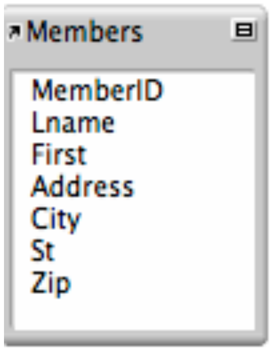
A classic database problem is that of the "many to many" relationship. This refers to a situation best described by an example.

Imagine you are the secretary of the Explorer's Club, and are keeping track of the club members and the club committees.

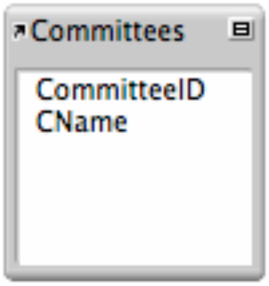
You have one table that contains the member information:
Last Name
First Name
Address
City, State, Zip

Add a unique member ID field for each record. This gives the following table structure:

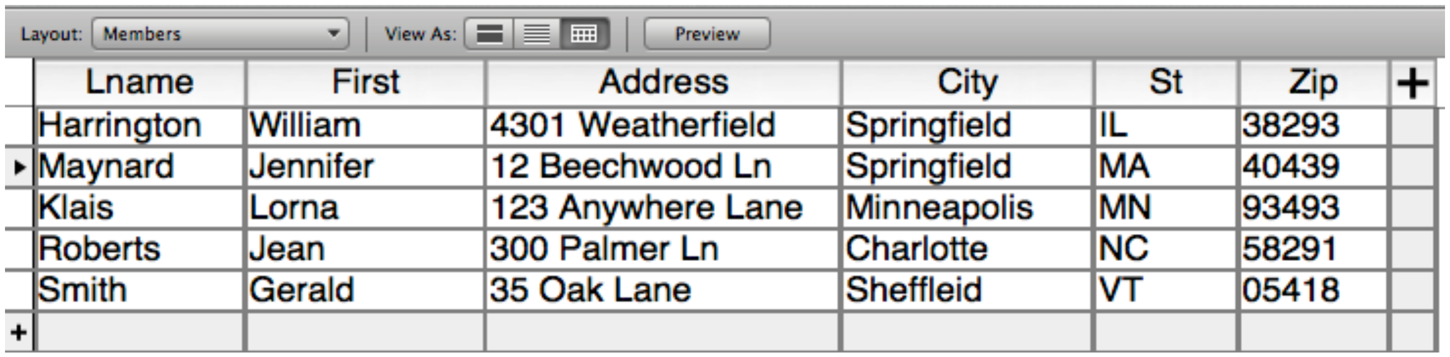


The MemberID field is a key field. It is a unique number automatically assigned by Filemaker when you add a new member.

Now we'll create the Committee table. This has just two fields, a key field CommitteeID and a field for the Committee name.

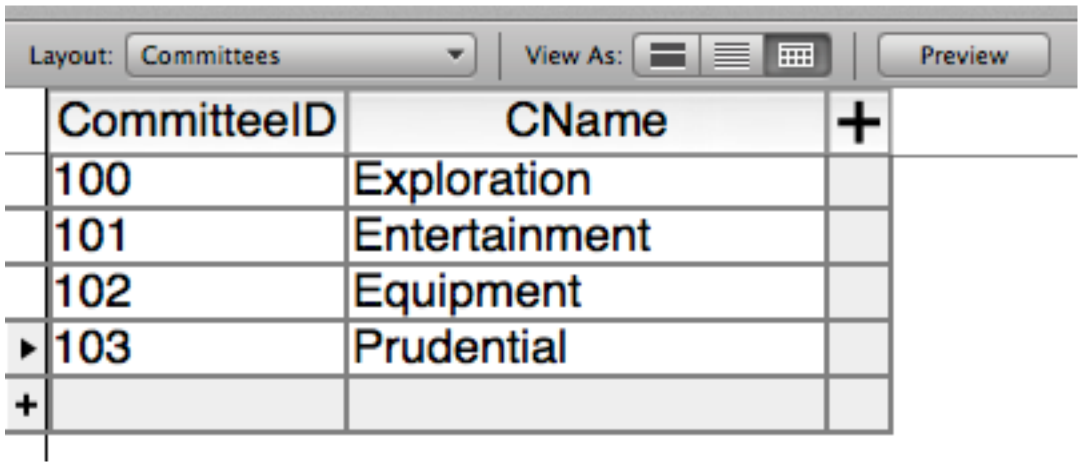


Now a few names to the names table. Note that the key field MemberID is not displayed, here.



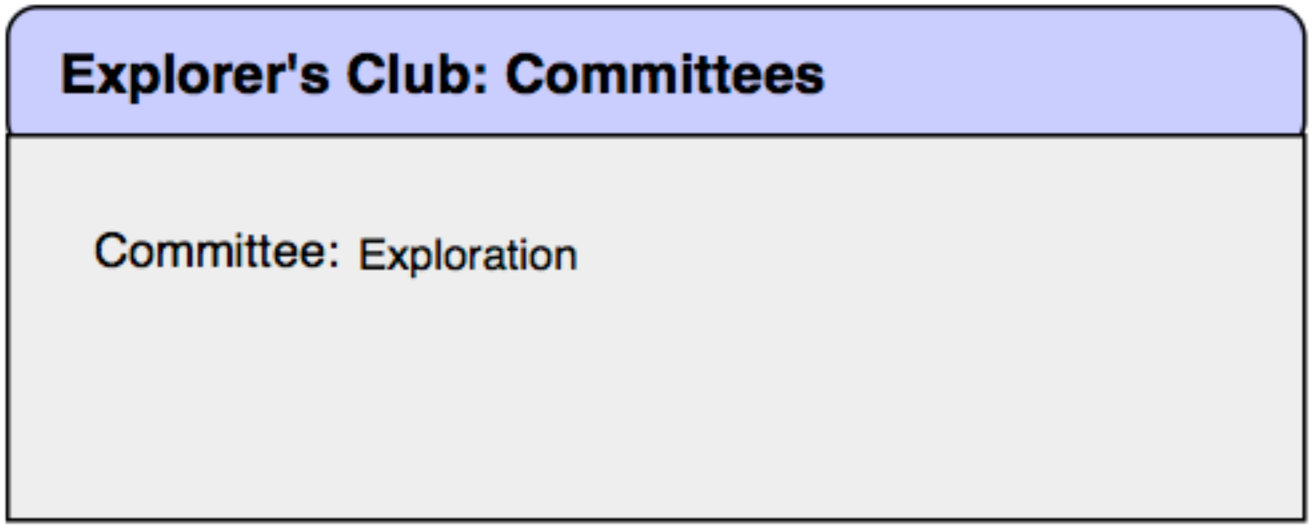
Lname	First	Address	City	St	Zip
Harrington	William	4301 Weatherfield	Springfield	IL	38293
Maynard	Jennifer	12 Beechwood Ln	Springfield	MA	40439
Klais	Lorna	123 Anywhere Lane	Minneapolis	MN	93493
Roberts	Jean	300 Palmer Ln	Charlotte	NC	58291
Smith	Gerald	35 Oak Lane	Sheffleid	VT	05418

We'll add a few committees for our club.



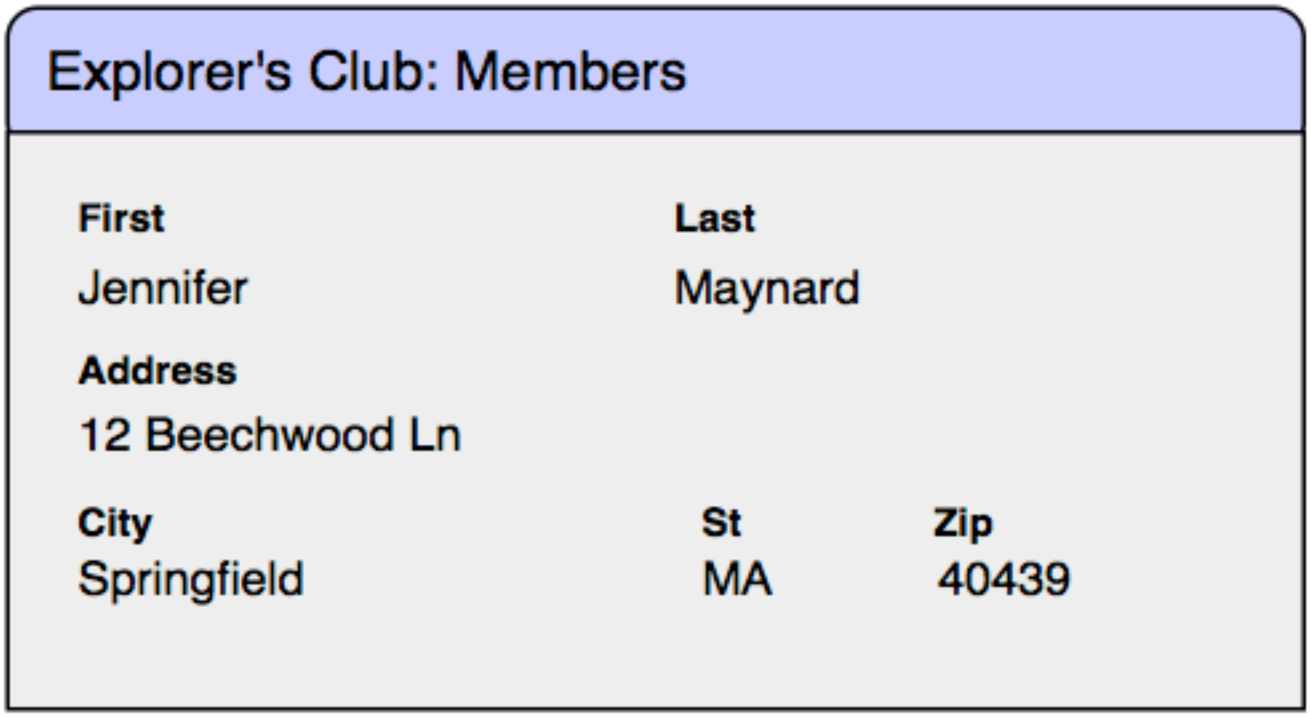
CommitteeID	CName
100	Exploration
101	Entertainment
102	Equipment
103	Prudential

We can create a nice data entry from for each table…. one for committees and one for members.



Explorer's Club: Committees

Committee: Exploration

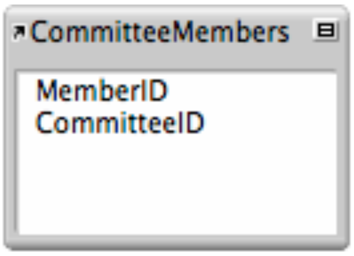

Explorer's Club: Members

First	Last	
Jennifer	Maynard	
Address		
12 Beechwood Ln		
City	St	Zip
Springfield	MA	40439

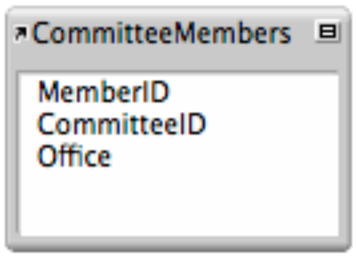
On the Committee form, there isn't any information other than the committee name. Obviously we'd like to see the list of the committee members, and their offices. Since a member can serve on multiple committees, we can't just enter a "committee" field on the member's record, as that would allow for only a single committee. Clearly, we can't just enter a single "member" field on the committee table either, as we expect to have more than a one member on a committee.

