Db2 For z/OS and Unicode – What you need to know

Chris Crone

Broadcom Distinguished Engineer – Database Technologies

cjc@broadcom.com

Central Canada Db2 User Group
September 2023

Agenda

- ASCII, EBCDIC, and Unicode – Oh My!
- Conversion
- Db2 Encoding Basics
- Additional Considerations
- Summary

ASCII, EBCDIC, and Unicode Oh My!

CCSID 37 EBCDIC

	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	¤	®	×	õ	ÿ	Õ	(EO)

EBCDIC Variant Characters – 37/500 example

	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	¤	®	×	õ	ÿ	Õ	(EO)

	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	¢	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	[	]	¦	:	«	ª	¡	¬	-	¹	²	³
-B	.	$	,	#	»	º	¿	|	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	!	^	?	"	±	¤	®	×	õ	ÿ	Õ	(EO)

CCSID 367 – 7 Bit ASCII, Unicode ”FOR SBCS DATA”

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	NUL	DLE	(sp)	0	@	P	`	p								
-1	SOH	DC1	!	1	A	Q	a	q								
-2	STX	DC2	"	2	B	R	b	r								
-3	ETX	DC3	#	3	C	S	c	s								
-4	EOT	DC4	$	4	D	T	d	t								
-5	ENQ	NAK	%	5	E	U	e	u								
-6	ACK	SYN	&	6	F	V	f	v								
-7	BEL	ETB	'	7	G	W	g	w								
-8	BS	CAN	(	8	H	X	h	x								
-9	HT	EM	)	9	I	Y	i	y								
-A	LF	SUB	*	:	J	Z	j	z								
-B	VT	ESC	+	;	K	[	k	{								
-C	FF	FS	,	<	L	\	l	|								
-D	CR	GS	-	=	M	]	m	}								
-E	SO	RS	.	>	N	^	n	~								
-F	SI	US	/	?	O	_	o									

CCSID 819 – Superset of 367 – Common in Linux

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	NUL	DLE	(sp)	0	@	P	`	p		DCS	RSP	°	À	Ð	à	ð
-1	SOH	DC1	!	1	A	Q	a	q		PU1	¡	±	Á	Ñ	á	ñ
-2	STX	DC2	"	2	B	R	b	r	BPH	PU2	¢	²	Â	Ò	â	ò
-3	ETX	DC3	#	3	C	S	c	s	NBH	STS	£	³	Ã	Ó	ã	ó
-4	EOT	DC4	$	4	D	T	d	t	IND	CCH	¤	´	Ä	Ô	ä	ô
-5	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
-6	ACK	SYN	&	6	F	V	f	v	SSA	SPA	¦	¶	Æ	Ö	æ	ö
-7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	·	Ç	×	ç	÷
-8	BS	CAN	(	8	H	X	h	x	HTS	SOS	¨	¸	È	Ø	è	ø
-9	HT	EM	)	9	I	Y	i	y	HTJ		©	¹	É	Ù	é	ù
-A	LF	SUB	*	:	J	Z	j	z	VTS	SCI	ª	º	Ê	Ú	ê	ú
-B	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
-C	FF	FS	,	<	L	\	l	|	PLU	STS	¬	¼	Ì	Ü	ì	ü
-D	CR	GS	-	=	M	]	m	}	RI	OSC	-	½	Í	Ý	í	ý
-E	SO	RS	.	>	N	^	n	~	SS2	PM	®	¾	Î	Þ	î	þ
-F	SI	US	/	?	O	_	o	DEL	SS3	ACP	¯	¿	Ï	ß	ï	ÿ

Unicode – What is it

- Before Unicode, there were many different systems, character encodings
  - Even for English, no single encoding covered all the letters, punctuation, and technical symbols
  - Pictographic languages, such as Japanese, were a challenge to support
  - Early character encodings often conflicted with one another (For Example ASCII and EBCDIC)
    - Two encodings could use the same number for two different characters, or numbers for the same character
  - Data passed between computers or encodings increased the risk of data corruption or errors
- Character encodings existed for a handful of “large” languages.
  - But many languages lacked character support altogether (for instance Apache, or Klingon)
- The Unicode Consortium started out to standardize character encoding
  - Derives its name from three main goals:
    - universal (addressing the needs of world languages)
    - uniform (fixed-width codes for efficient access)
    - unique (bit sequence has only one interpretation into character codes)
  - It has expanded to be far more than character encoding. Its work now includes
    - the character properties and algorithms (the ‘instructions’ for how characters work)
    - language and locale data for internationalization
    - production software libraries to make everything accessible to programs

CCSID 1208* -Unicode UTF-8

UTF-8 encodes code points in one to four bytes, depending on the value of the code point. In the following table, [illegible] e code point:

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	[b]U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

The first 128 code points (ASCII) need one byte.

The next 1,920 code points need two bytes to encode, which covers the remainder of almost all Latin-script alphabets, and also IPA extensions, Greek, Cyrillic, Coptic, Armenian, Hebrew, Arabic, Syriac, Thaana and N'Ko alphabets, as well as Combining Diacritical Marks.

Three bytes are needed for the remaining 61,440 code points of the Basic Multilingual Plane (BMP), including most Chinese, Japanese and Korean characters.

Four bytes are needed for the 1,048,576 code points in the other planes of Unicode, which include emoji (pictographic symbols), less common CJK characters, various historic scripts, and mathematical symbols.

