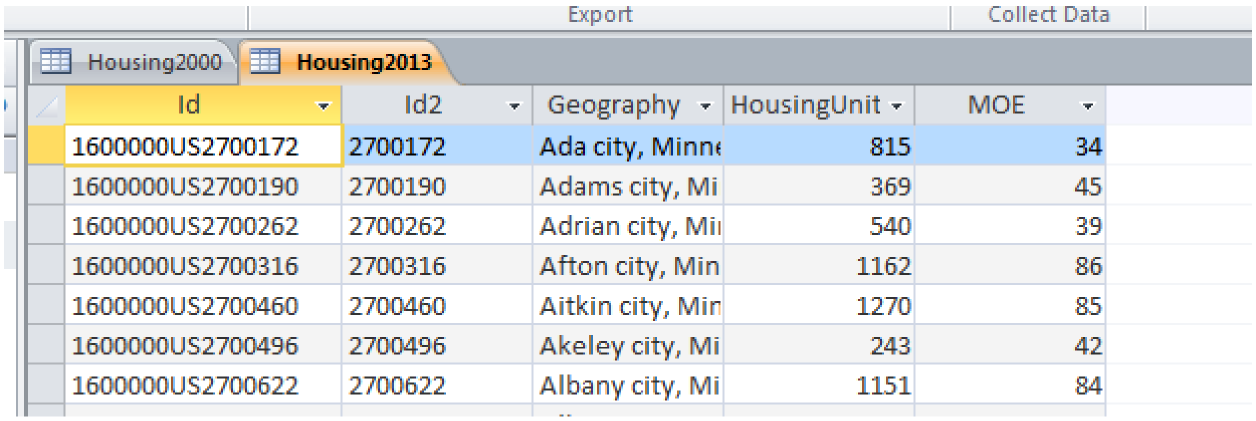
Basic Access joining exercise
Using Census data on housing units, by place
Created by @MaryJoWebster
August 2015

The goal of this exercise is to introduce how joining tables works in Microsoft Access. We're going to use two tables from the Census Bureau with the number of housing units in each "place" (incorporated cities) in Minnesota – one has data from 2000 and the other has data from 2013. I want to put them in the same file so it's easier to calculate the percentage change in housing units between the two points in time. Doing this kind of matching in Access is much easier than trying to line up two sets of data in Excel, especially in this situation because there are "places" that are in one of the datasets, but not in the other.

We're going to start with two .csv files that need to be imported to Access – housing2000.csv and housing2013.csv

Create a new Access database and then go to the External Data tab and choose "text", then follow the wizard to import the first file. Repeat to import the second file.

Let's look at the data. Both tables have a field called "geography," where it lists the name and the state; Both also have a field for the number of housing units. The 2013 data also has a MOE (Margin of error) field, because this data came from the American Community Survey 5-year data. The 2000 data is from the decennial census that year.



Id	Id2	Geography	HousingUnit	MOE
1600000US2700172	2700172	Ada city, Minne	815	34
1600000US2700190	2700190	Adams city, Mi	369	45
1600000US2700262	2700262	Adrian city, Mi	540	39
1600000US2700316	2700316	Afton city, Min	1162	86
1600000US2700460	2700460	Aitkin city, Mir	1270	85
1600000US2700496	2700496	Akeley city, Mi	243	42
1600000US2700622	2700622	Albany city, Mi	1151	84

You'll see there are two ID fields in these tables. They are basically the same – the first, though, has "1600000US" tacked on to the front. These are FIPS codes – 27 is for the state of Minnesota and the remaining five digits are the "place" FIPS codes.

How joining works:

Joining two tables together means that we want to line them up side-by-side . In this case, we want the 2000 data for Minneapolis to be on the same row as the 2013 data, then the same for St. Paul, and Bloomington and Duluth and on and on.

Some people confuse joining with “appending.” Appending is when you want to add more “rows” or records below an existing dataset.

Joining means we are essentially adding more “columns” to a table.

There are a couple circumstances when joining is necessary. The first is what we’re doing here. We have two datasets that came to us in separate files, but we want the information to be together.

The second is when you have a “relational” database where the data is stored in multiple tables.

