



# Unicode experiences with DB2



A R E M E M B E R S O F T H E K B C G R O U P

Jan Tielemans  
KBC ICT Services

# The KBC Group is expanding .... In Central, Eastern Europe and Russia



CENTRAL AND EASTERN EUROPE  
& RUSSIA

# Application Infrastructure

- Network Centric : Head office applications, KBC-Online...
- 3-Tier : Client – Midtier – Backend
  - Client : “browser based”, limited, no business logic
  - Midtier : webserver, applications presentation logic... (Unix, Intel)
  - Backend : application business logic, **data** (Mainframe)

- Applications are written in :



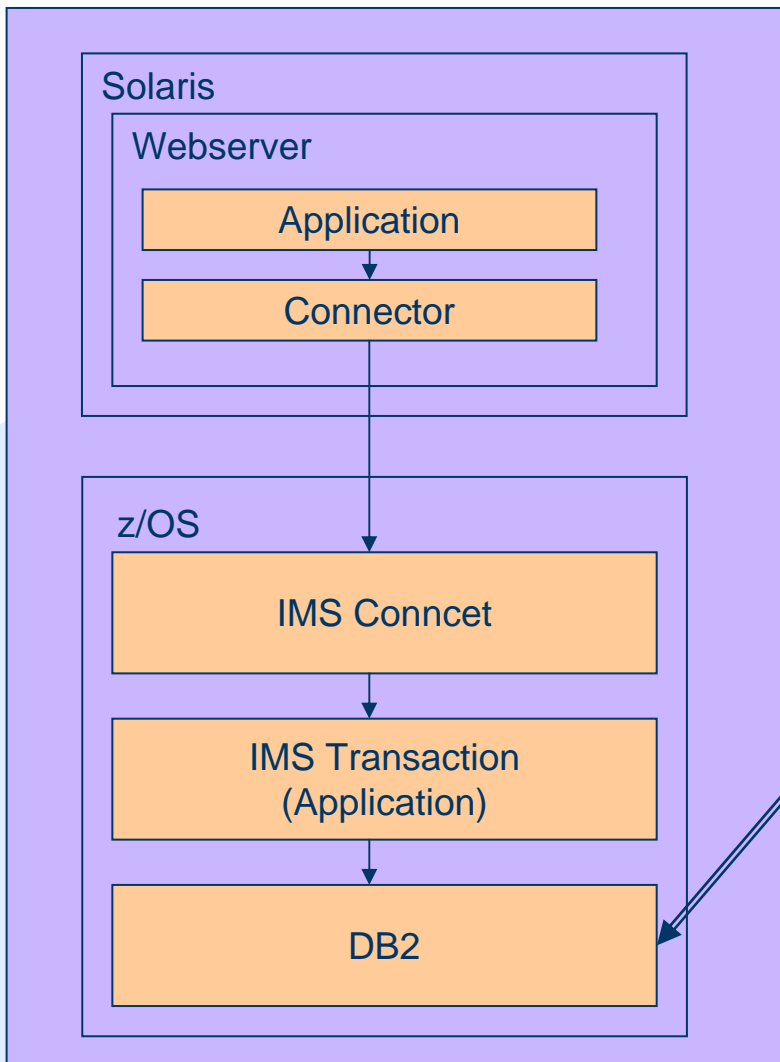
- Data in EBCDIC (codepage 500) :

- DB2 (since V1.2)
- IMS DB

## The Challenge

- Building new applications – Data models in Unicode
- Converting existing applications – Data models to Unicode
- Minimal impact on ‘Mainframe’ IT cost
  - Backend legacy systems are touched/changed
  - DB2 and the application
    - CPU usage
    - Storage
    - Memory
  - Development cost
  - .....