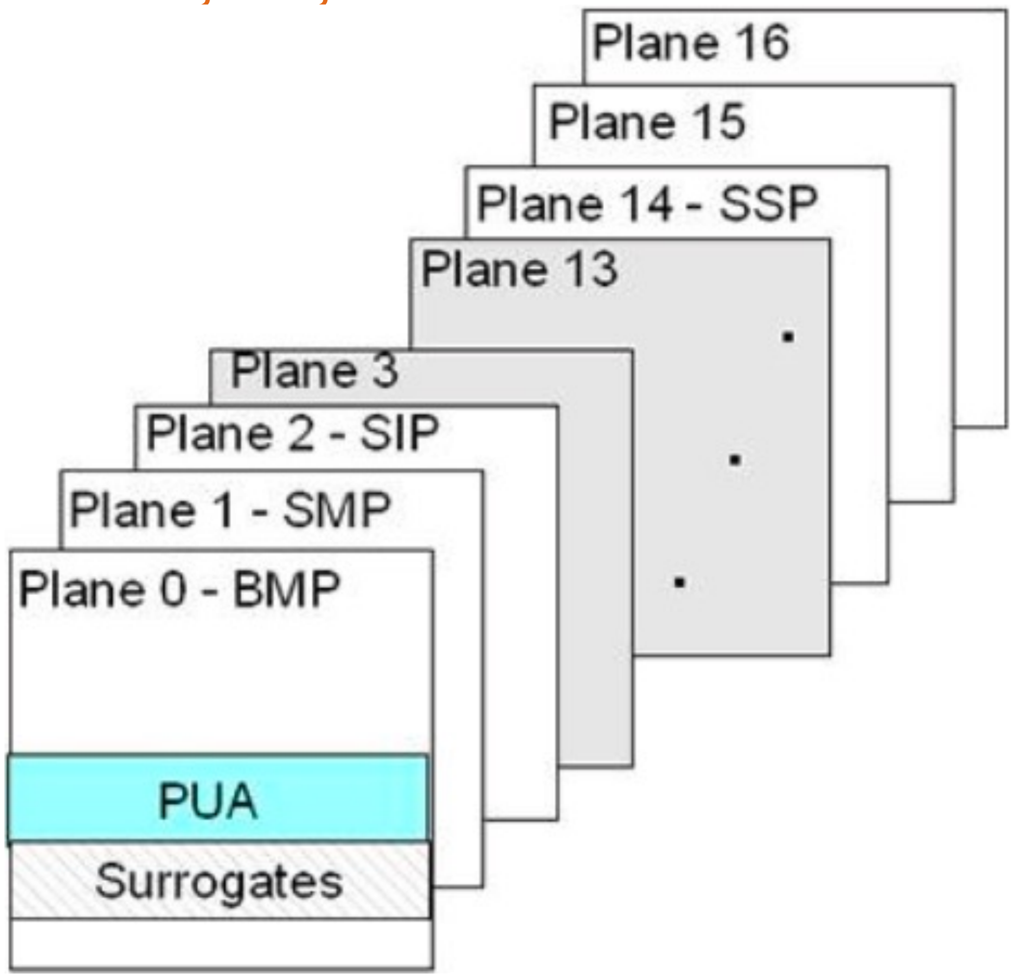
Source - Wikipedia

BROADCOM SOFTWARE

CCSID 1200* – Unicode UTF-16 – BMP-0

	000-	001-	002-	003-	004-	005-	006-	007-	008-	009-	00A-	00B-	00C-	00D-	00E-	00F-
-0	NUL	DLE	(sp)	0	@	P	`	p		DCS	(nbsp)	°	À	Ð	à	ð
-1	SOH	DC1	!	1	A	Q	a	q		PU1	¡	±	Á	Ñ	á	ñ
-2	STX	DC2	"	2	B	R	b	r	BPH	PU2	¢	²	Â	Ò	â	ò
-3	ETX	DC3	#	3	C	S	c	s	NBH	STS	£	³	Ã	Ó	ã	ó
-4	EOT	DC4	$	4	D	T	d	t	IND	CCH	¤	´	Ä	Ô	ä	ô
-5	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
-6	ACK	SYN	&	6	F	V	f	v	SSA	SPA	¦	¶	Æ	Ö	æ	ö
-7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	·	Ç	×	ç	÷
-8	BS	CAN	(	8	H	X	h	x	HTS	SOS	¨	¸	È	Ø	è	ø
-9	HT	EM	)	9	I	Y	i	y	HTJ		©	¹	É	Ù	é	ù
-A	LF	SUB	*	:	J	Z	j	z	VTS	SCI	ª	º	Ê	Ú	ê	ú
-B	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
-C	FF	FS	,	<	L	\	l	|	PLU	ST	¬	¼	Ì	Ü	ì	ü
-D	CR	GS	-	=	M	]	m	}	RI	OSC	-	½	Í	Ý	í	ý
-E	SO	RS	.	>	N	^	n	~	SS2	PM	®	¾	Î	Þ	î	þ
-F	SI	US	/	?	O	_	o	DEL	SS3	ACP	¯	¿	Ï	ß	ï	ÿ

UTF-16 – 17 Planes - 1,114,112 Characters



From - IBM Character Data Representation Architecture

Endianess

Big Endian

- pSeries (P8 and above can be Big and Little), zSeries, iSeries, Sun, HP Most significant byte is leftmost
  - For a 4 byte word - Byte order 0,1,2,3

Little Endian

- Intel based machines including xSeries Least significant byte is leftmost
  - For a 4 byte word - Byte order 3,2,1,0

UTF-8 - not affected by endianess issues

UTF-16 and UTF-32 are effected by endianess issues Big Endian

- 'A' = x'0041' for UTF-16 or x'00000041' for UTF-32 Little Endian
- 'A' = x'4100' for UTF-16 or x'41000000' for UTF-32

Note: A BYTE is always ordered as leftmost most significant bit to rightmost least significant bit. Bit order within a byte is always 7,6,5,4,3,2,1,0

BROADCOM SOFTWARE

Conversion

Conversion Overview

Three Basic Technologies Used on z
- ICONV (used by USS and FTP)
- SYSIBM.SYSSTRINGS – Db2 base conversion capability
- z/OS Unicode Conversion Services – Used for most new conversions post Db2 V

Two Essential Conversion Techniques
- Round Trip (RT)
  - In General – SYSIBM.SYSTRINGS uses RT
  - In General – ASCII/EBCDIC -> Unicode conversions are RT (Subset -> Superset)
- Enforced Subset (ES)
  - All Unicode -> ASCII/EBCDIC conversions are ES (Superset -> Subset)

CR+LF and NL
- Not usually an issue with Db2 data
- ICONV maps CR+LF -> NL (and vice versa) in ASCII<->EBCDIC Conversions
  - Necessary for C/C++ Source Code
  - Not compatible with Db2 conversions

BROADCOM SOFTWARE

ASCII to EBCDIC can be problematic (1252 -> 37)

	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	&	-	ø	Ø	°	µ	^	{	}	\	0
-1	(rsp)	é	/	É	a	j	~	£	A	J	÷	1
-2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
-3	ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
-4	à	è	À	È	d	m	u	©	D	M	U	4
-5	á	í	Á	Í	e	n	v	§	E	N	V	5
-6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
-7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
-8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
-9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
-A	¢	!	¦	:	«	ª	¡	[	-	¹	²	³
-B	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
-C	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
-D	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
-E	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
-F	|	¬	?	"	±	€	®	×	õ	ÿ	Õ	(EO)

	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(sp)	0	@	P	`	p	€		(rsp)	°	À	Ð	à	ð
-1	!	1	A	Q	a	q		‘	¡	±	Á	Ñ	á	ñ
-2	"	2	B	R	b	r	‚	’	¢	²	Â	Ò	â	ò
-3	#	3	C	S	c	s	ƒ	“	£	³	Ã	Ó	ã	ó
-4	$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
-5	%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ
-6	&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö
-7	'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
-8	(	8	H	X	h	x	ˆ	˜	¨	¸	È	Ø	è	ø
-9	)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù
-A	*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
-B	+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û
-C	,	<	L	\	l	|	Œ	œ	¬	¼	Ì	Ü	ì	ü
-D	-	=	M	]	m	}			-	½	Í	Ý	í	ý
-E	.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î	þ
-F	/	?	O	_	o	DEL		Ÿ	¯	¿	Ï	ß	ï	ÿ

BROADCOM® SOFTWARE

RT –vs – ES – What happens?