For example, a dataset that is commonly stored as a relational database is campaign finance records. Typically there is one table that has one record for the committee/candidate, with their basic info (address, party affiliation, position they are running for, etc). And then there is another table with one record for each donation that each of those committees received, but usually that table only contains the ID number of the committee. You need to join the two tables to line up the name of the committee/candidate receiving the money with the name and other info of the person/committee giving the money.

Most of the time, you’re going to be joining tables that were intended to go together. So for example, the campaign finance data has an ID number for the committee and that field shows up in both tables. We just need to tell Access to “match” the two tables using those fields.

In some rare circumstances you might want to do what’s called an “enterprise join” – where you are matching two tables that weren’t intended to go together, but have some fields that contain the same information. For example, you might have one table with names of school bus drivers and another table with names of people convicted of felonies and you want to see if the school district is failing its job of backgrounding new drivers. That’s more complex and we’ll save that for another tutorial.

Let’s get back to this Census data….

The first thing you need to figure out is what field(s) “match” between the two tables.

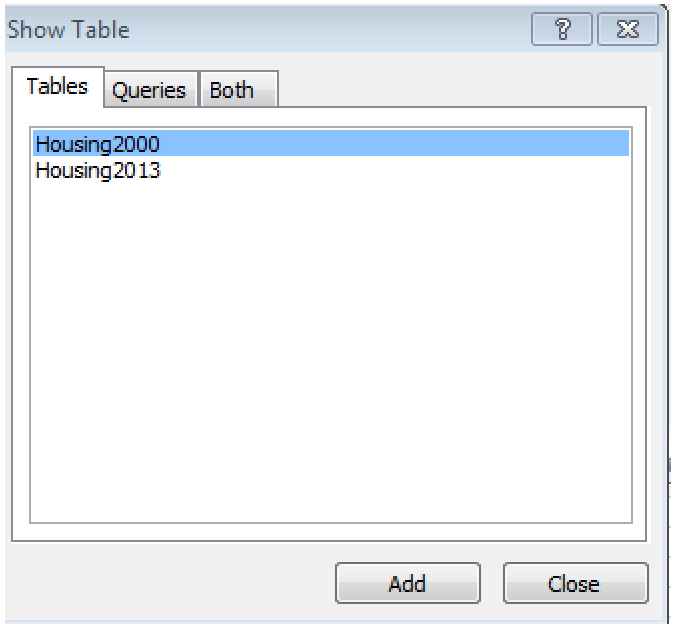
Your first inclination would probably be to join on the names, right? Seems like that would be the simplest way to go, but it’s not.

Names tend to change and they offer more opportunity for error (one little typo in the spelling, for example), so it’s far more likely that we’ll have trouble matching. The problem is that Access (and any data program, for that matter) cannot discern that “Minneapolis” and “Mpls” are the same thing or that “Minneapolis, Minnesota” and “Minneapolis, MN” are the same thing. The program wants it to be exactly the same – down to the punctuation and spaces.

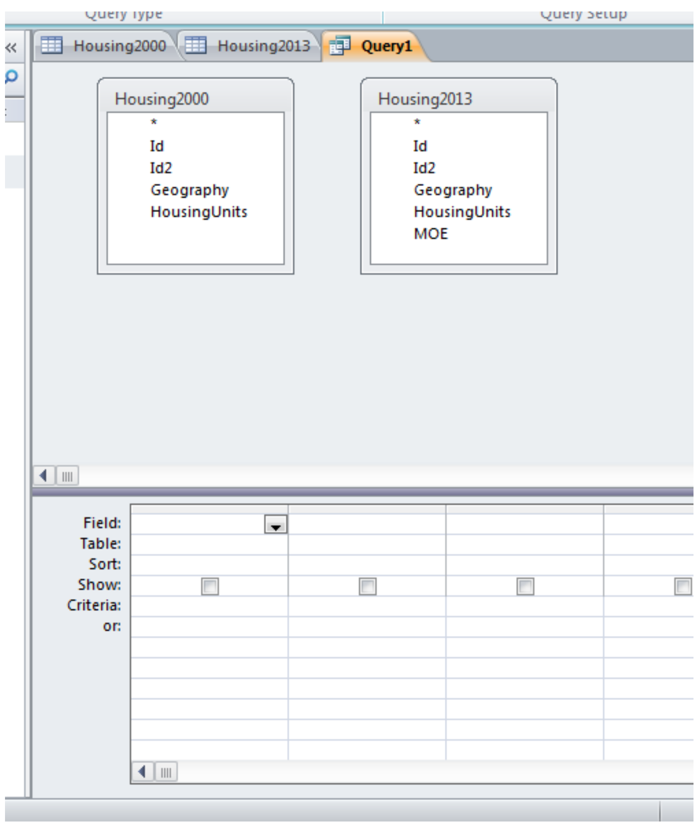
So the best way to join two tables is if you have ID numbers that are intended to be the same from one table to the next. FIPS codes are one of many examples out there of "codes" that show up in datasets for this very purpose—so you can match two datasets together.

