now, your migrations should be up-to-date and the cascading deletes should not occur.

Too bad I was not able to find any way to do this in Entity Framework Core 2.0.

Good luck!

answered Nov 28 '17 at 20:37

user1477388

18.1k ● 23 ● 116 ● 222

You can change your migration file to do so (without changing sql script), i.e. in your migration file you can set onDelete action to Restrict from Cascade – Rushi Soni Mar 17 '18 at 11:46

It's better to specify this using fluent annotations so that you don't have to remember to do this if you end up deleting and recreating your migrations folder. – Allen Wang May 1 '18 at 15:08

In my experience, the fluent annotations can be used and should be used (I use them) but they are often be quite buggy. Simply specifying them in the code doesn't always work produce the expected result. – user1477388 May 2 '18 at 10:21

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.