1252 to 37
Round Trip

00:	00	01	02	03	37	2d	2e	2f	16	05	25	0b	0c	0d	0e	0f
10:	10	11	12	13	3c	3d	32	26	18	19	3f	27	1c	1d	1e	1f
20:	40	5a	7f	7b	5b	6c	50	7d	4d	5d	5c	4e	6b	60	4b	61
30:	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	7a	5e	4c	7e	6e	6f
40:	7c	c1	c2	c3	c4	c5	c6	c7	c8	c9	d1	d2	d3	d4	d5	d6
50:	d7	d8	d9	e2	e3	e4	e5	e6	e7	e8	e9	ba	e0	bb	b0	6d
60:	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
70:	97	98	99	a2	a3	a4	a5	a6	a7	a8	a9	c0	4f	d0	a1	07
80:	20	21	22	23	24	15	06	17	28	29	2a	2b	2c	09	0a	1b
90:	30	31	1a	33	34	35	36	08	38	39	3a	3b	04	14	3e	ff
a0:	41	aa	4a	b1	9f	b2	6a	b5	bd	b4	9a	8a	5f	ca	af	bc
b0:	90	8f	ea	fa	be	a0	b6	b3	9d	da	9b	8b	b7	b8	b9	ab
c0:	64	65	62	66	63	67	9e	68	74	71	72	73	78	75	76	77
d0:	ac	69	ed	ee	eb	ef	ec	bf	80	fd	fe	fb	fc	ad	ae	59
e0:	44	45	42	46	43	47	9c	48	54	51	52	53	58	55	56	57
f0:	8c	49	cd	ce	cb	cf	cc	e1	70	dd	de	db	dc	8d	8e	df

ABC

TM

0123456789

SUB

1252 to 37
Enforced Subset

00:	00	01	02	03	37	2d	2e	2f	16	05	25	0b	0c	0d	0e	0f
10:	10	11	12	13	3c	3d	32	26	18	19	3f	27	1c	1d	1e	1f
20:	40	5a	7f	7b	5b	6c	50	7d	4d	5d	5c	4e	6b	60	4b	61
30:	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	7a	5e	4c	7e	6e	6f
40:	7c	c1	c2	c3	c4	c5	c6	c7	c8	c9	d1	d2	d3	d4	d5	d6
50:	d7	d8	d9	e2	e3	e4	e5	e6	e7	e8	e9	ba	e0	bb	b0	6d
60:	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
70:	97	98	99	a2	a3	a4	a5	a6	a7	a8	a9	c0	4f	d0	a1	07
80:	3f	3f	3f	3f	3f	15	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f
90:	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f	3f
a0:	41	aa	4a	b1	9f	b2	6a	b5	bd	b4	9a	8a	5f	ca	af	bc
b0:	90	8f	ea	fa	be	a0	b6	b3	9d	da	9b	8b	b7	b8	b9	ab
c0:	64	65	62	66	63	67	9e	68	74	71	72	73	78	75	76	77
d0:	ac	69	ed	ee	eb	ef	ec	bf	80	fd	fe	fb	fc	ad	ae	59
e0:	44	45	42	46	43	47	9c	48	54	51	52	53	58	55	56	57
f0:	8c	49	cd	ce	cb	cf	cc	e1	70	dd	de	db	dc	8d	8e	df

SUB

BROADCOM SOFTWARE

RT –vs – ES – What happens? (cont)

37 to 1252
Round Trip

```
00:00 01 02 03 9c 09 86 7f 97 8d 8e 0b 0c 0d 0e 0f
10:10 11 12 13 9d 85 08 87 18 19 92 8f 1c 1d 1e 1f
20:80 81 82 83 84 0a 17 1b 88 89 8a 8b 8c 05 06 07
30:90 91 16 93 94 95 96 04 98 99 9a 9b 14 15 9e 1a
40:20 a0 e2 e4 e0 e1 e3 e5 e7 f1 a2 2e 3c 28 2b 7c
50:26 e9 ea eb e8 ed ee ef ec df 21 24 2a 29 3b ac
60:2d 2f c2 c4 c0 c1 c3 c5 c7 d1 a6 2c 25 5f 3e 3f
70:f8 c9 ca cb c8 cd ce cf cc 60 3a 23 40 27 3d 22
80:d8 61 62 63 64 65 66 67 68 69 ab bb f0 fd fe b1
90:b0 6a 6b 6c 6d 6e 6f 70 71 72 aa ba e6 b8 c6 a4
a0:b5 7e 73 74 75 76 77 78 79 7a a1 bf d0 dd de ae
b0:5e a3 a5 b7 a9 a7 b6 bc bd be 5b 5d af a8 b4 d7
c0:7b 41 42 43 44 45 46 47 48 49 ad f4 f6 f2 f3 f5
d0:7d 4a 4b 4c 4d 4e 4f 50 51 52 b9 fb fc f9 fa ff
e0:5c f7 53 54 55 56 57 58 59 5a b2 d4 d6 d2 d3 d5
f0:30 31 32 33 34 35 36 37 38 39 b3 db dc d9 da 9f
```