# What to choose ?

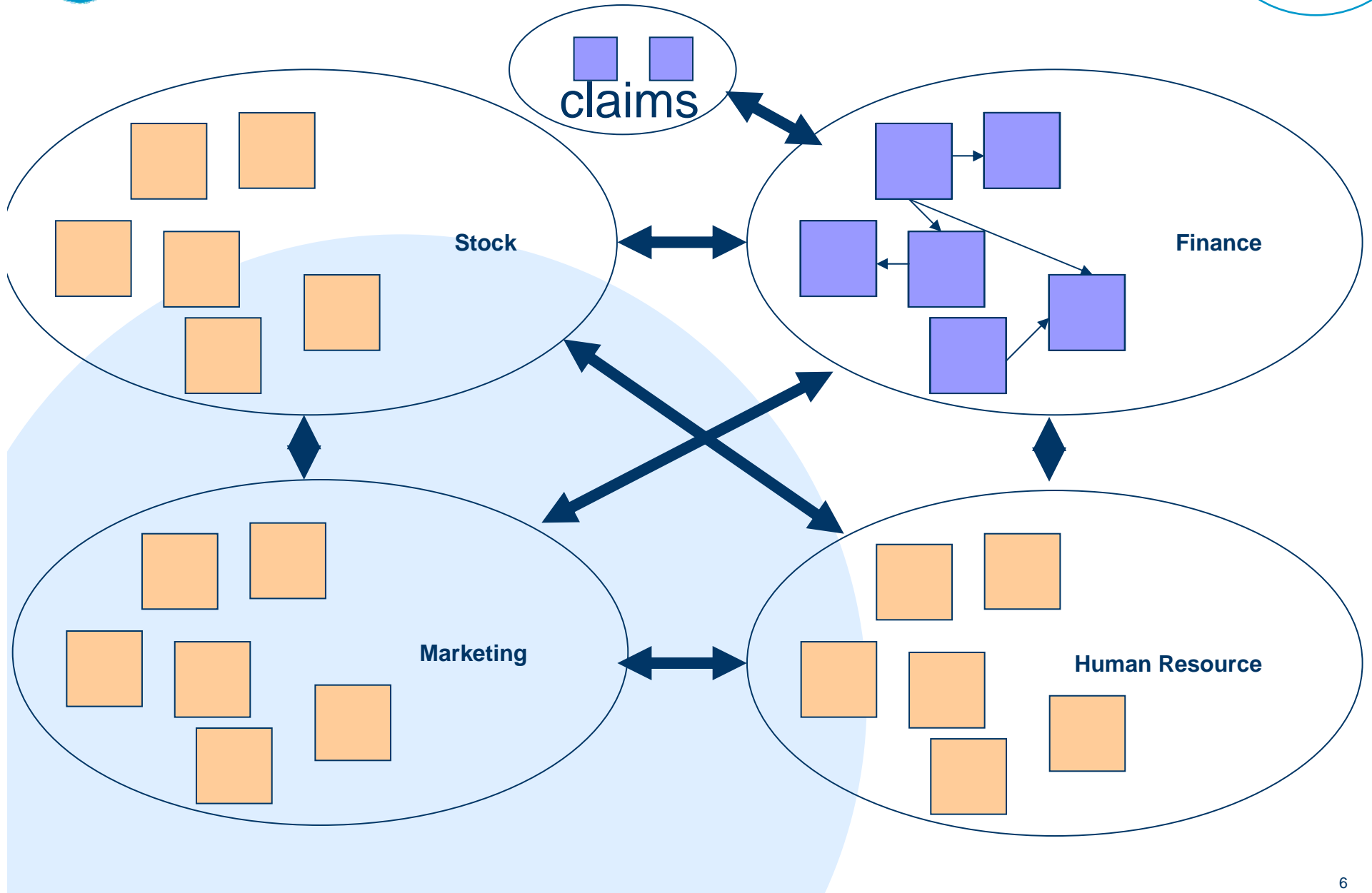


- UTF-8 or UTF-16 or a combination
- Column or Table based Unicode
- Function Based or Table based Unicode

Minimal impact on :

- Performance
- Development cost
- Resources usage

# Function based or Table based Unicode





# ● Column or Table based Unicode

- Pro Table based
  - Ease of programming, management, understanding,...
- Cons Table based
  - Overhead (DASD, CPU,...)
  - Only 5%-10% of the character based columns can functionally contain UNICODE data .....how much of these will actually contain 'special characters'

Table Emp  
Name Char(30)  
...  
Zipcode Char(6)  
  
CCSID UNICODE

# Unicode UTF-8 or UTF-16

## Pro UTF-8

- Optimal for space
- Inline with DB2 V8

## Con UTF-8

- More programming effort
  - Each character can be 1 or 2 bytes
    - Substr vs Substring

## Pro UTF-16

- Ease of programming
  - each character is 2 bytes
- Inline with programming language COBOL, Pic N()

## Con UTF-16

- Wasted space, more DASD space ?
- Bigger Bufferpool ?
- More CPU consumption ?
- More IO's – GetPages ?