As we mentioned above, there are two ID fields that are basically the same. For simplicity sake, I'm going to use "ID2". Conveniently, this field is named "ID2" in both tables (it doesn't always work that way and you <u>don't</u> need to have matching field names in order to join!)

Go to the Create ribbon and choose "Query design". In the Show Table box that comes up, click on Housing2000 and click Add. Then click on Housing2013 and click Add. Then close the Show Table box

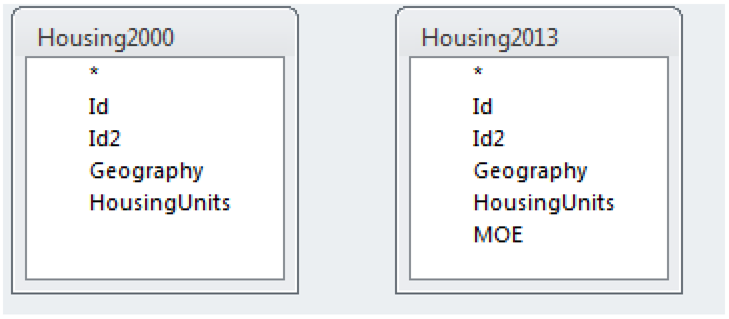


So now you'll have both tables in the Grid/Design view of the query, like this:

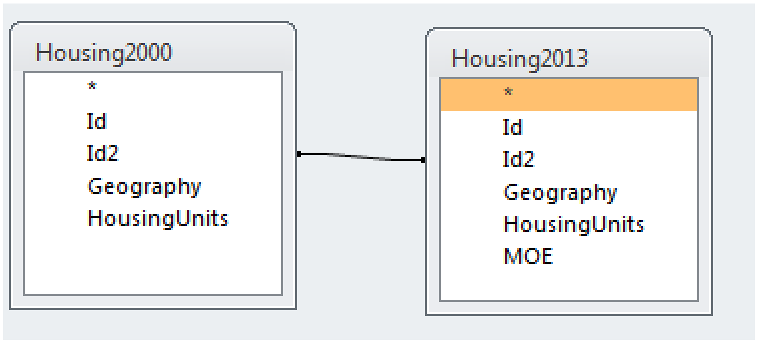


This is one of the rare circumstances where I PREFER the Design View/Grid View versus writing the SQL for a query. Joining tables in the design view is slick and easy.

In your query, you see both tables and the list of fields in each.



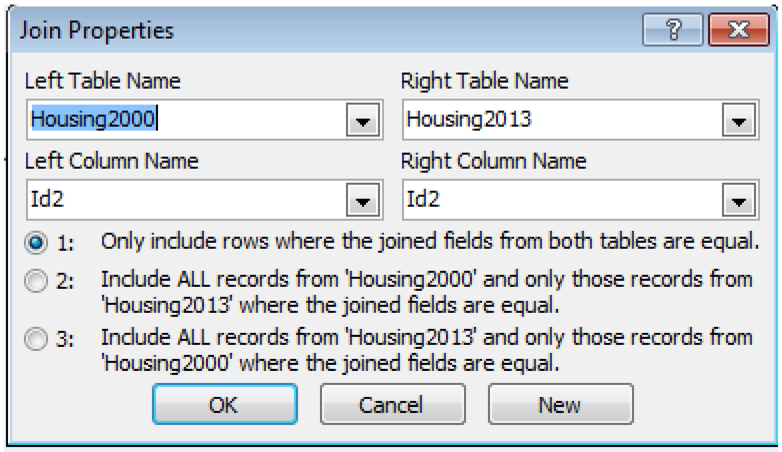
Click on "ID2" in one of the lists (doesn't matter which one), hold down the left-house key and drag across to the "ID2" field in the other table. When your cursor is hovering over the ID2 field, let go of the mouse and it will create a line between the two tables.



