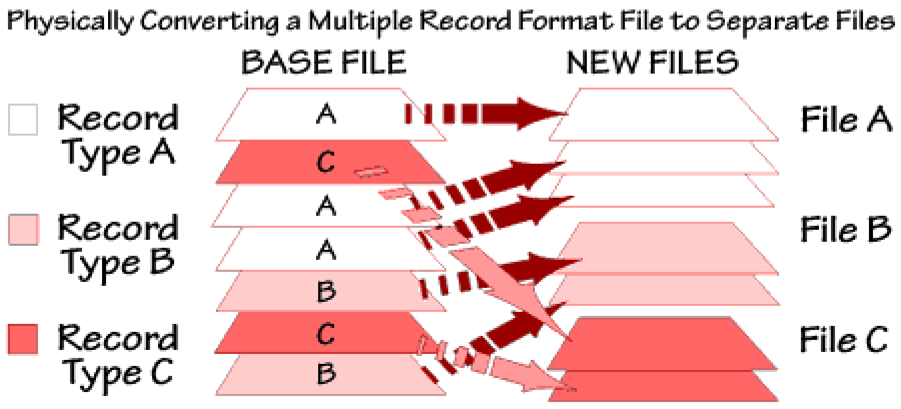
ODBC Liberates Non-Normalized Data

By Andy Forget

The BASIS ODBC Driver 1.1 has new capabilities that will enable you to access legacy files that have multiple record formats. You may recall, in the past, BASIS had encouraged you to convert these legacy files to a normalized format. We suggested that you create a new file for each record format in the base file. Although this is still a good idea, this solution is no longer required to use the BASIS ODBC Driver 1.1 with your existing data.



The Basics of a View

The mechanism that enables you to retain existing multiple record files without creating new files and rewriting existing code is a *relational view*. A relational view allows you to look at a file (also called a base table) via a virtual query.

Assume we have the following base table, EMPLOYEE:

EMP_ID	**EMP_NAME**	**EMP_DEPT**
0001	**Black**	**Sales**
0002	**Green**	**Sales**
0003	**Blue**	**Marketing**
0004	**Brown**	**Shipping**

We could create a view on the EMPLOYEE base table that only shows employees who belong to the Sales department:

```
CREATE VIEW SALES_EMP AS SELECT EMP_ID, EMP_NAME, EMP_DEPT
FROM EMPLOYEE WHERE EMP_DEPT='Sales'
```

We now have a new virtual table, or view called SALES_EMP:

EMP_ID	**EMP_NAME**	**EMP_DEPT**

0001	**Black**	**Sales**
0002	**Green**	**Sales**

This new view can be treated exactly like a base table. Your can do queries against the new view just as you would against a normal table. Views are *virtual*. They do not physically exist. At runtime, they retrieve data from the base table that they are associated with. Creating a view does not create a new physical file.

In addition to limiting rows in a table from a view, we can also limit the number of columns that can be seen from the view. Suppose we had an application that had no need for employee departments. We could create a view that does not contain department information:

```
CREATE VIEW SHORT_EMP AS SELECT EMP_ID, EMP_NAME FROM EMPLOYEE
```

This would result in the following SHORT_EMP view:

EMP_ID	**EMP_NAME**
0001	**Black**
0002	**Green**
0003	**Blue**
0004	**Brown**

Non-Normal Views

These previous simple examples have relied upon a normalized base file. It is possible to create a view that accesses a non-normalized file that has multiple record types. Suppose we had a multi-record type file that had information for both customers and orders in it. This physical file custord.dat has the following information in it:

```
BWW, 0, Bob's Western Wear, 12 West St., Albany
BWW, 1, Felt Hat, 21.95
BWW, 2, Leather Gloves, 4.80
ABS, 0, ACME Boot Shop, 9 Palm Ave., O'Fallon
ABS, 1, Leather Gloves, 4.80
ABS, 2, Stitched Boots, 50.00
```

The first field is the company code and the second field is the order number. If the order number is zero, then the record describes customer information and conforms to the following PRO/5 template:

```
COMPANY_CODE:C(3), ORDER_NUM:C(4), NAME:C(20*),
ADDRESS:C(20*), CITY(20*)
```

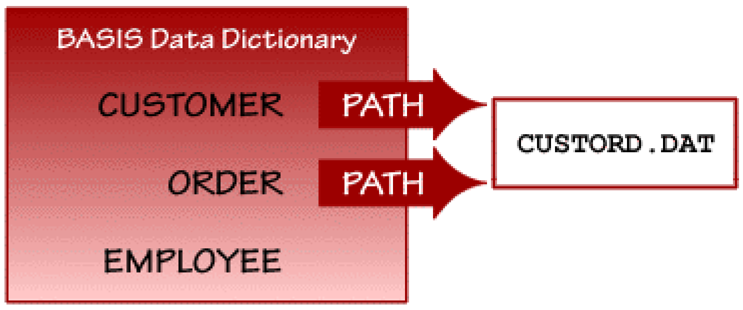
If the order number is non-zero, then the record describes order information and conforms to an order record:

```
COMPANY_CODE:C(3), ORDER_NUM:C(4), PRODUCT:C(20*), PRICE:N(8*)
```

When we look at the raw data, we see that we have two customers, Bob's Western Wear, and ACME Boot Shop. Each customer has two orders.

We can now create definitions for a "base" CUSTOMER table and a base ORDER table in our BASIS Data Dictionary. Each base table will point to the same physical file, custord.dat.

The CUSTOMER and ORDER Tables Each Point to the Same Physical File Via the Data Dictionary Path



At this point it is not possible to do queries against CUSTOMER and ORDER because they have different field types and different number of fields in each record. We must create two new relational views on our CUSTOMER and ORDER files that filter out the records we do not need.

```
CREATE VIEW V_CUSTOMER AS SELECT COMPANY_CODE, NAME, ADDRESS,
CITY FROM CUSTOMER WHERE ORDER_NUM=0

CREATE VIEW V_ORDER AS SELECT * FROM ORDER WHERE ORDER_NUM<>0
```

Since we have created our views, V_CUSTOMER, and V_ORDER, we now have two new virtual tables. Each table represents a single record type. There is no ambiguity as to which record type is associated with each view because the views WHERE clause limits the selection of records to those that match the correct record definition.

The V_CUSTOMER view now has the following data in it:

COMPANY_CODE	NAME	ADDRESS	CITY
BWW	Bob's Western Wear	12 West St	Albany
ABS	ACME Boot Shop	9 Palm Ave	O'Fallon

The V_ORDER view has the following information in it:

COMPANY_CODE	ORDER_NUM	PRODUCT	PRICE
BWW	1	Hat	21.95
BWW	2	Leather Gloves	4.80
ABS	1	Leather Gloves	4.80
ABS	2	Stitched Boots	50.00

This has been done without creating new physical files or altering the existing custord.dat physical file in any way. You do not have to modify existing PRO/5 code or change existing applications. All of the work has been done via the BASIS Data Dictionary and the BASIS ODBC Driver 1.1.

Conclusion

This technique can be applied to physical files that have any number of record types. Simply define a BASIS Data Dictionary definition for each record type and associate a filter view with each record type described in the BASIS Data Dictionary. Views allow you to "peel" individual record formats from a physical file and have them appear as if they were contained in a normalized file.

We have barely scratched the surface when it comes to the power of views and the BASIS ODBC Driver 1.1. A much more complete and sophisticated example of using the BASIS ODBC Driver 1.1 with non-normal data can be found on the BASIS web site, www.basis.com. See the white paper titled "Non-Normalized Data with the BASIS ODBC Driver".