# Transfer: Many-to-One or One-to-One?

Posted At : November 9, 2008 2:20PM | Posted By : Matt Quackenbush
Related Categories: ColdFusion, Transfer

This post is actually a response to a post by Ray Camden . I planned to add a comment to Ray's blog, but I felt it was too long to be considered merely a comment. Here's a snippet from Ray's post that I am specifically responding to:

> Ok, so the point of this entry was to talk about relationships, so it's about time we got to it. Transfer defines three types of relationships: ManyToOne, OneToMany, and ManyToMany. We want to link an employee to a department, so which do we choose?
>
> Here is where I run into one of my problems with Transfer. When I think of "Link an employee to a department", I naturally want to look for a OneToOne relationship. One employee belongs in one department, right?
>
> However, that's not how Transfer looks at it. Every single time I use Transfer I have to double check this because it just seems... wrong to me, but Transfer looks at it as a ManyToOne. Ie, many employees are in one department. Again, this just does not click for my brain.

**Many-to-One**

The reason Transfer looks at Employee > Department as Many-to-One is because it ****IS**** an m2o relationship. It would be impossible to have this as a One-to-One (o2o) relationship, because it simply isn't one. Think about your database tables for a moment...

A) Employee: How many Department records can a single Employee record be assigned to? One? Many?

B) Department: How many Employee records can be assigned to a single Department record? One? Many?

The only answer to (A) is One, and the only answer to (B) is Many. Therefore, in terms of relationships, we know we are working with either One-to-Many (o2m) or Many-to-One (m2o). The only question is, which way do we want to model it?

To answer this question, we want to think about the model from the perspective of the view. Paul Marcotte has an excellent post about this on his blog.

Using the Employee > Department example, the Employee view is almost guaranteed to need the Department, whereas the Department view, more than likely, will only occasionally care about the Employee. Knowing this, we'll want our relationship to be readily available to the view on our Employee object. This means that we're going to add an m2o relationship to our Employee.

As an aside, I would choose to utilize a bit of TQL that would allow the Department view to get information about the Employee side of the relationship, but that's an entirely different post.

**One-to-One**

So then, what would be a good example of a true One-to-One (o2o) relationship?

Keeping with the Employee theme, let's assume that our application has a User table, which is used by our login system. This would be our super class. Employee is a subclass of User. We'll ask the exact same two questions about these two tables:

(A) User: How many Employee records can a single User record be assigned to? One? Many?

(B) Employee: How many User records can be assigned to a single Employee record? One? Many?

This time, the answer to both (A) and (B) is One, making this a true o2o relationship.

**Summary**

It is correct that Transfer currently does not support One-to-One relationships out of the box, but it is critical that one understands the difference between Many-to-One and One-to-One.

Employee > Department = Many-to-One

User > Employee (or Superclass > Subclass) = One-to-One

HTH