ABC

SUB

1252 to 37
Enforced Subset

```
00:00 01 02 03 1a 09 1a 7f 1a 1a 1a 0b 0c 0d 0e 0f
10:10 11 12 13 1a 85 08 1a 18 19 1a 1a 1c 1d 1e 1f
20:1a 1a 1a 1a 1a 0a 17 1b 1a 1a 1a 1a 1a 05 06 07
30:1a 1a 16 1a 1a 1a 1a 04 1a 1a 1a 1a 14 15 1a 1a
40:20 a0 e2 e4 e0 e1 e3 e5 e7 f1 a2 2e 3c 28 2b 7c
50:26 e9 ea eb e8 ed ee ef ec df 21 24 2a 29 3b ac
60:2d 2f c2 c4 c0 c1 c3 c5 c7 d1 a6 2c 25 5f 3e 3f
70:f8 c9 ca cb c8 cd ce cf cc 60 3a 23 40 27 3d 22
80:d8 61 62 63 64 65 66 67 68 69 ab bb f0 fd fe b1
90:b0 6a 6b 6c 6d 6e 6f 70 71 72 aa ba e6 b8 c6 a4
a0:b5 7e 73 74 75 76 77 78 79 7a a1 bf d0 dd de ae
b0:5e a3 a5 b7 a9 a7 b6 bc bd be 5b 5d af a8 b4 d7
c0:7b 41 42 43 44 45 46 47 48 49 ad f4 f6 f2 f3 f5
d0:7d 4a 4b 4c 4d 4e 4f 50 51 52 b9 fb fc f9 fa ff
e0:5c f7 53 54 55 56 57 58 59 5a b2 d4 d6 d2 d3 d5
f0:30 31 32 33 34 35 36 37 38 39 b3 db dc d9 da 1a
```

0123456789

Some help on the Mainframe – Broadcom XCSD

```
------------ Character Conversion Service Utility ----------- 2023/07/25 11:11

Conversion Parameters:                                     64-bit Path ==> N (Y|N)
 Input CCSID       ==> 1208   Output CCSID     ==> 1200
 Force z/OS        ==> Y     (N - Quick Conversion,
                              Y - z/OS Conversion Service)
 SubAct            ==> Y     (N-No Sub Chars allowed. Error might be returned.
                              Y-Replace Sub Char(s) with output CCSID '?' mark.
                              S-Output CCSID Sub Char(s) are allowed)
 Input Lines       ==> 2     (Desired number of Input Lines 1-99)
 Input Format      ==> S     (T - Text, S - Hex String, V - 2-line Hex Vertical)
 Output Format     ==> D     (T - Text, D - Dump)
 Round Trip and    ==> N     (Convert Output back to Input and compare the
 Compare                      result with Input Data)

 (Terminal CCSID    ==> 00037)

 Input Data:
  486567656C73747261E1BA9E65
  ----------------------------------------------------------------------------

 Output Data:
  00480065 00670065 006C0073 00740072   |Hegelstr         |
  00611E9E 0065                         |a?e              |
******************************* BOTTOM OF DATA ********************************

Command ==>                                                     SCROLL ===> PAGE
 F1=HELP       F2=SPLIT      F3=END        F4=RETURN     F5=RFIND      F6=RCHANGE
 F7=UP         F8=DOWN       F9=SWAP       F10=LEFT      F11=RIGHT     F12=RETRIEVE
  *CSPCHRU
```

Variant Characters in EBCDIC multi-CCSID environments

EBCDIC has a limited number of invariant characters -

A LA020000	B LB020000	C LC020000	D LD020000	E LE020000	F LF020000	G LG020000	H LH020000	I LI020000	J LJ020000	K LK020000	L LL020000	M LM020000
N LN020000	O LO020000	P LP020000	Q LQ020000	R LR020000	S LS020000	T LT020000	U LU020000	V LV020000	W LW020000	X LX020000	Y LY020000	Z LZ020000
a LA010000	b LB010000	c LC010000	d LD010000	e LE010000	f LF010000	g LG010000	h LH010000	i LI010000	j LJ010000	k LK010000	l LL010000	m LM010000
n LN010000	o LO010000	p LP010000	q LQ010000	r LR010000	s LS010000	t LT010000	u LU010000	v LV010000	w LW010000	x LX010000	y LY010000	z LZ010000
0 ND100000	1 ND010000	2 ND020000	3 ND030000	4 ND040000	5 ND050000	6 ND060000	7 ND070000	8 ND080000	9 ND090000			
+ SA010000	< SA030000	= SA040000	> SA050000	% SM020000	& SM030000	* SM040000	" SP040000	' SP050000	(SP060000	) SP070000	, SP080000	_ SP090000
- SP100000	. SP110000	/ SP120000	: SP130000	; SP140000	? SP150000							

Character Set 00640

- Be aware of Variant Characters – Character set 640 documents the only characters that are invariant
- Noticeably missing are commonly used characters such as:
  - $, #, @ - which are special z/OS data set naming characters
- Other significant variant characters
  - Lowercase letters (a-z), |, ^, %, *, and even “

```
------------ Character Conversion Service Utility ----------- 2023/07/25 12:26

Conversion Parameters:                                  64-bit Path ==> N (Y|N)
 Input CCSID        ==> 00037  Output CCSID     ==> 0277
 Force z/OS         ==> Y     (N - Quick Conversion,
                               Y - z/OS Conversion Service)
 SubAct             ==> Y     (N-No Sub Chars allowed. Error might be returned.
                               Y-Replace Sub Char(s) with output CCSID '?' mark.
                               S-Output CCSID Sub Char(s) are allowed)
 Input Lines        ==> 2     (Desired number of Input Lines 1-99)
 Input Format       ==> S     (T - Text, S - Hex String, V - 2-line Hex Vertical)
 Output Format      ==> D     (T - Text, D - Dump)
 Round Trip and     ==> N     (Convert Output back to Input and compare the
 Compare                       result with Input Data)

 (Terminal CCSID   ==> 00037)

 Input Data:
  5B407C407B
  ------------------------------------------------------------------------------

 Output Data:
  67408040 4A                           *$ @ #          *
******************************* BOTTOM OF DATA ********************************
```

 |

BROADCOM SOFTWARE

Browsing and Editing ASCII in ISFP?

```
   Menu  Utilities  Compilers  Help
 BROWSE    .MISC.JCL(ASCIIEBC) - 01.07        Line 0000000000 Col 001 080
********************************** Top of Data ***************************
èçñë.ñë. +. ëäññ.<ñ+á.....................................................
This line is EBCDIC
******************************** Bottom of Data **************************
```

display line 1 5 cols 20 30 ccsid 819 (note can specify 1208 or 1200)

```
   Menu  Utilities  Compilers  Help
 BROWSE    .MISC.JCL(ASCIIEBC) - 01.07        Li      Converted data shown
********************************** Top of Data ***************************
THIS IS AN ASCII LINE                                             ........
This line is EBCDIC
******************************** Bottom of Data **************************
```

```
   File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 EDIT              MISC.JCL(ASCIIEBC) - 01.07          Columns 00001 00072
****** ***************************** Top of Data **************************
000001 èçñë ñë  +  ëäññ <ñ+á
000002 This line is EBCDIC
****** **************************** Bottom of Data ************************
```

source ascii

```
   File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 EDIT      .MISC.JCL(ASCIIEBC) - 01.07          Columns 00001 00072
****** ***************************** Top of Data **************************
000001 THIS IS AN ASCII LINE
000002 Òêëó@ôëòà@ëó@      @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
****** **************************** Bottom of Data ************************
```

source reset

```
   File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 EDIT              MISC.JCL(ASCIIEBC) - 01.07          Columns 00001 00072
****** ***************************** Top of Data **************************
000001 èçñë ñë  +  ëäññ <ñ+á
000002 This line is EBCDIC
****** **************************** Bottom of Data ************************
```