The solution to this problem is to create a <u>third</u> table with a purpose of holding information from our two existing tables. This is sometimes called a "join" table, because it joins records from the member and committee tables. Since this table "joins" two other table, We'll name the table by combining the names of its two parent tables, and call it CommiteeMembers.

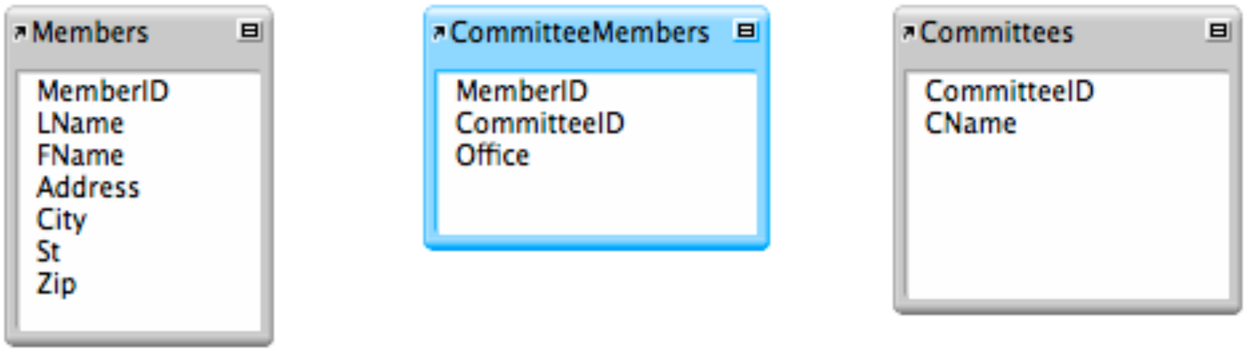
The join table will contain a key field from each of the two source tables. In this case, it will start with two fields, the key field copied from the members table (MemberID) and the key field copied from the committee table (CommitteeID).



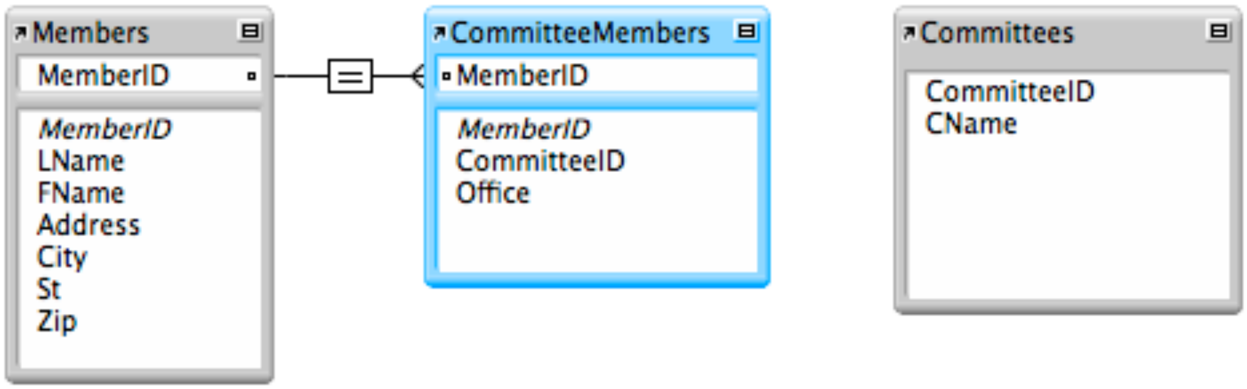
One further detail… We'd like to know the *office* that the member is holding on the committee. i.e. is the person the committee chair, vice-chair, member, etc. The office is unique to the <u>relationship</u>. A member can serve in a different office on different committees. So, we'll add an office field to our join table.

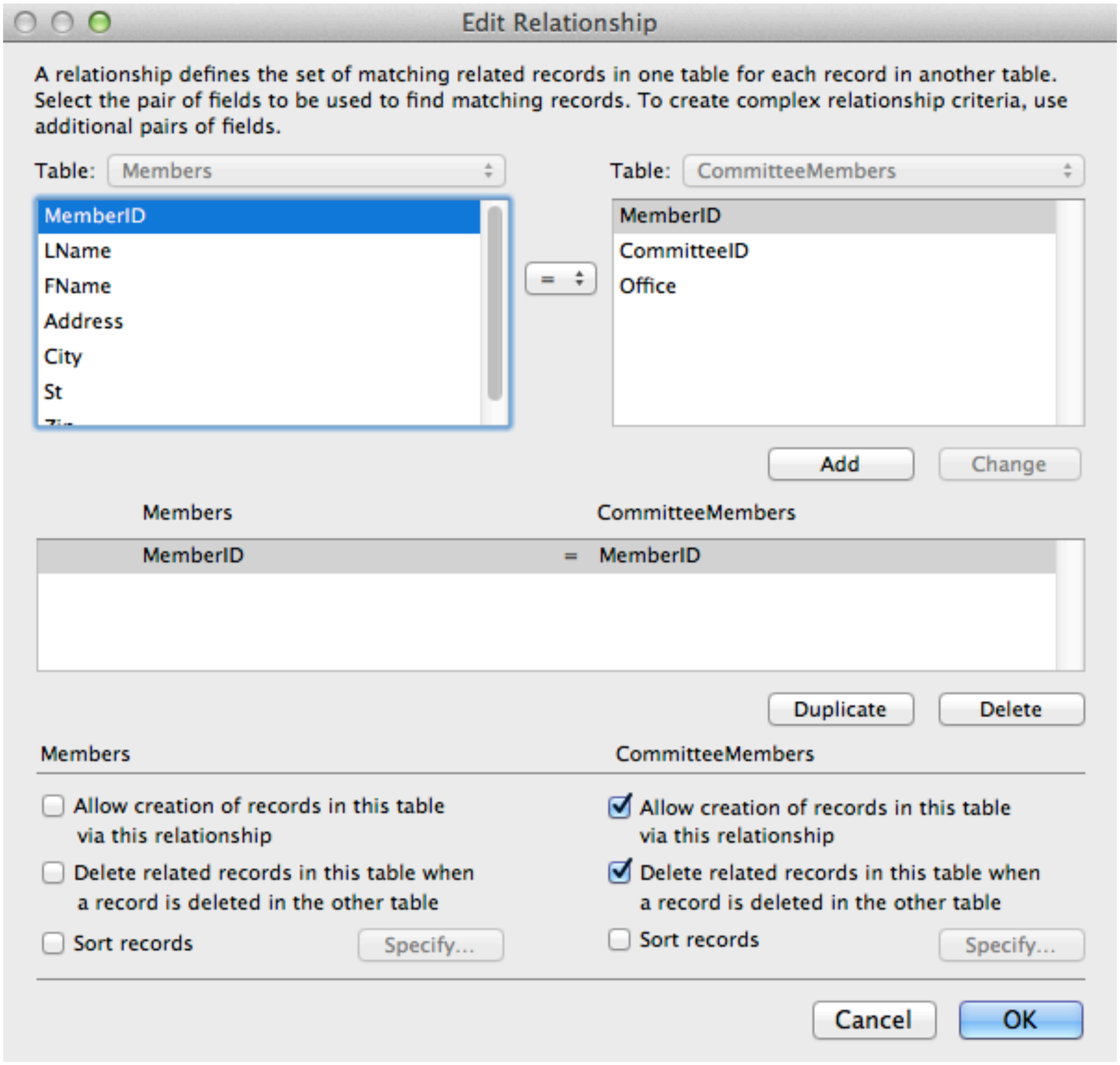


So now we have our three tables defined. We won't define forms or layouts for the join table as it will stay behind the scenes, and the user should never directly interact with it.



Now we'll establish the relationship between the members table and the join table. To do this, you click on the field in the source table Members, and draw a line to the corresponding field in the join table. This creates the one-to-many join between the two members. You can double-click on the box in the middle, to further refine this relationship.





A relationship defines the set of matching related records in one table for each record in another table. Select the pair of fields to be used to find matching records. To create complex relationship criteria, use additional pairs of fields.

