

Re: Query calculation question?

Source: <http://www.tech-archive.net/Archive/Access/microsoft.public.access.queries/2004-04/0772.html>

From: John Viescas (*JohnV_at_nomail.please*)

Date: 04/08/04

Date: Thu, 8 Apr 2004 08:03:36 -0500

Ah, well. I tried my ascending price solution – which I thought was correct – in Northwind. However, I had the comparison backwards. This almost works in Northwind:

```
PARAMETERS Budget Currency;
SELECT Products.ProductID, Products.ProductName, Products.UnitPrice,
[Budget] AS EnteredBudget, (Select Sum(UnitPrice) From Products As P2 Where
P2.UnitPrice <= Products.UnitPrice) AS TotSpent
FROM Products
WHERE [Budget] >= (Select Sum(UnitPrice) From Products As P2 Where
P2.UnitPrice <= Products.UnitPrice)
ORDER BY Products.UnitPrice;
```

The technique is to sum the price of all products whose price is less than or equal to the price in the current row. As long as that sum is less than or equal to the budget entered, the row is selected. This is not completely accurate, however, when multiple products have the same price, and adding just one or two of them would still keep you under the budget.

However, if your table has a unique identifier (like the ProductID field in Northwind Products), you can get an exact answer, including duplicate prices, like this:

```
PARAMETERS Budget Currency;
SELECT Products.ProductID, Products.ProductName, Products.UnitPrice,
[Budget] AS EnteredBudget, (Select Sum(UnitPrice) From Products As P2 Where
(P2.UnitPrice < Products.UnitPrice) Or ((P2.UnitPrice = Products.UnitPrice)
And (P2.ProductID <= Products.ProductID))) AS TotSpent
FROM Products
WHERE [Budget]>=(Select Sum(UnitPrice) From Products As P2 Where
(P2.UnitPrice < Products.UnitPrice) Or ((P2.UnitPrice = Products.UnitPrice)
And (P2.ProductID <= Products.ProductID)))
ORDER BY Products.UnitPrice;
```

Enter a budget of \$225, and you'll get two of the three products listed with a price of \$14.

microsoft.public.access.queries: Re: Query calculation question?

Let me guess – you're trying to find a slick way to get the most items for a fixed "upgrades" budget. <s>

```
--
John Viescas, author
"Microsoft Office Access 2003 Inside Out"
"Running Microsoft Access 2000"
"SQL Queries for Mere Mortals"
http://www.viescas.com/
(Microsoft Access MVP since 1993)
"Mike Pallos" <MPallos@gte.net> wrote in message
news:2067BE14-EF9E-49D7-8E45-E0F3A6ECE320@microsoft.com...
> Hello John,
>
> I am working on a program for my dissertation and do not wish to let the
"cat out of the bag" yet. The candy analogy works nicely representing the
logic I need.
>
> The candy table contains many kinds of candies at all different prices.
The user has a set budget and wishes to purchase as many candy as possible.
However, the point I may have failed to focus on before is that, the user
only wants one piece of each type of candy available that fit within his
budget, i.e., how many different candies could one obtain for a certain
budget?
>
> To get the most amount of candy, the logic needs to start at with the
least expensive piece, and keep purchasing additional pieces of different
types of candy, until the budget is exhausted.
>
> I hope that helps. I am from old school. Unix C back in the late 80s
doing embedded SQL. So I am more familiar with programs, than complex SQL
statements - although I would much prefer to pull this off in SQL, making
the logic more portable, and less tied to VB. Being an architect, I am
rusty to programming - now a days I just talk about architecture and draw
pictures mostly (Powerpointware!)
>
> Thanks for everything. The coding examples from your Access 2000 book are
wonderful (I am using 2000). I am able to learn quite well from reading the
book, and now working with your examples.
>
> I appreciate your time. I too hope to publish a book some day (let alone
three!!!!).
>
> Michael
>
```