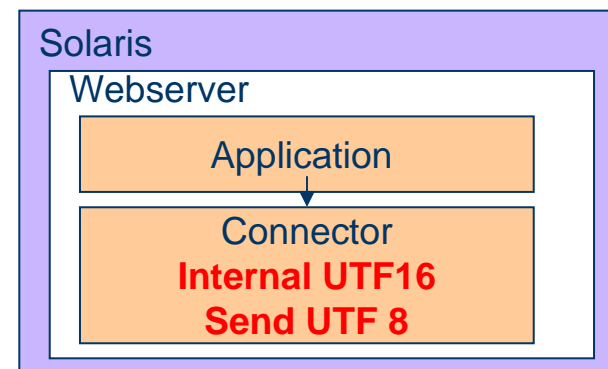
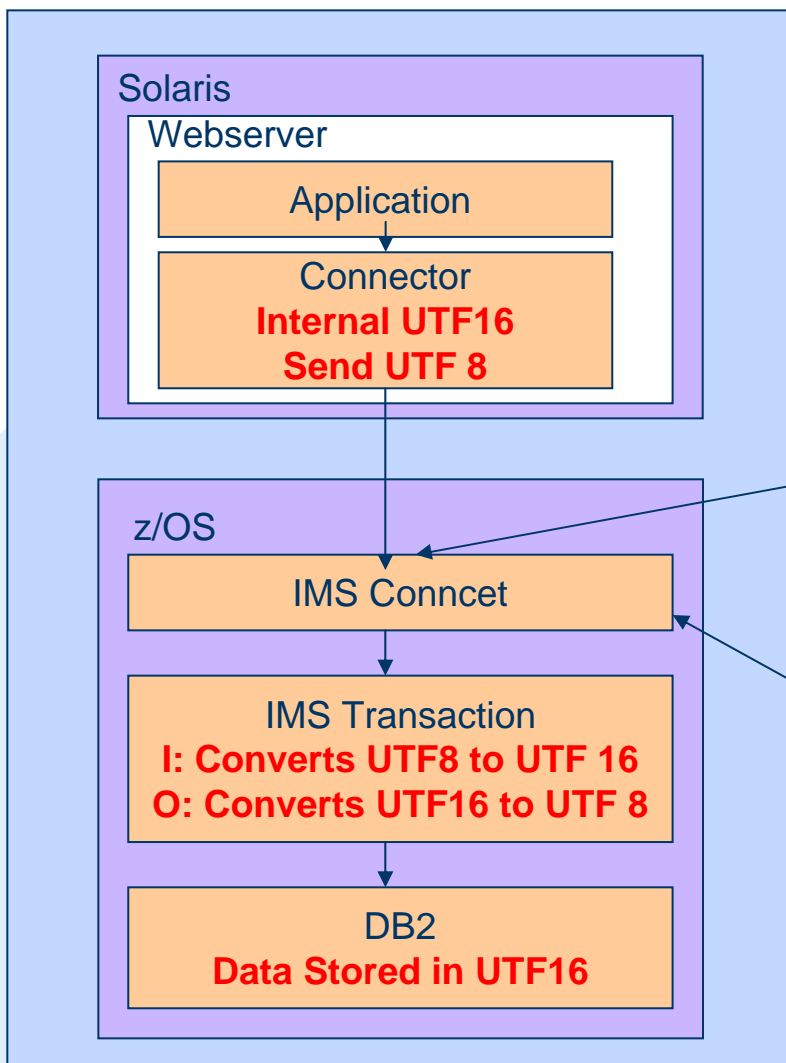


## ● KBC Unicode Policy

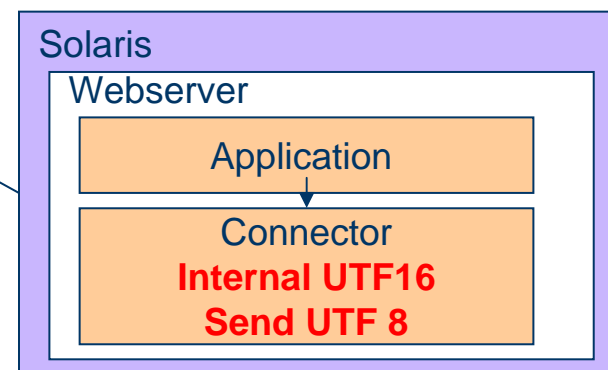
- UNICODE UTF-16 will be used
- All new applications will be developed in UNICODE UTF-16
- All character columns in a table will be in UNICODE UTF-16
- All tables in a 'function' will be converted to UNICODE
- No (or minimal) Joining between EBCDIC and UNICODE tables

# 'Enhanced 3-Tier Architecture Model'

Poland



**Network capacity :**  
 UTF-16 <-> UTF-8 : +40% gain + compressions





## ● DB2 Support for Unicode

- Migrated to DB2 V8 for 'better Unicode Support':
  - Catalog and Directory converted to UTF-8 (enfm mode)
    - **Still in EBCDIC** : SYSCOPY, SYSEBCDC, SCT02, DBD01, SYSLGRNX, SYSUTILX
  - Precompiling/DBRM converted to unicode UTF-8 (nfm mode)
    - Controlled NEWFUNC parameter in DSNHDECP
  - All SQL parsing is done in UTF-8
  - Allows mixed codepages in a SQL Join statement
  - An enhanced set of Unicode SQL Functions
    - Eg : substring

