All for One and One for All – Introducing Unicode (Part 2)

In the last issue we gave an introduction to Unicode and the Version 8 enhancements in this area, including the conversion of the DB2 Catalog to Unicode and Unicode parsing. As stated previously, these enhancements will impact everyone to some degree so in the second half of this article we will continue to explain these enhancements in more detail, focussing on their impact and providing information to assist with planning for Version 8.

The following topics will be covered.

- Health checking your existing code page configuration
- How the DB2 catalog changes in V8 will affect you
- Program preparation in V8
- Storing application data and writing applications in Unicode

Health checking your existing code page configuration

Prior to migrating your systems to V8, it is important to perform a code page health check to ensure no latent issues exist. Any discrepancies may cause problems in Version 8 once DB2 parses in Unicode and starts performing code page conversions more frequently. It is important to check the following in each of your DB2 systems :

1) Does your DB2 system have the correct CCSIDs specified and do these match your terminal emulators and local applications e.g. CICS?

2) Identify whether there are objects within the system with different CCSIDs. This is performed by a set of queries in job DSNTIJPM.

If either of these raise an issue at your site please contact DB2 Level 2 Support for advice on how to resolve this prior to migration to V8. We recommend you run the checks as soon as you start planning your V8 migration in order to schedule any appropriate changes.

What impact will the DB2 catalog changes have?

To enable Unicode names and literals in SQL, the DB2 catalog is being converted to UTF-8. This conversion is being done at the same time as long names support, which will allow for potential increases in catalog storage size, in addition to assisting with porting applications across the DB2 Family.

These changes will be performed during the Enable New Function Mode of the migration, which will perform a series of SHRLEVEL REFERENCE REORGs on the DB2 catalog.

There are a number of implications from these changes and you should assess the impact on your particular site.

1) Reading DB2 catalog data

For the most part, you will still be able to view the DB2 catalog using your normal methods. For example, when using SPUFI, the data will be displayed in a legible form (EBCDIC) so long as the character can be represented in the CCSID being used. Other applications will also display the catalog data correctly as DB2 will carry out a conversion, if required, based on the CCSID information specified on the application's precompile and bind options. As these will default to the CCSIDs specified when DB2 was installed, no application changes should be required.

The exceptions to this are the DB2 catalog columns that are defined as FOR BIT DATA, as conversion will not be performed for these columns. The columns affected can be identified by querying SYSIBM.SYSCOLUMNS where FOREIGNKEY='B'. However the main impact for most customers is that the SQL statements stored in the TEXT and STMT columns (SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT) will not be converted. These will be illegible unless a program or tool such as Visual Explain is used to perform an explicit conversion.

(Note that the SQL statements in the DBRMs and catalog will continue to be stored in EBCDIC when precompiled using the NEWFUN=NO parameter which disallows new functionality.)

Another area of potential impact are programs that access the consistency token information in the DB2 catalog (CONTOKEN) as it is also stored as FOR BIT DATA. This will also be returned in Unicode format unless an explicit conversion is performed.

Recommended actions:

- Review any applications or tools that access the DB2 catalog columns defined as FOR BIT DATA. Pay particular attention to tools accessing SQL statements.
- Ensure that all required conversions for local and distributed access have been configured in z/OS Conversion Services. Further information can be found in

Appendix A of the DB2 for z/OS V8 install Guide and Information APARs II13048 and II13049 . Program directory for Unicode Conversion Services?

2) Accessing the DB2 catalog remotely

Using tools and applications running on a workstation to access the DB2 catalog will display the data correctly because the DRDA requester will convert the data as appropriate. This is termed as 'Receiver Makes Right'. In the case of INSERTs and UPDATEs the host will perform the appropriate conversion. (Again, the exception to this is data stored as FOR BIT DATA)

3) Be aware that Unicode's collating sequence is different from EBCDIC

The collation sequence for Unicode is different from EBCDIC, as UTF-8 orders numbers before characters. This is equivalent to ASCII behavior, as the first 127 code points are identical to Unicode UTF-8.

Consequently, you should be aware that catalog queries may return results in a different sequence. However, this is only an issue if the columns being ordered contain a mix of numeric, upper case and lower case characters. There may also be a slight difference in behavior if the data contains accented and special characters as these may be sorted in a different order in Unicode to EBCDIC.

An example:

```
SELECT NAME FROM SYSIBM.SYSTABLES
ORDER BY NAME
```

```
NAME
++++++++++++++++++++++
TEST1
TEST2
TEST3                    Prior to V8 NFM, TESTA/B/C would be before TEST1/2/3
TESTA
TESTB
TESTC
```

Note that GROUP BY, range predicates and MIN/MAX functions may also be impacted where they span sets of characters. For example, the following statement will return TEST2, TEST3, TESTA and TESTB in Unicode but will return no rows in EBCDIC.

```
SELECT * FROM SYSIBM.SYSTABLES
WHERE NAME BETWEEN 'TEST2' and 'TESTB'
```

If the predicates are reversed, the query returns TESTB, TESTC, TEST1 and TEST2 in EBCDIC as letters are sequenced before numbers.

