IBM Software Group

DB2 for z/OS V8 Unicode: preparing today for tomorrow

DB2 Information Management Software

Cécile Benhamou
Technical Sales DB2 z/OS et Tools DB2
cecile_benhamou@fr.ibm.com

ON DEMAND BUSINESS

CCSID - Usage

- CCSID= Coded Character Set Identifier
- DB2 uses a set of three CCSIDs to describe data stored in DB2
  - EBCDIC, ASCII and Unicode
- DB2 supports specification of CCSIDs at DB2 subsystem level
  - DSNHDECP
  - Once set they should not be changed!

Conversion methods

- Native DB2 - SYSIBM.SYSSTRINGS
- z/OS support for Unicode
  - Central repository for z/OS system
    - ✓ Used by DB2, COBOL, ODBC Driver
- Must configure Unicode Conversion Services for z/OS
  - Unicode CCSIDs (367,1208,1200) <-> ASCII/EBCDIC CCSIDs
  - Client CCSIDs <-> Unicode CCSIDs (367,1208,1200)
  - Must also add to/from Unicode CCSIDs (367,1208,1200)
    - CCSID 37 for DBRMs provided with DB2 install
    - CCSID 500 for DRDA
    - CCSID 1047 for Unix System Services (USS)
  - Rebuild conversion image -> New image picked up by DB2 'on the fly'
- Configure for best conversion performance
  - z890 and z990 zSeries (hardware instructions instead of mcode/ucode) + z/OS R4

Conversion methods

- Configure Unicode Conversion Services for z/OS

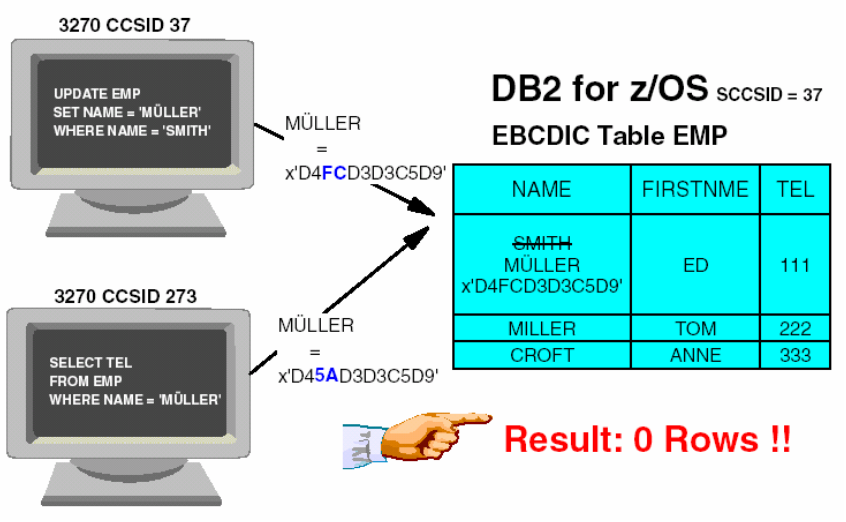
```
COMMAND INPUT ===> /d uni,all

    RESPONSE=SC63
     CUN3000I 02.37.19 UNI DISPLAY 752
      ENVIRONMENT: CREATED       04/17/2004 AT 10.08.19
                   MODIFIED      04/17/2004 AT 10.08.20
                   IMAGE CREATED 12/08/2003 AT 17.02.33
          SERVICE: CHARACTER      CASE           NORMALIZATION  COLLATION
          STORAGE: ACTIVE      424 PAGES
                   LIMIT     51200 PAGES
        CASECONV: NORMAL
       NORMALIZE: DISABLED
         COLLATE: DISABLED
      CONVERSION: 00850-01047-ER              01047-00850-ER
                  00037-01200(13488)-ER       01200(13488)-00037-ER
                  00037-01208-ER              01208-00037-ER
                  00437-01208-ER              01208-00437-ER
                  00037-00367-ER              01252-00037-ER
                  00037-01252-ER              00367-00037-ER
                  00500-01200(13488)-ER       01200(13488)-00500-ER
                  01047-01200(13488)-ER       01200(13488)-01047-ER
                  01047-01208-ER              01208-01047-ER
                  01208-01200-ER              01200-01208-ER
                  01383-01200-ER              01200-01383-ER
                  00932-01200-ER              01200-00932-ER
  .....
```

