When/Why to use Cascading in SQL Server?

Asked 16 years, 3 months ago Modified 2 years, 3 months ago

Viewed 120k times

181

When setting up foreign keys in SQL Server, under what circumstances should you have it cascade on delete or update, and what is the reasoning behind it?

This probably applies to other databases as well.

I'm looking most of all for concrete examples of each scenario, preferably from someone who has used them successfully.

sql-server database-design foreign-keys rdbms cascade

Share

Improve this question

Follow

edited Sep 16, 2008 at 18:47

Vinko Vrsalovic

340k ● 55 ● 340 ● 373

asked Sep 12, 2008 at 15:27

Joel Coehoorn

415k ● 114 ● 577 ● 813

7 This question doesn't seem strictly related to SQL Server and looks more like a theoretical, general question. It would be more usefull for the community if you remove the `sql-server` tag. – clapas Aug 23, 2013 at 15:40

Cascading actions take serializable locks. – Mitch Wheat Jul 24, 2019 at 0:07

16 Answers Sorted by: Highest score (default)

149

Summary of what I've seen so far:

- Some people don't like cascading at all.

Cascade Delete

- Cascade Delete may make sense when the semantics of the relationship can involve an exclusive *"is part of"* description. For example, an OrderLine record is part of its parent order, and OrderLines will never be shared between multiple orders. If the Order were to vanish, the OrderLine should as well, and a line without an Order would be a problem.
- The canonical example for Cascade Delete is SomeObject and SomeObjectItems, where it doesn't make any sense for an items record to ever exist without a corresponding main record.

- You should *not* use Cascade Delete if you are preserving history or using a "soft/logical delete" where you only set a deleted bit column to 1/true.

Cascade Update

- Cascade Update may make sense when you use a real key rather than a surrogate key (identity/autoincrement column) across tables.
- The canonical example for Cascade Update is when you have a mutable foreign key, like a username that can be changed.
- You should *not* use Cascade Update with keys that are Identity/autoincrement columns.
- Cascade Update is best used in conjunction with a unique constraint.

When To Use Cascading

- You may want to get an extra strong confirmation back from the user before allowing an operation to cascade, but it depends on your application.
- Cascading can get you into trouble if you set up your foreign keys wrong. But you should be okay if you do that right.
- It's not wise to use cascading before you understand it thoroughly. However, it is a useful feature and therefore worth taking the time to understand.

Share Improve this answer

Follow

edited May 12, 2015 at 3:35

Community Bot

1 ● 1

answered Sep 12, 2008 at 16:43

Joel Coehoorn

415k ● 114 ● 577 ● 813

3 Note that cascade updates are also often used where the "so-called" natural keys appear not to be these real effective unique keys. In fact I am convinced that cascade updates are needed only with poorly normalised database models, and they are an open gate to messy tables and messy code. – Philippe Grondier Oct 3, 2008 at 12:13

2 You are missing one important point, cascading can create huge performance issues if there are many child records. – HLGEM May 14, 2010 at 19:11

20 @HLGEM - I don't see the relevance. If a cascade operations causes a slow down, the equivalent manual process would either cause the same slow down or not be correctly protected in case the transaction needs to be rolled back. – Joel Coehoorn May 14, 2010 at 19:26

4 Why would it matter whether there's a cascade update on an IDENTITY or auto-increment column? I can see why it wouldn't be *necessary* because you shouldn't need to change those (arbitrary) values, but if one of them *did* change, at least the referential integrity would be intact. – Kenny Evitt May 27, 2010 at 20:11

76

Foreign keys are the best way to ensure referential integrity of a database. Avoiding cascades due to being magic is like writing everything in assembly because you don't trust the magic behind compilers.

What is bad is the wrong use of foreign keys, like creating them backwards, for example.

Juan Manuel's example is the canonical example, if you use code there are many more chances of leaving spurious DocumentItems in the database that will come and bite you.

Cascading updates are useful, for instance, when you have references to the data by something that can change, say a primary key of a users table is the name,lastname combination. Then you want changes in that combination to propagate to wherever they are referenced.