*Note – when browsing a dump, you can use ASCII/EBCDIC command in IPCS to see storage as basic ASCII/EBCDIC

BROADCOM SOFTWARE

To Covert or Not to Covert -That is the Question

Performance and Functional preference

- No Conversion
- Lossless Conversions
  - UTF-8<->UTF-16
  - Latin-1<->Latin-1(e.g., 37<->500)
  - ASCII/EBCDIC->Unicode
- Round Trip
  - Most useful in a 2-tier,homogenousenvironment
  - Watch out for characters that may not be in the “right” spot due to RT conversions
- Enforced Subset
  - Unicode->ASCII/EBCDIC
  - Latin-n->Latin-m

Db2 Encoding Basics

Db2 Encoding Basics (Mixed = NO System)

- Prior to Db2 V8, Db2 internals were EBCDIC based
  - The System EBCDIC CCSID (SCCSID in DECP) was the basis of processing
  - EBCDIC still used for most z/OS APIs
    - ESM (SAF) userid processing for instance
- Post Db2 V8 processing is Unicode UTF-8 Based
  - Parsing (Special EBCDIC Dependent Processing)
  - Catalog
  - Internal string representation
  - Some conversion for things like IFCIDs
- GETVARIABLE can be used to retrieve your CCSIDs

```
SET :hv3 = GETVARIABLE('SYSIBM.SYSTEM_EBCDIC_CCSID');   -- can be ASCII, EBCDIC or UNICODE
:hv3 = '37,65534,65534'
```


BROADCOM SOFTWARE

How is Data Stored?

EBCDIC and ASCII

- Data is stored in CHAR, VARCHAR or CLOB in the specified SYSTEM CCSID
  - Only one EBCDIC or ASCII CCSID per system

Unicode

- CHAR/VARCHAR/CLOB FOR SBCS DATA
  - CCSID 367 - 7 Bit ASCII
- CHAR/VARCHAR/CLOB FOR MIXED DATA -- Optional as this is the Default
  - CCSID 1208 - UTF-8
- GRAPHIC/VARGRAPHIC/DBCLOB
  - CCSID 1200 – UTF-16

XML is UTF-8 by default regardless of the associated base table

Controlling OBJECT Encoding

CREATE DATABASE … CCSID ASCII|EBCDIC|UNICODE

Defaults to ENSCHEME DECP Value

CREATE TABLESPACE … CCSID ASCII|EBCDIC|UNICODE

Defaults to Encoding Scheme of the DATABASE if not specified

CREATE TABLE … CCSID ASCII|EBCDIC|UNICODE

Must match TABLESPACE

All tables in a TABLESPACE must be same encoding

Other - specified on CREATE – for example

```
CREATE PROCEDURE
  my_sp( in in_parm1 char(10) ccsid unicode )
```


Controlling Encoding in an Application

DECLARE VARIABLE statement

- Mechanism to allow CCSID to be specified for host variables
- Example
  - EXEC SQL DECLARE :hv1 CCSID UNICODE;
  - EXEC SQL DECLARE :hv2 CCSID 37;

Precompiler directive to specify hostvar CCSID

Useful for PREPARE / EXECUTE IMMEDIATE statement text

- EXEC SQL PREPARE S1 FROM :hv2; - Statement text would be in EBCDIC 37

May be used with any string host variable on input or output

Controlling Encoding at Bind

- Application Encoding Scheme – ENCODING Parm in DECP
  - System Default
    - Determines Encoding Scheme when none is explicitly specified
  - Bind Option
    - Allows explicit specification of ES at an application level. Affects Static SQL
    - Provides default for dynamic SQL
    - System Default used if bind option not specified
  - Special Register
    - Allows explicit specification of ES at the application level.
    - Affects Dynamic SQL
    - Initialized with value from ENCODING Bind Option
  - DRDA
    - OPTION is ignored (for CCSID info) when packages are executed remotely
    - DRDA specified Input CCSID, Data flows as is to client
  - Used in SET and Multiple CCSID statements
- Be careful specifying ENCODING(UNICODE)

Literals

Character literals may be used for all string data

INSERT INTO T1 (C1) VALUES (‘A°’);

INSERT INTO T1 (G1) VALUES (‘abc’); -- converted to UTF-16

Graphic literals should only be used for Graphic data

INSERT INTO T1 (G1) VALUES (G’胢脷’); -- U+80E2 U+8137

Hex literals should only be used for character data

INSERT INTO T1 (C1) VALUES (X’3132’);

UX (UTF-16) and GX literals

INSERT INTO T1 (C1) VALUES (UX’80E28137’);

INSERT INTO T1 (G1) VALUES (GX’42C142C2’);

GX encoding is determined by Application Encoding Scheme

BROADCOM SOFTWARE

Multiple CCSIDs in SQL Statements

```
SELECT
a.name, a.creator, b.charcol, 'ABC', :hvchar, X'C1C2C3'
FROM sysibm.systables a, ebcdictable b
WHERE
a.name = b.name AND
 b.name > 'B' AND
a.creator = 'SYSADM'
ORDER BY b.name;
```

Result or Evaluated:
EBCDIC
Unicode
Application Encoding Scheme

BROADCOM SOFTWARE

General Rules for Multiple CCSID sets

Comparison and resulting data types for multiple CCSID sets...

If an expression or comparison involves two strings which contain columns with different CCSID sets, Drive to Unicode if necessary:

WHERE T1.C1 = T2.C1

If an expression or comparison involves two strings with different CCSID sets where only one of them contains a column, Drive to the column's CCSID set

WHERE T1.C1 = X'C1C2'

If an expression or comparison involves two strings with different CCSID sets and neither contains a column, Drive to Unicode

WHERE GX'42C142C2' = ‘ABC‘ -- GX literal and ‘ABC’ are different CCSIDs

String constants and special registers in a context by themselves use the application encoding scheme

SELECT ‘ABC‘ FROM T1 . . .

