Joining Tables in Access

Quite often, you'll get data from a government agency that is stored in separate tables. They do that in order to save space, which was a much bigger necessity when government databases were first being created. As a result, this practice still lingers. This is known as a "relational database."

For example, all of the pieces we want for public worker salary data (earnings, job title, years of service, name, etc) might actually be stored in two separate databases. One might be for human resources -- showing the person's job title (as of a certain date), their original hire date, the name of the bargaining unit they are in, possibly their base pay rate (i.e. an hourly rate) and things like that, which don't change as often.

Then they might have a payroll database where they keep track of the information from each paycheck; this database might only have an employee ID number (no name) and it would be necessary to join the two databases together to capture all the information.

This is a situation where the two tables were intended to go together. In some cases, the two tables will have the same number of records – one human resources record and one salary record. Other times it might be what's called a "one-to-many" relationship, where there is one human resources record but then many salary records (perhaps one for each paycheck or one for each year that they worked, etc)

When a government agency gives you data from a relational database, it's important that you know what this relationship is (one –to-one or one-to-many) and to know exactly what each record represents. For example, I've worked with salary data where they provided me one record for each portion of a person's paycheck – regular pay, overtime pay, sick pay, holiday pay, etc. So for one paycheck there might be 5 records; another might have 8 -- all depending on the variations of pay they had in that paycheck.

Other times, government agencies will give me salary data that has already been collapsed down into a total for the year – one record for each job the person held that year, with separate columns to tell me how much they were paid in regular pay, how much in overtime, and the grand total. There are times, though, that even this kind of table will have more than one record per person if they held multiple jobs (like somebody working for the Parks Service temporarily in the summer and then working for Public Works during the winter)

It's also important to make sure the data you get in from a relational database has a unique ID that will allow you to join all the tables together (that same unique ID field needs to exist in each table).

Also note that complex relational databases (with more than 2 tables) might have situations where one table joins to second table and then the third table only joins to the second one (not the first). In these situations, it's good to ask for a database "diagram" that shows how they all fit together.

We're going to practice working with a relational database using some St. Paul city salary data, along with a separate table that shows people who retired and what their pension is. The salary table has one record for each person who worked in 2011, showing their total salary, department, title, etc. The Retiree table has a unique ID that joins back to the salary table (it's the employee's unique ID for the city), plus the date they retired and their pension amount.

We're going to find out whether any of these recent retirees are making more in their pensions than they did in their last year of service with the city. We need to join the two tables together because their salary is contained in one table and their pension amount is contained in the other.

First, let's import the Excel file into Access.

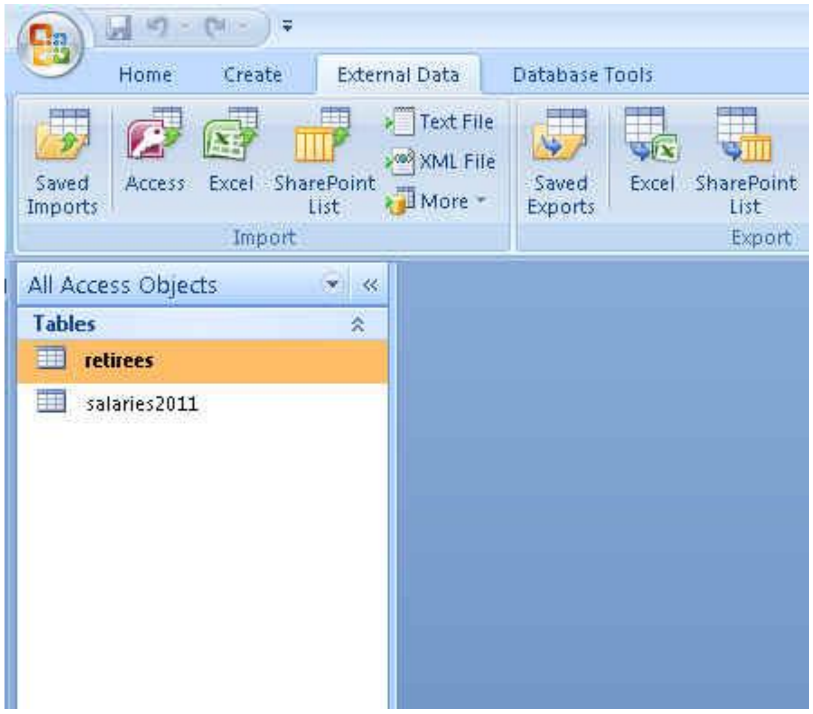
Create a new blank Access database and call it "St Paul pension"

Then go to the "External Data" ribbon and select "Excel" . Navigate to the **pensions.xls** spreadsheet.

When the wizard comes up, it's going to ask you which worksheet you want to import. There are two – **retirees and salaries2011** – that we need to import.

