November, 1999

Advisor Answers

Multiple Detail Band Reports

Visual FoxPro 6.0, 5.0, 3.0 and FoxPro 2.x

Q: I have a parent, child, child, child set of tables. The three child tables are all related to the parent, but do not relate to each other. I need to replace a hand-coded FoxPro/DOS report in Visual FoxPro (either 5.0 or 6.0). This report has the parent's information at the top, then the first child's records, then the second child's records and so forth.

This is an insurance program that prints all policy header information first, then all location information, then all the coverage detail, then all of the various addresses last. I need to somehow figure out how to finish the header info first, then SCAN and print all location records, then SCAN and print all coverage records, then SCAN and print all address information.

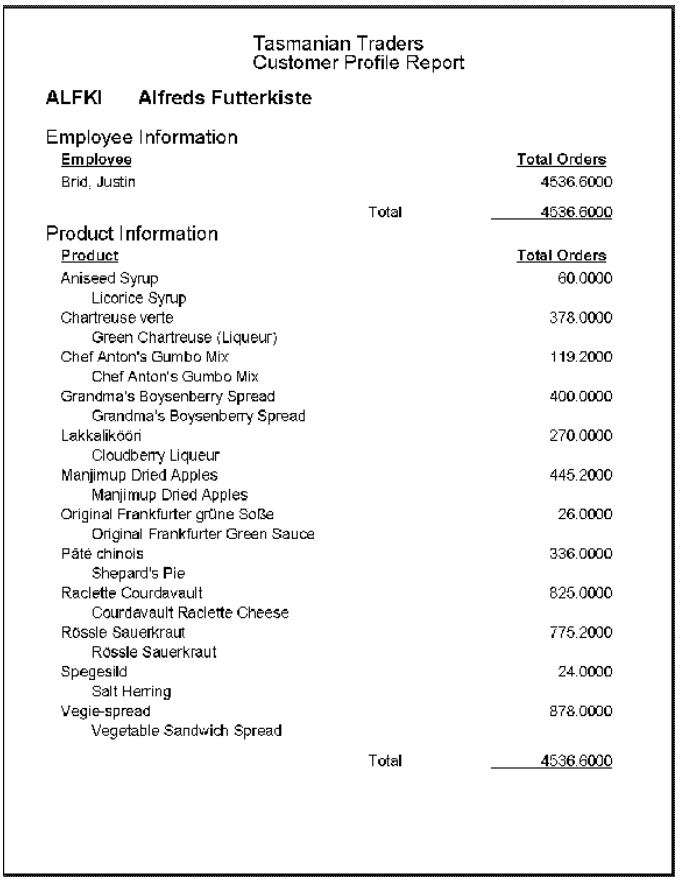
What I feel I need is a "Group Header" and a "Detail" set of lines for each different table. Can I get there from here, or do I have to tell my customer that it can't be done? Actually, I can do this by pre-processing, but really don't want to do that. Are there any other options?

—Dan Farnham (via Advisor.COM)

A: You have what I like to refer to as the "multiple unrelated siblings" challenge (it's a 90's kind of thing). I first addressed this problem in FoxPro 2.0 in the October 1993 issue of FoxPro Advisor. While the basic approach hasn't changed since, the specifics have, so it's worth looking at it again. (Also, note that Gary Zaika published an alternative approach in the July '95 issue.)

By design, the Report Designer can't handle multiple detail bands. But FoxPro's design has never stopped us from doing what we want. We can trick the RD into producing what look like multiple detail bands. However, you can't do it without doing some work up front. Since I almost always use queries (or views) to prepare the data for a report, I don't consider this a problem.

I'll use the TasTrade example data that comes with VFP to demonstrate. Suppose you want to create a "customer profile" report. The report shows, for each customer, a list of all employees with whom the customer has placed an order, and a list of all the products the customer has ever ordered. For each, the total of the orders is shown. Figure 1 shows the first page of the report. (This is a simplified version. In a real application, the report would probably include a lot of other information, especially about the customer.)



