



Active Blog
Access Trick

www.active-technologies.com
info@active-technologies.com

Query With Running Total

Displaying subtotals, grand totals, and otherwise summarizing data is a common report feature. You can display a running total in a report simply by setting a property. A *running total* is a cumulative sum that evaluates the previous rows and the current row. In other words, each row's running total is equal to itself plus the previous total. Unfortunately, not every object handles a running total as easily as a report.

The expression

There's no built-in property or function for generating a running total in a query. For that, you'll need a rather complex expression in the form
SELECT fieldlist,

```
(SELECT Sum(valuefield) AS Total  
  
FROM datasource  
  
WHERE datasource.sortfield <= T1.sortfield) AS Total  
  
FROM datasource AS T1
```

Table A explains the statement's arguments.

Table A

Argument	Explanation
<i>fieldlist</i>	The fields you want the query to return
<i>valuefield</i>	The field you're summing
<i>datasource</i>	The table (or query) that contains the values you're summing
<i>sortfield</i>	A field of unique and ascending values

Although this expression works well, there's an unforgiving catch: The table must have a column of unique values in ascending order. That's *sortfield* in the above syntax statement. An AutoNumber field gets the job done, but a numeric field of unique values in ascending order will also work. Sorting by the actual values you want to total works only if those values are unique and happen to be in ascending order. You should avoid using the actual values for this reason. Even if they start out that way, it's unlikely they'll stay that way.

Active Blog
Access Trick

www.active-technologies.com
info@active-technologies.com

An example query

Let's build a query that generates running totals for the values in the table shown in **Figure A**. (OrderID is an AutoNumber field and OriginalValue is a Number field.) You can use any table, but it must have the two numeric fields we've already discussed: a field of values to sum and a field of unique values in ascending order.

Figure A

	OrderID	OriginalValue
▶	1	1
	2	3
	7	2
	8	1
	9	7
*	(AutoNumber)	0

With the select query in Design view, choose SQL View from the View button to open the SQL window. This window displays a query's SQL statement. (You can't enter this SQL statement via the QBE grid.) In the SQL window, enter the following SQL statement:

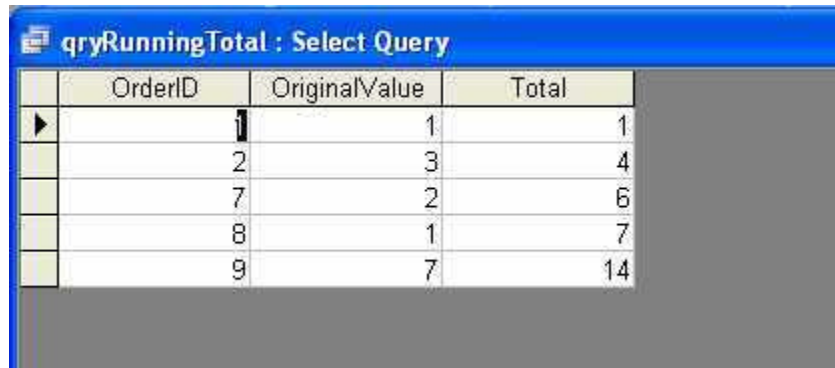
```
SELECT OrderID, OriginalValue,  
  
    (SELECT Sum(tblOriginalValues.OriginalValue) AS Total  
  
    FROM tblOriginalValues  
  
    WHERE tblOriginalValues.OrderID <= T1.OrderID) AS Total  
  
FROM tblOriginalValues AS T1
```

Active Blog
Access Trick

www.active-technologies.com
info@active-technologies.com

When you run the query, the Total field returns a cumulative sum for the values in the OriginalValue field, as shown in **Figure B**.

Figure B



OrderID	OriginalValue	Total
1	1	1
2	3	4
7	2	6
8	1	7
9	7	14

Because the OrderID value (the AutoNumber field) is in ascending order, the WHERE clause's condition is always True, which forces a new sum for every record. This SQL statement works similarly to an aggregate domain function, but it's much faster.

Worth noting

You might have noticed that the AutoNumber values in the example's *sortfield* (OrderID) skips values. Gaps between values won't affect the results. In truth, it isn't necessary for *sortfield* to be in ascending order. The only absolute is that the field contain unique values. If *sortfield* isn't in ascending order, add an ORDER BY clause in the form:
ORDER BY T1.*sortfield*

Doing so will most likely change the running totals, so if order matters, rely on an AutoNumber field.

Finally, this solution will slow down when evaluating large amounts of data. The good news is that it should still perform faster than using an aggregate domain in either a query expression or a report or form control.

Keep totaling

Generating a running total isn't intuitive to a query, but you can get the job done using a subquery. You'll need an extra numeric field with unique ascending values, which you can accomplish by simply adding an AutoNumber field to the table, if necessary. Once you've generated the running total in the query, you're free to view the list in any format you like.

If you require additional information or assistance with this item, please give us a call.