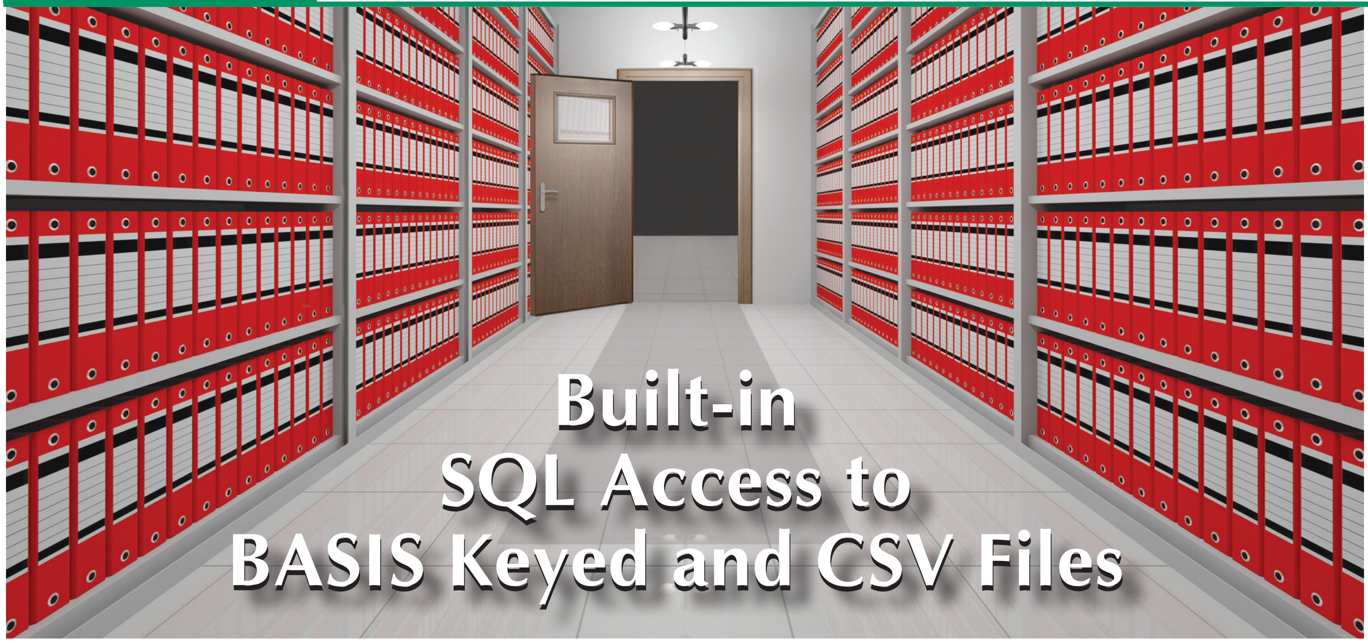


Built-in SQL Access to BASIS Keyed and CSV Files

Built-in stored procedures now provide SQL access to BASIS data files or CSV files with the familiar SQL 'CALL' syntax and forgo the typical prerequisite of a BASIS data dictionary or the need to write SQL syntax. Any SQL-enabled application can now access BASIS data files using this new functionality. Built-in SPROCs are an integral part of the BASIS RDBMS; the built-in BBJSYS database provides access to the new built-in SPROCs. The SQL 'CALL' syntax accesses built-in SPROCs through the BBJSYS or any other database.

Introducing the new SPROCs

These new built-in SPROCs, available in BBj® 12 and above, are always available and called from a connection to any available BBj database. If there is no database defined within BBj Services, use the system database to execute the stored procedure. The 'BBJSYS' database (built-in system database supplied with BBj Services) will execute the stored procedure without any need for a data dictionary.

BBj or third party applications can use the BBj ODBC Driver® or JDBC Driver to access the built-in SPROCs via the BBJSYS database. Here is a sample JDBC connection URL to the BBJSYS database:

```
jdbc:basis:localhost?DATABASE=BBJSYS&SSL=false&USER=admin&PASSWORD=admin123
```

Understanding how new SPROCs Work

Built-in stored procedures make use of a user-supplied file location and record template as parameters to the SQL CALL statement to return a result set based on the data from the file. The SQL syntax for use with the built-in stored procedure looks like this:

```
GET_RESULT_SET (file_path, template)
```

"File_path" is the path to the data file you wish to reference and "template" is the string template that describes the structure of the file.