Tasmanian Traders
Customer Profile Report

ALFKI Alfreds Futterkiste

Employee Information

Employee	Total Orders
Brid, Justin	4536.6000
Total	4536.6000

Product Information

Product	Total Orders
Aniseed Syrup	60.0000
Licorice Syrup	
Chartreuse verte	378.0000
Green Chartreuse (Liqueur)	
Chef Anton's Gumbo Mix	119.2000
Chef Anton's Gumbo Mix	
Grandma's Boysenberry Spread	400.0000
Grandma's Boysenberry Spread	
Lakkalikööri	270.0000
Cloudberry Liqueur	
Manjimup Dried Apples	445.2000
Manjimup Dried Apples	
Original Frankfurter grüne Soße	26.0000
Original Frankfurter Green Sauce	
Pâté chinois	336.0000
Shepard's Pie	
Raclette Courdavault	825.0000
Courdavault Raclette Cheese	
Rössle Sauerkraut	775.2000
Rössle Sauerkraut	
Spegesild	24.0000
Salt Herring	
Vegie-spread	878.0000
Vegetable Sandwich Spread	
Total	4536.6000

Figure 1 Customer Profile Report–The first "detail band" lists all the employees this customer has ordered from. The second shows all the products the customer has ordered.

So what's behind the smoke and mirrors? There are two parts: the query that assembles the data and the report that displays the query result.

The query is really one query for each of the child tables consolidated into a single result using the UNION clause. In the process, we add a field that tells which child table the particular record comes from. Here's a VFP 5 or 6 version of the query for the customer profile report. (In VFP 3 or FP 2.x, you need to use the WHERE clause for the join conditions rather than the JOIN clause.)

```
SELECT Customer.Customer_Id, Company_Name, ;
  PADR(TRIM(Last_Name) + ", " + First_Name, 40) AS cName, ;
  SPACE(50) AS cEngName, ;
  SUM(Quantity * Unit_Price) AS nTotal, ;
  "E" AS cType ;
 FROM Customer ;
   JOIN Orders ;
     JOIN Order_Line_Items ;
       ON Orders.Order_ID = Order_Line_Items.Order_ID ;
     ON Customer.Customer_ID = Orders.Customer_ID ;
     JOIN Employee ;
      ON Orders.Employee_ID = Employee.Employee_ID ;
 GROUP BY 1, 2, 3 ;
UNION ALL ;
SELECT Customer.Customer_ID, Customer.Company_Name, ;
  Products.Product_Name AS cName, ;
  Products.English_Name AS cEngName, ;
  SUM(Quantity * Order_Line_Items.Unit_Price) AS nTotal, ;
```

```
 "P" AS cType ;
FROM Customer ;
  JOIN Orders ;
    JOIN Order_Line_Items ;
      JOIN Products ;
      ON Order_Line_Items.Product_ID = Products.Product_ID ;
    ON Orders.Order_ID = Order_Line_Items.Order_ID ;
  ON Customer.Customer_ID = Orders.Customer_ID ;
GROUP BY 1, 2, 3 ;
ORDER BY 2, 6, 3 ;
INTO CURSOR CustomerProfile
```

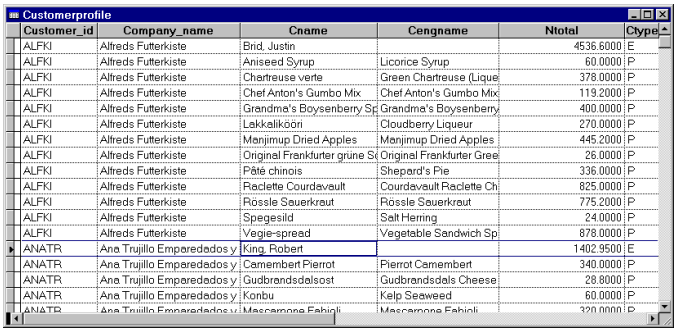
As with any query involving the UNION clause, the field lists of the individual queries must contain the same number of items and corresponding items must be of the same data type and size. That's why the first query (the Employees query) includes the line:

```
SPACE(50) AS cEngName
```

That field is a placeholder for the English_Name field in the Products query.

The cType field in the two queries tells us where a particular record originated. It's "E" for an employee record and "P" for a product record.

The ORDER BY clause, which applies to the overall result of a UNIONed query, sorts the records by customer, then by cType ("E" or "P"), then by the cName field (which is either employee name or product name). So, after the query, the cursor is in exactly the order we want for the report. Figure 2 shows a little of the data.



