Searching source physical file members using IBM OmniFind Text Search server for DB2 for i 7.1

Jian Li
Hui Ruan

March 14, 2013

This article explains how to use IBM® OmniFind® Text Search Server for DB2 for i to index and search source physical files with multiple members. This support provides search capabilities for this IBM i object type that are similar to what is commonly available on the World Wide Web.

Overview

The IBM OmniFind Text Search Server product for DB2 for i (5733-OMF) is a no-additional charge product that can be used to search for different types of documents that are stored in either a database column or in a supported IBM i object.

In IBM i 7.1, OmniFind added a solution to allow users to index and search the text data associated with IBM i spool files and integrated file system (IFS) stream files. Consult the references section of this article to learn about previous extensions to OmniFind capabilities.

A common business scenario is to have text data stored in the members of a source physical file. Some applications might need to retrieve specific members based on keywords and phrases within those members. As part of the Technology Refresh 5 (TR5) enhancements to DB2 for i, OmniFind now provides the capability to index and search multiple member source physical files.

It's important to understand that this article focuses on searching and indexing members of source physical files, rather than searching for rows in an SQL table, or for the individual records that exist in members of database physical files. Customers who are interested in searching for rows in an SQL table (including partitioned tables) should use the CONTAINS and SCORE built-in SQL functions. Although these functions can also be used to search database physical files that have a single member, the CONTAINS and SCORE built-in SQL functions do not currently support searching nonpartitioned, multiple member physical files. For more detail on how CONTAINS and SCORE work, refer to the OmniFind information center.

Software requirements

In order to use the functions described in this article, it is necessary to order and install OmniFind Text Search Server for DB2 for i V1R2 (5733-OMF) and apply DB2 PTF Group SF99701 Level

18 or later. In addition, OmniFind requires a few other products to be installed on the system. The required software products are documented in the OmniFind information center.

Definitions and terms

Before getting into how to index and search members of a source physical file, we need to explain some key terms and definitions.

Text search collection: A text search collection is an SQL Schema that contains tables for tracking the indexed objects, and the SQL procedures for administering and searching the index.

Object set: An object set defines a set of objects that will be included in the text search collection during the update process. If a source physical file is added to collection, the object set has been expanded to include the members of the physical file. Each member of the source physical file will be treated as an object.

Object: An object is treated as a single document to be indexed. If an object set is defined as the members of a source physical file, every member of the file will be recognized as an object that can be a result of the search. The search result returns all the members that contain the keywords that match the search criteria.

Scheduled update: Each collection can have a scheduled update associated with it. While creating text search collection, this attribute can be specified using the UPDATE FREQUENCY clause. The default behavior if NONE is specified is that no further index updates are made. The update must be started manually. This option might be useful for a text column in which no further changes are planned.
If the text search collection was created as scheduled, it will be updated following the schedule defined by user using the SYSPROC.SYSTS_UPDATE stored procedure.

Topics covered

This article will walk through an example that covers these tasks:

- Create a source physical file with several members that contain unstructured text.
- Create an empty text search collection.
- Add a source physical file object set to the text search collection.
- Update the text search collection.
- Search the text search collection.

Besides these main features discussed in this article, there are still some other topics presented.

There is a section that shows how to upgrade a text search collection from a previous release to the latest version.

The final section shows a code example in embedded SQL that uses these new features.

Source physical file

A source physical file contains source data needed to create objects such as control language (CL) source statements, which are used to create a CL program, or used as plain text directly.

