



# DBCS support in DB2 for Z/OS

-China customer experience and common problems

*Fan Jiong*



**DB2 for z/OS Technical Conference** **DB2实用技术论坛**  
**Sept 05-07, 2007 - Shanghai, China**  
**Sept 10-12, 2007 - Beijing, China**

# Chinese character standard (1)

- Chinese character
  - About 100,000 Chinese characters
  - 6 extensions
    - Base and extension 2 , 4 are simplified Chinese
    - Extension 1,3,5 are traditional Chinese
    - Extension 6 is from GB13000.1 Chinese used by Japan, Korea and Taiwan.
- GB2311-1990
  - ISO 2022:1986
- GB2312-1980
  - 6763 Chinese character and 682 symbols
- GB11383-1989
  - ISO 4873:1986
- GB12345-1990
  - Extension 1
- GBK-1995
  - 21,003 Chinese character
  - 883 symbols



# Chinese character standard (2)

- GB18030-2000
  - 2000/03/17
  - Adding 4 byte encoding
  - 1.5 million code point
  - 20,000+7,000
  - Single byte: 0X00-0X7F
  - Double byte: 0X81-0XFE||(0X40-0X7E|0X80-0XFE)
  - Four byte: 0X81-0XFE||0X30-0X39||0X81-0XFE||0X30-0X39
    - Extension A
  - 27, 484 Chinese character and symbols
  - Support 6,582 more Chinese character than GBK
- GB13000.1-1993
  - UCS, Unicode 3.1
  - ISO 10646.1-1993
  - CJK, extension A, extension B
  - 20, 902 Chinese, Korea and Japan character



# Encoding schema

- An encoding scheme is a collection of the code pages for various languages used on a particular computing platform. For example, the EBCDIC encoding scheme is typically used on z/OS and iSeries (AS/400). The ASCII encoding scheme is used on Intel-based systems (like Windows), and UNIX-based systems.
- ASCII stores the character 'A' as x'41' and the number '1' as x'31'. As mentioned above, EBCDIC stores the character 'A' as x'C1' and '1' as x'F1'. This results in a different collating sequence in EBCDIC and ASCII.
  - The collating sequence in ASCII is: space, numerics, upper case characters, lower case characters.
  - The collating sequence in EBCDIC is: space, lower case characters, upper case characters, numerics.
- You can create table spaces with different encoding schemes within one database. Within one table space, however, you can only create tables which use the same encoding scheme as the table space. For indexes, you cannot choose any specific encoding scheme. Indexes always inherit the table's encoding scheme.



# Code Page, CCSID and Code Point

- A *code page* is a set of assignments of characters to code points. Within a code page, a code point can have only one specific meaning.
- A *code point* is a location in a code page. In the IBM environment, the hexadecimal representation of the location is preferred. For example, code point x'C1' represents character 'A' in the EBCDIC code page 500, as shown in the visual above.
- A *Coded Character Set Identifier (CCSID)* is simply a number to identify a particular code page. For example, North Americans use the US-English code page denoted by a CCSID of 37. Germans use the CCSID 273. People in Turkey use CCSID 1026. Some of these include code points for specific characters in their language. Some have the same characters but are represented by different code points in different CCSIDs.



# Code Page, CCSID for Simplified Chinese

- EBCDIC

- Support thousand Chinese character for GB2312
  - MCCSID=935
  - SCCSID=836
  - GCCSID=837
- Support GB18030
  - MCCSID=1388
  - SCCSID=13124
  - GCCSID=4933

- ASCII

- Support thousand Chinese character for GB2312
  - MCCSID=1381
  - SCCSID=1115
  - GCCSID=1380
- Support thousand Chinese character for GB2312 and EUC
  - MCCSID=1383
  - SCCSID=367
  - GCCSID=1382
- Support GB18030
  - MCCSID=1386
  - SCCSID=5210
  - GCCSID=1385



## IBM product support for Simplified Chinese character

- Almost all product support GB2312
- PCOM
  - Support GB18030 very early
- TCP/IP
  - FTP support GB18030 since ZOS 1.4
- CICS
  - Normally has no sense for Chinese character, but may meet problem for COBOL program CICS translate phase
- DB2
  - Support GB18030 since DB2 V6 (added after GA)



