DB2 User Defined Functions and tableBASE

Summary: This document describes how to use DB2 user-defined functions (UDF) to access tableBASE.

Prerequisites:

- A basic understanding of tableBASE (access to the tableBASE Programmer's Guide and tableBASE CBT[1])
- Assistance from your System Administrator
- A basic understanding of SQL and writing applications for DB2

tableBASE data can now be accessed through a DB2 user-defined function (UDF). Using a standard DB2 query, tableBASE data can be returned in a relational database table. So go ahead and write DB2 applications that access tableBASE data using all the DB2 facilities to which you're accustomed!

What is a User-Defined Function?

A user-defined function (UDF) is very similar to a stored procedure (SP) or a host language program. However, a UDF is often the better choice for an SQL application as you can invoke a UDF in both Static and Dynamic SQL statements (like a built-in function in DB2).

UDFs are part of the DB2 object-relational extensions. These custom functions allow programmers to extend the DB2 built-in data-types and functions. With a UDF, you can write your own routine in any host language (C, C++, Assembler, COBOL and PL/1) and call these functions with an SQL statement. The UDF needs to be written only once and then stored in a central place. The operations that users can perform with those functions are limited by DB2 SQL privileges.

The UDF runs in the stored procedure's address space that is established in the Workload Manager (WLM) environment. The environment setup for UDF and Stored Procedure (SP) is the same.

Sourced vs. External UDF

There are two types of UDFs: sourced and external. Sourced UDFs are based on existing UDFs or DB2 built-in functions, and are usually written within the “CREATE FUNCTION” SQL statement. External UDFs can be written in any host language.

Scalar vs. Table Functions

UDFs can also be categorized as user-defined scalar functions and user-defined table functions. A user-defined scalar function returns a single-value after each invocation. A user-defined table function returns a table to the SQL statement that references it. External

[1] These are available free of charge to our maintenance customers. Please contact tableBASE@dkl.com or call us at 613-523-5588.

UDFs can be user-defined scalar functions or user-defined table functions. However, sourced UDFs cannot be user-defined table functions.

MainProgram vs. SubProgram

UDFs can be created as either a mainProgram or a subProgram. The way that you code your program must agree with the way you define the UDF in the “CREATE FUNCTION” SQL statement with the “PROGRAM TYPE MAIN” or “PROGRAM TYPE SUB” parameter.

A mainProgram will get program storage from LE/370 language environment. The language environment will close the files and dynamically release allocated storage after the main program has ended.

A subProgram will need to manage the storage and files by itself. However by controlling these resources, you will get better performance. Please note that an external user-defined table function must be coded as a subProgram.

Sourced User-defined scalar function	Return single value	Sub or main
External User-defined scalar function	Return single value	Sub or main
External User-defined table function	Return a table	Sub only

Creating a UDF

There are a number of steps involved in creating a UDF to access tableBASE.

1. Set up the environment for the UDFs. The UDF environment is the same as that for the WLM established address spaces for a stored procedure.

2. Set up the environment for a tableBASE VTS.

3. Write and prepare the UDF in a host language. Code the external program with tableBASE functions and use the “callType” variable to control the program flow.

4. Define and register the UDF with DB2 by executing the “CREATE FUNCTION” DDL statement against the target DB2 instance.

5. Grant execution authority to appropriate users.

Step 1

Your system administrator will be able to establish the appropriate environments for WLM and the tableBASE VTS, and provide you with the WLM-established address space name and tableBASE VTS name. The system administrator should set up the DB2 WLM-establish address space with recoverable resource services attachment facility (RRSAF).

For ease of implementation, all tableBASE tables that are to be accessed by the UDFs should be loaded into a VTS-TSR before the UDF is called.

Step 2

UDF for tableBASE can be written in assembler, C, C++, COBOL, or PL/1. Before you start to define the UDF to access tableBASE data through DB2, you must determine the characteristics of the UDF. The characteristics include the UDF name, schema, number and data types of the input parameters, and the types of values that will return from tableBASE and get converted into DB2 data types.

If you discover after you define the function that any of these characteristics is not appropriate for the function, you can use an “ALTER FUNCTION” statement to change the information or use a “DROP SPECIFIC FUNCTION” to delete the function.

UDF Create Function SQL Example

```
CREATE FUNCTION DKEXAMP (CHAR(4), CHAR(8))
       RETURNS TABLE (LASTNAME CHAR(20),
       FIRSTNAME CHAR(14), DIVISION CHAR(8),
         DEPARTMENT CHAR(8), SEX CHAR(1),
         CHARITABLE_DONA INTEGER,
         DATE_OF_CONTRIB CHAR(10))
         FENCED
       SPECIFIC DKEXAMP
        EXTERNAL NAME 'DK1UDFEX'
        NO SQL
        PARAMETER STYLE DB2SQL
        LANGUAGE C
         WLM ENVIRONMENT DEFAULT
        PROGRAM TYPE SUB
        NO COLLID
        NO FINAL CALL
         DISALLOW PARALLEL
        NO EXTERNAL ACTION
         DETERMINISTIC
         SCRATCHPAD
```

UDF Grant SQL Example

```
GRANT EXECUTE ON SPECIFIC FUNCTION AARON0.DKEXAMP TO PUBLIC
```

Since tableBASE data is external to DB2, we will use an external UDF written in C as an example throughout this document to explain how to use a UDF to access tableBASE. The next section discusses how to write a user-defined table function in a C sub program to access tableBASE table and return it as a DB2 table.