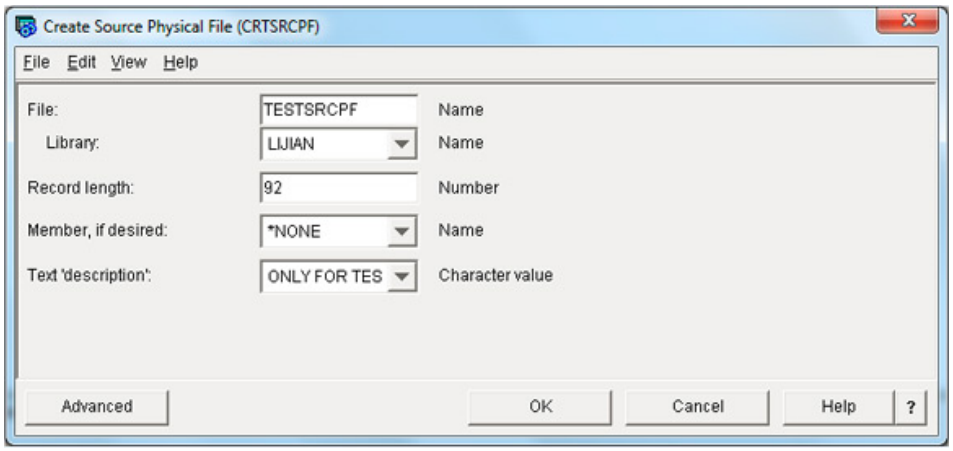
The following steps show how to create a source physical file. A source physical file can contain one or more members. The maximum number of members that can be added to the file is specified when the file is created using the Create Source Physical File (CRTSRCPF) CL command.

Command to create a source physical file:

```
CRTSRCPF FILE(LIJIAN/TESTSRCPF) TEXT('ONLY FOR TEST') MAXMBRS(*NOMAX)
```

The IBM Navigator for i product provides a graphical interface, which can be used instead of the command line, as shown in Figure 1.

Figure 1. Create source physical file



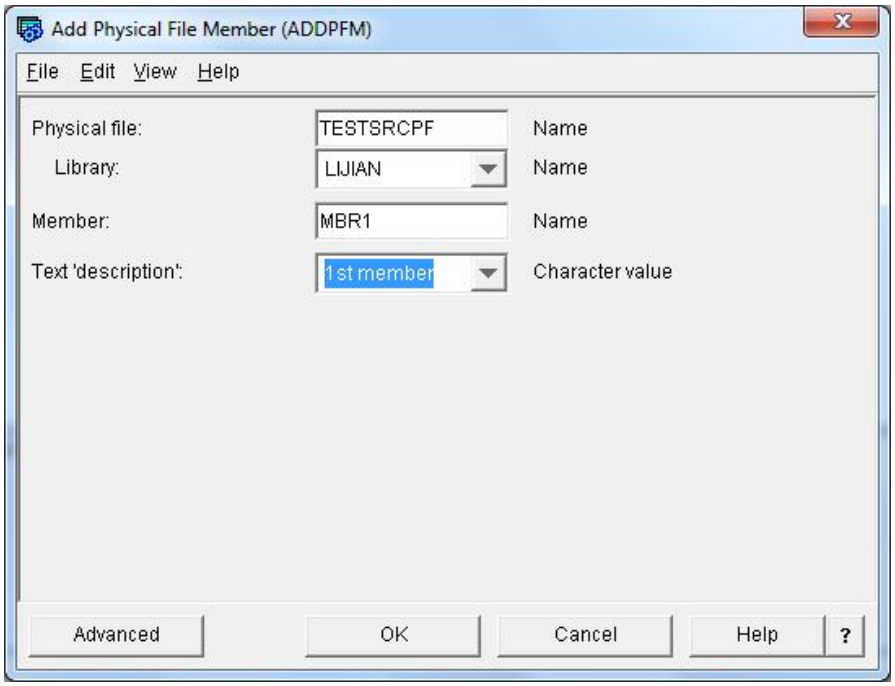
The source physical file was created as an empty file without members. Then new members can be added to the source physical file.

The following command adds a member named MBR1 to the source physical file.

```
ADDPFM FILE(LIJIAN/TESTSRCPF) MBR(MBR1)
```

The IBM i Navigator alternative function is shown in ↱Figure 2.

Figure 2. Add a physical file member



For the purposes of this example, three members have been created and a single sentence has been added to each member. This member contents are shown in ↱Listing 1.

Listing 1. Content of multiple members of source physical file

```
MBR1: "You gave me a big surprise!"
MBR2: "OmniFind supports linguistic features."
MBR3: "Searching mouse, all documents include mice also will be returned."
```

The sections that follow will discuss how to index and search for this data that has been added to the members.

Creating a text search collection

Before indexing and searching documents, you must create a text search collection.

Creating a text search collection is performed by invoking the SYSPROC.SYSTS_CRTCOL stored procedure.

Listing 2. Creating a text search collection

```
CALL SYSPROC.SYSTS_CRTCOL('COLLECTION_FOR_USER');
```

The CALL in ↱Listing 2 causes an SQL schema COLLECTION_FOR_USER to be created on the system. The schema contains all of the IBM DB2® objects associated with this text search, including the catalogs and administrative procedures.