BROADCOM SOFTWARE

Performance Consdierations

In general, UTF-8 will perform similarly to EBCDIC for SBCS data

- 2, 3, and 4-byte characters may cause some processing and storage overhead

Mixing Unicode and EBCDIC at the API Layer generally not a problem

- More than 20 years of JDBC accessing EBCDIC tables

Multi-CCSID statements can have varying effects

- Literal value converted once at bind time – negligible
- Host Variables converted at OPEN – negligible
- SELECT list item converted row after row – negligible for DRDA, Expensive for Local
- Join predicate conversion
  - Depends on Join Type and number of rows
- GROUP BY or ORDER BY
  - Could suffer both a materialization AND conversion cost

TEST, TEST, TEST

Expanding and Contracting Conversions

Example – Å

CCSID	HEX Representation
819	x'C5'
1208	x'C385'
1200	X'00C5'

The conversion of the character Å from CCSID 819 to CCSID 1208 (UTF-8) is an expanding conversion.

The conversion of the character Å from CCSID 1208 to CCSID 819 is a contracting conversion

Question – Is conversion from 819 to 1200 (UTF-16) an expanding conversion?

Answer – No, because Å is a single code unit in UTF-16 (taking 1 GRAPHIC or VARGRAPHIC character).

Character Based Functions

CHARACTER_LENGTH

CHAR

CLOB

DBCLOB

GRAPHIC

INSERT

INSTR

LEFT

LOCATE

LOCATE_IN_STRING

OVERLAY

- POSITION
- REGXP_* (IDAA Passthru)
- RIGHT
- STRLEFT
- STRRIGHT
- SUBSTRING
- VARCHAR
- VARGRAPIC
- CAST Specification (Code Units can be specified)

Character Based Function Examples

Function	Returns	
SUBSTR(‘Hegelstraße’,1,10)	‘Hegelstra?’	– x’486567656C73747261E1’
SUBSTRING(‘Hegelstraße’,1,10,OCTETS)	‘Hegelstra ’	– x’486567656C7374726120’
SUBSTRING(‘Hegelstraße’,1,11,OCTETS)	‘Hegelstra ’	– x’486567656C737472612020’
SUBSTRING(‘Hegelstraße’,1,11,CODEUNITS16)	‘Hegelstraße’	– x’486567656C73747261E1BA9E65’
SUBSTRING(‘Hegelstraße’,1,11,CODEUNITS32)	‘Hegelstraße’	– x’486567656C73747261E1BA9E65’

Assume that T1 is a Unicode table with column C1 (VARCHAR(10)) with one row with the value of the mathematical bold capital $\mathbf{A}$ (X'F09D9080’) – it looks like a bold “**A**”, but it’s not.

The following similar queries return different answers:

```
SELECT CHARACTER_LENGTH(C1,CODEUNITS32) FROM T1;      -- Returns 1
SELECT CHARACTER_LENGTH(C1,CODEUNITS16) FROM T1;      -- Returns 2
SELECT CHARACTER_LENGTH(C1,OCTETS) FROM T1;           -- Returns 4
```

The following similar queries return different answers:

```
SELECT HEX(SUBSTRING(C1,1,1,CODEUNITS32)) FROM T1;    -- Returns X'F09D9080’
SELECT HEX(SUBSTRING(C1,1,1,CODEUNITS16)) FROM T1;    -- Returns X'20’
SELECT HEX(SUBSTRING(C1,1,2,CODEUNITS16)) FROM T1;    -- Returns X'F09D9080’
SELECT HEX(SUBSTRING(C1,1,1,OCTETS)) FROM T1;         -- Returns X'20‘ (blank)
SELECT HEX(SUBSTR(C1,1,1)) FROM T1;                   -- Returns X'F0'
```


ASCII_STR and EBCDIC_STR

Enable Unicode data to be returned without substitution in ASCII or EBCDIC. Data not directly convertible is “escaped”

Assuming T1.C1 contains Unicode string – "Hi, my name is Андрей”

SELECT C1 FROM T1;

Returns (to 3270 CCSID 37 screen)

‘Hi, my name is ......’

SELECT ASCII_STR(C1)FROM T1; -- Note EBCDIC_STR would return the same result

Returns

‘Hi, my name is \0410\043D\0434\0440\0435\0439’ -- Escaped UTF-16

UNISTR or UNICODE_STR

Accepts “escaped” Unicode input and converts it to UTF-8 or UTF-16.

Assuming T1.C1 contains “Андрей”

```
SELECT ASCII_STR(C1)FROM SYSIBM.SYSDUMMYE;
 Returns '\0410\043D\0434\0440\0435\0439' – Escaped UTF-16

SELECT HEX(UNISTR(ASCII_STR(C1))) FROM SYSIBM.SYSDUMMYE;
 Returns 'D090D0BDD0B4D180D0B5D0B9' – HEX of UTF-8

SELECT HEX(UNISTR(ASCII_STR(C1),UTF16)) FROM SYSIBM.SYSDUMMYE;
 Returns '0410043D043404400435 0439' – HEX of UTF-16
```


COLLATION_KEY

CREATE TABLE T1 (C1 VARCHAR(6)) CCSID UNICODE;

INSERT INTO T1 VALUES('Cat’);

INSERT INTO T1 VALUES('cat’);

SELECT C1 FROM T1
ORDER BY C1 ;

Cat
cat

SELECT C1 FROM T1
ORDER BY
COLLATION_KEY(C1,'UCA410_LEL_CL’);

cat
Cat

COLLATION_KEY with Locales

```
CREATE TABLE T1 (C1 VARCHAR(6) ) CCSID UNICODE;
INSERT INTO T1 VALUES('cote',1);
INSERT INTO T1 VALUES('côté',2);
INSERT INTO T1 VALUES('côte',3);
INSERT INTO T1 VALUES('coté',4);
```