C sub program

First, we have to initialize some tableBASE data structures with appropriate values before passing it to the TBLBASE assembler routine. You can find out more detailed information about these structures and TBLBASE in tableBASE Programmer's Guide. Then the program has to get the parameters from the SQL statement and

convert them into a format that tableBASE expects. As tableBASE expects data in fixed length and uppercase characters, your program needs to append space or truncate the data and convert them into uppercase.

An external UDF is controlled by DB2. DB2 will invoke the subprogram repeatedly based on the different execution stages, and each time pass a different call type parameter to the subprogram. The external subprogram needs to follow the callType parameter to control the program flow. Because of this the program needs to execute a Change Status (CS) tableBASE command to enable the tableBASE setting each time the function is invoked.

Since all SPs and UDFs can run under the same instance of DB2. Any SP or UDF can change the tableBASE status-switches while the other SP or UDF is running. If the UDF is executed many times by DB2 to get each row of table, the status-switches may get changed from one call to the next. There are a number of ways to handle the status switches being changed between calls. The tableBASE administrator can set up the default switch in their tableBASE DB2 interface [the source code is comes with tableBASE in TBASE.V600.I14.SRC(DK1T1434)] then tableBASE will also use the same status-switches settings with all calls. Or if any program wants to change the status-switch, they can do an LS command and store the original status-switches, and before they exit each execution stage, they can then restore the status-switch to the original setting.

Note: Please refer to the tableBASE Programmer's Guide for details on these status-switches. The ABEND switch should be turned off, so that the program would not ABEND by tableBASE while DB2 is executing tableBASE commands. Otherwise, this may cause DB2 to rollback.

The overall program flow is based on the “callType” variables from the UDF parameter to indicate which section of code should be executed at each different stage of execution. If the “FINAL CALL” parameter is specified in the “CREATE FUNCTION” SQL statement, then the “callType” variable will pass “SQLUDF_TF_FIRST”, “SQLUDF_TF_OPEN”, “SQLUDF_TF_FECTH”, “SQLUDF_TF_CLOSE”, “SQLUDF_TF_FINAL” and “SQLUDF_TF_FINAL_CRA”. Otherwise, it will only pass the “SQLUDF_TF_OPEN”, “SQLUDF_TF_FECTH”, “SQLUDF_TF_CLOSE” and “SQLUDF_TF_FINAL”. These values are used to help the programmer specify which part of the program should be executed.

Scratchpad is a feature of DB2 UDF that allows a program to save information between different stages of invocation. Specifically, we saved part of the tableBASE COMMAND-AREA in scratchpad ,in order to get faster access to tableBASE through the saved COMMAND-AREA on the next pass. Also, you should set the appropriate value in the sqlstate after each invocation. The “SQLUDF_TF_FETCH” is called by DB2 multiple times until the program returns a sqlstate “02000” or other error code. Other callType parameters will only be called once while the program is executed. You can look at the example to get more details of the program flow.

The subprogram example (DK1UDFEX) is provided as an additional download.

The user-defined table function can use the IBM supplied procedure (DSNHC) to precompiled, compile, pre-link and link-edit the program, and then the load module needs to be moved to the dataset specified for the workload managed stored procedure address space.

The JCL and PROC to compile the program example (DSNHC[2] and DB2TCMPS) are provided as an additional download.

Step 3 and Step 4

To register the UDF information in the DB2 catalog (SYSROUNTINE), execute a "CREATE FUNCTION" statement. In order for the DB2 UDF to locate the tableBASE library, use tableBASE with VTS. By doing this you won't have to change the SYSLIB in the DB2 startup JCL or store the library list in the UDF.

Step 5

Finally, grant the execution authority to appropriate users.

The DB2 UDF can now be invoked by any SQL statement in any DB2 application and used to access tableBASE data. Because this facility is being used through DB2, all of the data type conversions are done for you and the data can easily be moved to distributed platforms through DB2 tools such as DB2 Connect.

Notes:

1. The IBM procedure (DSNHC) must be changed from the compiler EDCDC120 to CBCDRVR.

2. UDF sub program must start with this pragma macro.
   `#pragma linkage (your_function_name, fetchable)`

3. The DSNALI in the linkedit step is needed to include support for the language environment interface module with your program. If you did not specify this in the linkedit step, the linkedit will still successfully complete but it will not be invoked. The UDF is running under the DB2 subsystem WLM-established address space, it requires the RRSAF (Recoverable Resource Services Attachment Facility), so remember to change the SYSLIB from DSNALI to DSNRLI.

4. Storing the TB-PARM and COMMAND-AREA to the scratchpad may not be ideal since if the UDF is running under WLM-established address space under goal mode the tableBASE region may get moved to different address space by WLM. If this happens, the UDF will not know that the region has changed and will try to use the same TB-PARM to access tableBASE and this may cause unpredictable results.

[2] DSNHC is an IBM provided program.

5. About the UDF error handling; DB2 uses SQLSTATE to indicate errors. Follow this convention to warn users that an error has occur during the UDF execution. SQLSTATE is 5 bytes long characters.

- "00000" 5 zeros in character format. If everything is working perfectly in the program, the SQLSTATE should set to "00000"
- "01xxx" zero and one with 3 alphanumeric characters indicates warnings.
- "38xxx" indicates UDF errors. 38001-38004 are already defined in the DB2 manual. "381xx" was used to indicate the tableBASE errors. For example, if tableBASE error code is 2, then the user-defined table function error code will be "38102". The only exception is tableBASE error "1072 VTS is not available". The error code for tableBASE error 1072 is "39172".