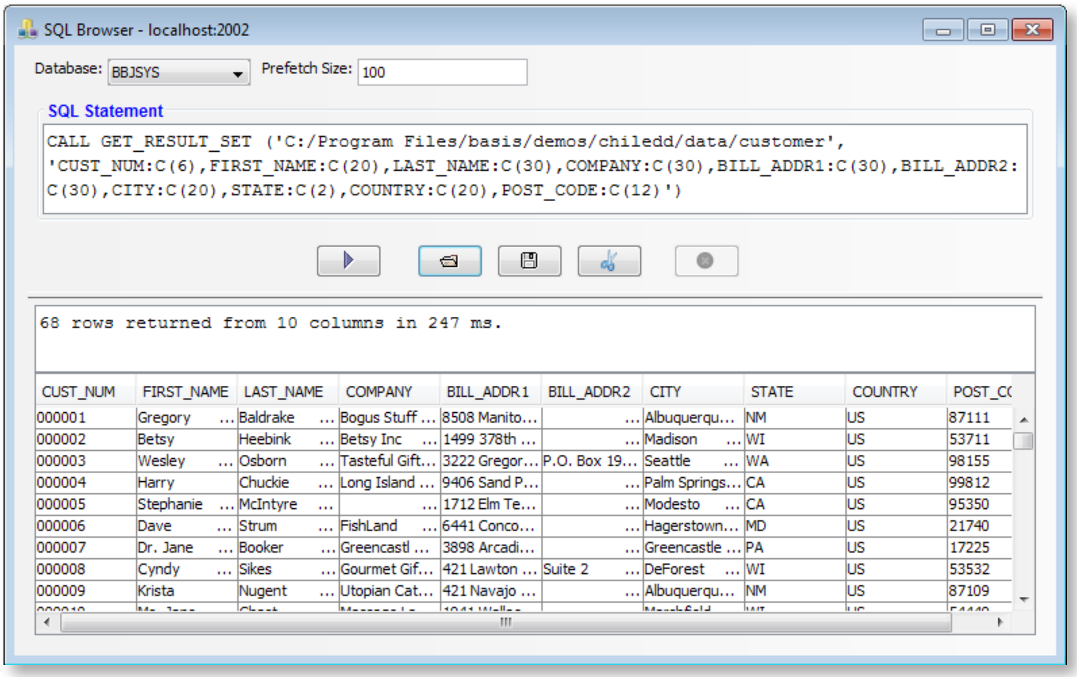
Using the new SPROCs

The following simple example retrieves information from the ChileCompany's CUSTOMER file (an MKEYED file) located in the <BBjHome>\demos\chiledd\data directory:

```
CALL GET_RESULT_SET
('C:/Program Files/basis/demos/chiledd/data/customer','CUST_NUM:C(6),FIRST_NAME:C(20),LAST NAME:C(30),
COMPANY:C(30),BILL_ADDR1:C(30),BILL_ADDR2:C(30),CITY:C(20),STATE:C(2),COUNTRY:C(20),POST_CODE:C(12)')
```

By Robert Del Prete
Quality Assurance Engineer

Executing this SQL statement in Enterprise Manager returns the data from the CUSTOMER file as an SQL result set, as shown in **Figure 1**.



```
CALL GET_RESULT_SET ('C:/Program Files/basis/demos/chiledd/data/customer',
'CUST_NUM:C(6),FIRST_NAME:C(20),LAST_NAME:C(30),COMPANY:C(30),BILL_ADDR1:C(30),BILL_ADDR2:
C(30),CITY:C(20),STATE:C(2),COUNTRY:C(20),POST_CODE:C(12)')
```

```
68 rows returned from 10 columns in 247 ms.
```

CUST_NUM	FIRST_NAME	LAST_NAME	COMPANY	BILL_ADDR1	BILL_ADDR2	CITY	STATE	COUNTRY	POST_C
000001	Gregory …	Baldrake …	Bogus Stuff …	8508 Manito…	…	Albuquerqu…	NM	US	87111
000002	Betsy	Heebink …	Betsy Inc …	1499 378th …	…	Madison …	WI	US	53711
000003	Wesley …	Osborn …	Tasteful Gift…	3222 Gregor…	P.O. Box 19…	Seattle …	WA	US	98155
000004	Harry	Chuckie …	Long Island …	9406 Sand P…	…	Palm Springs…	CA	US	99812
000005	Stephanie …	McIntyre …	…	1712 Elm Te…	…	Modesto …	CA	US	95350
000006	Dave …	Strum …	FishLand …	6441 Conco…	…	Hagerstown…	MD	US	21740
000007	Dr. Jane …	Booker …	Greencastl …	3898 Arcadi…	…	Greencastle …	PA	US	17225
000008	Cyndy …	Sikes …	Gourmet Gif…	421 Lawton …	Suite 2 …	DeForest …	WI	US	53532
000009	Krista	Nugent …	Utopian Cat…	421 Navajo …	…	Albuquerqu…	NM	US	87109