# Why DB2 need to know the code page?

- Multiple language support
  - DB2 need support data in multiple encoding schema and code page in same subsystem
  - The language of the application source and underlying data could be different, which is required for most developer and vender tools.
- Cross platform
  - DB2 need convert the data between the source and the target if their encoding schema or code page is different
- Java
  - Everything is Unicode except treat as binary



## Where is the parameter for the code page in DB2?

- DB2 subsystem level
  - DSNHDECP
    - MIXED
    - SCCSID, MCCSID, GCCSID
    - ASCCSIID, AMCCSID, GCCSID
    - USCCSID, UMCCSID, UGCCSID
    - ENSCHEME
    - APPENSCH
- DB2 DDL
  - CREATE DATABASE...CCSID(ASCII|EBCDIC|UNICODE)
  - CREATE TABLESPACE...CCSID(ASCII|EBCDIC|UNICODE)
  - CREATE TABLE...CCSID(ASCII|EBCDIC|UNICODE)
- Application level
  - BIND/REBIND...ENCODING(ASCII|EBCDIC|UNICODE|ccsid)
- CICS level
  - Translate
- Cobol level
  - Pre-compile CCSID option



# The normal definition in China customer

- Sample 1:
  - MCCSID is 935
  - MIXED DAT=YES
  - ENSCHEME=EBCDIC
  - APPENSCH=EBCDIC
- Sample 2:
  - MCCSID is 1388
  - MIXED DAT=YES
  - ENSCHEME=EBCDIC
  - APPENSCH=EBCDIC
- Sample 3:
  - MCCSID is 1388
  - AMCCSID is 1386
  - MIXED DAT=YES
  - ENSCHEME=ASCII
  - APPENSCH=ASCII



## But what did we store the data really?(1)

- Some customer and vender product use DB2 as a file container, so the data stored is not match what we defined to DB2. And only the specific application know their format and encoding.
- Sample 1:
  - For single byte string we store real EBCDIC string, but may not match the SCCSID
  - For columns with Chinese characters (defined as EBCDIC CHAR subtype MIXED) we store ASCII string
  - For password column (defined as EBCDIC CHAR subtype MIXED) we store binary string
  - CHAR/VARCHAR with double byte (defined as EBCDIC CHAR subtype MIXED) , application will padding to full length with x'20'
- Sample 2:
  - We use CHAR to store the number
- Sample 3:
  - For CHAR/VCHAR column, we insert data with CHAR+Decimal value+CHAR directly from COBOL program
- Sample 4:
  - We create table for 935 or 1388, but we insert data through PCOM with code page setting as 037.



## But what did we store the data really?(2)

- The EBCDIC SBCS data is inserted after gateway convert program, but no one can tell where is the convert table from and whether it is right. The result is the converted data may not belong to any code page or CCSID.
- When we store ASCII data in DB2 EBCDIC MIXED CHAR column, no one can tell whether it is belong to ASCII CCSID 1381 or 1386, or their own one too.
- The result is the data can only be understand by that specific environment and program. The data stored is not match what we tell DB2.



# The mismatch?

- Sample 1:
  - 935 only support thousand Chinese character, 1388 is the biggest EBCDIC code page for Chinese character. Some customer define as 935, but actually store the data in 1388.
  - The encrypted password is binary data, but we defined with CHAR by subtype MIXED.
  - We store Chinese string with ASCII format, but we defined these column with EBCDIC CHAR by subtype MIXED
  - What's the real CCSID? Is it match the real data?
- Sample 2:
  - Numbers was stored by CHAR/VCHAR
- Sample 3:
  - The source code was written in CCSID 037, the data is defined as CCSID 935, we could meet problem in literal, as they are in different CCSID.
- Anything else?



# Where is the problem?

- When conversion was triggered by DB2, DB2 will use wrong source CCSID, then data could be lost or broken, the SQL results could have logical errors.
- When conversion should be triggered but did not, we would have the same result.



