Since there is a single country column, only one country can appear in a single Article record.  I was actually suggesting that we create a table to contain all country codes, specify a query against that table for the row source of a combo box, and Article.Country as the control source of the combo box, where Article is the record source of the form that the combo box is a member of.

So, there are two data sources:  one that populates the list of countries that the user can select from and one that is saved in an Article record when the user makes a selection in the country combo box.

To demonstrate how this works, I created an example using the following steps.  I always create data objects (tables) on the SQL server so that all data are shared by all users at all times.  Also, all combo box row sources should be queries against a SQL table, so that changes to code lists are made available to all users without having to distribute new versions of the app.  Rule:  data belong in the SQL database, instructions belong in the app.

1. Create a Codes table (I have already done this in the OEGM SQL database)

SQL syntax:

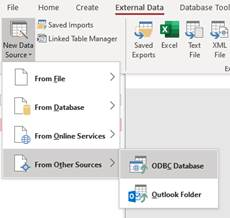
mysql> create table Codes(Class varchar(20) not null, Code varchar(100) not null, primary key(Class, Code));

Two columns:  Class (Country, OutcomeDomain, etc.) and Code (U.S., Canada, etc.).  To query country codes, we will use something like select Code from Codes where Class=’Country’ in a combo box row source.

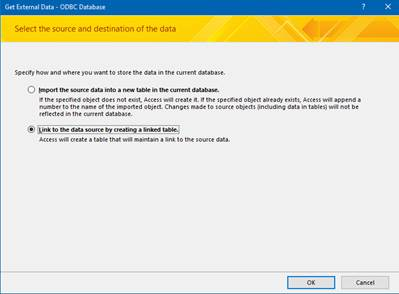
Note that, I believe, a better approach is to create separate tables for code classes, so that additional columns (latitude, longitude, continent) can be added with a guarantee that they apply to all codes in the particular code table.  Ancillary columns can be required entries when all codes are of the same class (imagine requiring lat and long for OutcomeDomain codes).  Furthermore, instead of storing actual codes in article records, I believe, it is better to auto assign (DB feature) code IDs and have those saved in records, so that a single ID points to all data elements (columns) related to a code.  The ID is saved in an article record and any changes to the associated code (spelling, corrected lat, etc.) are implemented by updating the single code record containing the data to be corrected.  No article records ever need modification.  Also, foreign keys (that would prevent a code from being deleted if specified in an article record) are difficult to implement when all code classes are combined in a single code table.  This is because two classes can have a single code (classes FavoriteLetter and HighestScore can each have an A record and deleting the FavoriteLetter code leaves a valid A code, but not related to FavoriteLetters).  For many reasons, combining code classes in a single file is problematic.  However, individual code tables, code IDs, and such require considerable additional effort.  For starters and simplicity, I suggest beginning with a simple strategy of a single codes table, but keep the individual code table model on your list of things to do to improve data and code integrity.

2. Link the Codes table within your Access app (examples, here, use the attached sample app)

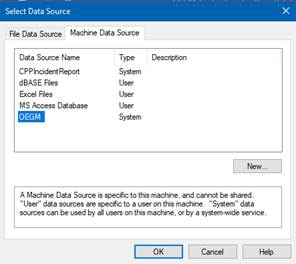
Click External Data, New Data Source, From Other Sources, ODBC Data Sources:



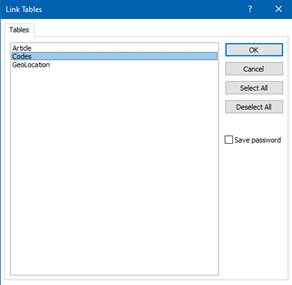
Click Link to Data Source:



Select Machine Data Source, OEGM:



Click Codes, then OK:



You should now have a link to the SQL Codes table in your Access Object Explorer:



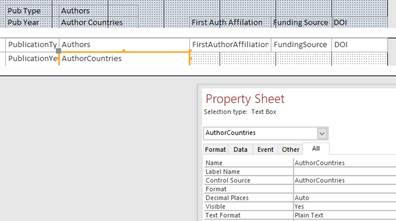
Double click the Codes table in the Access Object Explorer and you can now add/edit entries in it:



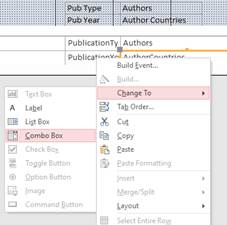
The attached app has a sample linked Codes table.

3. Specify a query against the Codes table as the row source of a combo box

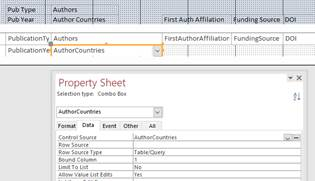
Open the Article form in design view and select the AuthorCountries text box:



Right click the AuthorCountries text box, roll over Change To, and select Combo Box:

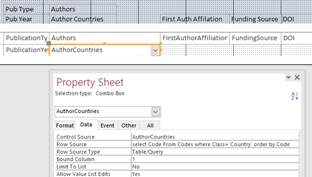


AuthorCountries now has a pull down icon, a Row Source, and a Row Source Type



Row Source Type contains Table/Query, which is good.  In the Row Source, enter:

select Code From Codes where Class=”Country” order by Code



Note that the control source of AuthorCountries is AuthorCountries.  This instructs that any entry made in this form control will be placed in the AuthorCountries column of the form’s record source, which happens to be Article.  The row source query limits code retrieval to code table entries with a class of Country.  Using this configuration , it is clear that all entries in Article.AuthorCountries are limited to countries from the corresponding codes for countries.

Save the Article form then double click it to be opened in view and entry mode.  Click on the AuthorCountries pull down icon:



Note that all codes (countries) in the Country class appear in the list in alphabetical order (order by).  I often include an AppearanceOrder column to order by, to give additional control over appearance order.  We can add this if you are interested.

4.  Add countries to the Code table

Double click the Codes table, so that it is in review/entry mode:



Position the cursor in the Class column of the \* record, type “Country,” tab to the Code column and enter the country of interest:



Press Enter, close the Code table entry form, re-open the Article form, and verify that the new country appears in the pull down list:

