Using Stored Procedures to Add Business Logic to the Database

By Jeff Ash

Developers use a variety of methods and tools for building BBj® applications. However, the applications all consist of one or more programs that provide various functionality to the complete project. In some cases, it is beneficial to move some of the business logic into the database so that it is also accessible from third party ODBC or JDBC applications. BBj 6.0 provides this ability using stored procedures.

What is a Stored Procedure?

A stored procedure consists of a function or procedure written in BBj and embedded in the database. A new data dictionary file contains stored procedure information such as a definition of the parameters and features of the procedure, and points to a file containing the source code for the procedure. To access a stored procedure, a program executes an SQL CALL statement that specifies the name of the procedure and any necessary parameters. Using SQL makes it possible to call the procedure from BBj or any third party ODBC/JDBC application and get the same results.

Real-World Example

The best way to understand stored procedures and how they work in BBj is to walk through a simple real-world example. This example uses the Chile Company database included in the BBj installation. Suppose an application needs to acquire a list of customers who have ordered a particular item. Then suppose both a BBj application and a Web-based application need to access this information. One solution would be to provide all users with the appropriate SQL query so that the BBj and Web applications could use it. This would work fine until a change occurs in the query or the logic becomes more complex such that the query would not be able to provide the required result set. We could provide the changes to everyone so that they can update their code or we could use a stored procedure and centralize the logic.

The name of the stored procedure sample will be `LIST_CUSTOMERS` and will take a single argument – the item number for a product. Start by creating a new stored procedure in the Chile Company database:

1. Log in to the BBj Enterprise Manager.
2. Expand the **Databases** tree folder.
3. Locate the **Chile Company** database and expand its folder.
4. Locate the **Stored Procedures** folder and expand it.
5. Right click on the **Stored Procedures** folder and select **New Stored Procedure**.
6. Enter `LIST_CUSTOMERS` when prompted for the name of the procedure.
7. Click on the newly created `LIST_CUSTOMERS` node under the Stored Procedures folder.
8. In the right hand pane, enter `(DICTIONARY)list_customers.prc` for the Source Location field.
9. Place a check in the "Has Result Set" check box.
10. In the Parameter Specification list, click in the first cell and type the parameter name `ITEM_NUM`, set the SQL Type to `CHAR`, set the Direction to `IN`, and set Precision to `6`.
11. Click the [Save Changes] button.
12. Use the BASIS IDE or another text editor to save the following code sample in **Figure 1** (available for download) in your dictionary directory with the file name `list_customers.prc`:

continued...

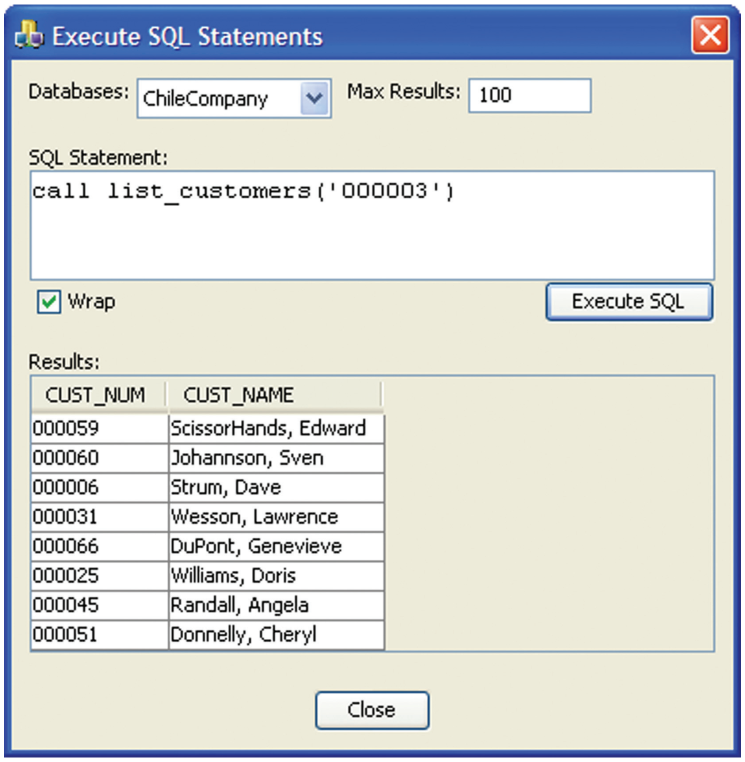
```
rem Get an object containing the parameter values passed into the procedure.
sp! = BBJAPI().getFileSystem().getStoredProcedureData()

rem Get the item number passed into the procedure
itemNum$ = sp!.getParameter("ITEM_NUM")

rem Open an SQL channel to run the query on. We want to use the same database.
chan = sqlunt
sqlopen(chan, mode="PROCEDURE")sp!.getDatabaseName()

rem Run the query that will get the appropriate information
sqlprep(chan)"select c.cust_num, cvs(c.last_name,3) + ', ' + cvs(c.first_name,3)
:    cust_name from customer c, order_header o, order_line l
:    where l.item_num = ? and o.order_num = l.order_num and c.cust_num = o.cust_num"

sqlexec(chan)itemNum$

rem Set the returning result set to be the result of the query.
sp!.setResultSet(chan)
```

Figure 1. Code sample `list_customers.prc`

Jeff Ash
Software Engineer

Partnership
Language/Interpreter
DBMS
Development Tools
System Administration

To test the example, open Microsoft Query or another third party ODBC/JDBC application and enter the SQL statement shown in the dialog box in **Figure 2**.



```
call list_customers('000003')
```

CUST_NUM	CUST_NAME
000059	ScissorHands, Edward
000060	Johannson, Sven
000006	Strum, Dave
000031	Wesson, Lawrence
000066	DuPont, Genevieve
000025	Williams, Doris
000045	Randall, Angela
000051	Donnelly, Cheryl

Figure 2. SQL statement

To call the procedure from a BBj application, execute `sptest.bbj` in **Figure 3** (also available for download):

```
rem Open up an SQL channel to the ChileCompany Database
rem and call the LIST_CUSTOMERS stored procedure

chan = sqlunt
SQLOPEN(chan) "ChileCompany"
SQLPREP(chan) "CALL LIST_CUSTOMERS('000010')"
SQLEXEC(chan)
DIM REC$:SQLTMPL(chan)

rem Print out the result set from the stored procedure
WHILE 1
    REC$=SQLFETCH(chan,end=*BREAK)
    PRINT REC$
WEND
```

Figure 3. Code sample `sptest.bbj`

Summary

Stored procedures offer a powerful new way for developers to add value and ease of maintenance to their BBj applications. Moving some of the common business and data access logic out of the BBj program into the database centralizes functionality in one place, making it accessible to the BBj program as well as any third party JDBC/ODBC application. Developers can extend the simple stored procedure example employed in this article to include complex BBj processing including CALL's to other BBj programs. All the powerful functions and features of the BBj language become available via an SQL CALL statement. A stored procedure's server-side processing delivers significant performance improvements over executing complex queries across the network as only the result set passes back across the network to the calling SQL statement. We encourage you to explore the myriad of new opportunities that you can take advantage of with this new BASIS DBMS feature.

Download the sample programs referenced in this article from
www.basis.com/advantage/mag_v10n1/sprocs.zip