# Sample 1

- Code point of character '/'
  - X'E0' in 037
  - X'B2' in 935
- Environment and the setting:
  - DSNHDECP MCCSID=935 MIXED DATA=YES
  - CREATE TABLE TBA... CCSID(EBCDIC)
  - The TBA has COLUMN COLA CHAR(1)
- Insert
  - We insert character '/' by SPUFI from PCOM, the PCOM setting is 935.
  - We have a record with column hex value x'B2'
- Select
  - We have a COBOL program to SELECT COLA from TBA where COLA='/'
  - But the COBOL was written in PCOM with the setting is 037
  - So actually we try to find COLA with hex value x'E0' rather than x'B2', and the result is no row return.
  - But from application logic, we just try to find a column, which have character '/', and it did have.
- Summary: if we use different CCSID between data and program, then you should be careful the CSSID difference. For SBCS, some character's code point is same in CCSID 037 and 935, for example A-Z, a-z,0-9. If possible application can only use that part to avoid the problem of this sample.



# Sample 2

- DRDA

- Environment and the setting:

- DB2 for Z/OS
      - DSNHDECP MCCSID=935 MIXED DATA=YES
      - CREATE TABLE TBA... CCSID(EBCDIC)
      - The TBA has COLUMN COLA CHAR(1)
    - DB2 UDB for Window
      - It's code page is 1381. It has DRDA connect to DB2 for Z/OS.

- Insert

- Although we defined column COLA as EBCDIC CHAR subtype MIXED, but we actually store ASCII data.
    - We put ASCII character 'L' into this column from CICS/DB2 program by purpose, as the program know this column will store ASCII data although the definition is EBCDIC CHAR. So the hex value of this COLA is x'4C'.
    - In EBCDIC 935 CCSID it is character '<'

- Select

- We issue "SELECT COLA from TBA" by DRDA connection
    - There is convert from 935 to 1381, as DRDA do not know this data actually is ASCII data already, he only knows this is EBCDIC table space and CCSID is 935.
    - So after DRDA conversion, client will receive '<'



## Why we do not have such problem today in DB2 V7?

- Generally DB2 will not do convert for local application.



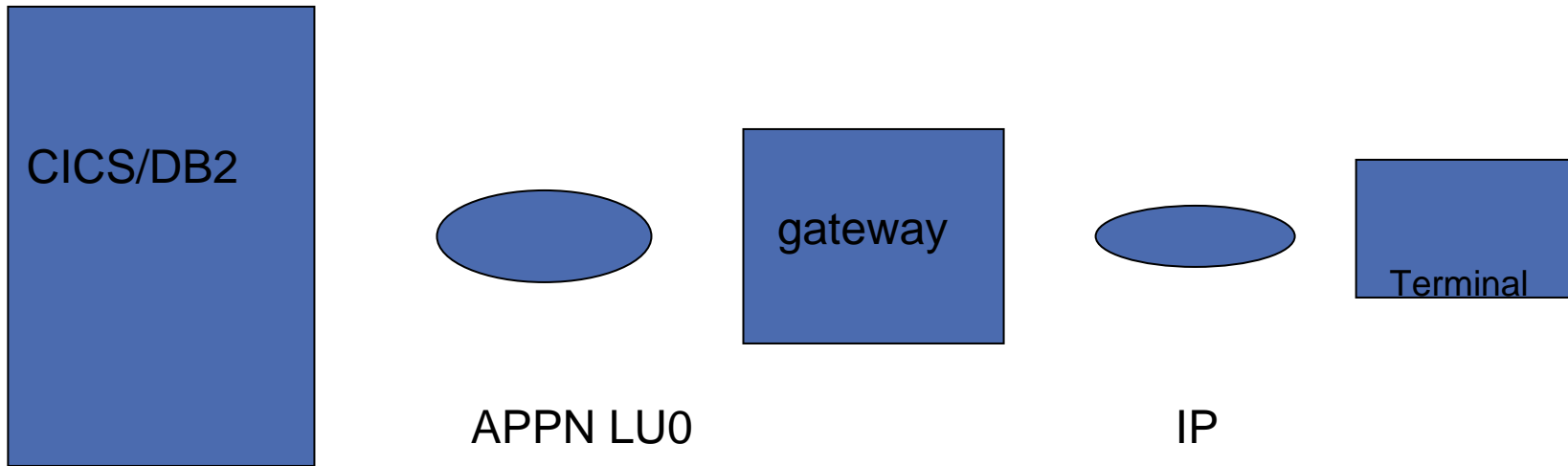
# When could we meet problem in V7?