Table: Members

- MemberID
- LName
- FName
- Address
- City
- St

Table: CommitteeMembers

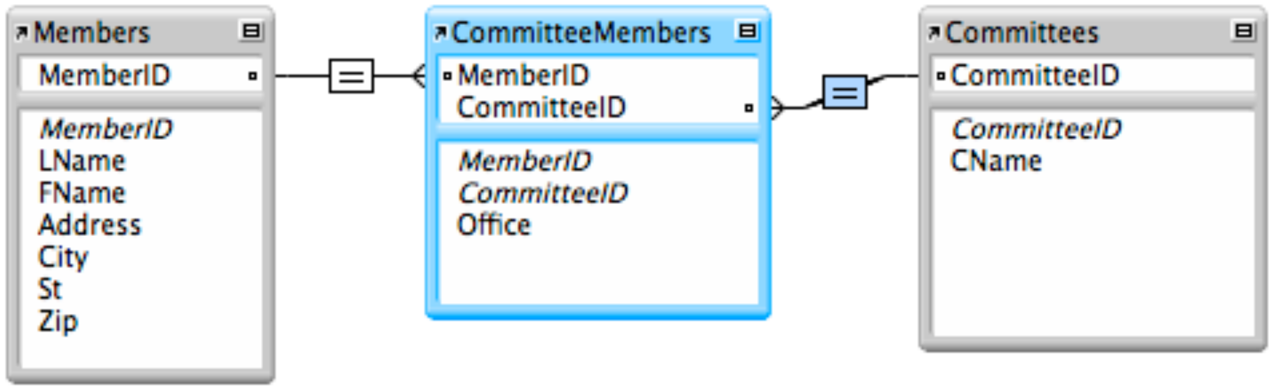
- MemberID
- CommitteeID
- Office

Members		CommitteeMembers
MemberID	=	MemberID

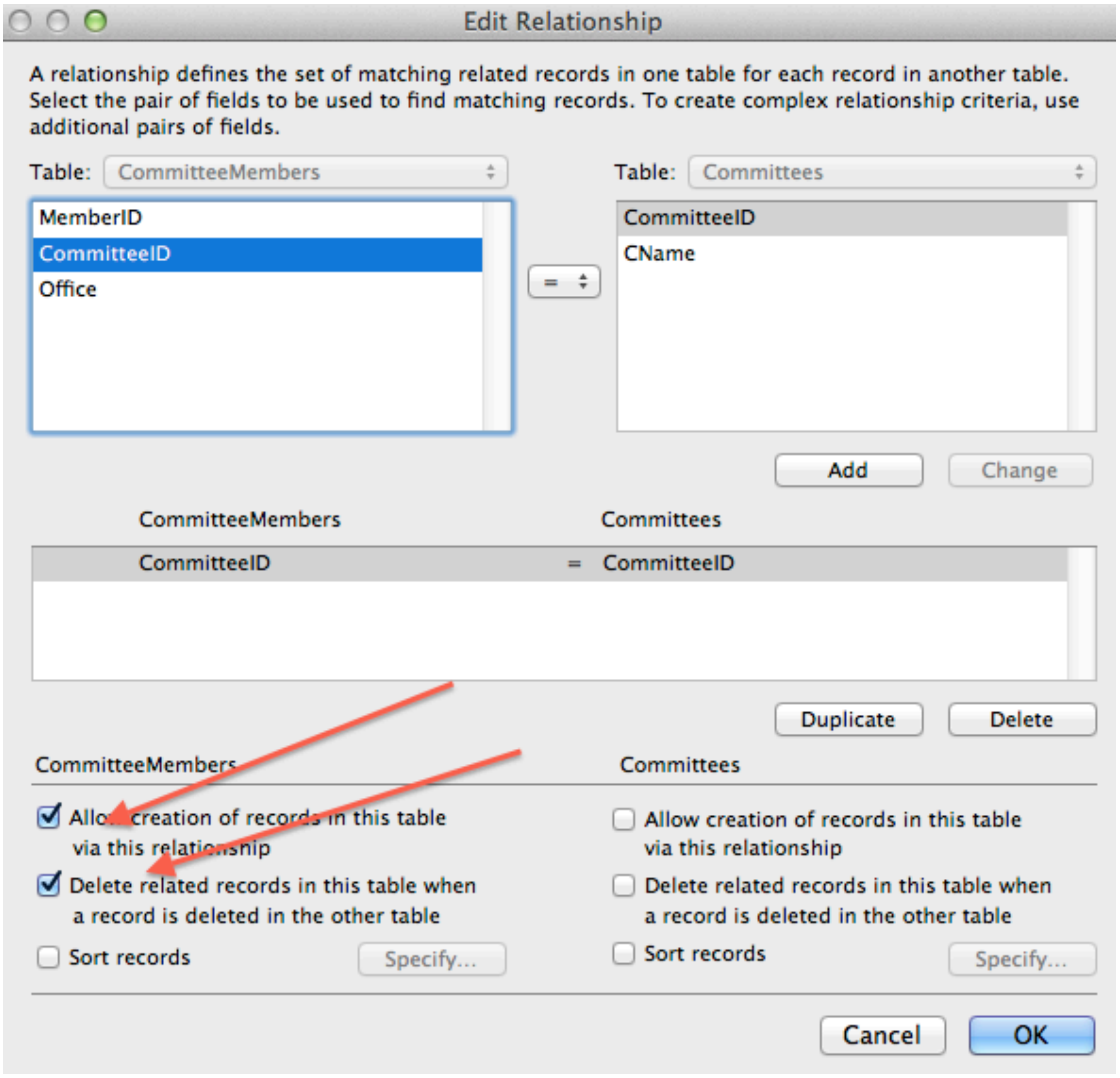
Members	CommitteeMembers
☐ Allow creation of records in this table via this relationship	☑ Allow creation of records in this table via this relationship
☐ Delete related records in this table when a record is deleted in the other table	☑ Delete related records in this table when a record is deleted in the other table
☐ Sort records	☐ Sort records

This rather confusing dialog box shows the relationship between the two tables. You need to check the boxes to allow new records to be created in the join table. This happens when you add a new member to a committee.

The process is repeated in the other direction with the Committee table.



Again, you have to check the boxes to allow records to be created in the join table from the committee side… whenever a new committee is added.


Edit Relationship

A relationship defines the set of matching related records in one table for each record in another table. Select the pair of fields to be used to find matching records. To create complex relationship criteria, use additional pairs of fields.

Table: CommitteeMembers — fields: MemberID, CommitteeID (selected), Office

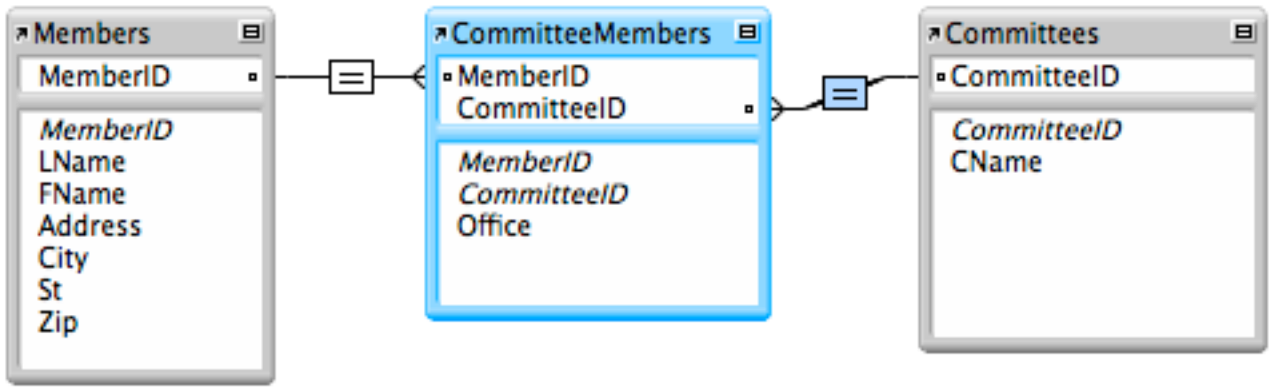
Table: Committees — fields: CommitteeID, CName

CommitteeMembers		Committees
CommitteeID	=	CommitteeID

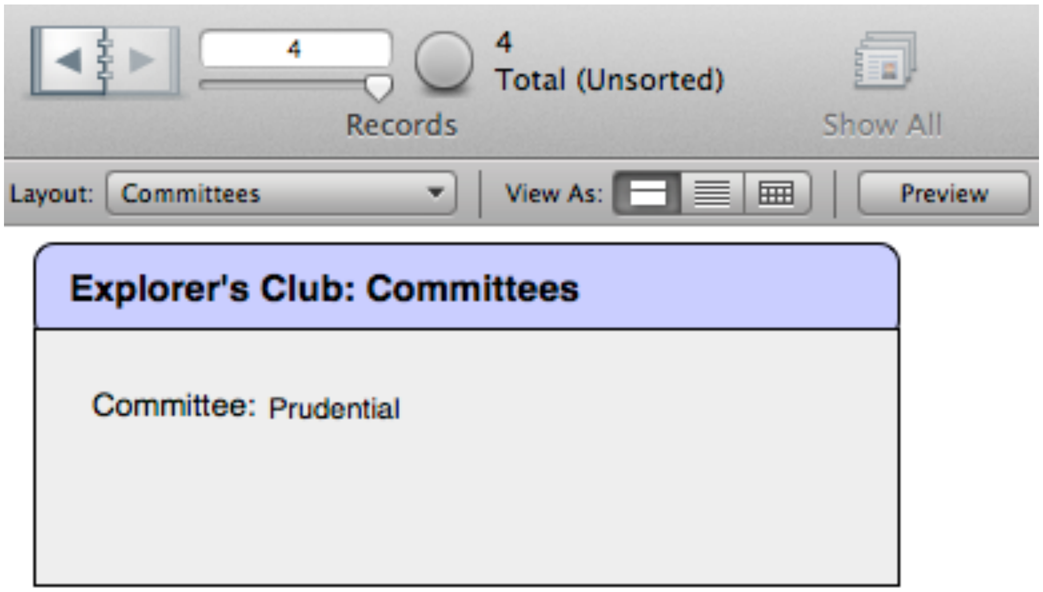
CommitteeMembers
- [✓] Allow creation of records in this table via this relationship
- [✓] Delete related records in this table when a record is deleted in the other table
- [ ] Sort records

Committees
- [ ] Allow creation of records in this table via this relationship
- [ ] Delete related records in this table when a record is deleted in the other table
- [ ] Sort records

In part one of this explanation, we created two tables, *members* and *committees* to keep track of the membership of the Explorers Club. We
also created a *join table* to keep allow us to assign members to various committees, and to allow a single member to join more than one committee. This is a classic example of the "Many to Many" relationship that is found database systems.



We created two forms (called "layouts" in FileMaker Pro parlance), a contact form for the members, and a committee form.
Now we are going to modify the committee form to allow us to assign members from the members table to committees.