  - Brochure 'Support for Unicode: Using Conversion Services' (SA22-7649)

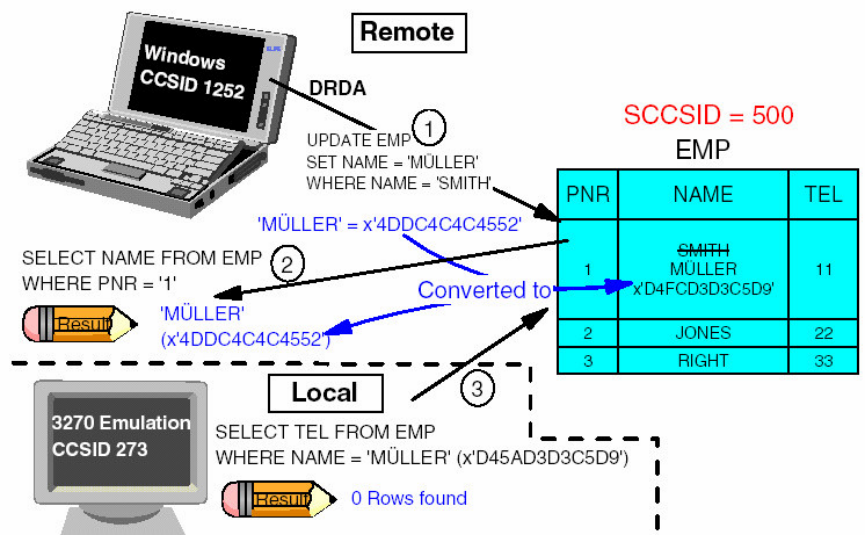
When does conversion occurs?

- Local applications: General case, no data conversion
  - Terminal emulators should have a compatible code page
- Local applications: sometimes, conversion occurs
  - Dealing with ASCII/Unicode tables
  - Specified by application:
    - ✓ Application ENCODING bind option (V7)
    - ✓ CURRENT APPLICATION ENCODING SCHEME special register (V7)
- Remote applications: Automatic conversion for remote access when needed
  - Done by DRDA receiver

Storing Characters by Local Applications



Remote (DRDA) Applications



Preparation for DB2 V8: CCSID is it a problem?

- Potential for incorrect character interpretation (i.e., data corruption) if multiple CCSIDs within one encoding scheme
  - Terminal emulator using incompatible code page
  - Incorrect CCSID specification
  - Change of CCSID specification
- Must be fixed and residual corruption cleaned up ahead of V8
- V7 APAR PQ56697 introduces checks for non-zero valid CCSIDs and issues warning when trouble found
- V7 APAR PQ89018 will provide CCSID detection if DECP value does not match the emulator value
- Run DSNTIJP8 (V7) or DSNTIJPM (V8) ahead of migration to check that there are not multiple CCSIDs within one encoding
- V8 start up checks for:
  - Non-zero, valid CCSIDs
  - EBCDIC<->Unicode round trip conversion

Unicode: what is it?

