Take A Ride With The BASIS ODBC Driver 2.0

By Don Andersen and Jeff Ash

The challenge of accessing and retrieving data from multiple databases on different platforms and consolidating it into a single format can be intimidating. Though retrieving this data is often done with considerable time and effort, many developers may not realize that by using the proper tools, this need not be the monumental task it appears to be. Using the features in the new version of the BASIS ODBC Driver®, any ODBC-compatible application can access data stored in files described in a BASIS data dictionary.

What Is ODBC?

Developed as a component of Microsoft's Windows Open Services Architecture (WOSA), ODBC (Open Database Connectivity) is the standard Application Programming Interface (API) that presents differences in database management systems to the user as a single, transparent data source. Because the ODBC driver is platform independent, highly customized front-end applications are unnecessary. In addition, as an open API, ODBC isolates the user from any changes or upgrades to the underlying data source.

For developers, this means an application can be developed using tools such as Microsoft Access to retrieve data from a BASIS, ORACLE, or Access data source without writing three different sets of code. By using ODBC, the task of compiling information from a database can be moved from the developer to the user because familiar spreadsheet or report writer applications can be used to access data that may have been entered using a BBx® application.

What's New In Version 2.0?

New SQL Grammar

Though Structured Query Language (SQL) is for the most part an industry standard, variations or additions in syntax exist that developers adopt for their implementations of SQL. The new BASIS ODBC Driver includes additions and enhancements to the SQL grammar to increase the power and flexibility of the SQL engine, while making it compatible with a wider variety of applications. Several features have also been added to increase the versatility of the BASIS ODBC Driver. With queries and updates, for example, nested SELECT statements, table abbreviations, outer joins, **`table_name.*`**, and **`{d 'date value'}`** help smooth data access and retrieval.

The most notable enhancement to the BASIS ODBC Driver is the ability to perform subqueries using nested SELECT statements. To illustrate, the following sample query builds an "imaginary table" containing all the values in PROD_CAT, where the COST_ACCOUNT is 5004. It then selects from the ITEM table all records where the value of PROD_CAT is present and returns the value of the DESC column.

```
SELECT item.desc
FROM   item
WHERE  item.prod_cat IN
       (SELECT category.prod_cat
        FROM   category
        WHERE  category.cost_account = '5004')
```

Subqueries can also be applied when retrieving lists of records with values greater than or less than the average of all the values in the table. Here, the average COST is computed from all records in the table, after which the ID, DESCRIPTION, and COST are retrieved from all the records in that table where the COST is greater than the average COST.

```
SELECT p.id, p.description, p.cost
FROM   product p
WHERE  p.cost >
       (SELECT AVG(cost)
        FROM   product)
```

Table Abbreviations

Another capability added to version 2.0 of the BASIS ODBC Driver is table abbreviations. For example, the following two queries generate the same output, but table abbreviations eliminate the need to manually enter lengthy table names with each query.

The following query example uses no abbreviations:

```
SELECT item.desc, item.prod_cat, category.desc
FROM   item, category
WHERE  item.prod_cat = category.prod_cat
```

By using abbreviations, the same example can be written as

```
SELECT i.desc, i.prod_cat, c.desc
FROM   item i, category c
WHERE  i.prod_cat = c.prod_cat
```

This feature saves time and effort when lengthy queries, using multiple tables and nested subqueries, are executed. Table abbreviations may also be required with some nested subqueries.

Outer Joins

For version 2.0, outer joins have been added to the SQL grammar. Outer joins preserve unmatched rows from tables, while inner joins disregard rows in which the specified search conditions are not met. Using outer joins with non-normalized data can provide more complete results. For example:

Table TAB1 contains the following data:

```
COL1  COL2
 15    20
 40    32
```

Table TAB2 contains the following data:

```
COL1  COL2
 15    35
 19    55
```

For a query executing an inner join such as

```
SELECT *
FROM   tab1, tab2
WHERE  tab1.col1 = tab2.col1
```

the result would be

```
COL1  COL2  COL1  COL2
 15    20    15    35
```

However, if the following outer join query is executed,

```
SELECT *
FROM   {OJ tab1 LEFT OUTER JOIN tab2
        ON tab1.col1 = tab2.col1}
```

the result would be

```
COL1  COL2  COL1  COL2
 15    20    15    35
 40    32  <null><null>
```

table_name.*

To increase the compatibility of the driver with some Microsoft database front-end applications, the **`table_name.*`** syntax, used by some Microsoft products such as Access and Visual InterDev, has been added to the grammar in the SQL engine. It is now possible to execute queries, such as the following, that select all the columns from one table and a few chosen columns from a second table:

```
SELECT item.*, category.desc
FROM   item, category
WHERE  item.prod_cat = category.prod_cat
```

The addition of this feature increases the flexibility and range of applications of the BASIS ODBC Driver.

{d 'date value'}

When using date constants in SQL queries, the **`{d 'date value'}`** syntax is no longer needed for date comparisons. For example, the following query

```
SELECT *
FROM   invoice
WHERE  invoice_date > {d '1993-10-15'}
```

can now be written as

```
SELECT *
FROM   invoice
WHERE  invoice_date > '1993-10-15'
```

Overall, these new additions to the grammar of the SQL engine significantly increase the power, flexibility, and compatibility of the BASIS ODBC Driver.

Additional Scalar Functions

Scalar functions allow the SQL engine of the ODBC driver to process the information returned by a query before it is actually returned to the executing application. The scalar functions added to the new version of the BASIS ODBC Driver improve the power and functionality of executed queries. As an example, the scalar function SOUNDEX allows the user to compare string values based not on the actual strings themselves but by the phonetic sound of the words, as shown:

```
SELECT customer.cust_num, customer.first_name,
       customer.last_name
FROM   customer
WHERE  SOUNDEX(customer.first_name) = SOUNDEX('JOHN')
```

This example returns those customers whose first names are John, Jon, Jawn, etc., because the SOUNDEX function returns a phonetic representation of the word. This function can be a tremendous time-saver, especially when used for name lookups in large databases.

New features included with the BASIS ODBC Driver version 2.0, such as nested SELECT statements and outer joins, cut development time dramatically. In addition, significant advancements in the execution time of ORDER BY clauses, combined with a driver that functions in a web environment with products such as Microsoft's Visual InterDev, make the BASIS ODBC Driver a valuable tool to increase the productivity of developers and users alike.

ODBC provides interconnectivity regardless of platform or application, and the new BASIS ODBC Driver contains many new features, including additional mathematical, date, and string functions to give data access and retrieval a customized look. In all, more than 100 scalar functions are now available for use in selects, updates, and inserts. A complete list of functions is available in the online documentation included with each new BASIS ODBC Driver. So, put the top down and go for a drive. The BASIS ODBC Driver makes data easier to get and just might put the fun back in driving.