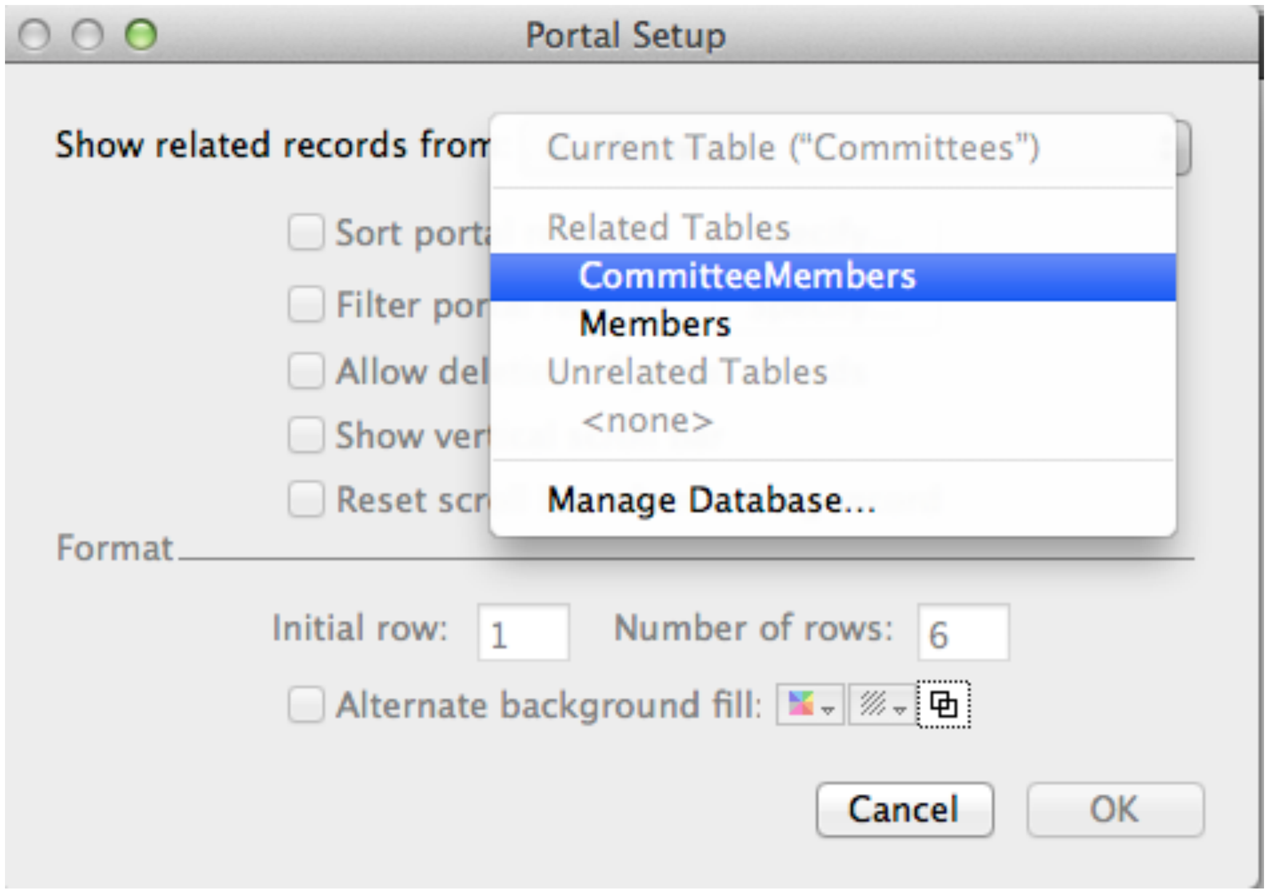


Explorer's Club: Committees

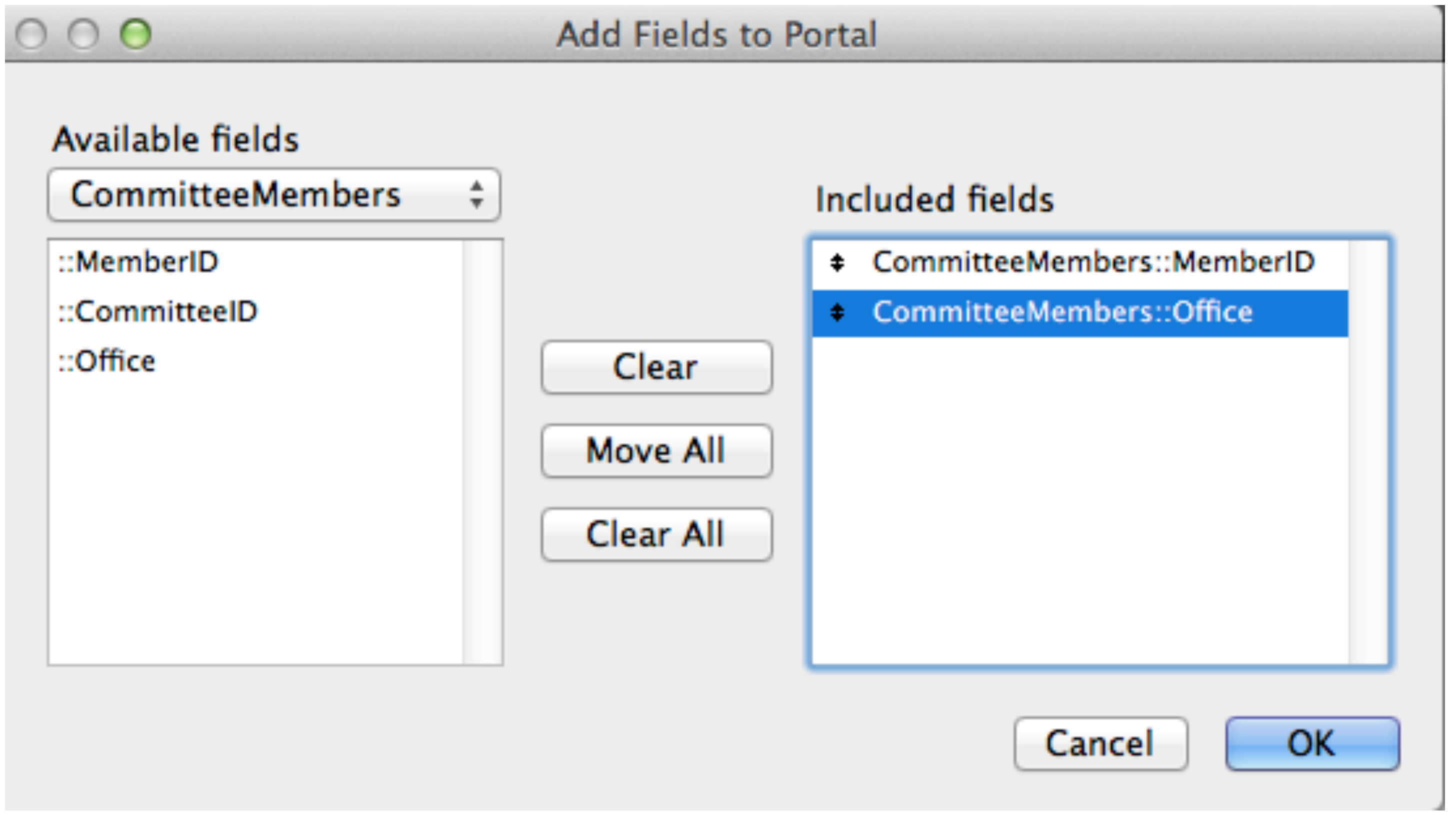
Committee: Prudential

At the moment, our form contains a single field, the field for committee name or cname.

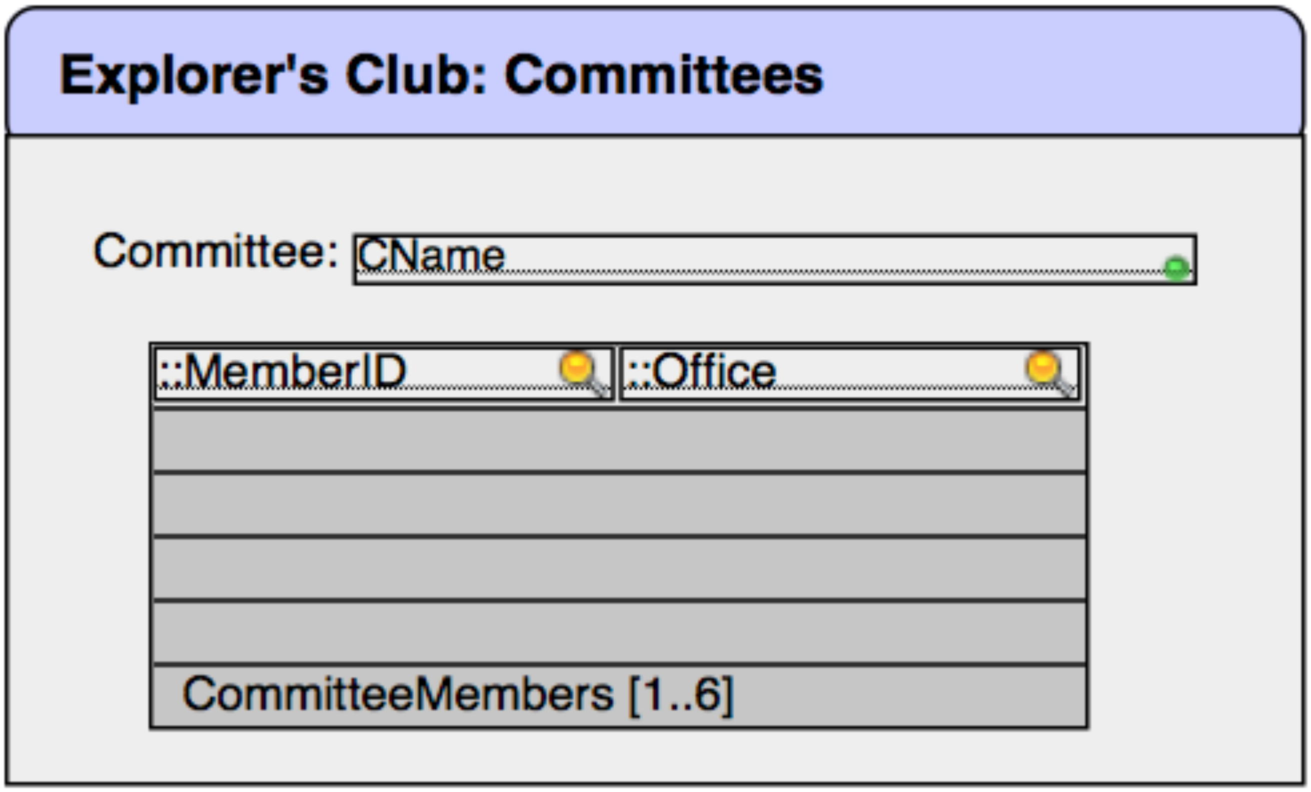
1. Place the portal on the form. In the portal setup dialog, choose CommitteeMembers (the join table) for the portal.



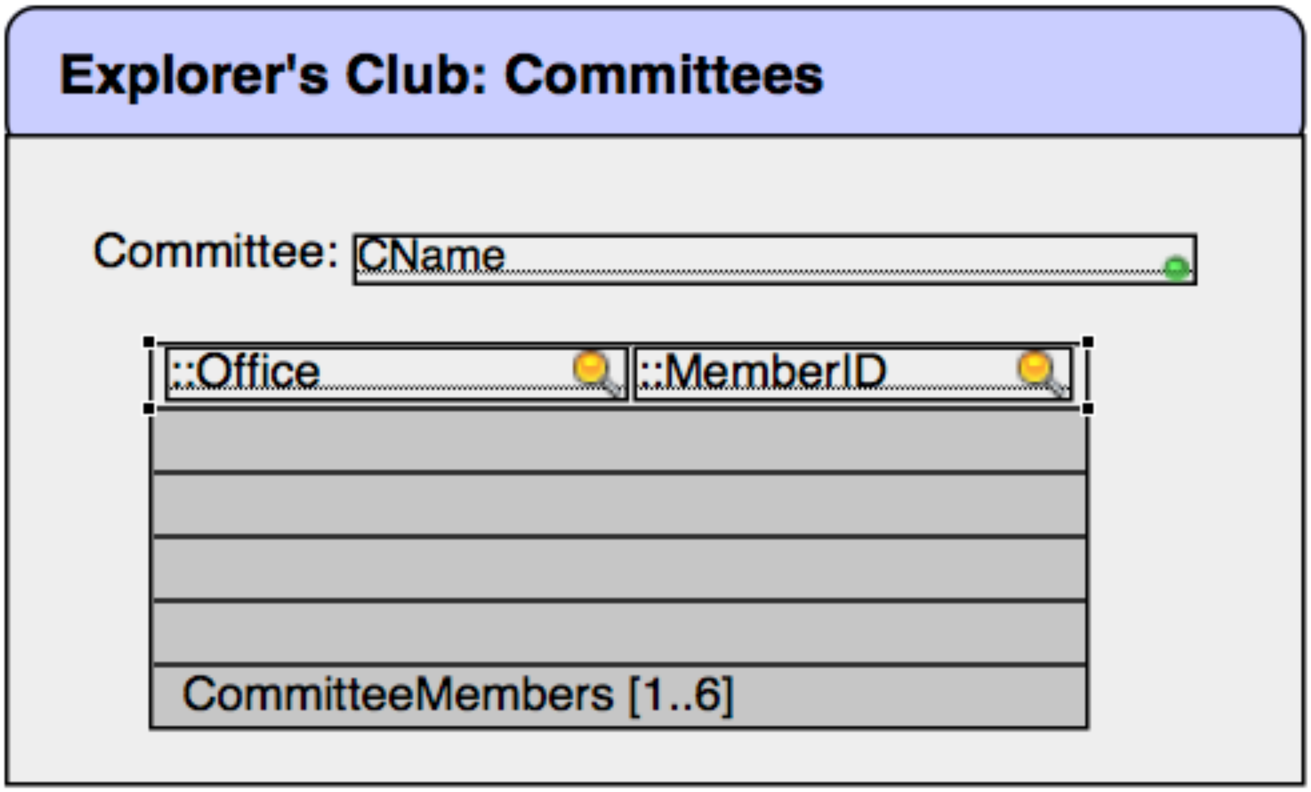
In the next dialog choose two fields: MemberID, and Office to be displayed in the portal.