Go through the steps and import each one. Make sure the "**person_ID**" field in retirees and "**empID**" in salaries are both set as "text" data type. It's crucial that the join field has the same data type when you are trying to join the tables together.

So now you should have two tables in your database, like this:



Let's spend a little time looking at our data.

In the **retirees** table, we have the person ID number, name (all in one field), original employment date, retire date and pension amount. Note: there are some records where there is not a pension (I added this from separate pension records, which could only be joined by name. The missing ones are situations where I couldn't find a record for that name or the name was too common to be certain I was matching the right person)

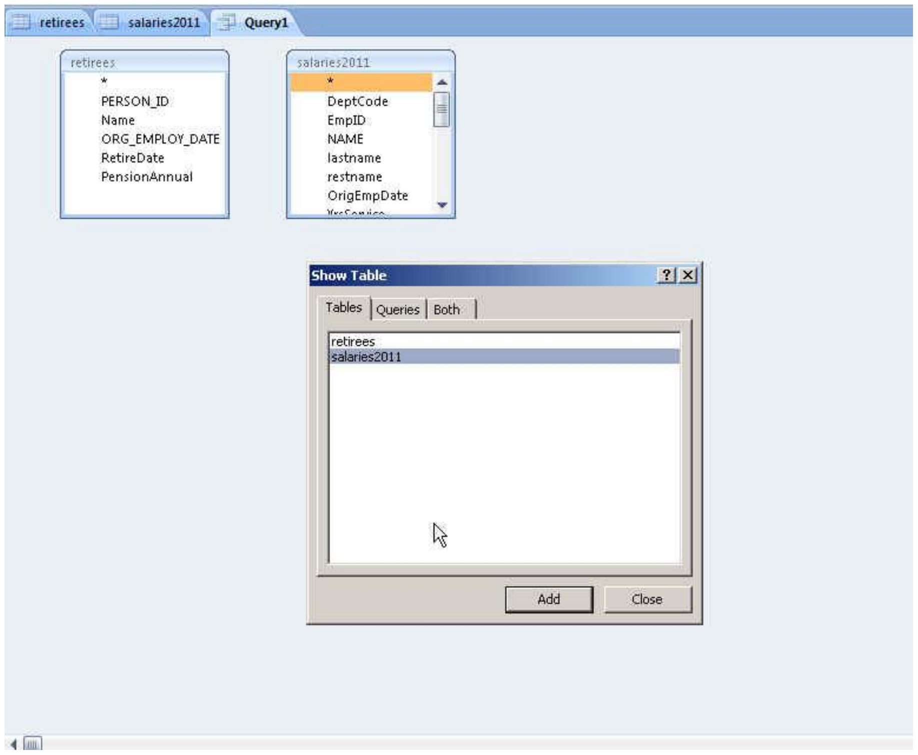
In the **salaries2011** table, we have a department code number (the department name is in a separate field), the EmpID (which is the person's employee ID number that matches the Person_ID in the retirees table), the name displayed together and separated out into last and rest, original employment date, years of service, their bargaining unit, base pay rate (base type indicates whether it's hourly or biweekly), total wages, total hours, overtime pay (which is a subset of total wages) and overtime hours (which is a subset of total hours). There's also a field indicating that this is calendar year 2011 data and on the far right are job title and department.

The key thing with a join is to know which fields match each other. In this case it's "person_ID" and "empID". It's fine that they aren't named the same. The only thing that matters is that they both have the same data type (in this case, "text")

Let's launch a query and see how this join works….

When you get the "Show Table" box, choose both tables.

You'll see that both of them show up in the Design View – like in this picture.

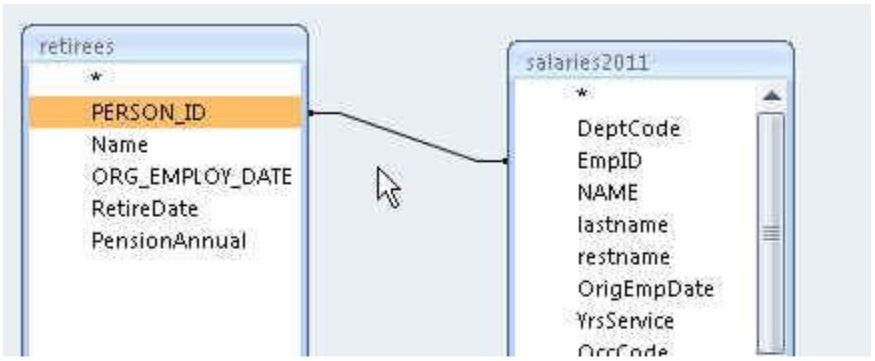


Joining tables is one of those occasions where I find the Design View is much, much easier than the SQL view – at least for the first step we need to do.

Close the Show Table box and stay in the Design View.