Figure 1. The built-in SPROC via SQL in Enterprise Manager

The call in **Figure 1** to the GET_RESULT_SET SPROC returns the first ten fields, as defined by the provided string template, from all records within the CUSTOMER file. This is extremely helpful when a data dictionary is not available for the data file. Utilizing iReport together with the built-in GET_RESULT_SET SPROC allows developers to provide stunning reports without the overhead of creating a data dictionary or writing SQL syntax for their customers' traditional file system-based data.

Understanding Advanced Data Manipulation

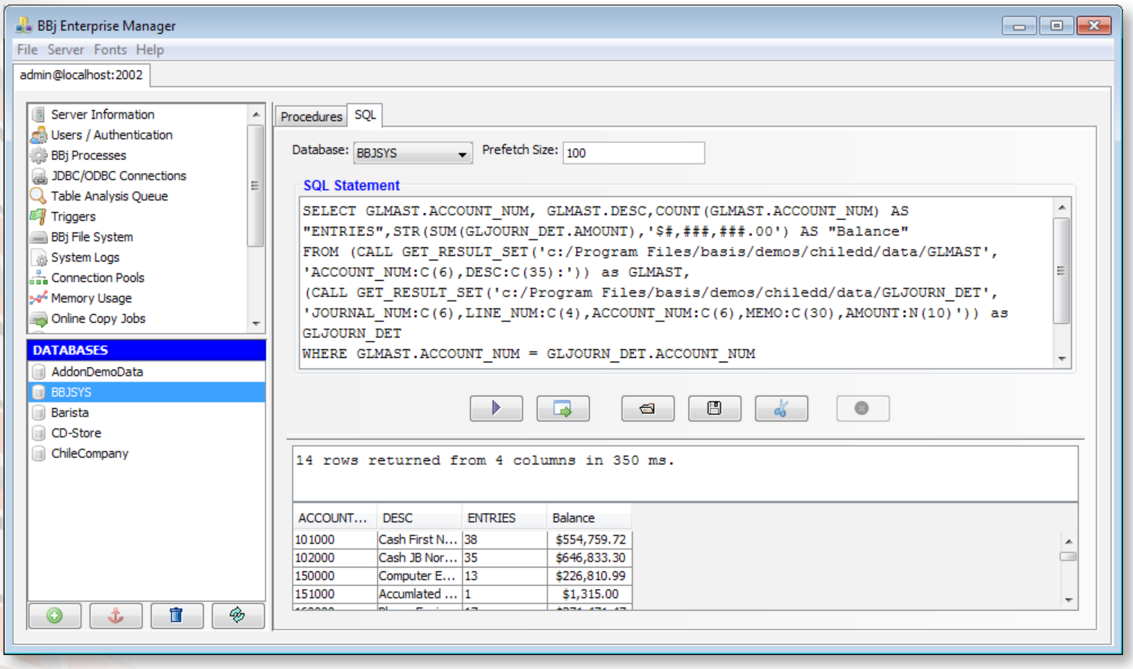
BASIS' SPROCs are very versatile and developers can utilize the SQL language to treat a SPROC's result set as a table to join, group, filter, and order the returned data. SQL clauses such as WHERE, ORDER BY and JOIN allow programmers to maximize their control over the returned data to suit the needs of the application and end users.

The example in **Figure 2** uses the GET_RESULT_SET stored procedure to pull data from two different files. The statement joins the results of two CALLs to GET_RESULT_SET SPROC, utilizing the WHERE clause to match the account numbers in the two tables. It then uses a GROUP BY to group the returned data by ACCOUNT_NUM and DESC, finally ordering the result set by account number.

```
SELECT
 GLMAST.ACCOUNT_NUM, GLMAST.DESC,COUNT(GLMAST.ACCOUNT_NUM) AS "ENTRIES",
 STR(SUM(GLJOURN_DET.AMOUNT),'$#,###,###.00') AS "Balance"
FROM
 (CALL GET_RESULT_SET
   ('c:/Program Files/basis/demos/chiledd/data/GLMAST',
     'ACCOUNT_NUM:C(6),DESC:C(35):')) as GLMAST,
     (CALL GET_RESULT_SET
       ('c:/Program Files/basis/demos/chiledd/data/GLJOURN_DET',
         'JOURNAL_NUM:C(6),LINE_NUM:C(4),ACCOUNT_NUM:C(6),MEMO:C(30),AMOUNT:N(10)'
       )
     ) AS GLJOURN_DET
WHERE
 GLMAST.ACCOUNT_NUM = GLJOURN_DET.ACCOUNT_NUM
GROUP BY
 GLMAST.ACCOUNT_NUM, GLMAST.DESC
ORDER BY
 GLMAST.ACCOUNT_NUM
```

