Unleashing the Power of SPROCs Without SQL

By Jeff Ash

Stored procedures are a powerful and very flexible part of the SQL engine inside BBj®. They give developers a way to embed callable program logic in the database, thereby making it available to BBj and any other third party ODBC/JDBC application. One of the features stored procedures provide is the ability to return a result set to the client application. Prior to BBj 7.0, developers could return any result set based on an SQL query from an SQL channel. Now, stored procedures can return an SQL record set or the new SQL-less Memory Record Set, further expanding the power and flexibility of stored procedures. This article focuses on how to create a memory record set from information generated by an SQL-less query. Additionally, this article illustrates how to create a stored procedure programmatically.

What is an SQL Result Set?

Stored procedures have the option of returning nothing, a single value, or an entire result set of data to the client application. An SQL result set consists of zero or more records where each record is comprised of one or more columns of information such as name, address, phone number, etc.

What is a Memory Record Set?

The new Memory Record Set, created from BBjAPI: createMemoryRecordSet, makes it possible for a stored procedure to return a result set made up of arbitrary information that the stored procedure builds in any way the developer sees fit. For example, a stored procedure could use existing business logic with READ RECORD file operations to gather its information. Then, this information can be placed into a memory record set and sent back to the client application. The client application sees this information in the same way as a result set created from an SQL query. Thus, any BBj or third party ODBC/JDBC application can use it just like an SQL SELECT result.

Creating and Using a Memory Record Set

Creating a memory record set requires that the developer define the layout of the records to be contained in the record set. To do this, use the following string template:

```
rs! = BBJAPI().createMemoryRecordSet("ITEM_NUM:C(6)")
```

Now, adding values to the Memory Record Set is easy. This code snippet inserts a single record into the Memory Record Set:

```
data! = rs!.getEmptyRecordData()
data!.setFieldValue("ITEM_NUM", "000234")
rs!.insert(data!)
```

If the record set has more than one column defined in it, simply add more setFieldValue calls to set the additional column values. When the stored procedure has finished populating the record set, it must return that record set to the SQL engine so that the client application can receive the information. To do this, simply call the new setRecordSet method on the BBjStoredProcedureData object.

A Sample Stored Procedure Utilizing a Memory Record Set

The following CREATE PROCEDURE call creates a stored procedure named GET_ITEMS in the ChileCompany sample database that demonstrates using READ RECORD statements to populate a Memory Record Set, and returns that record set to the client application as an SQL result set. To create the stored procedure, execute the SQL code in **Figure 1** using the BBj Enterprise Manager. Right-click on the ChileCompany database, select the "Execute SQL" option, and paste in the code.

continued...

Jeff Ash
Software Engineer

```
CREATE PROCEDURE get_items (prod_cat CHAR(2) IN) RESULT_SET
{_BEGIN_}
    REM Open the file
    chan = UNT
    OPEN (chan) "C:\Program Files\basis\demos\chiledd\data\ITEM"

    REM Get the parameter value specified by the calling program
    sp! = BBJAPI().getFileSystem().getStoredProcedureData()
    prod_cat$ = sp!.getParameter("PROD_CAT")

    REM Create a memory record set to hold the results of
    REM our read operations.
    rs! = BBJAPI().createMemoryRecordSet("ITEM_NUM:C(6)")

    REM Iterate over the file and find the items that
    REM have the specified PROD_CAT
    TMPL$ = "ITEM_NUM:C(6),DESC:C(30),PROD_CAT:C(2),STOCK_UOM:C(3),"
    TMPL$ = TMPL$ + "COST:N(12),WEIGHT:N(12),WT_UNIT:C(2),PRICE:N(12)"
    DIM REC$:TMPL$
    while (1)
        READ RECORD (chan, ERR=eof) rec$
        if (rec.prod_cat$ = prod_cat$) then
            REM Found a match, so add it to the Record Set
            data! = rs!.getEmptyRecordData()
            data!.setFieldValue("ITEM_NUM", rec.item_num$)
            rs!.insert(data!)
        endif
    wend

eof:
    CLOSE (chan)

    REM Set the returned result set value to the record set.
    sp!.setRecordSet(rs!)
{_END_}
```

Figure 1. Sample stored procedure

Now that the database contains the stored procedure, it is ready for use. To test this stored procedure, simply execute the following SQL CALL statement directly from the Enterprise Manager's SQL Browser dialog:

```
CALL get_items('CH')
```

The results appear as a list of the items in the database with a PROD_CAT of **CH** as shown in **Figure 2**.

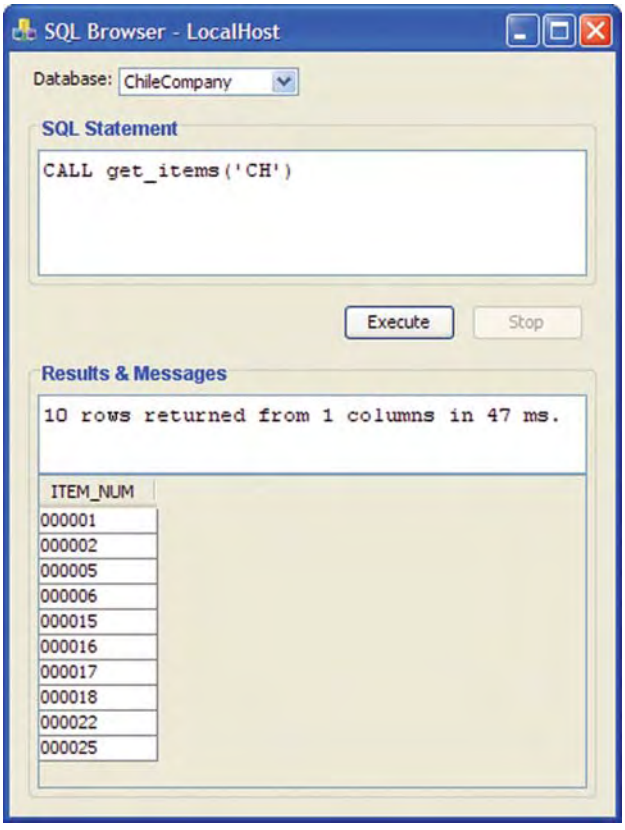

Figure 2. Result of the SQL CALL statement

Summary

Stored procedures offer a method of executing SQL on the server side of a client-server application, delivering enormous performance benefits when sorting through large volumes of data. Oftentimes, the developer can sort or process that data more efficiently using the fast READRECORD constructs of the BASIS language. Now using memory record sets, developers can more easily reuse existing business logic and data structures created in BBj, in their stored procedures, extending the power of the BBj language to the world of third party applications.