@Aidan, That clarity you refer to comes at a high cost, the chance of leaving spurious data in your database, which is *not small*. To me, it's usually just lack of familiarity with the DB and inability to find which FKs are in place before working with the DB that foster that fear. Either that, or constant misuse of cascade, using it where the entities were not conceptually related, or where you have to preserve history.

edited Sep 14, 2008 at 1:37

answered Sep 12, 2008 at 15:46

Vinko Vrsalovic
340k ● 55 ● 340 ● 373

9 Using that sort of 'natural' primary key is a really poor idea in the first place. – Nick Johnson Oct 19, 2008 at 11:39

1 The idea was to show an example about the cascading updates, I agree it's not the best example though. File locations may be a better example. – Vinko Vrsalovic Oct 19, 2008 at 15:10

4 RE: Comment directed to Aidan. No, leaving off CASCADE on an FK does not increase the chance of leaving spurious data. It decreases the chance that more data will be impacted by a command than was expected and increase code. Leaving out FKs entirely leave a chance of spurious data. – Shannon Severance Jul 18, 2009 at 17:08

6 Having at least twice in my career seen the business-threatening consequences of a misunderstood cascade delete I'm very disinclined to use them myself in all except the most clear cut cases. In both cases data had been deleted as a result of a cascade that really should have been retained but wasn't - and that it was missing was not detected until the normal backup cycle had lost the possibility of an easy restore. Vinko is correct from a purely logical point of view, however in the real world using cascades exposes one to the human fallibility and unforeseen consequences more than I'd like. – Cruachan May 27, 2010 at 20:43

7 @Cruachan: The rule, in my view, is simple. If the data is not as strongly related as to be useless without the parent data, then it doesn't warrant a cascade relationship. This I what I tried to address in the last phrase on my answer. – Vinko Vrsalovic May 28, 2010 at 0:11

20

I never use cascading deletes.

If I want something removed from the database I want to explicitly tell the database what I want taking out.

Of course they are a function available in the database and there may be times when it is okay to use them, for example if you have an 'order' table and an 'orderItem' table you may want to clear the items when you delete an order.

I like the clarity that I get from doing it in code (or stored procedure) rather than 'magic' happening.

For the same reason I am not a fan of triggers either.

Something to notice is that if you do delete an 'order' you will get '1 row affected' report back even if the cascaded delete has removed 50 'orderItem's.

Share Improve this answer

Follow

answered Sep 12, 2008 at 15:36

Loofer

6,969 ● 9 ● 65 ● 106

37 Why not get rid of primary keys, too? You would get the clarity of ensuring unique values in your code.
– MusiGenesis Nov 10, 2008 at 16:51

5 @MusiGenesis, Aidan was not advocating removing the FK. The FK still protects the data, but without CASCADE ON .... unexpected magic does not happen. – Shannon Severance Jul 18, 2009 at 17:05

9 @Vinko: Delete and update have well defined default semantics. Changing the behavior via a cascade or trigger to do more work leaves a chance more was done than intended. No, I don't work without testing and yes my databases are documented. But do I remember every piece of documentation while writing code? If I want higher level semantics, like delete parent & children, than I'll write and use an SP to do that. – Shannon Severance Jul 23, 2009 at 23:17

4 @Vinko. the problem of magic is not with competent developers or DBA's, it's with Joe interen 5 years later who's been given a 'simple'maintenance task when the DBA is on holiday and who then screws up corporate data without anyone realising it. Cascades have their place, but it's important to consider the full circumstances, including human factors, before deploying them. – Cruachan May 27, 2010 at 20:55

5 @Vinko: Why 'Gasp' SPs? SPs are defiantly the way to go where the database is a critical corporate asset. There's a strong argument in the sort of circumstances were talking about to restrict *all* data accesses to SPs, or at the least all but Select. See my answer under stackoverflow.com/questions/1171769/… – Cruachan May 29, 2010 at 9:32

13

I work a lot with cascading deletes.

It feels good to know whoever works against the database might never leave any unwanted data. If dependencies grow I just change the constraints in the diagramm in Management Studio and I dont have to tweak sp or dataacces.

That said, I have 1 problem with cascading deletes and thats circular references. This often leads to parts of the database that have no cascading deletes.

Share Improve this answer Follow

edited Oct 25, 2008 at 16:07

answered Sep 16, 2008 at 19:07

Mathias F

15.9k ● 25 ● 93 ● 165

12

I do a lot of database work and rarely find cascade deletes useful. The one time I have used them effectively is in a reporting database that is updated by a nightly job. I make sure that any changed data is imported correctly by deleting any top level records that have changed since the last import, then reimport the modified records and anything that relates to them. It save me from having to write a lot of complicated deletes that look from the bottom to the top of my database.

I don't consider cascade deletes to be quite as bad as triggers as they only delete data, triggers can have all kinds of nasty stuff inside.

In general I avoid real Deletes altogether and use logical deletes (ie. having a bit column called isDeleted that gets set to true) instead.