Figure 2. Advanced SQL query utilizing the built-in SPROC to pull data from two files

This SQL statement returns the account numbers in ascending order along with the description for each account, the number of entries, and the current balance on the account without the use of a data dictionary, as shown in **Figure 3**.



Database: BBJSYS
Prefetch Size: 100

```
SELECT GLMAST.ACCOUNT_NUM, GLMAST.DESC,COUNT(GLMAST.ACCOUNT_NUM) AS
"ENTRIES",STR(SUM(GLJOURN_DET.AMOUNT),'$#,###,###.00') AS "Balance"
FROM (CALL GET_RESULT_SET('c:/Program Files/basis/demos/chiledd/data/GLMAST',
'ACCOUNT_NUM:C(6),DESC:C(35):')) as GLMAST,
(CALL GET_RESULT_SET('c:/Program Files/basis/demos/chiledd/data/GLJOURN_DET',
'JOURNAL_NUM:C(6),LINE_NUM:C(4),ACCOUNT_NUM:C(6),MEMO:C(30),AMOUNT:N(10)')) as
GLJOURN_DET
WHERE GLMAST.ACCOUNT_NUM = GLJOURN_DET.ACCOUNT_NUM
```

```
14 rows returned from 4 columns in 350 ms.
```

ACCOUNT...	DESC	ENTRIES	Balance
101000	Cash First N...	38	$554,759.72
102000	Cash JB Nor...	35	$646,833.30
150000	Computer E...	13	$226,810.99
151000	Accumlated ...	1	$1,315.00

Figure 3. SQL using the built-in SPROC in Enterprise Manager

Retrieving Data From a CSV File

A second type of built-in SPROC is also available for returning data from a standard CSV (comma-separated values) file using this syntax:

```
GET_RESULT_SET_CSV (file_path, template, delimiter)
```

As with the previous built-in SPROC, the file_path parameter denotes a fully qualified path to the CSV file. The template used for a CSV file simply describes the fields for each row and the delimiter determines the separator between each row of data in the text file. The SPROC expects the actual bytes of the delimiter rather than a string representation of the characters. In the case of a linefeed, a CHAR(10) would be specified in the SQL statement instead of the common '\n' string representation of the newline character. The same sorting and filtering capabilities are available to this SPROC as well. Consider the following CSV file format:

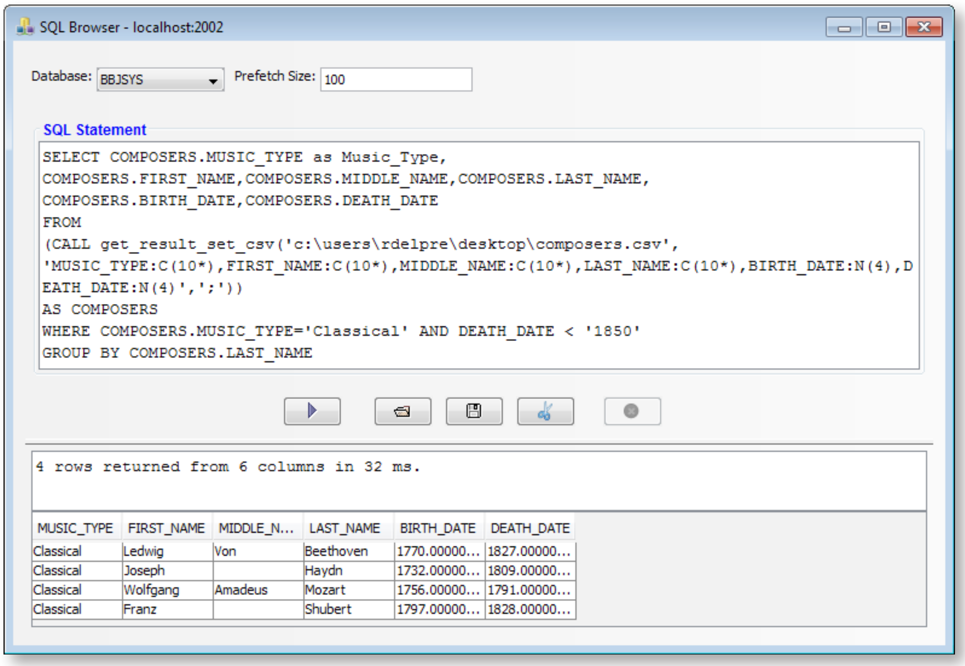
```
Baroque,George,Fredric,Handel,1685,1759
Classical,Joseph,,Haydn,1732,1809
Romantic,Carl,Maria,von Weber,1786,1826
```