You can right-mouse click on the line (sometimes you have to drag the tables apart from each other – making the line longer – in order for this to work) and click "Join Properties" and it will show you the details of your join.

It shows the table names and the column names that it matched (yeah, we did it right cause it's "ID2" in both tables!)

It also gives you three options for the type of join. The default (#1) is what's known as an "Inner Join." This means that the results you're going to get back are ONLY records where a "mate" is found.


Join Properties

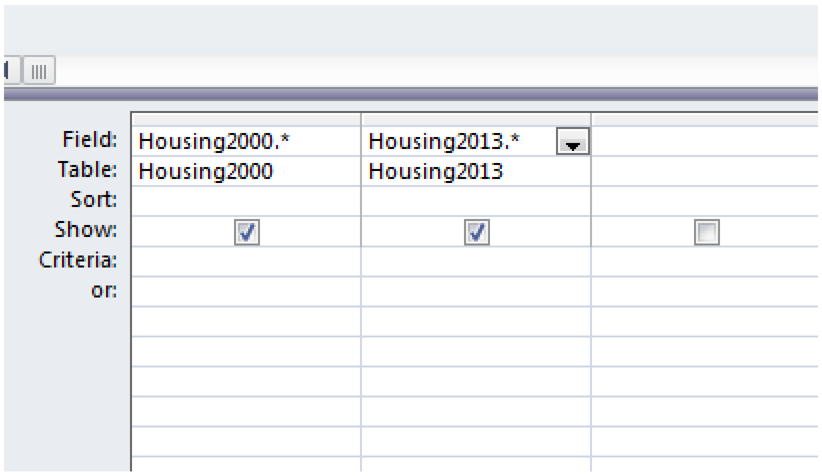
Left Table Name: Housing2000
Right Table Name: Housing2013
Left Column Name: Id2
Right Column Name: Id2

[✓ 1: Only include rows where the joined fields from both tables are equal.]
2: Include ALL records from 'Housing2000' and only those records from 'Housing2013' where the joined fields are equal.
3: Include ALL records from 'Housing2013' and only those records from 'Housing2000' where the joined fields are equal.

The other two options let you choose to get all the records back from one table, plus whatever mates it finds in the other table.

Let’s leave it on the default and see what we get.

Next, we need to add some fields to our query so we get some output. Since we have so few fields, let’s just put all of them in the query. Double-click on the asterisk at the top of each table (where you did the join) and it will put all the fields from both tables into the designer at the bottom.



Field:	Housing2000.*	Housing2013.*	
Table:	Housing2000	Housing2013	
Sort:			
Show:	✓	✓	☐
Criteria:			
or:			

Run the query.

You’ll see that we got 863 records back.

First thing we should do is go into our original tables and see how many records we’re supposed to have. The 2013 table has 905 records. The 2000 table has 867 records.

So we’ve got a situation here where we’re not going to get a perfect match. So what would the IDEAL match be?

Our choices here are:

1) Include only cities that existed in both 2000 and 2013. It will exclude any cities that have data in one year, but not the other.
2) Get a table back that has all the cities that existed in 2013, showing the 2000 data as well if the city existed back then, but it would be blank for ones that did not. This approach would exclude any cities that existed in 2000, but are not in the 2013 data (maybe it lost its city status and is now a township, for example – or maybe it merged with another city)
3) Get a table back that has all the cities that existed back in 2000, with data for 2013 if that city is still around. This one would exclude any cities that did not exist in 2000, but are in the 2013 data

Your choice is going to be dependent on your purpose with this analysis. But I also think it’s best to see what you’re missing, so I usually like to go with option 2 or option 3—at least initially.

Go back to the Design View of your query and right-mouse click on the line between the two tables to go back to Join Properties. Click on the option that lets you "include ALL records from 'Housing 2013'…"

Run your query again. This time we get 905 records. The same number as in the 2013 table. So the join did what it said…gave us all the 2013 records.