Share Improve this answer

answered Sep 12, 2008 at 15:51

Follow

Martynnw
10.9k ● 5 ● 31 ● 27

2 You got me curious to learn some more. Why do you strongly prefer logical deletes? Does the data you're working with have anything to do with it? – Tim Lovell-Smith Jul 4, 2018 at 0:26

9

One example is when you have dependencies between entities... ie: Document -> DocumentItems (when you delete Document, DocumentItems don't have a reason to exist)

Share Improve this answer

Follow

answered Sep 12, 2008 at 15:33

juan
81.8k ● 52 ● 164 ● 198

6

ON Delete Cascade:

When you want *rows in child table to be deleted* If *corresponding row is deleted* in parent table.

If *on cascade delete* isn't used then an error will be raised for *referential integrity*.

ON Update Cascade:

When you want *change in primary key* to be updated in *foreign key*

Share Improve this answer

Follow

answered Jan 12, 2014 at 16:03

Durai Amuthan.H

32.3k ● 11 ● 162 ● 241

5

Use cascade delete where you would want the record with the FK to be removed if its referring PK record was removed. In other words, where the record is meaningless without the referencing record.

I find cascade delete useful to ensure that dead references are removed by default rather than cause null exceptions.

Share Improve this answer

Follow

answered Mar 1, 2013 at 11:19

testpattern

2,478 ● 1 ● 26 ● 30

5

I have heard of DBAs and/or "Company Policy" that prohibit using "On Delete Cascade" (and others) purely because of bad experiences in the past. In one case a guy wrote three triggers which ended up calling one another. Three days to recover resulted in a total ban on triggers, all because of the actions of one idjit.

Of course sometimes Triggers are needed instead of "On Delete cascade", like when some child data needs to be preserved. But in other cases, its perfectly valid to use the On Delete cascade method. A key advantage of "On Delete cascade" is that it captures ALL the children; a

custom written trigger/store procedure may not if it is not coded correctly.

I believe the Developer should be allowed to make the decision based upon what the development is and what the spec says. A carpet ban based on a bad experience should not be the criteria; the "Never use" thought process is draconian at best. A judgement call needs to be made each and every time, and changes made as the business model changes.

Isn't this what development is all about?

Share Improve this answer

Follow

edited Jul 4, 2018 at 0:33

Tim Lovell-Smith

16.1k ● 16 ● 79 ● 94

answered Jun 23, 2014 at 14:29

BrianBurkill

135 ● 1 ● 2 ● 7

I didn't think it would delete *everything*... you mean the feature actually does what it says it does? ...
– Joel Coehoorn Jun 23, 2014 at 20:05

3

One reason to put in a cascade delete (rather than doing it in the code) is to improve performance.

Case 1: With a cascade delete

```
DELETE FROM table WHERE SomeDate < 7 years ago;
```

Case 2: Without a cascade delete

```
FOR EACH R IN (SELECT FROM table WHERE SomeDate < 7 y
  DELETE FROM ChildTable WHERE tableId = R.tableId;
  DELETE FROM table WHERE tableId = R.tableid;
  /* More child tables here */
NEXT
```

Secondly, when you add in an extra child table with a cascade delete, the code in Case 1 keeps working.

I would only put in a cascade where the semantics of the relationship is "part of". Otherwise some idiot will delete half of your database when you do:

```
DELETE FROM CURRENCY WHERE CurrencyCode = 'USD'
```

Share Improve this answer

Follow

edited May 27, 2010 at 20:47

JohnFx

34.9k ● 18 ● 107 ● 166

answered Oct 19, 2008 at 11:31

WW.

24.3k ● 15 ● 97 ● 124

6 Not knowing which datbase you use, I would suggest that your manual delete performs worse than cascading delete because it is not set based. In most datbases you can delete based on a join to another table and so have a set-based, much faster delete than looping through records. – HLGEM Nov 10, 2008 at 19:28

2

I try to avoid deletes or updates that I didn't explicitly request in SQL server.

Either through cascading or through the use of triggers. They tend to bite you in the ass some time down the line, either when trying to track down a bug or when diagnosing performance problems.

Where I would use them is in guaranteeing consistency for not very much effort. To get the same effect you would have to use stored procedures.

Share Improve this answer

Follow

answered Sep 12, 2008 at 15:34

pauliephonic

2,127 ● 1 ● 13 ● 9

2

I, like everyone else here, find that cascade deletes are really only marginally helpful (it's really not that much work to delete referenced data in other tables -- if there are lot of tables, you simply automate this with a script) but really annoying when someone accidentally cascade deletes some important data that is difficult to restore.