# Can be handy....

```

BROWSE      DB2TS.LIB.DSN810.SDSNDBRM(DSN@CCC4)           Line 00000000 Col 001 080
Command ==> Display ccsid 1208 or display utf8           Scroll ==> CSR
***** Top of Data *****
DBRM...µW98COMP DSNACCC4.ÄÄd.Âé\..4.....1..ØLL
..
DBRM...m.....j.....ÿàää< êá.!íèíèè..áíèè!ê.ã!ê.ëá<áääè.ãñèèñ+äè.èèèñ&...ââ+ (
á...è.....ãê!(.ëßëñâ(...ëßèè â<á& êè.ïçáéá.èè!êèß&á....á.. +à.îä è+ (á.....
..... +à.ââ+ (á.....äë+ââ....í+ñ!+.ëá<áääè.ãñèèñ+äè.èèèñ&...â...ââ+ (á...è...
.....ãê!(.ëßëñâ(...ëßëñ+ääì& êè. ...ëßëñâ(...ëßëñ+ääìáè.â.ïçáéá.èè!êèß&á....á..
+à. ...ñî+ (á...â...+ (á. +à. ...îä è+ (á.....ã!ê.ãääèäç.!+<ß.ïñèç.íè.

```

```

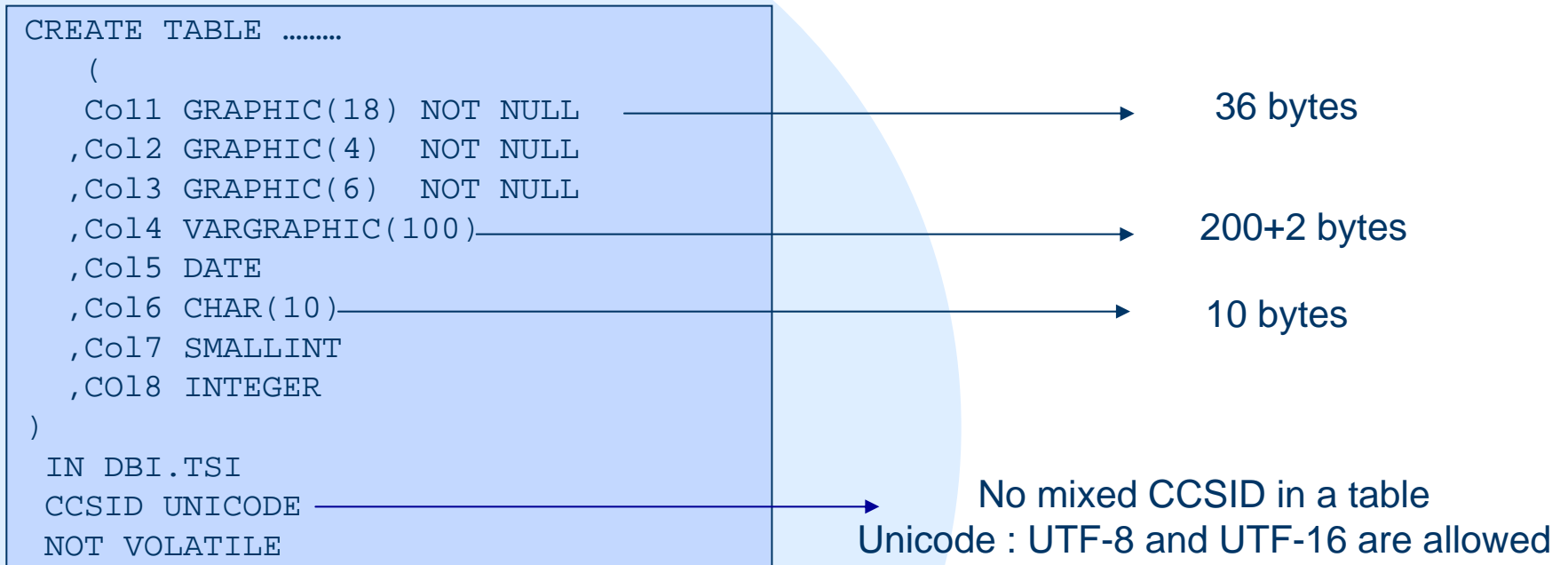
BROWSE      DB2TS.LIB.DSN810.SDSNDBRM(DSN@CCC4)           Converted data shown
Command ==>                                           Scroll ==> CSR
***** Top of Data *****
.....Ö.@..Ãô.cc..bQ.....
..@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