You'll see that there are a few that have blanks in the fields where you'd expect to find the 2000 data. (For example, look at Angle Inlet CDP). You'll see that there are a bunch of records for these CDP entities – I think these are places that used to be treated as townships, but more recently they've been identified as "places," so now they show up in this data.

There are a couple that aren't CDP's – Elko New Market and Columbus. What's going on with these? I would want to make sure there is a legitimate reason my join didn't work on these (and not a case of a typo). So typically I would try to do a manual search in the 2000 table and just make sure it's not really there. If you go looking for Elko New Market in the 2000 data, you'll find "Elko city" and "New Market City" and a little research in past news stories will show you that these two cities merged sometime between 2000 and 2013. If you go looking for Columbus, you won't find anything. The news clips, though, will show you that this used to be a township and it incorporated a few years back.

So we've now confirmed that all the places that don't match up have legitimate reasons for not matching. In this case, I'd probably go back to the Inner join (option 1) and just take the records that match.

Let's look at the SQL for this join:

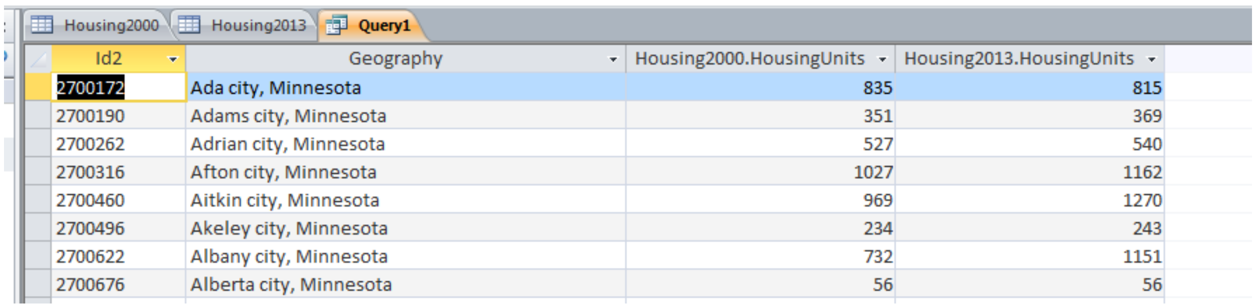
```
SELECT Housing2000.*, Housing2013.*
FROM Housing2000 INNER JOIN Housing2013 ON Housing2000.Id2 = Housing2013.Id2;
```

You'll see that it's doing its work in the FROM line. Notice also that it is putting the table name in front of the field names, with a period separating (Access does this by default, but if you are writing your own SQL you only need to do it in cases where the field names are the same between the two tables – like "ID2" and "ID2")

The Select line is where we put what fields we want to show in our answer. And once you make the join, you can pick and choose whichever fields from the two tables you want. In this case, we're pulling all of them. But maybe we don't want all of them?

For example, we could set it up so that we just have the ID and Name from the 2013 data, and then the housing unit counts from both tables, like this:

```
SELECT Housing2013.Id2, Housing2013.Geography, Housing2000.HousingUnits,
Housing2013.HousingUnits
FROM Housing2000 INNER JOIN Housing2013 ON Housing2000.Id2 = Housing2013.Id2;
```



Id2	Geography	Housing2000.HousingUnits	Housing2013.HousingUnits
2700172	Ada city, Minnesota	835	815
2700190	Adams city, Minnesota	351	369
2700262	Adrian city, Minnesota	527	540
2700316	Afton city, Minnesota	1027	1162
2700460	Aitkin city, Minnesota	969	1270
2700496	Akeley city, Minnesota	234	243
2700622	Albany city, Minnesota	732	1151
2700676	Alberta city, Minnesota	56	56

Now you can save this as a table to do additional work on it, or export it to Excel.

To save as a table, go back to the Design View of the query and go up to the Ribbon and select "Make Table" from the QueryTools/Design ribbon. This will bring up a little dialog box where you can name the new table. Then run the query and you'll have a new table.

To export to Excel, this one is small enough that you can just highlight the query results and copy and paste out to Excel. (If it's big, you can save the query and then in the left-hand pane where it lists all the tables and queries, right-mouse on the query and choose "Export")