Checking for Related Records on Deletion

Posted At : August 23, 2007 2:55 PM | Posted By : Steve
Related Categories: DataMgr

After covering features, I wanted to share a real-world scenario with **DataMgr** (**Read about DataMgr**). In this case, the need to prevent the logical deletion of a record where related records exist.

In this system, we have volunteers and referrals (among other things). We want our administrators to be able to manage the referrals (how volunteers heard about the site). This includes the usual add/edit/delete actions. Except that we don't want them to be able to delete a referral value that is in use. Which is to say that they can't delete a referral that was chosen by a volunteer in the system.

To make matters more complicated, we don't actually delete volunteer records from the database. Instead, we use a logical delete. We have an "isDeleted" column that indicates if the record has been deleted (ideally we would have used a date field, but that's another story).

Typically, actions such as preventing a delete if related records exist, can be done a few places depending on the degree of certainty/flexibility needed.

- The database level - typically with a "before" trigger
- The code level - with some sort of conditional check before performing the delete
- The interface level - hiding the delete option from the user

Clearly, it is a good idea from a usability standpoint to hide the delete option from the user if they shouldn't be able to delete the record - no point in showing them actions that they can't use.

The database level provides the greatest degree of confidence. In order to add the constraint at that level, however, you must be absolutely certain that you will *never* want to delete the record if another record exists. You must also be using a hard delete (or write some relatively complicated logic for an "update" trigger).

Assuming that you don't want to use the database level (either you aren't sure enough that you will *never* have an exception or you don't want to deal with the complexity of triggers with soft deletes), then you will have to handle this in your ColdFusion code.

The general problem with preventing (or cascading) a deletion in code is that you have to remember to do that anywhere in the code that the deletion can occur.

DataMgr can handle this from a code-level, but one that is very close to the database level and keeps your code **DRY**. It does so by using relation fields (see "**Introduction to Relation Fields**").

In this case, an aggregate relationship (specifically "count"). First, I need to add a relation field to the referrals table. I could define this in XML if using **DataMgr's ActiveSchema functionality** or I could use the following code to add the relation field to the existing referrals table:

```
<cfscript>
sRelation = StructNew();
sRelation["type"] = "count";
sRelation["table"] = "volunteers";
sRelation["field"] = "volunteer_id";
sRelation["join-field"] = "referral_id";
sRelation["onDelete"] = "Error";
Application.DataMgr.setColumn(
  tablename="referrals",
  columnname="NumVolunteers",
  Relation=sRelation
);
</cfscript>
```

This will add a relation field named "NumVolunteers" to the referrals table that holds the number of volunteers that have selected that referral (by counting the "volunteer_id" field where the "referral_id"

fields of each table match).

By using this, I can display a delete button only for records where this value is zero, preventing the user from seeing a delete option where they shouldn't be able to delete a record.

The "OnDelete" key with a value of "Error" will also prevent the record from being deleted in cases where any matching records exist in the volunteers table.

When the deleteRecord() method is called against a table, DataMgr checks for any related records in any relation fields (where "onDelete" is "Error" or "Cascade"). If it finds any records for relation fields with OnDelete="Error", it throws an error and prevents the deletion.

This covers prevention of the delete action as well as hiding that option from the user. What it doesn't cover, however, is deletion by logical delete. Fortunately, it is easy to have DataMgr perform a logical delete instead of a hard delete.

All I need to do to accomplish this is to set the "isDeleted" field to have a "Special" attribute of "DeletionMark" in DataMgr:

```
<cfscript>
Application.DataMgr.setColumn(
  tablename="referrals",
  columnname="isDeleted",
  Special="DeletionMark"
);
</cfscript>
```

Now when DataMgr deletes a record, it will mark the "isDeleted" column as true instead of deleting the record (at which point it won't show up in any getRecords query or other relation field).

If I wanted to use a date field and have any field with a date in the field show as deleted instead of using a boolean field, I just do the same thing on a date field instead. Using a "DateDeleted" field as an example:

```
<cfscript>
Application.DataMgr.setColumn(
  tablename="referrals",
  columnname="DateDeleted",
  Special="DeletionMark"
);
</cfscript>
```

So, DataMgr can make preventing (or cascading) deletions easy to handle. It will provide data to allow you to hide a deletion option from a user and it will provide an error to prevent the deletion of a record - for both hard deletes and logical deletes (if DataMgr is used for all deletetions against that table).

DataMgr is open source and free for any use. You can **download it from RIAForge**.