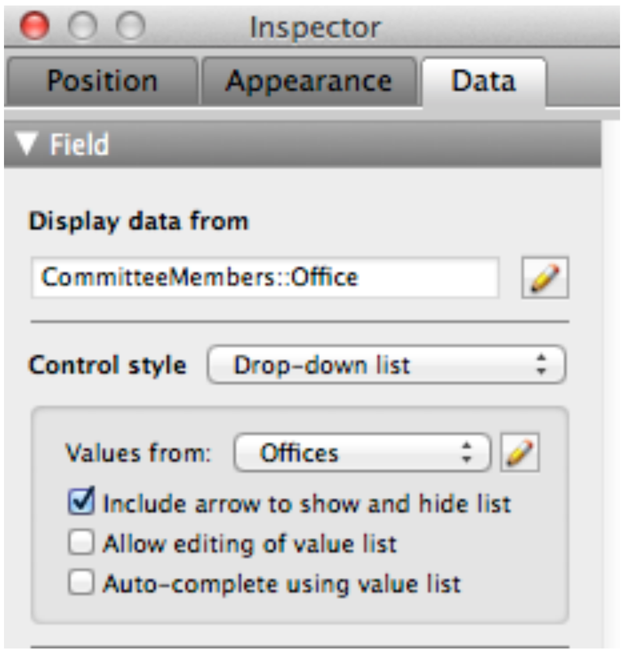
Since this portal is on the Committees form, you don't need to include he CommitteeID field in the portal, because you are *adding members to a committee* and the CommitteeID is already going to be part of any new record created within the portal. (Read that sentence again, if it is confusing… it took me awhile to grasp.)



You should see that the layout looks similar to the picture above.
Since I prefer to see the office listed first, and then the member's name, I'll reverse these in the layout.



Committee members can hold one of five positions on a committee… Chair, Vice-Chair, member or president. I want these to be made available from a drop-down list.
So I'll change the field type for the office from an edit box to a drop down listand include the arrow on the end which allows me to click to choose from the list.
These changes are made by clicking on the field itself, and modifying the inspector settings.



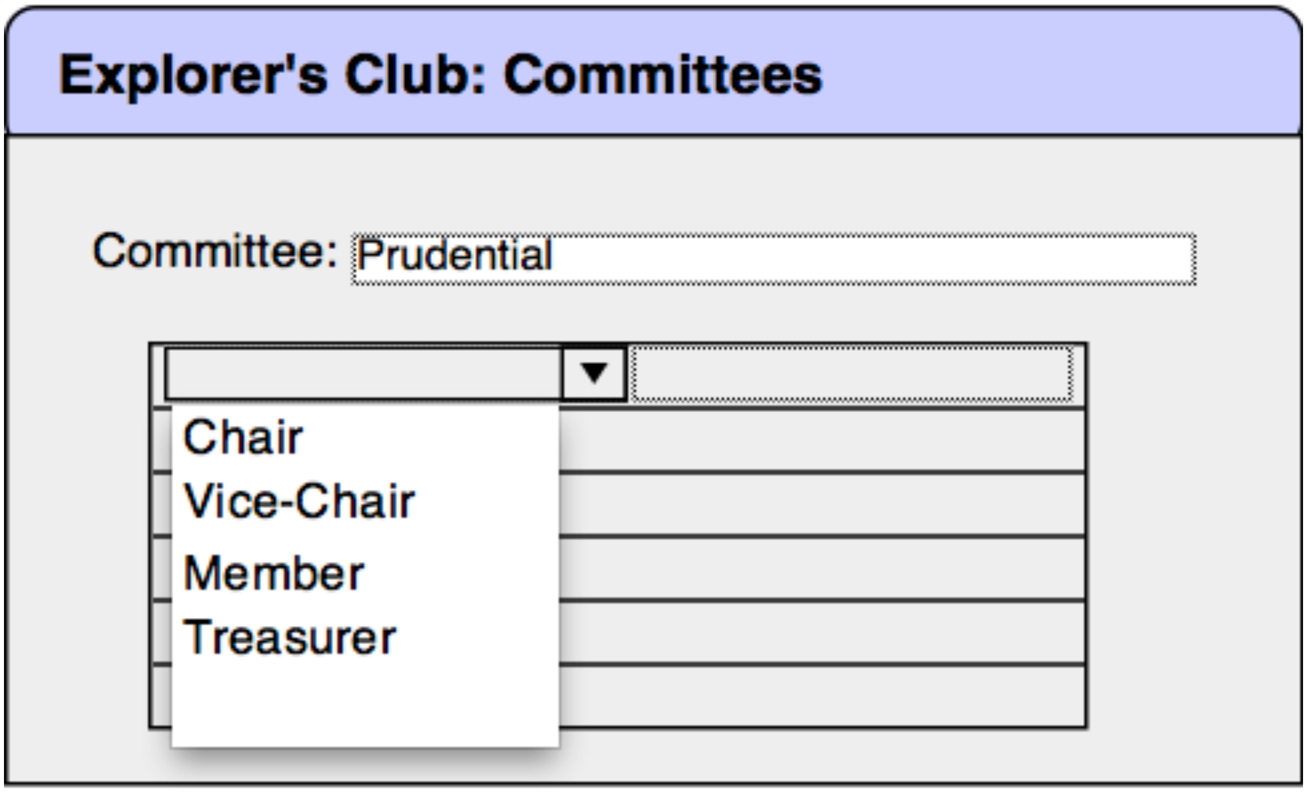
Display data from: CommitteeMembers::Office

Control style: Drop-down list

Values from: Offices

[✓] Include arrow to show and hide list
[ ] Allow editing of value list
[ ] Auto-complete using value list

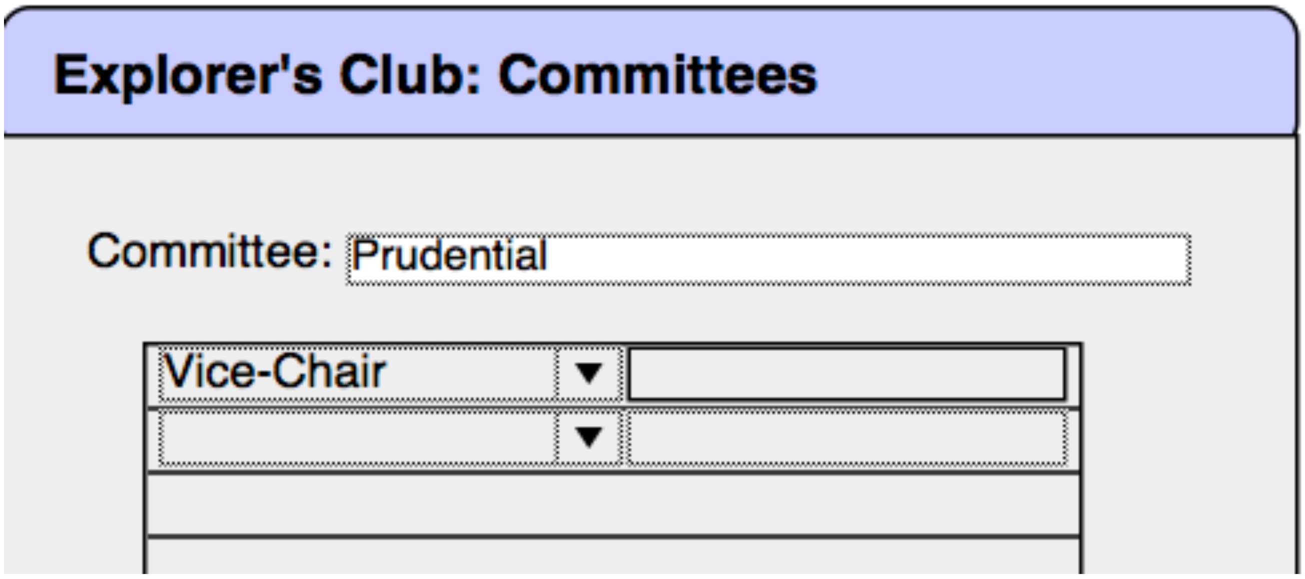
Testing this out, it looks promising. I can now choose an office.



Explorer's Club: Committees

Committee: Prudential

And here is how it looks after I've made my choice.



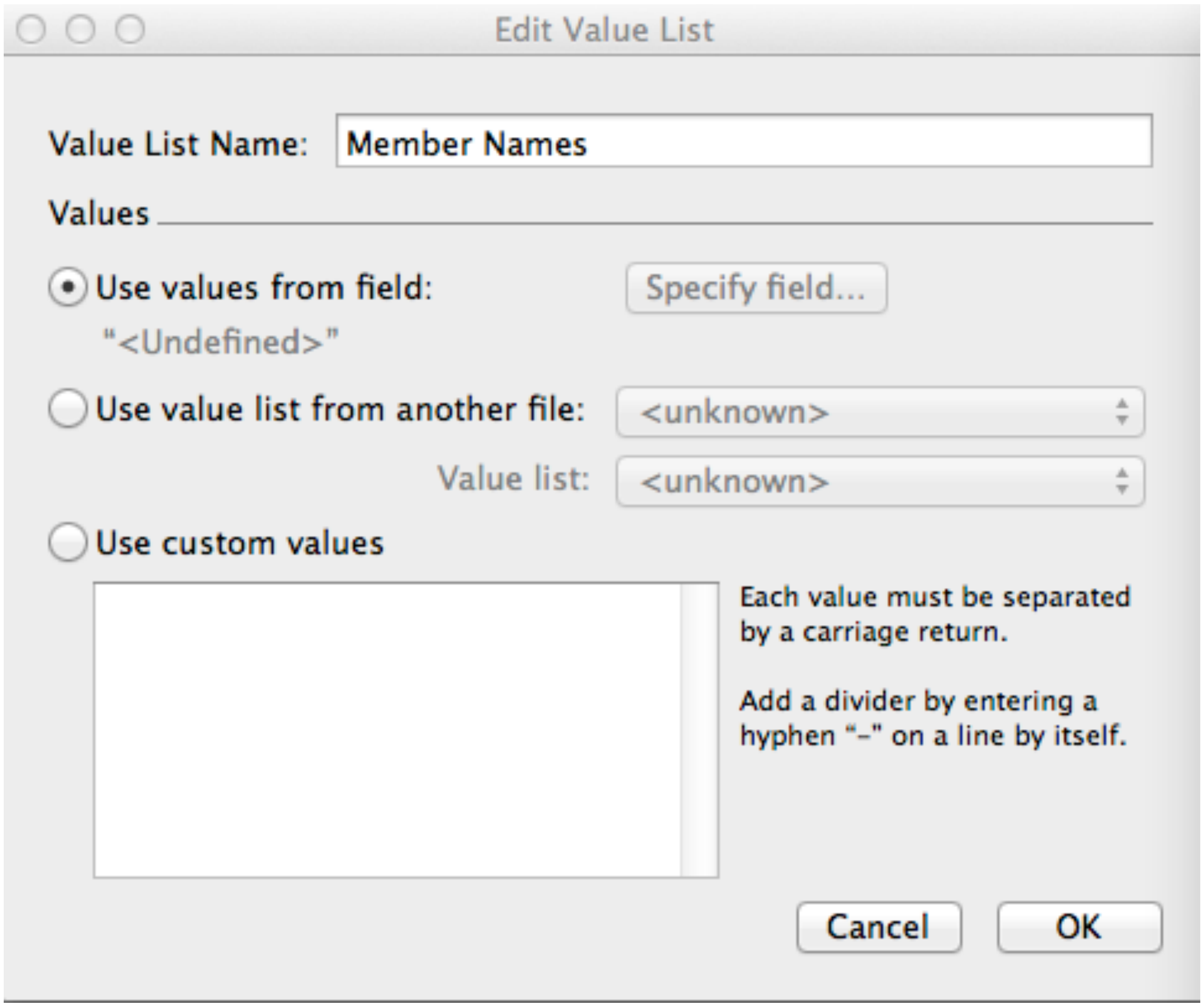
Explorer's Club: Committees

Committee: Prudential

Vice-Chair

Now we have to deal with choosing the member's name for the position.