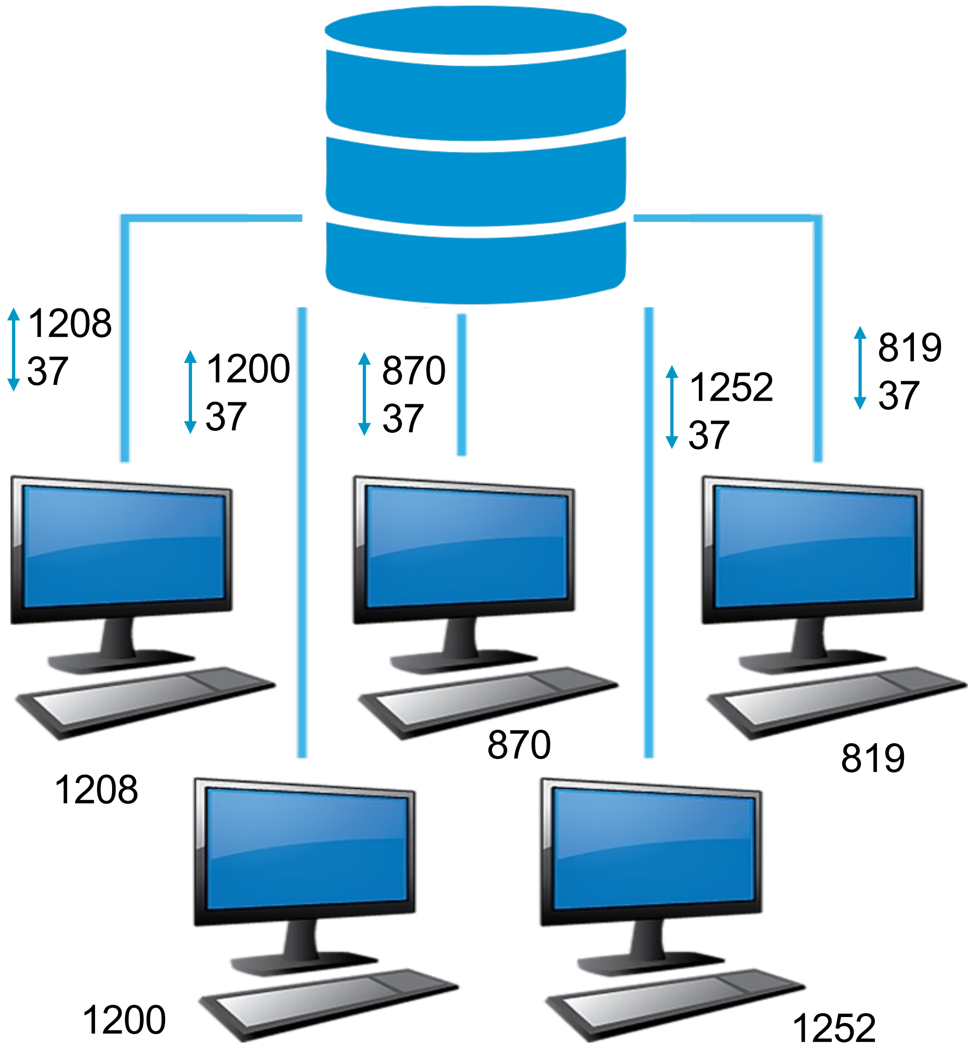
```
SELECT C1 FROM T1
  ORDER BY C1;
```

cote
coté
côte
côté

```
SELECT C1 FROM T1
  ORDER BY
COLLATION_KEY(C1,'UCA410_LFR_FO');
```

cote
côte
coté
côté

Db2 - EBCDIC CCSID 37



 |

BROADCOM SOFTWARE

COBOL

Enterprise COBOL V3R1+ Supports Unicode

NATIONAL is used to declare UTF-16 variables
MY-UNISTR pic N(10). -- declares a UTF-16 Variable
N and NX Literals

```
N'123'
NX'003100320033'
```

National Groups*

```
01 Alpha-Group-1.
  02 Group-1.
    04 Month PIC 99.
    04 DayOf PIC 99.
    04 Year PIC 9999.
  02 Group-2 GROUP-USAGE NATIONAL.
    04 Amount PIC 9(4).99.
```

*Can be subordinated under an alpha group, but alpha groups cannot be subordinated under a national group.

Conversions

```
NATIONAL-OF Converts to UTF-16
DISPLAY-OF Converts to specific CCSID
DECLARE Greek-EBCDIC pic X(10) value "ΞΣΦΛΘΖΔΓΩ".
UTF16STR pic N(10).
UTF8STR pic X(20).
Move Function National-of(Greek-EBCDIC, 00875) to UTF16STR.
Move Function Display-of(UTF16STR, 01208) to UTF8STR.
```


COBOL - Db2 Doing the Conversion

```
EXEC SQL BEGIN DECLARE SECTION; 01 HOST-VARS.

05 GREEK-EBCDIC PIC X(10) VALUE "ΞΣΦΛΘΖΔΓΩ".
05 UTF16STR PIC N(10).*
05 UTF8STR PIC X(20).

EXEC SQL DECLARE :UTF8STR VARIABLE CCSID 1208.

EXEC SQL END DECLARE SECTION;

INSERT INTO T1 (C1) VALUES(:GREEK-EBCDIC) END EXEC.**

EXEC SQL
 SELECT C1, C1 INTO :UTF16STR, :UTF8STR END EXEC.
```

* COBOL will use an implicit DECLARE VARIABLE for PIC N data.
** This example assumes T1 is a table encoded in EBCDIC CCSID 875 and that the ENCODING bind option for this appl is also CCSID 875.

BROADCOM SOFTWARE

PL/I

```
%PROCESS CODEPAGE(277), WIDECHAR(BIGENDIAN);
DCL UTF16STR WIDECHAR(10) VARYING;
DCL uOneTwoThree WCHAR(3);
DCL eOneTwoThree CHAR(3);
uOneTwoThree = WX’003100320033’;                  /* UTF-16 ‘123’.              */
eOneTwoThree = ‘123’;                             /* EBCDIC ‘123’ – x’F1F2F3’ */

IF uOneTwoThree = eOneTwoThree THEN.                    /* Evaluates False       */
...
IF uOneTwoThree = WIDECHAR(eOneTwoThree) THEN /* Evaluates True         */
...
UTF16STR = WIDECHAR(‘ABC@’);
/* note ‘@’ is assumed to be in CCSID 273 position (x’B5’) because of the CODEPAGE(273)
specification. UTF16STR now = x’0041004200430040’  - @ is at x’0040’  /*
```


BROADCOM SOFTWARE

PL/I – USING DESCRIPTOR to specify CCSID

```
DCL STMT1 CHAR(100) VARYING INIT('INSERT INTO T1 VALUES (?,?) ');
...
DCL DA1 CHAR(16+(2*44));
/* ALLOCATE SPACE FOR 2 SQLDA ENTRIES */
EXEC SQL INCLUDE SQLDA;
SQLDAID = 'SQLDA+ ';                          /* Note the "+"          */
SQLN = 2;                                     /* Allocated SQLVARS     */
SQLD = 2;                                     /* Used SQLVARS          */
SQLVAR(1).SQLTYPE = 468;                      /* Graphic – not null    */
SQLVAR(1).SQLLEN1 = 3;                        /* Length = 6 bytes      */
SQLVAR(1).SQLDATA = ADDR(uOneTwoThree);       /* Address of host var   */
SQLVAR(1).SQLNAME = '0000048000000000'X;      /* CCSID 1200            */
SQLVAR(2).SQLTYPE = 452;                      /* Character – not null  */
SQLVAR(2).SQLLEN1 = 3;                        /* Length = 3 bytes      */
SQLVAR(2).SQLDATA = ADDR(eOneTwoThree);       /* Address of host var   */
SQLVAR(2).SQLNAME = '0000011100000000'X;      /* CCSID 273             */

EXEC SQL PREPARE S1 FROM :STMT1;              /* Prepare Statement     */
EXEC SQL EXECUTE S1 USING DESCRIPTOR :SQLDA;  /* Execute Stmt          */
```

 |

COPROCESSOR – CCSID Options

COPROCESSOR must be used for Unicode applications in COBOL or PL/I

- COBOL
  - SQLCCSID (default)
  - NOSQLCCSID
- PL/I
  - CCSID0 (default)
  - NOCCSID0

DCLGEN

DBCSSYMBOL

- Specifies the symbol used to denote a graphic data type in a COBOL PICTURE clause.
  - (G) Graphic data is denoted using G.
  - (N) Graphic data is denoted using N.