Customerprofile

Customer_id	Company_name	Cname	Cengname	Ntotal	Ctype
ALFKI	Alfreds Futterkiste	Brid, Justin		4536.6000	E
ALFKI	Alfreds Futterkiste	Aniseed Syrup	Licorice Syrup	60.0000	P
ALFKI	Alfreds Futterkiste	Chartreuse verte	Green Chartreuse (Lique	378.0000	P
ALFKI	Alfreds Futterkiste	Chef Anton's Gumbo Mix	Chef Anton's Gumbo Mix	119.2000	P
ALFKI	Alfreds Futterkiste	Grandma's Boysenberry Sp	Grandma's Boysenberry	400.0000	P
ALFKI	Alfreds Futterkiste	Lakkalikööri	Cloudberry Liqueur	270.0000	P
ALFKI	Alfreds Futterkiste	Manjimup Dried Apples	Manjimup Dried Apples	445.2000	P
ALFKI	Alfreds Futterkiste	Original Frankfurter grüne S	Original Frankfurter Gree	26.0000	P
ALFKI	Alfreds Futterkiste	Pâté chinois	Shepard's Pie	336.0000	P
ALFKI	Alfreds Futterkiste	Raclette Courdavault	Courdavault Raclette Ch	825.0000	P
ALFKI	Alfreds Futterkiste	Rössle Sauerkraut	Rössle Sauerkraut	775.2000	P
ALFKI	Alfreds Futterkiste	Spegesild	Salt Herring	24.0000	P
ALFKI	Alfreds Futterkiste	Vegie-spread	Vegetable Sandwich Sp	878.0000	P
ANATR	Ana Trujillo Emparedados y	King, Robert		1402.9500	E
ANATR	Ana Trujillo Emparedados y	Camembert Pierrot	Pierrot Camembert	340.0000	P
ANATR	Ana Trujillo Emparedados y	Gudbrandsdalsost	Gudbrandsdals Cheese	28.8000	P
ANATR	Ana Trujillo Emparedados y	Konbu	Kelp Seaweed	60.0000	P
ANATR	Ana Trujillo Emparedados y	Mascarpone Fabioli	Mascarpone Fabioli	320.0000	P

Figure 2 Collecting Customer Profile information–A UNIONed query lets us consolidate data from multiple unrelated siblings into a single cursor for reporting. The cType field indicates the original source.

Now we're ready to create the report. The secret to "multiple detail bands" is using an extra level of grouping based on the cType field. To make the appropriate group headers and footers, use a combination of IIF() and Print When conditions. (FoxPro/DOS didn't have Print When, so IIF() was the only tool available for this part.) Figure 3 shows the Customer Profile report in the Report Designer.

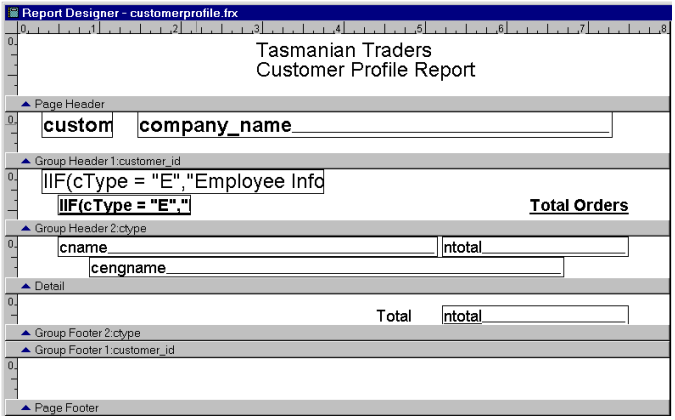


Figure 3 Designing "Multiple Detail Bands"–The cType Group provides the extra level of looping needed to report on each child separately. IIF() is used in the group header to change the heading for each child. Since the cursor used for the report is also ordered by this field (within customer), all of the employee information and all of the product information appears together for each customer. Otherwise, you could see multiple employee and product headings for each customer.

The first expression in the cType group header is:

```
IIF(cType = "E","Employee Information", ;
    "Product Information")
```

The others are similar. The cEngName field uses Print When's Remove Line if Blank option. In other cases, other Print When options may be useful. You'll find the query and the report on this month's PRD.

This approach works best when the list of fields is similar, that is, when the various "detail bands" look more or less alike in structure. When the child tables are wildly different, you can use lots of dummy fields and Print When's or try another approach, such as the one in Zaika's article.

—Tamar