.....xDECLARE OUTUSR1 CURSOR FOR SELECT DISTINCT STRIP ( DBNAM
E , T , ' ' ) FROM SYSIBM . SYSTABLEPART WHERE STORTYPE = 'E' AND VCATNAME <> '0
0000001' AND DBNAME <> 'DSNDB07' UNION SELECT DISTINCT STRIP ( B . DBNAME , T ,
' ' ) FROM SYSIBM . SYSINDEXPART A , SYSIBM . SYSINDEXES B WHERE STORTYPE = 'E'
AND A . IXNAME = B . NAME AND A . VCATNAME <> '00000001' FOR FETCH ONLY WITH UR
.....@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```



# DB2 Support for Unicode

- Supports UTF-8 and UTF-16 equally
- Defined on the Table(space)
  - Char/VarChar/CLOB for UTF-8
  - Graphic/VarGraphic/DBCLOB for UTF-16



## ● Myths or Reality

### ● UTF-16 :

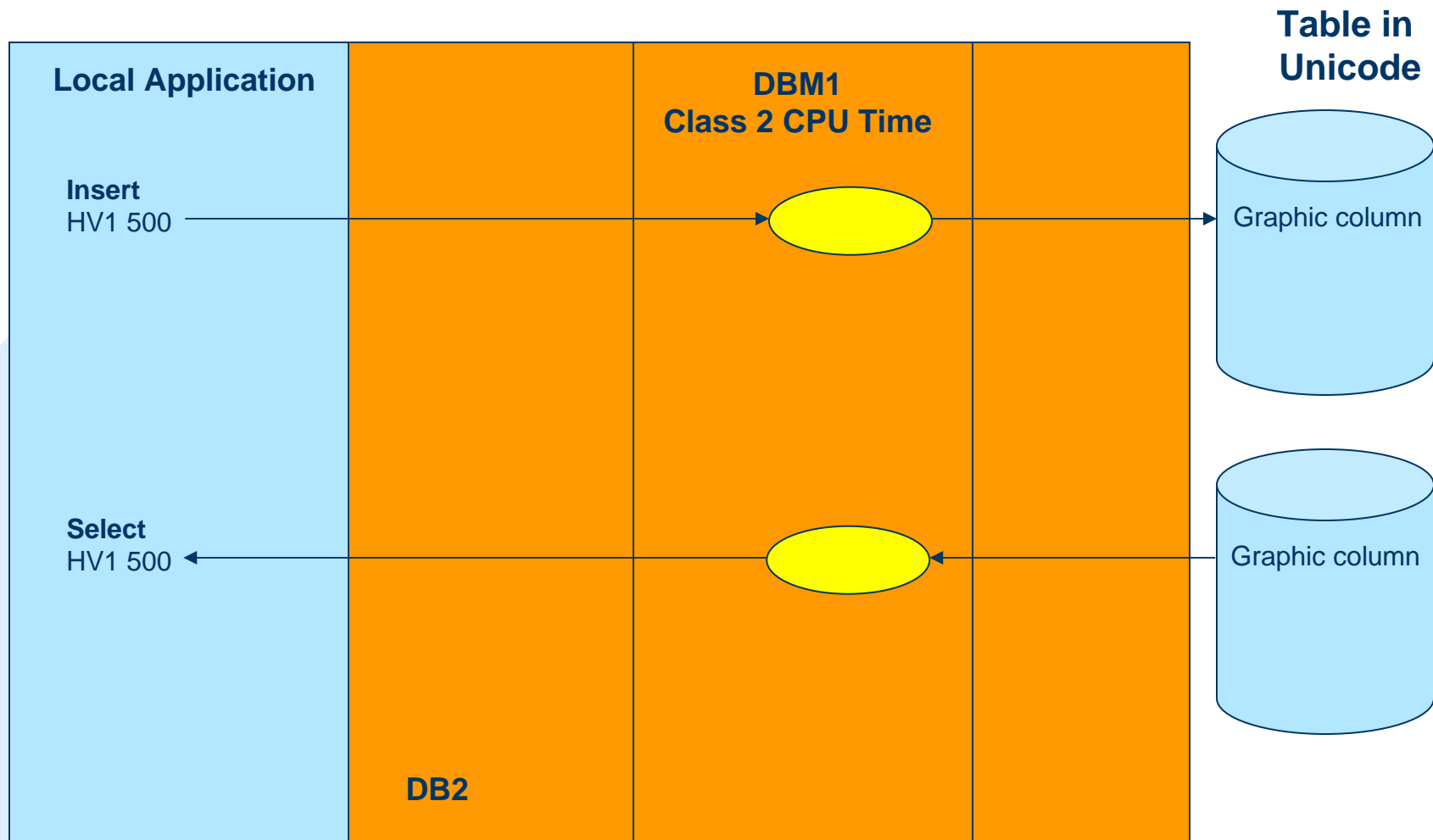
#### ● CPU increase

- Conversion costs (EBCDIC vs UNICODE)
- Bytes moved from DB2 to the application times 2
- It takes twice the time/cost to do the compare/move
- Twice as much bytes to sort
- Data space is doubled, need bigger bufferpools
- Sortspace(DSNDB07) times 2, bigger sort bufferpool