A possible workaround for the ORDER BY issue

An option to simulate previous behaviour of an ORDER BY is to use the CAST function to convert the data to EBCDIC format, as this will result in the original sort order

It should be noted that this is only an option for the ORDER BY issue as predicate evaluation will occur in Unicode.

Recommended action:

- Review applications and tools that query the DB2 catalog to determine if they will be impacted. Consider using the CAST function where appropriate.

4) Joining DB2 catalog tables to EBCDIC tables

V8 has introduced the ability for a single SQL statement to access tables in multiple encoding schemes. Therefore it will be possible to continue to join the DB2 catalog tables with user tables. You should be aware of some differences which could lead to different results being returned following the catalog conversion. These include the following:

- Where two columns or strings with different CCSIDs are compared, the evaluation will be performed in Unicode.
- Where a column is compared with a string which has a different CCSID, the evaluation will be performed in the column's CCSID. E.g. UNICODECOL= :hv will be evaluated in Unicode and EBCDICCOL= :hv will be evaluated in EBCDIC
- Ordering will be impacted by any conversions that have been performed for evaluation.

These issues will only be important where range predicates or ordering is important and the data contains a mixture of upper case, lower case and numeric characters.

5) Unicode data may take more space

It is important to note that with Unicode, the storage size no longer always equals the displayed size. You may need to allow extra space for the DB2 catalog data sets due to the potential increase in storage size for some characters. The amount will depend on your existing character set but Unicode can increase disk storage costs by three to four times, with an associated processor memory increase.

6) Are any program changes required?

Although data strings may be longer when stored in Unicode, host variables should not generally need to be changed as DB2 will convert the data to the programs CCSID based on the application encoding scheme and the precompiler CCSID. This makes the conversion of the database to Unicode transparent to the application.

However you should consider that the change to VARCHAR(128) for many of the DB2 columns may have an impact. For example, in Version 8 tables can be created with names longer than the 18 characters currently allowed and your program may need to be changed to display complete object names.

7) Are any strings close to the limit of 255 bytes?

It is possible that strings which were valid in V7 will be invalid in V8 due to the potential increase in number of bytes following conversion to Unicode. In most cases this is only a problem in Compatibility Mode, as on entering New Function Mode the 255 byte limit is raised to 32704. It should be noted that the limit for the VALUES clause on CREATE and ALTER INDEX statements will continue to be limited to 255 bytes. In most cases though this should not present a problem since you can change to table-controlled partitioning and specify the values on the table definition instead.

Program preparation in Version 8

It is important to recognize that the V8 precompiler parses SQL in Unicode and has some new input parameters.

The first is the NEWFUN parameter which, when set to YES, has a dual impact: not only does it allow the program to use new functions, but it also results in Unicode DBRMs and Unicode statements in the DB2 catalog. This will be in addition to the rest of the Unicode data in the catalog which will have been converted during the Enable New Function Mode phase of the migration.

Setting NEWFUN=NO prevents programs from using new functionality by rejecting V8 SQL. It will also cause the DBRMs and statements in the DB2 catalog to be written as EBCDIC. Note that even with this option, the precompiler will parse SQL in Unicode.

As the Unicode DBRMs will not be legible using a standard ISPF browser, you may wish to use tools to view the DBRMs. For example the DB2 Path Checker tool will convert the statements to EBCDIC.

The second new precompiler parameter is CCSID, which tells DB2 the code page of the program source so that the SQL statements and literals can be parsed correctly. It should be used in conjunction with the application encoding bind option to ensure that the appropriate conversions are performed, ensuring a match between DB2 and the program's code pages.

Note that when executing packages remotely, the DRDA receiver will carry out the appropriate conversion based on the CCSID in the DRDA flow (rather than the application encoding bind option).

Storing application data and writing applications in Unicode

In addition to the DB2 catalog being converted to Unicode, you may wish to consider storing some user data as Unicode. This has been possible since Version 7, but V8 removes limitations, making it a more feasible solution for some customers. The considerations raised earlier in relation to the DB2 catalog apply for any Unicode data but some additional information and guidance follows.

Deciding to store data as Unicode

The decision whether to store user data as Unicode in the database should be considered carefully on a case-by-case basis. Obviously the main benefit to be realized is the ability to store multilingual data in a single DB2 system, database or even table. Some customers using Unicode-based technologies such as Java and XML may also wish to store data as Unicode to avoid conversion issues. (It should be noted that as they use UTF-16 as their base code page, your installation may wish to use graphic data types to avoid a conversion cost. This is discussed in more detail later.)