The create text search collection procedure has several interesting options available for it that mirror the options used to create a DB2 text search index. The stored procedure parameters allow the user to configure the index update frequency, language, text format, and coded character set identifier (CCSID) of the text. The syntax and explanation of each option is described in detail in the OmniFind information center.

Listing 3 shows how to create a text search collection that gets updated every day, starting at midnight.

Listing 3. Create scheduled text search collection

```
CALL SYSPROC.SYSTS_CRTCOL('COLLECTION_FOR_USER',
'UPDATE FREQUENCEY D(*) H(0) M(0)');
```

Setting the path to the current collection

The OmniFind administrative procedures are created inside of the schema for the text search collection. The SET PATH SQL statement can be used to indicate which collection is being used. By specifying the path, you can avoid the need to explicitly qualify each procedure call with a schema. For simplicity, the other examples in this article assume the SQL path is set to above value.

```
SET CURRENT PATH COLLECTION_FOR_USER;
```

Adding object sets

After creating the text search collection, the documents that will be indexed must be added to the text search collection.

OmniFind supports three types of object sets. These object sets can co-exist within the same text search collection:

- Spool files within an output queue
- IFS stream files within an IFS directory
- Members within a source physical file

Adding an object set does not update the index with the text data from that object set. OmniFind updates the index with the text from these objects during the next update, which will happen when the UPDATE stored procedure is called or when a scheduled update runs.

One source physical file can be added to a text search collection as an object set. Each member of this physical file will be added into the text search collection as an object. After invoking the UPDATE stored procedure, the content of the members are indexed.

Adding source physical file object sets

After the text search collection is created, the `ADD_SRCPF_OBJECT_SET` procedure can be used to add the source physical file's members to the text search collection.

As shown in Listing 4, calling the `ADD_SRCPF_OBJECT_SET` procedure will add the source physical file `LIJIAN/TESTSRCPF`.

Listing 4. Add a source physical file object set

```
CALL ADD_SRCPF_OBJECT_SET('LIJIAN','TESTSRCPF');
```

↱Listing 5 shows the same procedure call, except this call includes an output parameter 'setid'. This parameter is an output integer value that returns an identifier for the object set that was added; the object set identifier can be used to remove the object set at a later time.

Listing 5: Add an object set with output variable

```
SET CURRENT SCHEMA COLLECTION_FOR_USER;
SET CURRENT PATH COLLECTION_FOR_USER;
CREATE VARIABLE SETID INT DEFAULT 0;
CALL ADD_SRCPF_OBJECT_SET ('LIJIAN', 'TESTSRCPF', SETID);
```

While adding a source physical file object set to a text search collection, OmniFind verifies that the source physical file exists. If the source physical file does not exist, an error message is returned and the object set is not added. If the file specified is not a source physical file, the procedure call fails with an error.

Note that if the source physical file is deleted after adding the object set to the text search collection, the subsequent call to the UPDATE stored procedure will ignore the indexing of the removed source physical file. There is no error returned. The search result will still contains the members of that source physical file that was removed until user removes the query object set.

The following statement shows how to remove an object set.

```
CALL REMOVE_OBJECT_SET (setid);
```

Updating the text search collection

The object sets included in the text search collection are not indexed until an update is performed. This can either be a scheduled update that is configured when the text search collection is created (as shown in ↱Listing 3) or a manual update by calling the UPDATE stored procedure.

```
CALL UPDATE;
```

The update processing will determine which objects are new or changed on the system and index the text data from those objects. The initial update will index all the objects included in the object set. Updates after the initial update are incremental; unchanged objects that have already been indexed will not be indexed again by incremental update. Some processing time will be spent at the beginning of each update process to determine which objects have been created, deleted, or changed.

After the UPDATE process has been completed, the text search collection is now ready to be searched.

Searching the text search collection

Searching the text search collection is done by calling the SEARCH stored procedure. The search expression works a lot like the web search syntax that users are familiar with. There are some advanced features that are documented in the IBM i 7.1 Information Center for the programmers who require advanced capabilities, but the basic syntax is intuitive.

```
CALL SEARCH ('give');
```

No matter whether the variation of the search term is "gave", "given" or "give", MBR1 will be returned since OmniFind includes linguistic variations in the search.