The first thing we'll do is modify the field for MemberID in the inspector, and turn it into a drop down list as well.
When we change the field to a drop down list, we have to say where we're going to get the values for the field. These will come from the Members data file.
Create a new value list


Edit Value List

Value List Name: Member Names

Values

[✓ Use values from field:] "<Undefined>"

Use value list from another file: <unknown>

Value list: <unknown>

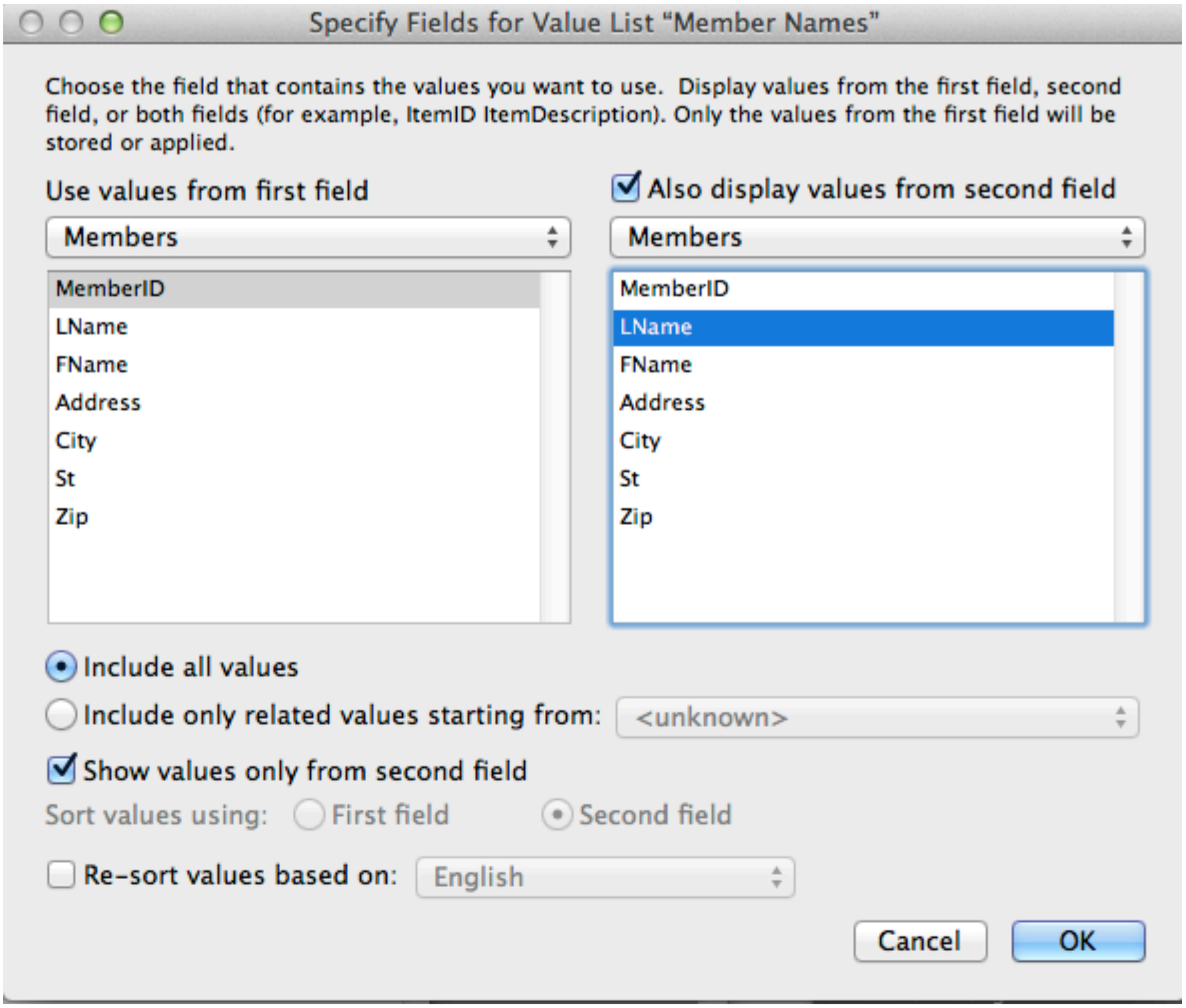
Use custom values

Each value must be separated by a carriage return.

Add a divider by entering a hyphen "-" on a line by itself.

Give it a name, and choose the second option…. 'Use values from the field…

Click on Specify field


Specify Fields for Value List "Member Names"

Choose the field that contains the values you want to use. Display values from the first field, second field, or both fields (for example, ItemID ItemDescription). Only the values from the first field will be stored or applied.

Use values from first field: Members

- [✓ MemberID]
- LName
- FName
- Address
- City
- St
- Zip

✓ Also display values from second field: Members

- MemberID
- [✓ LName]
- FName
- Address
- City
- St
- Zip

[✓ Include all values] Include only related values starting from: <unknown>

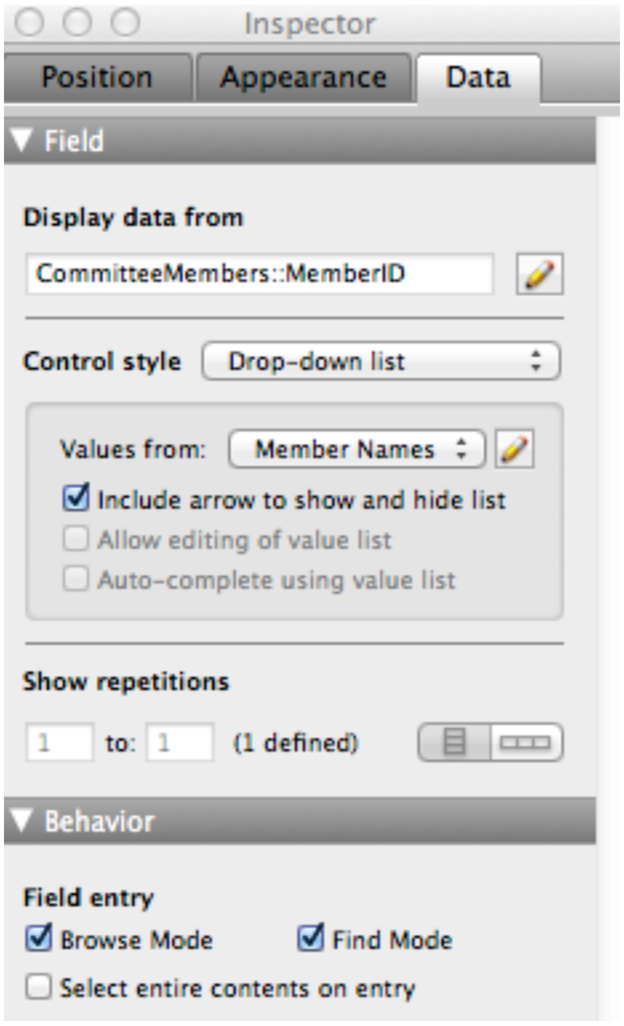
✓ Show values only from second field

Sort values using: First field [✓ Second field]

☐ Re-sort values based on: English

This allows you to choose the contents of one field, based on the contents of another field in the same record.
Since MemberIDs aren't going to be very intuitive….we want to choose instead by last name. Using the settings shown above, we will see a drop down list last names, but the program will actually enter the MemberID.

So, the settings show that we're going to insert a MemberID into our new record, but we will choose which one by looking at the member's last name.



Display data from: CommitteeMembers::MemberID

Control style: Drop-down list

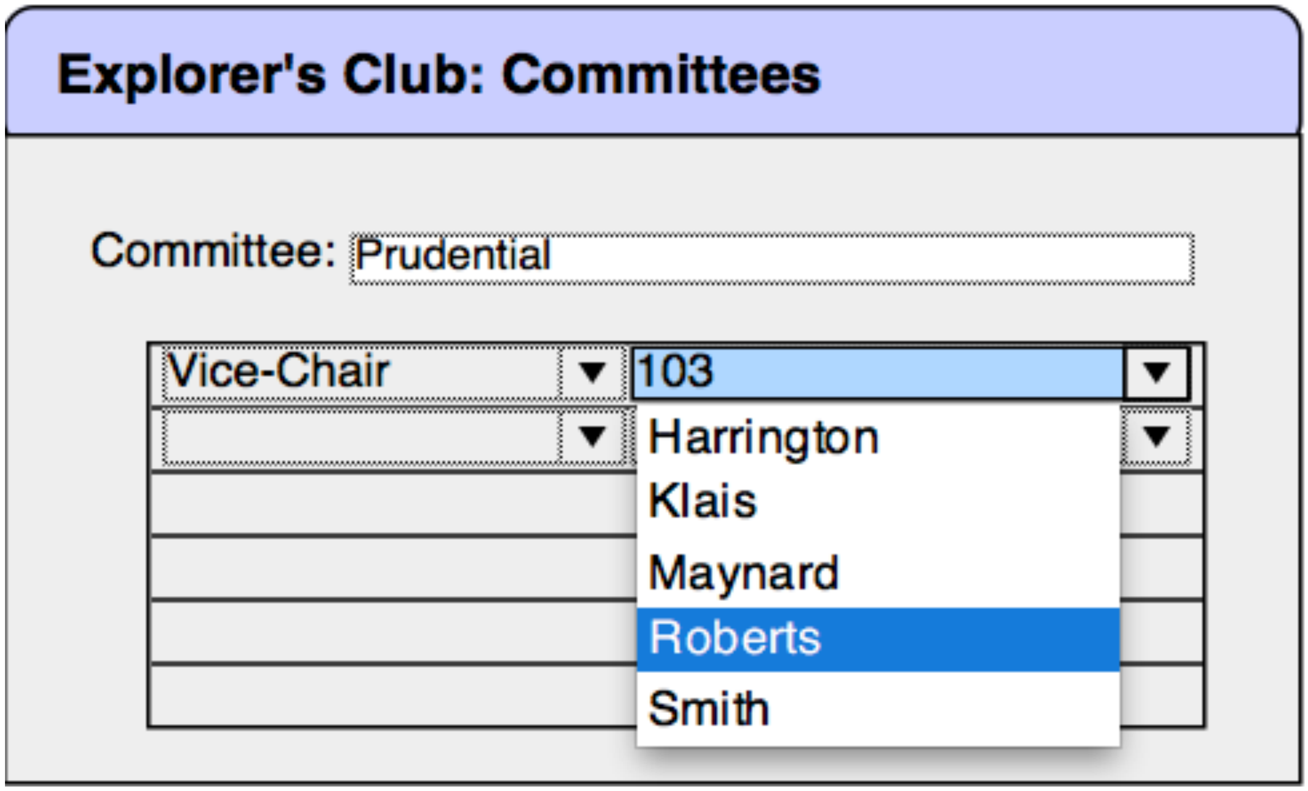
Values from: Member Names

[✓] Include arrow to show and hide list
[ ] Allow editing of value list
[ ] Auto-complete using value list