- When the convert happen
  - Java application
  - DRDA
  - Literal in SQL statement
  - Specified by application
    - CCSID override in SQLDA
    - Declare variable
    - Application ENCODING bind option
    - CURRENT APPLICATION ENCODING SCHEME special register



# Why we do it this way?

-the best guess



- The client and terminal need ASCII string. If we store all data in EBCDIC, then we need a big gateway to convert all string back and forth. Only store single byte string in EBCDIC, columns with Chinese characters are in ASCII, then the gateway only need convert single byte string, which is much simple. It is possible to use ASCII table space, then no conversion was needed, and less risk to cheat DB2. We could meet difficulty when we use ASCII literal in the program and write the program in EBCDIC environment if the default application encoding schema is ASCII.

- Application programmer and Vender product may still keep the habit for file system and IMS, so they use DB2 as a container and do not realized DB2 has more intelligent for data conversion.



## CCSID of SQL statement, Literal and Host Variable in DB2 V7

- CCSID of the SQL statement
  - Statistic SQL statement was interpreted by SYSTEM EBCDIC CCSID of DECP . It used to be 935/1388 in China customer
  - Dynamic SQL statement was interpreted by BIND ENCODING. It used to be 935/1388 in China customer
  - Most China customer's CCSID of DECP, BIND ENCODING are same as the table space.
- Literal (from DB2 point of view)
  - In statistic SQL, it is SYSTEM EBCDIC
  - In dynamic SQL, It will be interpreted by BIND ENCODING. It is used to be 935/1388 in China customer.
- CCSID of host variable (from DB2 point of view: host variable in statistic SQL or parameter marker in dynamic SQL)
  - Decided by BIND ENCODING as default
- In V7, if the BIND ENCODING and Table space CCSID is same, we will not do much check and conversion, the data is hex in and hex out. We did do very limited check for some special character (for example the 0x0E, 0x0F pair).