```
Select name, adress, departnm From UEmp a,  
UDept b  
Where a.departcode = b.departcode  
Order by adress, departnm
```

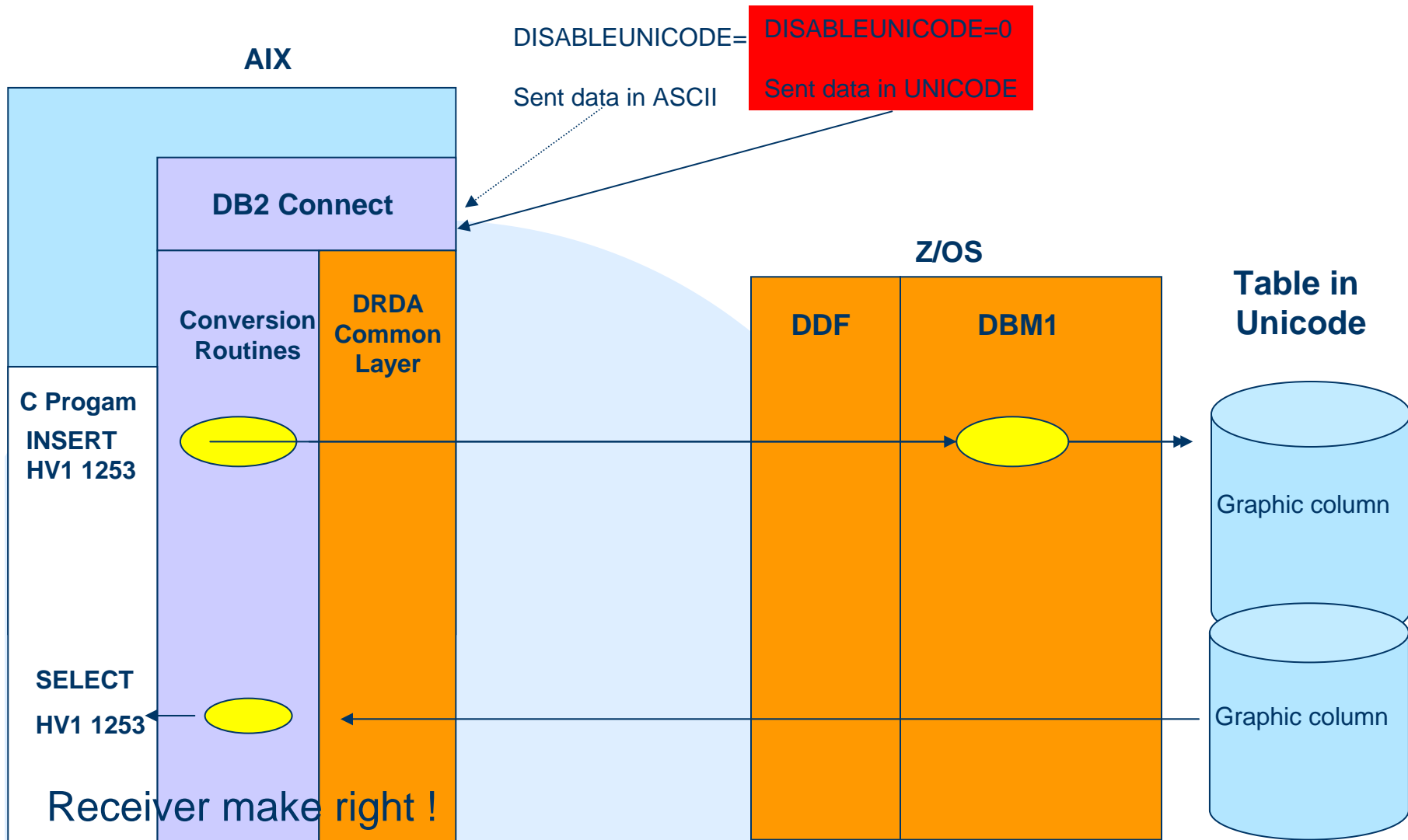


# ● Conversion Cost : Mainframe Program





# Conversion Cost : DRDA Protocol





# ● z/OS Conversion Services

- What is this :
  - Central repository (conversion tables) for z/OS system used by :
    - ODBC Driver
    - COBOL
    - DB2
- High performance conversion method :
  - **Uses HW** instructions available in zSeries 800, 900, 990, z9, z10
    - CU12 (alias for CUTFU): Convert from UTF-8 to UTF16
    - CU21 (alias for CUUTF): Convert from UTF16 to UTF-8
    - TROO: Translate from one byte to another one byte (used to convert single-byte codes)
    - TRTO: Translate from two bytes to one byte (used to convert from UTF16 to EBCDIC)
    - TROT: Translate from one byte to two bytes (used to convert from EBCDIC to UTF-16)
    - TRTT: Translate two bytes to two bytes (used to convert between UTF16 and double-byte EBCDIC).
