override constraint from no action to cascading at runtime

Asked 16 years, 3 months ago Modified 5 years, 10 months ago

Viewed 373 times

0

I feel like I have a very basic/stupid question, yet I never saw/read/heard anything in this direction.

Say I have a table *users(userId, name)* and a table *preferences(id, userId, language)*. The example is trivial but could be extended to a situation with multi-level relations and way more tables..
When my UI requests to delete a user I first want to show a warning stating that also its preferences will be deleted. If at some point the database gets extended with more tables and relationships, but the software isn't adapted accordingly (the client didn't update) a generic message should be shown.

How can I implement this? The UI cannot know about the whole data structure and should not be bothered to walk down all the relations to manually delete all the depending records.

I would think this would be with constraints.
The constraint would be *no action* at first so the constraint will throw an error that can be caught by the UI. After the

UI receives a confirmation, the constraint should become a *cascade*.

Somehow I'm feeling like I'm getting this all wrong..

sql constraints cascade

Share

Improve this question

Follow

edited Feb 5, 2019 at 6:56

DJo

2,169 ● 4 ● 30 ● 48

asked Sep 17, 2008 at 9:50

Boris Callens

93.2k ● 86 ● 210 ● 308

2 Answers

Sorted by: Highest score (default)

1

What I would do is this:

1. The constraint is CASCADE
2. The application checks if preferences exist.
3. If they do, show the warning.
4. If no preferences exist, or the warning is accepted, delete the client.

Changing database relationships on the fly is not going to be a good idea!!

Cheers,

RB.

Share Improve this answer

Follow

answered Sep 17, 2008 at 9:53

RB.

37.1k ● 13 ● 100 ● 134

0

If you are worried about the user not realising the full impact of their delete, you might want to consider not actually deleting the data - instead you could simply set a flag on a column called say "`marked_for_deletion`". (the entries could then be deleted a safe time later)
The downside is that you need to remember to filter out the marked rows in other queries. This can be mitigated by creating a view on the table with the marked rows filtered out, and then always using the view in your queries.

Share Improve this answer

Follow

answered Sep 17, 2008 at 10:06

hamishmcn

7,981 ● 11 ● 42 ● 46