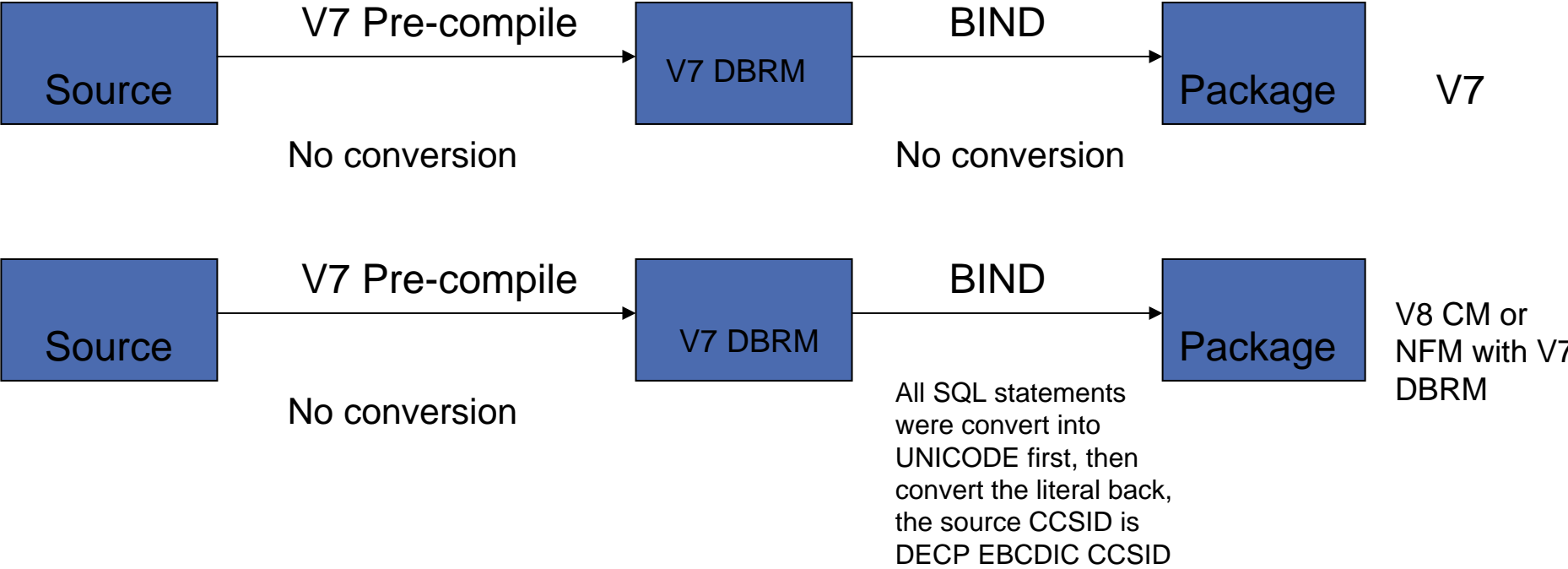


# More risk in DB2 V8

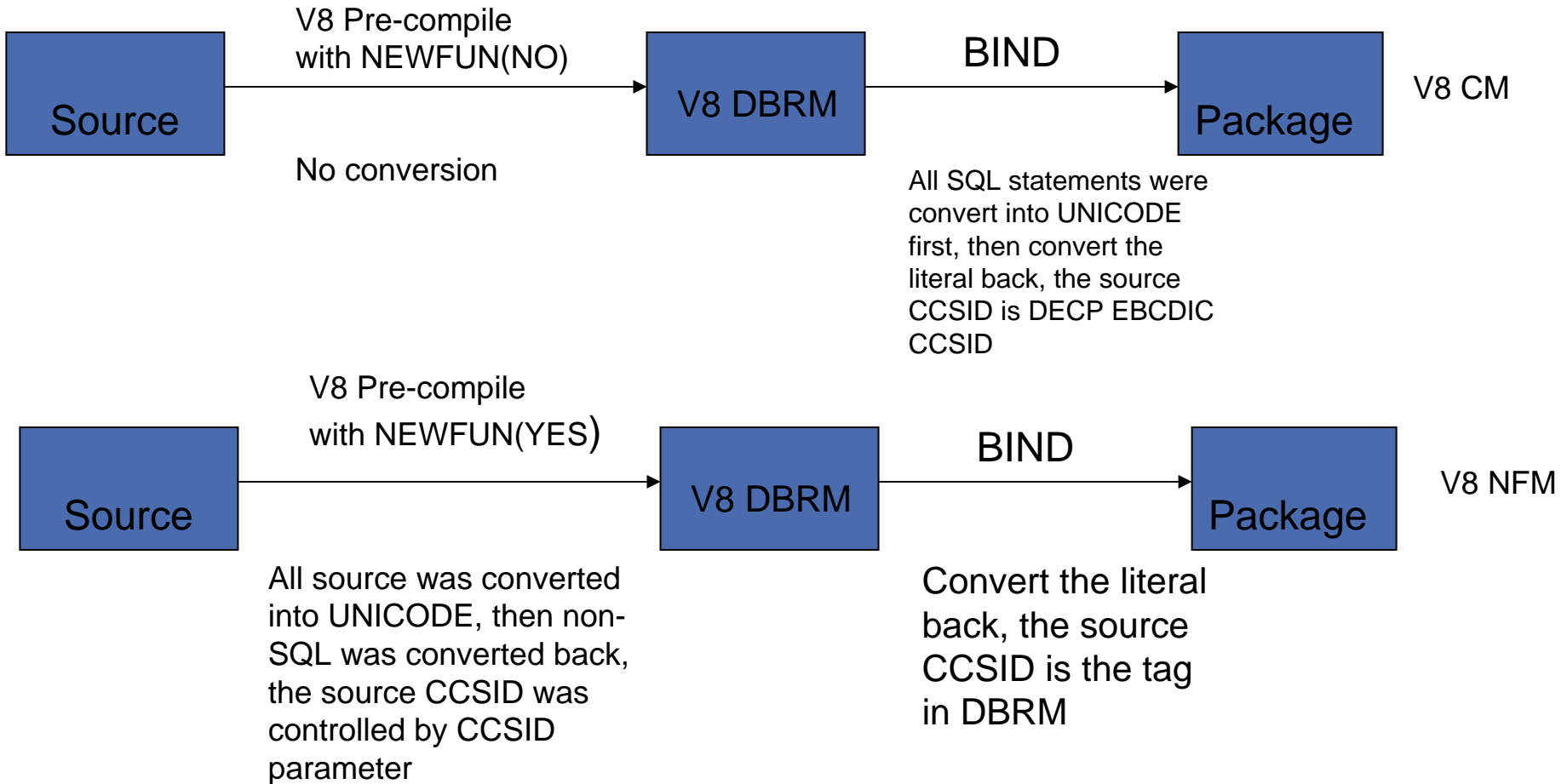
- Here we assume the application encoding and table space encoding has same CCSID. They are either 935 or 1388. And all table space is EBCDIC