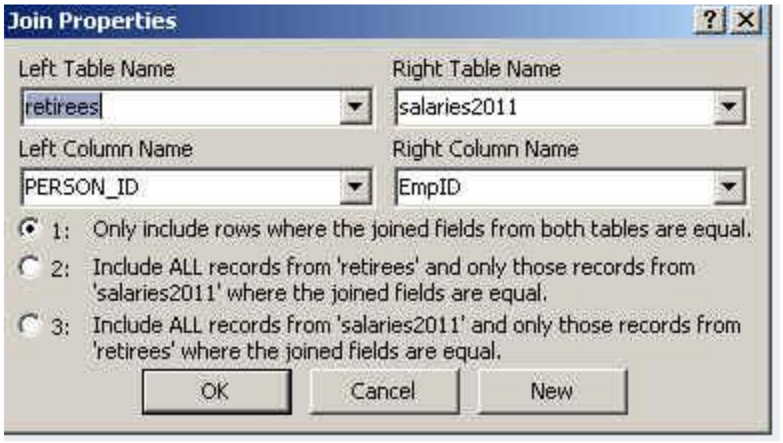
Notice that you can see all the fields from the two tables (in those little boxes that appeared in the Design View area).

If you click your mouse on “Person_ID” in the retirees table, hold down your mouse and then drag across and hover over the “EmpID” field in the salaries2011 table. Then let go when your cursor is hovering over the EmpID field. It will draw a line from one field to the next.



Right-mouse click on the line and choose “join properties.”

There you will see what it has done.



Join Properties

Left Table Name: retirees
Right Table Name: salaries2011
Left Column Name: PERSON_ID
Right Column Name: EmpID

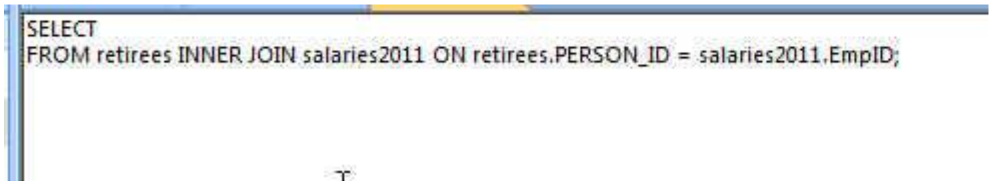
[✓ 1: Only include rows where the joined fields from both tables are equal.]
2: Include ALL records from 'retirees' and only those records from 'salaries2011' where the joined fields are equal.
3: Include ALL records from 'salaries2011' and only those records from 'retirees' where the joined fields are equal.

It shows the names of the columns that it matched (this is a good way to make sure you connected the correct fields) and then it has 3 radio button options.

This is for the type of join. The default is what’s known as an “inner join” – it will only return records that matched between the two tables. In other words, if there’s no retiree record for one of the people who has a salary record, the answer we get back in our query will not include that person.

The other two options are what’s known as “left join” and “right join” – depending on which table is in the left position and which is in the right (in this case, retirees is the left table).

Now flip over to SQL View and see what it's done:



```
SELECT
FROM retirees INNER JOIN salaries2011 ON retirees.PERSON_ID = salaries2011.EmpID;
```

It's written the SQL for us, putting both tables in the FROM line and adding some syntax indicating how they are joined together. It's a lot of typing to do a join in SQL – that's the reason I prefer using the Design View for this little task.

Once you have your two tables joined together like this, then you can start asking questions just like we did on single-table queries. Access thinks of these two tables as one table now. So you can use any of the fields – from either table – to craft your query.

The first thing you should do is just see how well the match works. In this case, we need to know that we have 100 people in the retirees table. All of them retired sometime in 2011 or early 2012. So presumably, they should have a pay record in the salaries table, since that shows total pay for 2011. All of them worked at least part of the year.

Since we are doing an "inner join" – the answers we get back from our queries are only going to include the people who showed up in both tables. Our first query SHOULD give us 100 records – that's if the join works properly and there are salary records for all the retirees.

One more thing before we get started….a join query requires some extra syntax in the SQL in cases where we have fields using the same name in both tables. For example, both tables have fields called "name" . So if we are going to use one of those fields in our query (in any way), then we need to specify which table we are getting the field from. We do that by saying "tablename.fieldname" – so it would be "retirees.name" or "salaries2011.name"

Let's try this one:

```
SELECT retirees.name, salaries2011.name, jobtitle, retiredate
FROM retirees INNER JOIN salaries2011 on retirees.PERSON_ID =salaries2011.EmpID
```

Check out the total number of records (at the bottom of the query). We got 100! Perfect. That means our join worked and we have matches for all the retirees.

It's also a good idea to peruse through these records and just make sure the names are matching up (what if they gave you screwed up employee ID numbers??). You just never know. It's worth checking a little.