The only case where I'd use is if the data in the table table is highly controlled (e.g., limited permissions) and only updated or deleted from through a controlled process (like a software update) that has been verified.

Share Improve this answer

Follow

answered Sep 16, 2008 at 19:05

Jen A

1

A deletion or update to S that removes a foreign-key value found in some tuples of R can be handled in one of three ways:

1. Rejection
2. Propagation
3. nullification.

Propagation is referred to as cascading.

There are two cases:

‣ If a tuple in S was deleted, delete the R tuples that referred to it.

‣ If a tuple in S was updated, update the value in the R tuples that refer to it.

Share Improve this answer

Follow

answered Jan 25, 2016 at 0:40

Mayank Chopra

11 ● 1

0

If you're working on a system with many different modules in different versions, it can be very helpful, if the cascade deleted items are part of / owned by the PK holder. Else, all modules would require immediate patches to clean up their dependent items before deleting the PK owner, or the foreign key relation would be

omitted completely, possibly leaving tons of garbage in the system if cleanup is not performed correctly.

I just introduced cascade delete for a new intersection table between two already existing tables (the intersection to delete only), after cascade delete had been discouraged from for quite some time. It's also not too bad if data gets lost.

It is, however, a bad thing on enum-like list tables: somebody deletes entry 13 - yellow from table "colors", and all yellow items in the database get deleted. Also, these sometimes get updated in a delete-all-insert-all manner, leading to referential integrity totally omitted. Of course it's wrong, but how will you change a complex software which has been running for many years, with introduction of true referential integrity being at risk of unexpected side effects?

Another problem is when original foreign key values shall be kept even after the primary key has been deleted. One can create a tombstone column and an ON DELETE SET NULL option for the original FK, but this again requires triggers or specific code to maintain the redundant (except after PK deletion) key value.

answered Aug 5, 2014 at 20:21

Erik Hart

1,314 ● 1 ● 14 ● 34

0

Cascade deletes are extremely useful when implementing logical super-type and sub-type entities in a physical database.

When separate super-type and sub-type tables are are used to physically implement super-types/sub-types (as opposed to rolling up all sub-type attributes into a single physical super-type table), there is a one-to-one relationship between these tables and the issue then becomes how to keep the primary keys 100% in sync between these tables.

Cascade deletes can be a very useful tool to:

1) Make sure that deleting a super-type record also deletes the corresponding single sub-type record.

2) Make sure that any delete of a sub-type record also deletes the super-type record. This is achieved by implementing an "instead-of" delete trigger on the sub-type table that goes and deletes the corresponding super-type record, which, in turn, cascade deletes the sub-type record.

Using cascade deletes in this manner ensures that no orphan super-type or sub-type records ever exist, regardless of whether you delete the super-type record first or the sub-type record first.

edited Dec 9, 2016 at 20:51

answered Dec 9, 2016 at 20:39

Mark Allen

1 ● 1

Good example. In JPA, it's InheritanceStrategy Joined Table. For 1): Normally, you are using a persistence layer framework (EclipseLink, Hibernate, ...), that implements the deleted sequence for a joined entity to first delete the joined part, then the super part. But if you have more basic software in place, like an import or archive job, it's convenient to be able to just delete the entity by issuing a delete on the super part. Regarding 2): agree, but in that case the client should be aware already that he is working on a joined/sub part of the entity. – leo Dec 5, 2018 at 11:07

0

I would make a distinction between

- Data integrity
- Business logic/rules

In my experience it is best to enforce integrity as far as possible in the database using PK, FK, and other constraints.

However business rules/logic IMO is best implemented using code for the reason of cohesion (google "coupling and cohesion" to learn more).

Is cascade delete/update data integrity or business rules? This could of course be debated but I would say it is usually a logic/rule. For example a business rule may be

that if an `Order` is deleted all `OrderItems` should be automatically deleted. But it could also be that it should never be possible to delete an `Order` if it still have `OrderItems`. So this may be up to the business to decide. How do we know how this rule is currently implemented? If it is all in code we can just look at the code (high cohesion). If the rule is maybe implemented in the code or maybe implemented as cascade in the database then we need to look in multiple places (low cohesion).

Of course if you go all-in with putting your business rules only in the database and use triggers, stored proc then cascade may make sense.

I usually consider database vendor lock-in before using any stored proc or triggers. A SQL database that just stores data and enforces integrity is IMO easier to port to another vendor. So for that reason I usually don't use stored proc or triggers.

answered Aug 26, 2022 at 8:26

Jonas Kello

1,402 ● 2 ● 16 ● 29