- Unicode is a single character set that encodes all world scripts
- Provides a cross-platform, cross-vendor method of encoding data that enables lossless representation and manipulation
  - Required by many modern standards (Java, XML, LDAP, ..)
  - Required by many software vendors (SAP, PeopleSoft, Siebel, ..)
- 3 forms of Unicode
  - UTF-8: Unicode Transformation Format in 8 bits
  - UTF-16: Unicode Transformation Format in 16 bits
  - UTF-32: Unicode Transformation Format in 32 bits

Unicode: DB2 Usage

- UTF-8: Unicode Transformation Format in 8 bits
  - Uses 1 to 4 bytes for one character representation
  - Standard characters (A,B,C, ...) : still on one byte
  - Specific characters (é, à, ç, ...): on 2 bytes
- UTF-16: Unicode Transformation Format in 16 bits
  - Used for GRAPHIC/VARGRAPHIC
- Most internal processing is done in Unicode
  - Parsing: literal in parse tree are in Unicode
  - Precompiler: source converted to Unicode and parsed in Unicode
  - DBRM: Unicode in NFM
  - Optimization: catalog lookup in Unicode
  - Special registers: stored in Unicode
  - Tracing: option to output trace data in Unicode
  - …

Unicode: Conversion

- Conversion of SQL statements and derived metadata to Unicode
  - Minor Conversion (fast)
    - ✓ Optimization for most common single byte CCSIDs (MIXED=NO)
      i.e., simple alphanumeric characters
    - ✓ Most common case ASCII/EBCDIC->UTF-8, can go other way
    - ✓ Internal 256 byte translation tables used by DB2 (TR/TRT)
  - Major Conversion (slower)
    - ✓ Offloaded to Unicode Conversion Services for z/OS
    - ✓ Used if EBCDIC/ASCII->UTF-8 cannot be performed inline
    - ✓ Always for ASCII/EBCDIC->UTF-16 (no optimization)
- Particular conversion not available -> SQLCODE -332
- Use zparm UIFCIDS option to have IFCID trace data in Unicode UTF-8 format
  - Default = NO
  - Use YES to move conversion overhead off-line e.g., DB2PE

Unicode: Catalog migration

- Catalog is converted to Unicode during migration
  - 18 Tablespaces converted to Unicode
  - Some Tablespaces which must interface with MVS external names stay in EBCDIC
    - ✓ SYSCOPY
    - ✓ DBD01, SCT02, SYSLGRNX, SYSUTILX
  - New tablespace called SYSEBCDC contains a single row/byte to support SYSDUMMY
  - 2 Tables dropped
    - ✓ SYSLINKS
    - ✓ SYSPROCEDURES
- DSNTIJNE job (ENFM) can be restarted from the beginning without change
  - Objects reorganized in a specific order
- Up to 10% increase in disk space requirement for Cat/Dir
  - Some objects shrunk by 50% because they had never been reorganized

Unicode: Impacts on Catalog

- SQL parsed and metadata derived from SQL in Unicode UTF-8 format
- Most character columns converted to Unicode VARCHAR(128)
- Trailing blanks not stripped
- Special characters expand beyond single byte (e.g., #)
- Increase in space: CHAR -> VARCHAR (length bytes)
- Decrease in space: DB2 defined indexes converted to NOT PADDED
- Consider converting user defined indexes to NOT PADDED
- Compatibility (COMPAT) mode
  - SQL parsed in Unicode UTF-8 format
  - Metadata derived from SQL converted back to EBCDIC
- New Function Mode (NFM) mode
  - Catalog converted to Unicode UTF-8 format
  - Metadata derived from SQL no longer converted back to EBCDIC

Unicode: Impacts on Catalog …

- If precompile application with NEWFUN(NO)
  - SQL statements stored in EBCDIC in DBRMs
  - Disables use of new SQL function for static SQL
  - Go back through SQL Parser during BIND processing
    - ✓ To get Unicode UTF-8 format
    - ✓ Incur extra overhead
- After entry to NFM, default for NEWFUN changes to YES
  - SQL statements stored in UNICODE in DBRMs
  - DBRMs switch to Unicode UTF-8 format
- At BIND time
  - Anything from DBRM in Unicode UTF-8 format goes into SYSSTMT and SYSPACKSTMT in Unicode UTF-8 format
  - Anything from DBRM in EBCDIC format goes into SYSSTMT and SYSPACKSTMT in EBCDIC format

