ACCESS

By Patricia Cox

Master Budget Project: Budgeted Beginning Inventory

In order to calculate the Production Budget, we need the values for both Beginning and Ending Inventory each period. Last month we created a query for Ending Inventory. This month we will focus on two queries we need for the Beginning Inventory values. The first records the beginning inventory value for the first period. The second does the same for all other periods by using the ending value from the previous period.

Query for the First Period

The first query sets the first Beginning Inventory values and appends them to the Budget Table. This involves the Reporting Period and Product tables and criteria for 2012 quarter 1.

Create a query. Add the Reporting Period Table and the Product table to the query layout, but don't link the tables. Include fields for Reporting Year, Reporting Quarter, Product Name, Type, Beginning Inventory, and Value (see Table 1). Set the criteria for Budget Year to 2012 and Budget Quarter to 1. Calculate Value by multiplying Beginning Inventory times Selling Price. Change the query to an append query, and append the fields as shown in Table 1. Save the query as "Append First Quarter Beginning Inventory."

Table 1. First Quarter Beginning Inventory Query Elements

Field	Append to	Criteria
Reporting Year	Budget Year	2012
Reporting Quarter	Budget Quarter	"1"
Product Name	Budget Item	
Type: "Beginning Inventory"	Budget Type	
Beginning Inventory	Count	
Value: [Beginning Inventory]*[Selling Price]	Amount	

Query for All Other Periods

The next query uses the value from the first quarter to build the other values. This query takes the ending inventory values from the previous period and makes them the beginning inventory values in the next period. It then appends them to the Budget table.

To do this, create a query and add the Budget and Product tables. Link Budget Item to Product Name: From the Budget table, click on Budget Item and drag it over to Product Name in the Product table. Now add IIF functions for Next Budget Year (to increase the year by 1 when the quarter is 4) and Next Budget Quarter (to increase the quarter by 1 or, if it's the fourth quarter, reset it to 1). Table 2 shows the IIF statements and other elements of the query. Note that BudgetType2 is "Beginning Inventory" and that the Budget Type field from the Budget table is used to select "Ending Inventory" values. This, combined with increasing the period by 1, brings the ending inventory value forward to be the beginning inventory value for the next period. Save the query as "Append Beginning Inv Budget Values for All Products."

Update the Macro

Open the Budget Process Streamlined macro in design view. Insert two new rows above the second Set Warning row

Table 2. Beginning Inv Budget Values for All Products Query Elements

Field	Append to	Criteria
Product Name	Budget Item	
Next Budget Year: IIF([Budget Quarter]=4,[Budget Year]+1,[Budget Year])	Budget Year	
Next Budget Quarter: IIF([Budget Quarter]=4,1,[Budget Quarter]+1)	Budget Quarter	
BudgetType2: "Beginning Inventory"	Budget Type	
Budget Count: Count	Count	
Amount: [Budget Count]*[Selling Price]	Amount	
Budget Type		"Ending Inventory"

and add OpenQuery actions for the two queries you created. Save the macro. One important thing to do is to make sure the actions are in the correct order (see Figure 1). The macro needs to run the append query for the first period first. Keep in mind that a complex macro can include all the right steps but still get the wrong result if they aren't in the correct order. That's why it's important to always test the macro every time you change a process or when building a particularly complex macro involving a series of steps.

Best Practice

As we work on complex processes in Access, it can seem at first as if everything is a unique new problem. As you get more comfortable, however, you'll begin to notice that we repeat certain steps again and again. For example, when we make a change to the process, we always update the macro and test it. Each month going forward, I will include a best-practice note. These will be the principles that we operate under when using a database to solve problems. We are following a cohesive set of guiding practices, but it's easy to lose sight of this as we work on miniscule details each month. Ultimately, these best practices will help you as you work on any database project.

Sometimes you can't achieve what you want done in a single step or with one query. Often, this is where people give up. This month's best practice is to always try breaking down complex problems into multiple steps. Eventually you'll be able to develop a set of steps that gets you the solution you need. And no matter how many steps you end up with, they can easily be run in order with a macro. **SF**

Patricia Cox has taught Excel and Access to management accounting students and other college majors and has consulted with local area businesses to create database reporting systems since 1998. She is also a member of IMA's Greater Milwaukee Chapter. To send Patricia a question to address in the Access column, e-mail her at kathrynmann@tds.net.

Figure 1. Updated Macro

Budget Process Streamlined

Condition	Action	Arguments
	SetWarnings	No
0	OpenForm	Budget Assumptions Entry Form, Form, , , , Normal
	OpenQuery	Empty the Budget Table, Datasheet, Edit
	OpenQuery	Append Budget Values for All Products, Datasheet, Edit
	OpenQuery	Append First Quarter Beginning Inventory, Datasheet, Edit
	OpenQuery	Append Beginning Inv Budget Values for All Products, Datasheet, Edit
	SetWarnings	Yes
	MsgBox	Process Complete, Yes, None,