Show repetitions: 1 to: 1 (1 defined)

Field entry
[✓] Browse Mode [✓] Find Mode
[ ] Select entire contents on entry

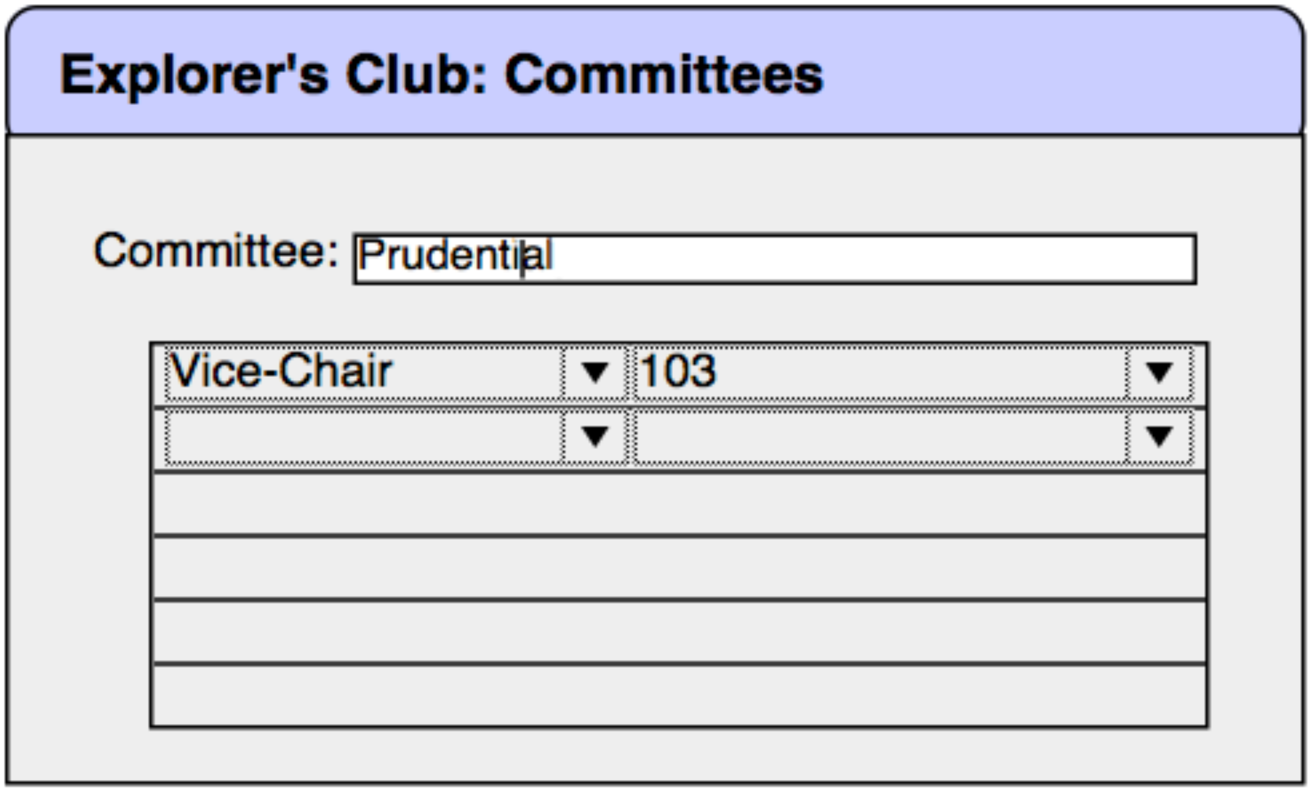
This works fine, as far as it goes….. but, wait…it only ends up displaying the MemberID!


Explorer's Club: Committees

Committee: Prudential

Vice-Chair	103

Once the choice is made… you see….

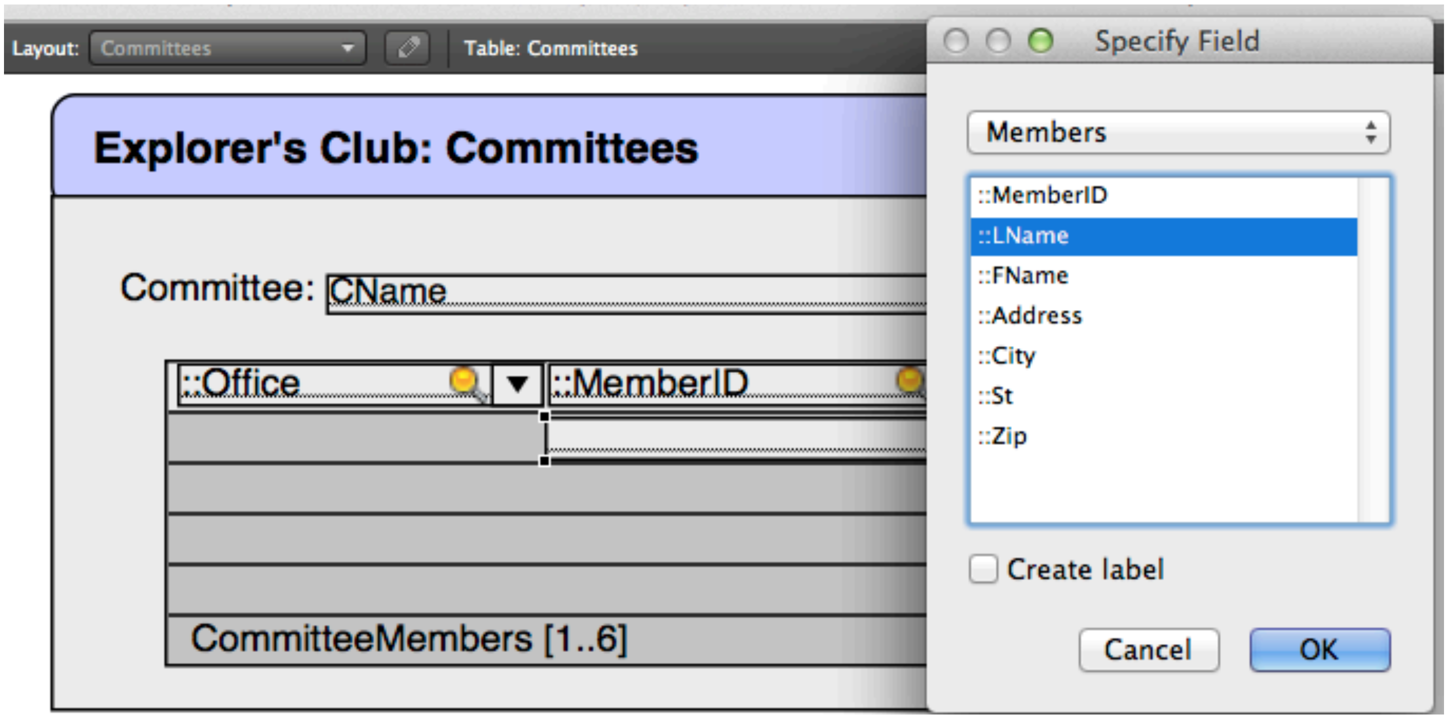

Explorer's Club: Committees

Committee: Prudential

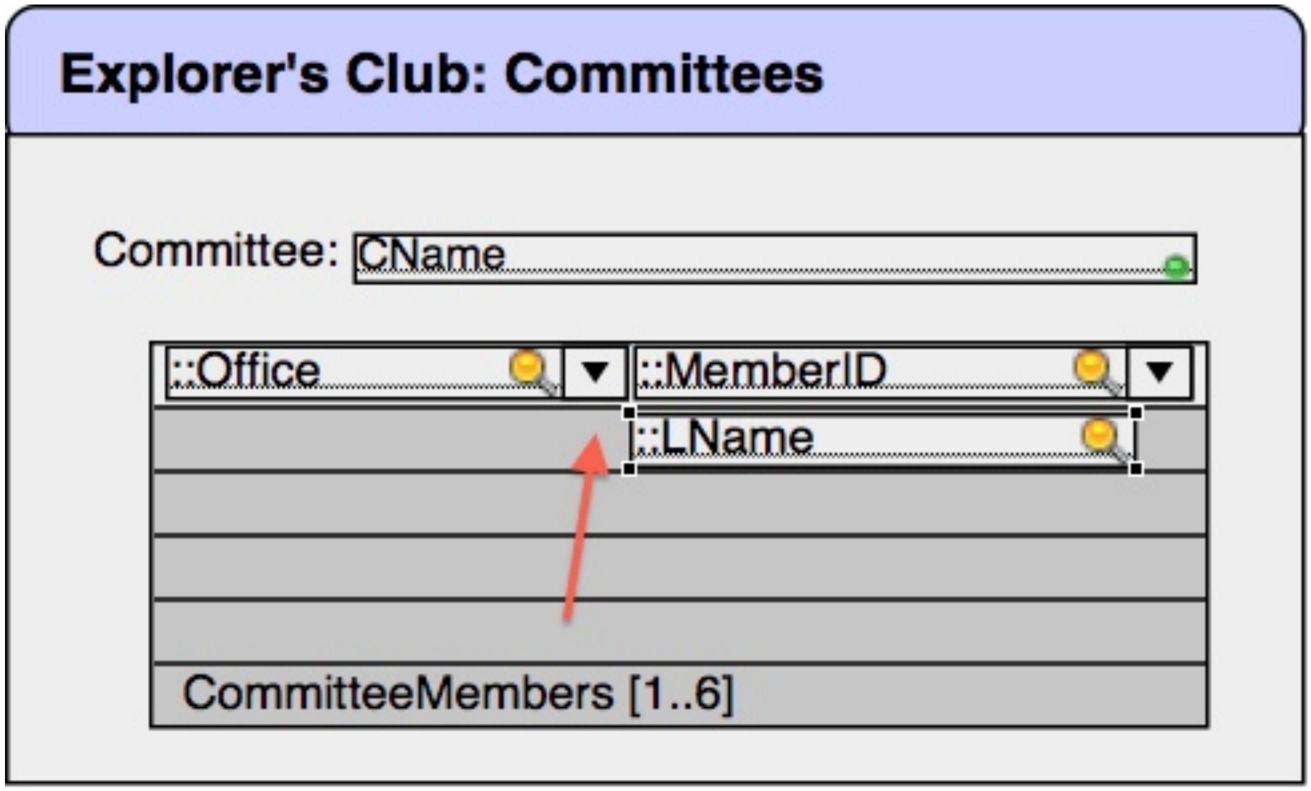
Vice-Chair	103

Next comes the magical part, for which I'm grateful from suggestions and help from the denizens of the FileMaker TechNet online forum.

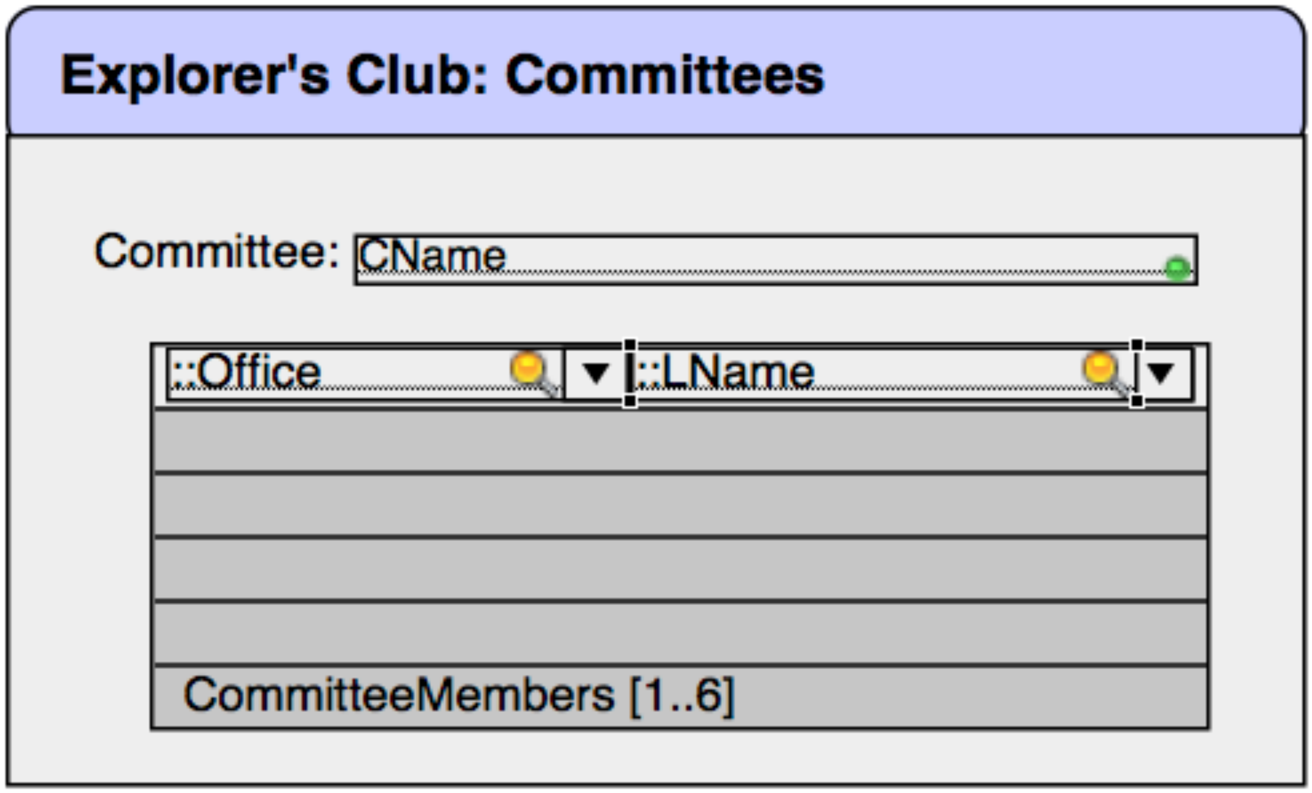
Add a new field, but this time choose the field from the *Members* table.



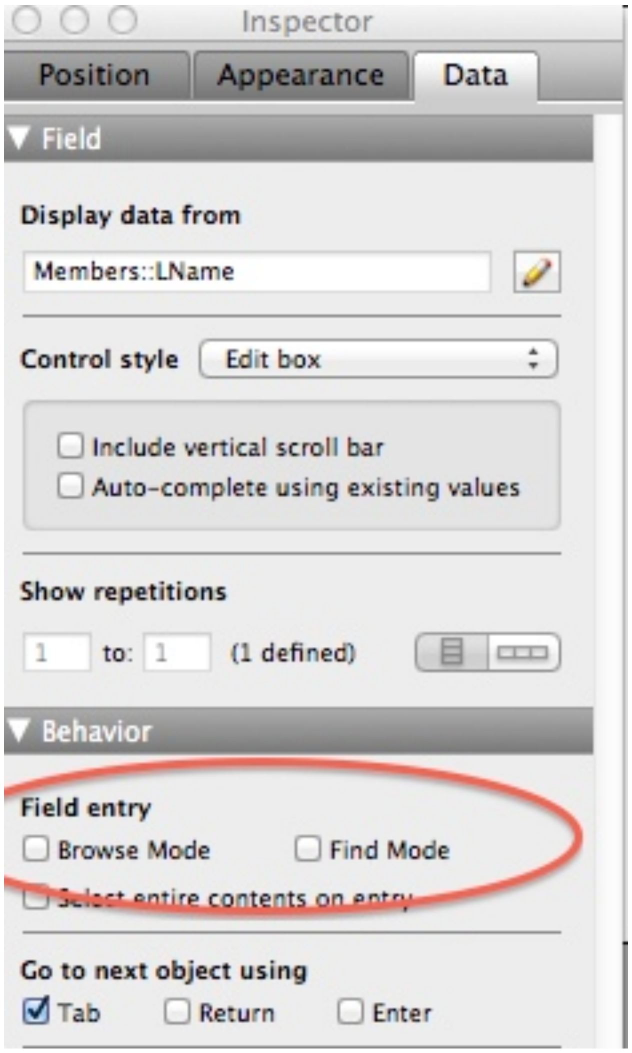
Take the new field and *slide it on top of* the MemberID field.



I position to last name field directly on top, but adjust the length to show the drop-down arrow from the underlying field.



Making sure that you have still selected the LNAME field after positioning it, go to the Inspector, and "turn off" Browse and Find in the Behavior panel.



Display data from

Members::LName

Control style: Edit box

☐ Include vertical scroll bar
☐ Auto-complete using existing values

Show repetitions

1 to: 1 (1 defined)

Field entry

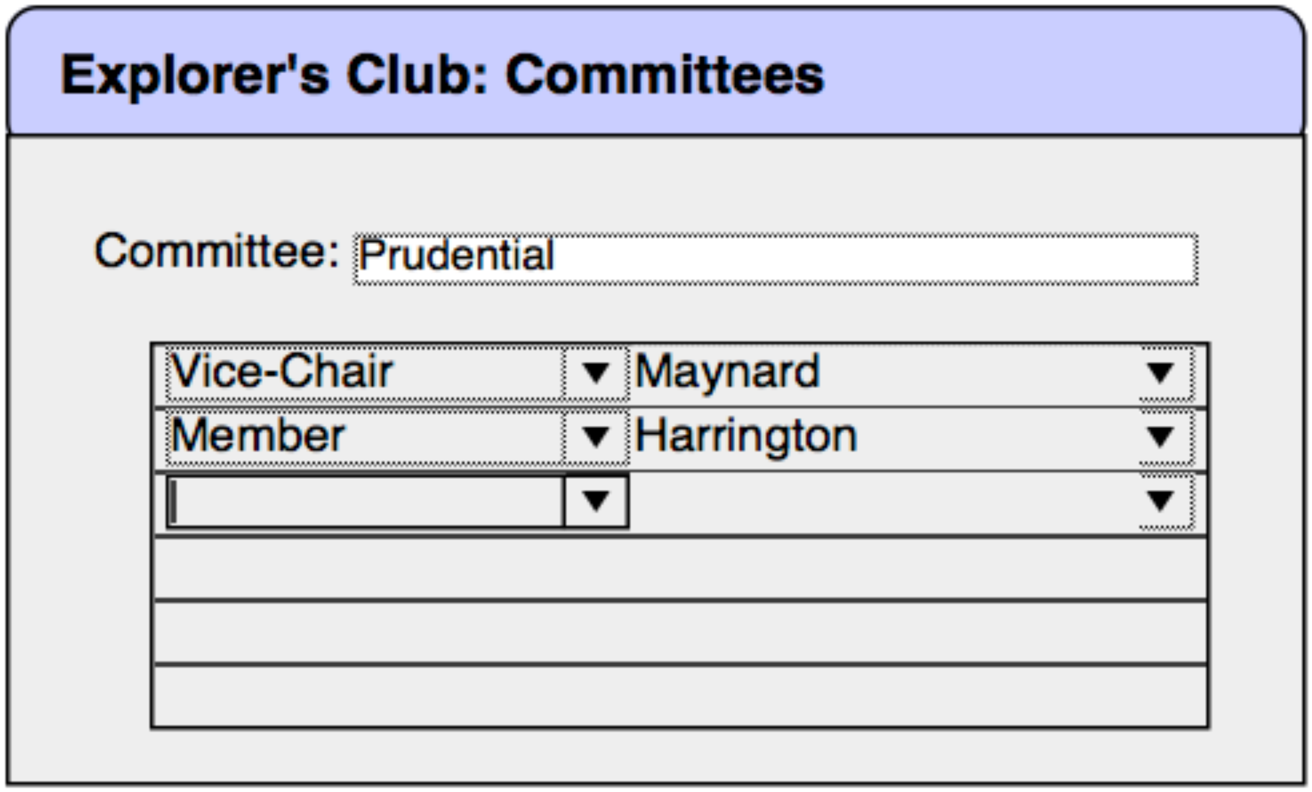
☐ Browse Mode ☐ Find Mode

☐ Select entire contents on entry

Go to next object using

☑ Tab ☐ Return ☐ Enter

This now means that when you click on the stacked fields, you will activate the bottom field, (the drop-down list). However, once the selection is made, you will see the person's name displayed from the top field.



Explorer's Club: Committees

Committee: Prudential

Vice-Chair	Maynard
Member	Harrington

Two Refinements.

1. Create a calculated field within the member table that concatenates the first name and last name. Call this FullNameFL, and use that as the display field instead of just the last name.

Use this for a calculation
("FirstName & \" \" & LastName" ; [FirstName ; LastName])

2. Create a calculated field FullNameLF within the member table that reverses this, and concatenates last name + first name. This allows you to search by last name, and even type in the first few letters to move the drop-down list quickly.

Use this for a calculation
("LastName & \" \" & FirstName" ; [LastName ; FirstName])