It is important to recognize that storing data as Unicode will not be transparent to the developers and DBAs. Storage size does not equal displayed size so columns will need to be increased in size to cater for maximum stored lengths. In some cases, additional complexities may arise as limits are reached. For example, the maximum CHAR/VARCHAR column size of 255/32704 bytes may no longer be sufficient. The use of varying length columns will become more prevalent as fixed length data no longer applies. LOBs may also be required more often as the varying length string limit of 32K bytes may not be adequate.

In addition to potential column length increase, datasets may grow in size requiring additional disk storage. The amount will depend on the nature of the data as discussed previously for the catalog. However for user tables ESA compression can be used to reduce the impact.

Other considerations include those given in the earlier section entitled 'What impact will the DB2 catalog changes have?' These include the change in collating sequence potentially impacting the results of range predicates and ORDER BYs. This could have a significant impact where data contains a mixture of upper case, lower case and numeric characters. Additional factors to takes into account include the impact of Unicode data on routines such as Field procedures and Edit procedures. When considering converting existing data to Unicode, there is also an availability issue as the data will need to be unloaded from its current EBCIDC or ASCII format and loaded back into the table.

Performance is also an important factor in the decision although measurements are still being established. The key recommendation is to have an understanding of the CCSIDs being used by the various components of the system so that the decision is made as part of the overall design rather than at the DB2 level. In this way the impact of conversion costs can be assessed and weighed up against the other considerations.

Defining Unicode tables

The Unicode encoding scheme can be defined at the database, table space or table level, depending on your requirements. Defining a database with CCSID UNICODE will cause all tables and tablespaces within that database to default to Unicode.

Once a table has been defined as Unicode, columns defined as CHAR, VARCHAR and CLOB will use UTF-8 by default. This results in mixed data with characters taking one to four bytes each. It is possible to ensure that the data is stored in single bytes only by specifying FOR SBCS DATA on the column definition. However, this will restrict the characters stored to 7-bit ASCII and is therefore not expected to be widely used.

Some customers may prefer to store their data as UTF-16 by defining their columns as GRAPHIC, VARGRAPHIC and DBCLOB. This includes the Chinese, Japanese and Koreans as UTF-16 stores many of their characters as two bytes.

A number of applications may also prefer to use the graphic data types as UTF-16 is the base code page for programming languages such as Java and Visual Basic. This will avoid the performance overhead of converting between UTF-8 and UTF-16 that incurs a standard conversion cost.

Writing Unicode applications

As stated earlier, applications are independent of the server code page as DB2 will convert the data as appropriate, based on the precompiler CCSID and bind application encoding option. These will default to the EBCDIC CCSID and application encoding CCSID parameters specified on the DB2 installation panels.

The application encoding bind option defines the CCSID for the host variables in the program. For dynamic SQL this can be overridden using the APPLICATION ENCODING special register.

It is also possible to specify the CCSID at a more granular level than the plan/package level by using the CCSID clause in the DECLARE VARIABLE statement. This gives a good degree of flexibility when coding programs accessing data in different CCSIDs.

Writing programs to cater for multiple CCSIDs

Customers wishing to write programs which cater for different CCSIDs so that they can be run in various countries have a number of options:

1) Use the SET CURRENT APPLICATION ENCODING SCHEME =:hv statement at the start of the program to specify the CCSID for each particular invocation. This is a good solution for dynamic SQL but it should be noted that this special register is not used for static SQL.

2) Bind a package with each different application encoding CCCSID and use the SET CURRENT PACKAGESET statement to ensure the correct package is invoked.

3) A CCSID could be specified in each SQLVAR entry in the SQLDA.

4) The use of the CCSID clause on DECLARE VARIABLE statements could be considered but as a host variable specification is not currently allowed for this statement, a text pre-processor would need to set the appropriate value.

5) Consider using your programming languages Unicode support. For example COBOL Version 3 provides the ability to specify a code page to the coprocessor on the use of the Unicode support

Summary

I hope that this paper has given you some understanding of Unicode and the impact of the enhancements in DB2 for z/OS Version 8. As the Early Support Program continues and the product becomes generally available, more detailed information and experiences will be published.

Sarah Ellis
EMEA Product Introduction Centre
IBM UK
sellis@uk.ibm.com

21st November 2003

Further Information

Unicode Consortium : www.unicode.org

DB2 for z/OS V8 Installation Guide and Application Programming Guide

Section 6.2 of Redbook SG24-6871

Reprinted with permission of the IDUG Solutions Journal