# Difference in V7 and V8 for local application: Literal in statistic SQL (1)



# Difference in V7 and V8 for local application: Literal in statistic SQL (2)



# What's new risk in V8

- DB2 parser is UNICODE now
  - When we use V7 DBRM, BIND will convert into UNICODE first, then convert the literal back to TS encoding schema, which is an extra task to exposure the mismatch problem.
    - It will convert the literal from 935 to UNICODE, then translate back to 935, so one issue would be invalid code point if data and literal actually is 037 or something else.
  - In NFM, V8 pre-compiler will convert whole program into UNICODE, extract the SQL statement, then convert back
    - The question would be how to define the CCSID parameter in Pre-compile. To keep the same behavior of V7, then we may have to use default, which will get from DECP. Invalid code point issue could happen too, but now the scope will expand from SQL statement into whole program.
  - These extra convert will bring extra check for code point for literal and source program, also it bring the chance to exposure the mismatch problem.



# Sample 1-1

- Code point of /
  - X'E0' in 037
  - X'B2' in 935
  - COLA CHAR(1) CCSID 935 TS, but we store 037 '/', then the hex value is x'E0'. 935 SPUFI display is '\$'
  - SQL: Select \* from TBA where COLA="/" (coding in 037 PCOM, so the hex is x'E0')
- V7
  - Literal was passed as is, so DB2 return one match row.
  - V8 CM with V7 DBRM
    - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'E0'.
    - There is potential invalid character detect
    - DB2 return one qualify row
  - V8 NFM with Pre-Compile CCSID 037
    - Pre-compiler will convert '/' from 037 to UNICODE, then translated back to 935. The value would be x'B2'
    - There is potential invalid character detect
    - DB2 return +100
    - **Different from V7**
- V8 NFM with Pre-Compile CCSID 935
  - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'E0'.
  - There is potential invalid character detect
  - DB2 return one qualify row



# Sample 1-2

- Code point of /
  - X'E0' in 037
  - X'B2' in 935
  - COLA CHAR(1) CCSID 935 TS, but we store 037 '/', then the hex value is x'E0'. 935 SPUFI display is '\$'
  - SQL: Select \* from TBA where COLA="/" (coding in 935 PCOM, so the hex is x'B2')
- V7
  - Literal was passed as is, so DB2 return +100
  - V8 CM with V7 DBRM
    - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'B2'.
    - There is potential invalid character detect
    - DB2 return +100
  - V8 NFM with Pre-Compile CCSID 037
    - Pre-compiler will convert '/' from 037 to UNICODE, then translated back to 935. The value would be x'E0'
    - There is potential invalid character detect
    - DB2 return one qualify row
    - **Different from V7**
- V8 NFM with Pre-Compile CCSID 935
  - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'B2'.
  - There is potential invalid character detect
  - DB2 return +100