The SEARCH stored procedure returns a result set to the client application, with the most relevant results being ordered first.

The result set contains following columns:

```
OBJTYPE    object type
OBJATTR                  object attribute
CONTAINING_OBJECT_LIB          containing object library
CONTAINING_OBJECT_NAME          containing object name
OBJECTINFOR    object information
MODIFY_TIME    modification time
SC                          score value
```

These columns provide additional information about the search result and can be used by the application to filter results. They are described in detail in the OmniFind information center.

The most interesting column in the result set is the OBJECTINFOR column, which contains the location of the indexed object. XML was chosen as the data type for this column, both because of its flexible structure and also because of the wide variety of tools and parsers available for working with XML data.

XML is human readable, a matching source physical file location could look like this:

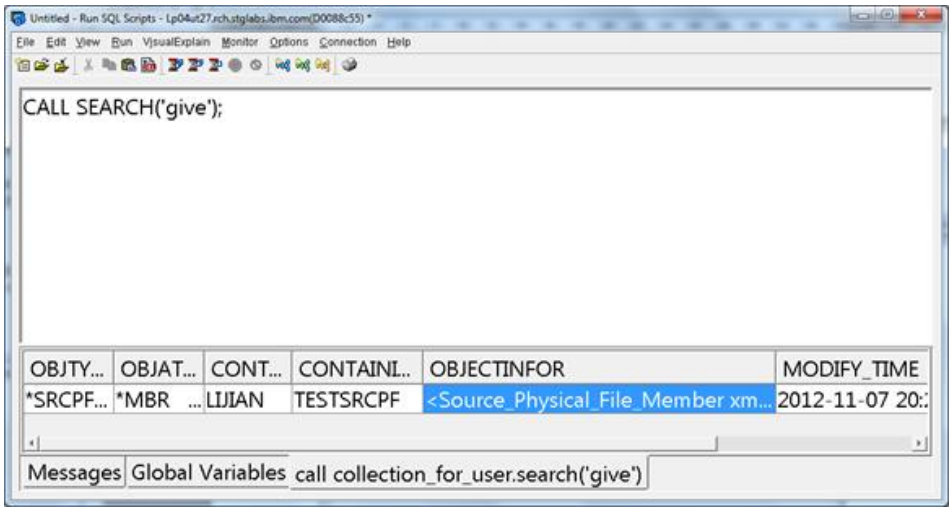
Listing 6. Source physical file object information

```
<Source_Physical_File_Member xmlns="http://www.ibm.com/xmlns/prod/db2textsearch/obj1">
    <file_library>LIJIAN    </file_library>
    <file_name>TESTSRCPF </file_name>
    <member_name>MBR1      </member_name>
</Source_Physical_File_Member>
```

Using Listing 6, the member that contains the search words can be identified.

It's easy to view the result set by calling the SEARCH stored procedure from the IBM Navigator for i Run SQL Scripts tool. The result set will appear in a tabbed window at the bottom of the tool's main window. ↱↱Figure 3 shows how to use IBM Navigator for i to return the result.

Figure 3. IBM Navigator for i output



Upgrading a text search collection

Source physical file object sets were not supported in V1R2 until the PTF Group SF99701 Level 18 became available. If a text search collection was created before applying this PTF group, the text search collection does not contain the procedures to add a source physical file object set to it. For this reason, there is a program that can be called to upgrade text search collections, instead of dropping and recreating the text search collection.

The CL command in ↱Listing 7 can be used to upgrade all text search collections created prior to moving to DB2 PTF Group SF99701 Level 18.

Listing 7. Upgrade all collections

```
CALL QDBTSLIB/QDBTSUPGRD;
```

When this program is invoked without input parameter, all the text search collections created in previous releases will be upgraded to the latest version.

Also, Listing 8 shows how to upgrade a specific text search collection.

Listing 8. Upgrade one collection

```
CALL PGM(QDBTSLIB/QDBTSUPGRD) PARM('MYCOL');
```

If a text search collection name is specified as the input parameter, then only the specified text search collection is upgraded. If the specified text search collection does not exist, an error message is returned. If the text search collection is already the latest version, the procedures are recreated if they already exist.

