Save What Has The FK

Posted At : June 25, 2008 4:03AM | Posted By : Matt Quackenbush
Related Categories: ColdFusion, Transfer

Wow! I cannot believe it has been 14 months since I last posted about Transfer . One would think that this was a result of me not using Transfer , which is actually not the case. I've been using it for nearly a year-and-a-half now, but I've been so busy with other things that I just haven't found the time to really dig into it the way that I want (need?) to. Well, diggin' time has arrived.

There are so many ways to model your database that it's a somewhat daunting task in the beginning, unless you have just a basic database schema. I still consider myself to be a n00b when it comes to ORM, so I'm no expert on the topic. With that said, here's a quickie example of a common database relationship and how you might model it in Transfer.

```
1. // put on expert hat
2. Quack.setExpert(
3. ex: "has-been" ,
4. spurt: "a drip under pressure"
5. );
```

Company > Address = One to Many

A company often has multiple addresses associated with it. Let's say that Acme Manufacturing has a delivery address (e.g. UPS/FedEx), a mailing address (e.g. a PO Box), and a corporate office address. There are two ways that we could model these relationships:

1. set a many-to-one element on the Address table definition (Many Addresses have A [as in 1] Company as its parent)
2. set a one-to-many element on the Company table definition (A [as in 1] Company has Many Addresses)

In my opinion, the latter option is better suited for this relationship, so I define a one-to-many (o2m) relationship on the Company table.

Now that we have that part done, let's say that Acme decides to add a new distribution center. How do we get that new address into the database? Do we save the Company? Do we save the Address?

I was pretty sure that I had it all figured out, but decided to double-check with Mr. Transfer himself, Mark Mandel . Thankfully, I actually had it right, but Mark gave me this gem to remember:

> "save what has the FK is the rule of thumb"

Man, that is an AWESOME way to put it! And it really stuck in my head as soon as he said it. I doubt I'll ever forget it.

So, knowing that we want to save what has the FK [foreign key], let's get back to our task of adding a new address for Acme's distribution center.

```
1. <cffunction name= "addAddress" hint= "I add a new Address" >
2. <cfargument name= "company"
```

```
hint= "The Company TransferObject"
required= "yes"
type= "any" />
<cfargument name= "args"
hint= "A struct of values to populate the Address with"
required= "yes"
type= "struct" />
<cfscript>
var address = getTransfer().new( "company.Address" );
getBeanPopulator().populate(
bean: address,
valueStruct: arguments.args
);
address.setParentCompany( arguments.company );
getTransfer().save( address );
</cfscript>
</cffunction>
```

For the purpose of this post, which is to make sure you remember to "save what has the FK", the important lines are the last two before the closing cfscript tag. We are first telling the Address that the Company object provided to the function is its parent, and then we are saving the Address, which, of course, is the object that has the foreign key.

Hopefully this short example will help you the next time you're wondering aloud, "Hmmmm, which damn object should I be saving?"