# Sample 1-3

- Code point of /
  - X'E0' in 037
  - X'B2' in 935
  - COLA CHAR(1) CCSID 935 TS, but we store 037 '/', then the hex value is x'E0'. 935 SPUFI display is '\$'
  - SQL: Select \* from TBA where COLA="\$" (coding in 935 PCOM, so the hex is x'E0')
- V7
  - Literal was passed as is, so DB2 return one qualify row
  - V8 CM with V7 DBRM
    - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'E0'.
    - There is potential invalid character detect
    - DB2 return one qualify row
  - V8 NFM with Pre-Compile CCSID 037
    - Pre-compiler will convert '/' from 037 to UNICODE, then translated back to 935. The value would be x'B2'
    - There is potential invalid character detect
    - DB2 return +100
    - **Different from V7**
- V8 NFM with Pre-Compile CCSID 935
  - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'E0'.
  - There is potential invalid character detect
  - DB2 return one qualify row



# Sample 2-1

- Code point of /
  - X'E0' in 037
  - X'B2' in 935
  - COLA CHAR(1) CCSID 935 TS, and we store 935 '/', then the hex value is x'B2'.
  - SQL: Select \* from TBA where COLA="/" (coding in 037 PCOM, so the hex is x'E0')
- V7
  - Literal was passed as is, so DB2 return +100.
- V8 CM with V7 DBRM
  - BIND will convert the '/' from 935 to UNICODE, then translate back to 935. The value is still x'E0'.
  - There is potential invalid character detect
  - DB2 return +100
- V8 NFM with Pre-Compile CCSID 037
  - Pre-compiler will convert '/' from 037 to UNICODE, then translated back to 935. The value would be x'B2'
  - There is potential invalid character detect
  - DB2 return one qualify row
  - **Different from V7**
- V8 NFM with Pre-Compile CCSID 935
  - BIND will convert the '¥' from 935 to UNICODE, then translate back to 935. The value is still x'E0'.
  - There is potential invalid character detect
  - DB2 return +100



# Sample 2-2

- Code point of /
  - X'E0' in 037
  - X'B2' in 935
  - COLA CHAR(1) CCSID 935 TS, and we store 935 '/', then the hex value is x'B2'.
  - SQL: Select \* from TBA where COLA='¥' (coding in 037 PCOM, so the hex is x'B2')
- V7
  - Literal was passed as is, so DB2 return one match row.
- V8 CM with V7 DBRM
  - BIND will convert the '¥' from 935 to UNICODE, then translate back to 935. The value is still x'B2'.
  - There is potential invalid character detect
  - DB2 return one qualify row
- V8 NFM with Pre-Compile CCSID 037
  - Pre-compiler will convert '¥' from 037 to UNICODE, then translated back to 935. The value would be x'5B'
  - There is potential invalid character detect
  - DB2 return +100
  - **Different from V7**
- V8 NFM with Pre-Compile CCSID 935
  - BIND will convert the '¥' from 935 to UNICODE, then translate back to 935. The value is still x'B2'.
  - There is potential invalid character detect
  - DB2 return one qualify row



# Suggest and Caution

- Do not cheat DB2
  - If you cheat DB2, then you need make sure the cheat is cover all place and future version
  - Highlight the rules used now to prevent problem when we cheat with DB2, enforce it into all coding people and system programmer.
  - For single byte EBCDIC, use the common set code point between CCSID (037, 935, 1388) may be much safe if you can not enforce people use 1388 all the time.
- May consider M CCSID=1388 if 935 was used now
  - 935 could be a problem for future DRDA access or JAVA application
- May alter the mismatch column into CHAR by subtype FOR BIT if you do not want DB2 convert the data and let application handle it by itself.
  - Be careful about the padding issue of FOR BIT
  - Need more test for these change and V8 environment
- Could consider ASCII TS and BIND ENCODING with ASCII but may have difficulty with ASCII literal if the BIND ENCODING is ASCII
- Thinking UNICODE in future application