Creating the SQL statement to access data in the CSV text file via the SPROC is easy. Insert the file path and create the string template by following the format of the CSV file. Include the delimiter for the records, in this case a line feed represented by CHAR(10). Filter the composers using MUSIC_TYPE and exclude any born after 1800. Lastly, group the results according to the composer's last name. The SQL statement code in **Figure 4** uses the built-in SPROC for a CSV file, selects all of the fields, sorts by music type and birth date, and then orders by the last name.

```
  SELECT
  MUSIC_TYPE,FIRST_NAME,MIDDLE_NAME,
  LAST_NAME,BIRTH_DATE,DEATH_DATE
FROM
  (CALL GET_RESULT_SET_CSV
    ('c:\users\data\desktop\composers.csv',
      'MUSIC_TYPE:C(10*),FIRST_NAME:C(10*),MIDDLE_NAME:C(10*),'+
      'LAST_NAME:C(10*),BIRTH_DATE:I(4),DEATH_DATE:I(4)',
      CHAR(10)
     )
  ) AS COMPOSERS
WHERE
  MUSIC_TYPE='Classical' AND BIRTH_DATE < '1800'
GROUP BY
  LAST_NAME
```

Figure 4. Sample of SQL using the built-in SPROC for a CSV file

The result of the code sample appears in **Figure 5.**



Database: BBJSYS   Prefetch Size: 100

SQL Statement

```
SELECT COMPOSERS.MUSIC_TYPE as Music_Type,
COMPOSERS.FIRST_NAME,COMPOSERS.MIDDLE_NAME,COMPOSERS.LAST_NAME,
COMPOSERS.BIRTH_DATE,COMPOSERS.DEATH_DATE
FROM
(CALL get_result_set_csv('c:\users\rdelpre\desktop\composers.csv',
'MUSIC_TYPE:C(10*),FIRST_NAME:C(10*),MIDDLE_NAME:C(10*),LAST_NAME:C(10*),BIRTH_DATE:N(4),D
EATH_DATE:N(4)',';'))
AS COMPOSERS
WHERE COMPOSERS.MUSIC_TYPE='Classical' AND DEATH_DATE < '1850'
GROUP BY COMPOSERS.LAST_NAME
```

```
4 rows returned from 6 columns in 32 ms.
```

MUSIC_TYPE	FIRST_NAME	MIDDLE_N...	LAST_NAME	BIRTH_DATE	DEATH_DATE
Classical	Ledwig	Von	Beethoven	1770.00000...	1827.00000...
Classical	Joseph		Haydn	1732.00000...	1809.00000...
Classical	Wolfgang	Amadeus	Mozart	1756.00000...	1791.00000...
Classical	Franz		Shubert	1797.00000...	1828.00000...

Figure 5. SQL using the built-in GET_RESULT_SET_CSV SPROC

Summary

The new built-in SPROCs allow access to any BASIS or CSV formatted data files regardless of the presence of a data dictionary. Developers can leverage this new feature in many ways, supplying new access to data by end users. Being able to filter and sort the data using SQL syntax also provides additional flexibility for the distribution and presentation of that data. ■

- Read more about built-in SPROCs in the online documentation at links.basis.com/bisprocs
- Check out these articles
  - *Using Stored Procedures to Add Business Logic to the Database* at links.basis.com/06sprocs
  - *Unleashing the Power of SPROCs Without SQL* at links.basis.com/07sprocs
  - *Recipes for Successful Report Writing* at links.basis.com/09reportwriting

BARISTA®
Rapid
Application
Development
BARISTA® the ultimate RAD tool for database-driven software projects built with BBj®, the dynamic object-oriented programming language for Java
BASIS International
www.basis.com