Unicode: Impacts on Catalog …

- ORDER: How to influence it?
  - SELECT NAME FROM SYSIBM.SYSTABLES
    WHERE NAME LIKE 'T%'
    ORDER BY NAME
    - ✓ V7 (EBCDIC): returns TA, TB, T1, T2
    - ✓ V8 (UNICODE): returns T1, T2, TA, TB
  - SELECT CAST(NAME AS CCSID EBCDIC) AS E_NAME
    FROM SYSIBM.SYSTABLES
    WHERE NAME LIKE 'T%'
    ORDER BY E_NAME
    - ✓ returns TA, TB, T1, T2
    - ✓ Column NAME is converted into EBCDIC

Some Examples if using Unicode in V8

- Select rows from ET1, UT1 where C1 = 'ABC'
  - Application encoding scheme: EBCDIC

Coded SQL statement:

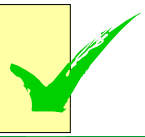
```
SELECT ET1.C1, UT1.C1
FROM ET1,UT1
WHERE ET1.C1 = X'C1C2C3'
AND UT1.C1 = X'414243'
```

Interpreted as:

```
SELECT ET1.C1, UT1.C1
FROM ET1,UT1
WHERE ET1.C1 = 'ABC'
AND UT1.C1 = ' âä'
```

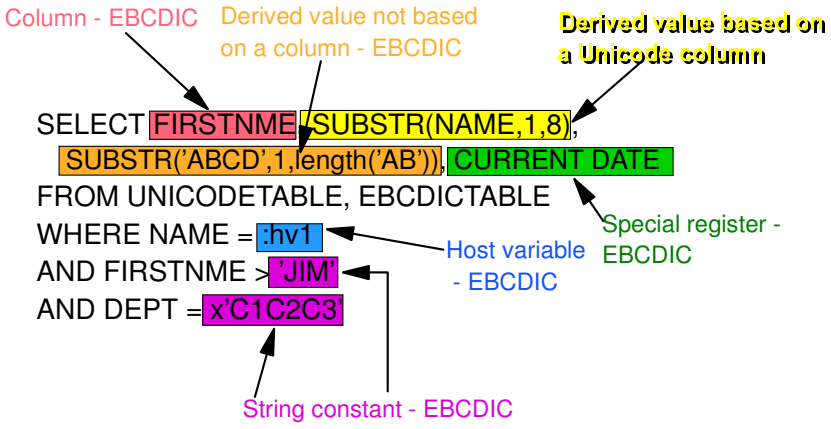
Correct encoding:

```
SELECT ET1.C1, UT1.C1
FROM ET1,UT1
WHERE ET1.C1 = X'C1C2C3'
AND UT1.C1 = X'C1C2C3'
```



Some Examples if using Unicode in V8

- Application encoding scheme: EBCDIC



Some Examples if using Unicode in V8

- UTILITIES
  - Utility Control Statements can be specified in EBCDIC or UNICODE
    - ✓ DB2 detects which encoding scheme is being used
    - ✓ Objects names in messages will be in EBCDIC
- EBCDIC
  - //SYSIN DD *

    COPY TABLESPACE A.B

    /*
- UNICODE
  - //SYSIN DD *

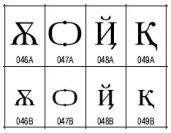
    “à|&$”èà<àë&äà””â”””””””””

    /*

Conclusion

- Don't be afraid

Cyrillic

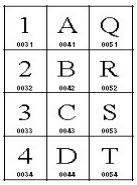


Mongolian

CJK

- But be prepared

Basic Latin



- Don't wait until V8 migration: Start now

Acknowledgments

- This presentation is based on :
  - a presentation by John J Campbell and Florence Dubois
  - a presentation by Chris Crone
  - the following 'Redbook': Redbooks
    - ✓DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More (SG24-6079)