Searching a text search collection with embedded SQL

Applications can access the result set through embedded SQL, JDBC, or ODBC interfaces. ↱Listing 9 shows how to retrieve the result set of SEARCH using embedded SQL from C.

Listing 9. Working with a result set of a stored procedure

```
//declare a RESULT_SET_LOCATOR type.
SQL TYPE IS RESULT_SET_LOCATOR rs_loc1;

//Associate this new defined result set locator to specific stored procedure
EXEC SQL ASSOCIATE RESULT SET LOCATOR (:rs_loc1)
WITH PROCEDURE collection_for_user.SEARCH(VARCHAR);

//Allocate a cursor for this result set locator, then fetch this cursor to get data
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :rs_loc1;
```

There is more detail information about ASSOCIATE LOCATOR and ALLOCATE CURSOR described in the SQL information center.

In Listing 10, an embedded SQLC program demonstrates how to execute a search, and retrieve data.

Listing 10. Searching a text search collection

```
EXEC SQL SET OPTION SQLCA = *YES, COMMIT = *CHG, NAMING = *SQL;
#include "stdio.h"
#include <sqlca.h>


int main(int argc, char * argv[]){

    struct sqlca sqlca = {0x0000000000000000};

    EXEC SQL BEGIN DECLARE SECTION;
    //declare variables used later
    long int setid;
    short sqlCode;                /* SQL Return Code */
    VARCHAR libName[10];
    VARCHAR objName[10];
    VARCHAR objType[10];
    VARCHAR objAttr[10];
    VARCHAR timeStamp[27];
    SQL TYPE IS XML AS CLOB(5000) objInfo;
    //Since SEARCH stored procedure will return a result set as search result
    //we define a RESULT_SET_LOCATOR variable to receive the results
    SQL TYPE IS RESULT_SET_LOCATOR rs_loc1;
    double sc;
    VARCHAR searchWord[32704];
    EXEC SQL END DECLARE SECTION;
    EXEC SQL DECLARE :objInfo VARIABLE CCSID 37;
    EXEC SQL DECLARE :searchWord VARIABLE CCSID 37;

    strcpy(searchWord.data,"give");
    searchWord.len=strlen(searchWord.data);
    //CALL PROCEDURE SEARCH
    EXEC SQL CALL collection_for_user.SEARCH(:searchWord);
    if (sqlca.sqlcode==466) {
        //warning sqlstate returned since SEARCh return result set
```

```
        printf("%s\n",
               "[SQL0466]1 result sets are available from procedure");
    }
    //associate result set locator to stored procedure
    EXEC SQL ASSOCIATE RESULT SET LOCATOR (:rs_loc1)
    WITH PROCEDURE collection_for_user.SEARCH(VARCHAR);
    EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :rs_loc1;

    if (sqlca.sqlcode == 0) {
        //open cursor successful
        do {
            //fetch search result back
            EXEC SQL FETCH C1 INTO :objType, :objAttr,
            :libName, :objName,
            :objInfo, :timeStamp, :sc  ;
            if (sqlca.sqlcode == 0) {
                //fetch row successful
                /*
                OBJINFO is a XML column which contains the detail
                information of the members of source physical
                file. With this information, user can find the
                member which has the search words.
                */
                objInfo.data[objInfo.length]='\0';
                printf("%s \n",objInfo.data);
            } else if (sqlca.sqlcode == 100) {
                //do nothing, this do-while will end soon
            } else {
                printf("Fetch from cursor failed for %d \n",
                       sqlca.sqlcode);
            }
        }while (sqlca.sqlcode!=0);
        //close cursor after get all the results
        EXEC SQL CLOSE C1;
    } else {
        //open cursor failed
        printf("Open cursor failed for sqlcode %d \n",
               sqlca.sqlcode);
    }
}
```

Conclusion

This article explained how to use the new OmniFind stored procedures to perform the following operations:

- Create a text search collection
- Add one or more object sets of source physical file members
- Search text search collections
- Update old version text search collections to the new version
- Demonstrate how SQL can be used to consume OmniFind results

With the addition of being able to search for members in source physical files, the OmniFind search facility usefulness has taken yet another meaningful step forward.

Resources:

- V1R2 OmniFind information center

- Exploring the IBM OmniFind Text Search Server white paper
- Searching Spool Files and IFS Stream Files