DCLBIT

- Specifies if DCLGEN should be sensitive to the declaration of FOR BIT DATA items
  - (NO) – backward compatible behavior
  - (YES) – Causes DCLGEN to create a DECLARE VARIABLE statement for columns in DB2 that were declared with the FOR BIT DATA clause

DCLGEN EXAMPLE

Wrong Way

```
*******************************************************
* DCLGEN TABLE(ADMF001.T1)                            *
* LIBRARY(USER.DBRMLIB.DATA(T4))                      *
* LANGUAGE(COBOL)                                     *
* QUOTE                                               *
* ...                                                 *
* IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STMTS  *
*******************************************************
EXEC SQL DECLARE ADMF001.T1 TABLE
    ( NAME                     VARGRAPHIC(15),
      ADDRESS                  VARGRAPHIC(25),
      CITY                     VARGRAPHIC(20),
      STATE                    GRAPHIC(2),
      ZIP                      GRAPHIC(50),
      PASSWORD                 CHAR(8)
    ) END-EXEC.
*******************************************************
* COBOL DECLARATION FOR TABLE ADMF001.T1.             *
*******************************************************
     01 DCLT1.
       10 NAME.
        49 NAME-LEN            PIC S9(4) USAGE COMP.
        49 NAME-TEXT           PIC G(15) USAGE DISPLAY-1.
       10 ADDRESS.
        49 ADDRESS-LEN         PIC S9(4) USAGE COMP.
        49 ADDRESS-TEXT        PIC G(25) USAGE DISPLAY-1.
       10 CITY.
        49 CITY-LEN            PIC S9(4) USAGE COMP.
        49 CITY-TEXT           PIC G(20) USAGE DISPLAY-1.
       10 STATE                PIC G(2) USAGE DISPLAY-1.
       10 ZIP                  PIC G(5) USAGE DISPLAY-1.
       10 PASSWORD             PIC X(8).

*******************************************************
* THE NUMBER OF COLUMNS DESCRIBED  IS 6               *
*******************************************************
```

Right Way

```
*******************************************************
* DCLGEN TABLE(ADMF001.T1)                            *
* LIBRARY(USER.DBRMLIB.DATA(T1))                      *
* LANGUAGE(COBOL)                                     *
* QUOTE                                               *
* DBCSSYMBOL(N)                                       *
* DCLBIT(YES)                                         *
* ...                                                 *
* IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STMT   *
*******************************************************
EXEC SQL DECLARE ADMF001.T1 TABLE
 ( NAME                          VARGRAPHIC(15),
   ADDRESS                       VARGRAPHIC(25),
    ...
   PASSWORD                      CHAR(8)
 ) END-EXEC.
*******************************************************
* DECLARED VARIABLES FOR 'FOR BIT DATA' COLUMNS       *
*******************************************************
 EXEC SQL DECLARE
   :PASSWORD  VARIABLE FOR BIT DATA END-EXEC.
*******************************************************
* COBOL DECLARATION FOR TABLE ADMF001.T1              *
*******************************************************
   01 DCLT1.
        10 NAME.
             49 NAME-LEN         PIC S9(4) USAGE COMP.
             49 NAME-TEXT        PIC N(15).
        10 ADDRESS.
              49 ADDRESS-LEN     PIC S9(4) USAGE COMP.
              49 ADDRESS-TEXT              PIC N(25).
         10 CITY.
               49 CITY-LEN       PIC S9(4) USAGE COMP.
               49 CITY-TEXT      PIC N(20).
          10 STATE  PIC N(2).
          10 ZIP                 PIC N(5).
          10 PASSWORD            PIC X(8).
*******************************************************
* THE NUMBER OF COLUMNS DESCRIBED IS 6                *
*******************************************************
```


BROADCOM SOFTWARE

Additional Considerations

Single or Multiple Encoding Applications

Single or Multiple Encodings?

- Most existing applications
- One Encoding from end to end
- One Encoding in DB, Second Encoding in App

Multiple encodings greatly increases complexity

Messaging technology typically is mono-encoding and string based – applications that are multi-encoding will have to convert to a single encoding on send/receive message

Tactical -> Strategic approach

Unicode Columns in EBCDIC Tables

Db2 12 added support for one or more Unicode (UTF-8 or UTF-16) columns in an EBCDIC table.

Many Restrictions

Every customer I ever talked to about Unicode had "one" column they needed to be Unicode, everything else was fine as EBCDIC

- See previous page with considerations

Transliteration

Transliteration is the conversion of letters from one script to another without translating the underlying words.

In many cases an additional column is added to a table containing names to provide a phonetic representation of the name using a Latin-1 alphabet:

- Андрей -> Andrei – is a Translation from Russian to English
- Андрей -> Andrey – is a phonetic Transliteration

Cultural Conventions

Numeric

1,234 or 1.234?

Date/Time

- 07/27/1975 – obvious
- 03/11/2003 – March 11th, or November 3rd?

Calendar

Gregorian, Islamic(lunar), Chinese(lunisolar)

Workweek(M-F,S-Th,???)

Time Issues

Timezones
- UTC
  - Some sites run LPAR at UTC, Some Local
  - LRSN is UTC based
- Timestamp is, by default local
  - May be able to use “CURRENT_TIMESTAMP + CURRENT TIME ZONE” to get UTC
- Timestamp with Timezone
  - Newer Data Type – eg. ‘2023-08-01-15.43.28.332409198242-04:00’
  - Application can set “SESSION TIMEZONE”
  - Application can use ”TIMESTAMP_TZ” function to create a Timestamp with Timezone.
  - Timezone information is normalized to UTC for comparison

BROADCOM SOFTWARE

Chris Crone
cjc@broadcom.com