  - See also article in z/Journal : **April/May 2006 DB2 for z/OS Version 8: Improved Unicode Conversion** By [Laxminarayan Sriram](#)



# ● Character Conversion support in DB2 V8

- DB2 catalog table SYSIBM.SYSSTRINGS
  - For EBCDIC <-> ASCII conversions
- For Unicode conversions DB2 uses **z/OS conversion services**
  - To ensure that conversions between EBCDIC and **UTF-8** are as fast as possible, in some cases DB2 V8 performs so-called "**inline conversion**" instead of calling the z/OS Conversion Services. As a general rule, inline conversion can be done when a string consists of single-byte characters in **UTF-8**. This conversion enhancement is not available in V7, nor is it available for conversions between EBCDIC and UTF-16 and vice versa.
  - In addition, **z/OS** V1.4 has improved the performance of EBCDIC to UTF-8 (and vice versa) conversions by streamlining the conversion process, and V1.4 dramatically outperforms z/OS 1.3 conversions.
  - On top of that, zSeries machines have **hardware instructions** that assist CCSID conversion. These instructions were first implemented on the z900, and have been enhanced on the z990 machines.

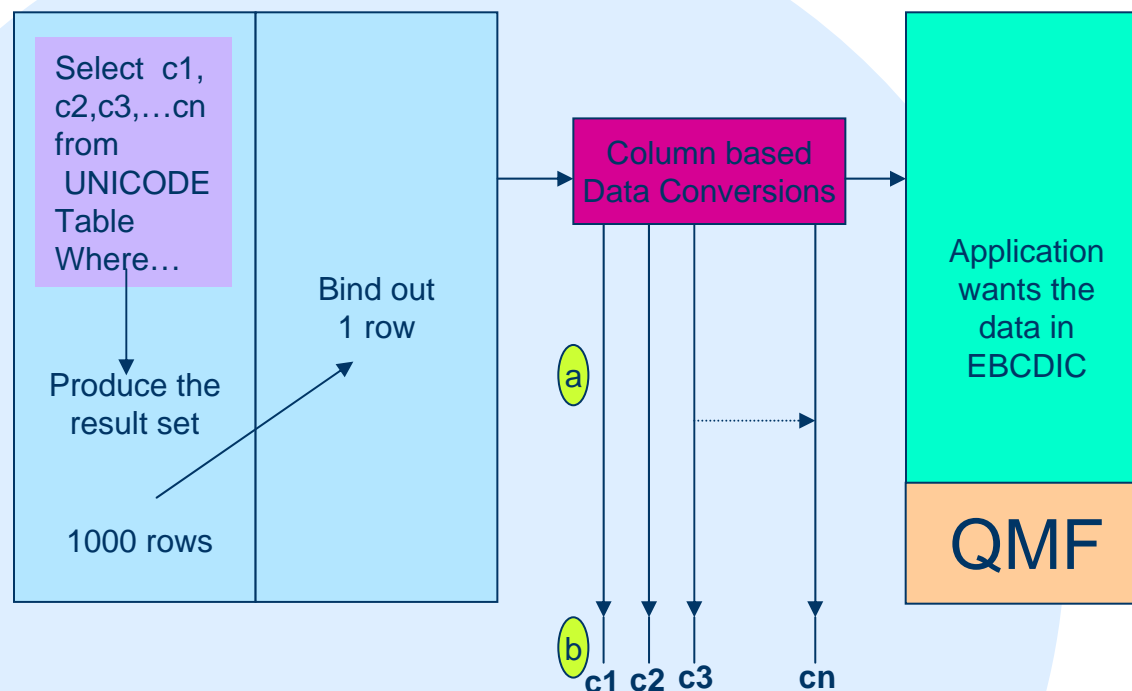
## ● z/OS support for Unicode

- Implication of the HW instruction set is ....
  - Conversion from EBCDIC to UTF-8
    - Step 1: EBCDIC to UTF-16 (via TROT)
    - Step 2: UTF-16 to UTF-8 (via CU21)
  - All access to the DB2 catalog !
    - Spufi : Select \* From SYSIBM.SYSTABLES
  - DB2 V8 performs so-called "inline conversion"

# ● Conversions requested by DB2

## ○ Bind out Process

- Moves data out of the DB2 addressspace into the application addressspace
- Row by Row – Column by Column

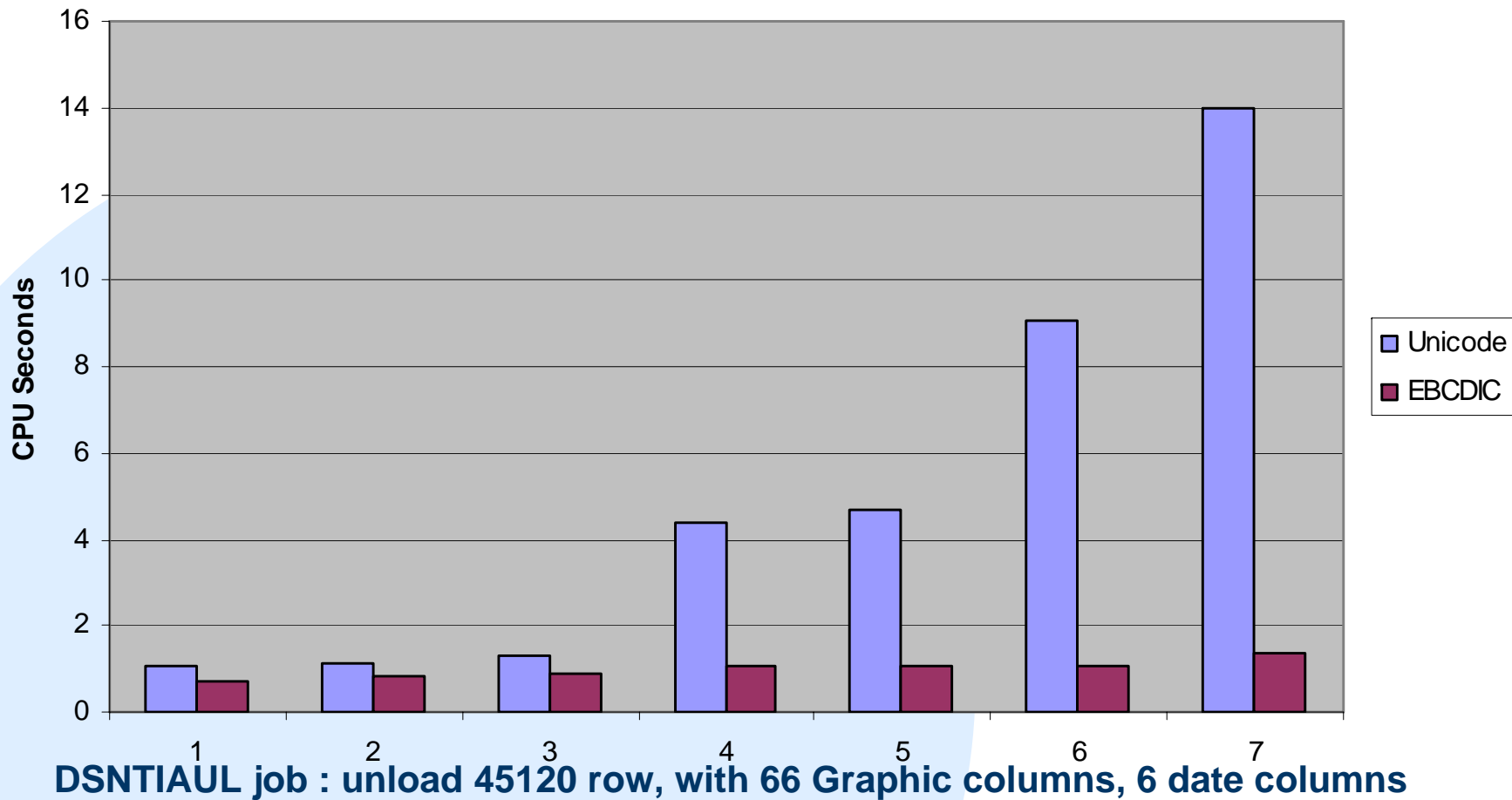


- Setup Cost dominates the conversion cost (less size dependent)
- For **each** row sent to the application, column conversions are done !
- 2 Columns – 1000 rows
  - 2000 conversions !
- (a) Setup Cost (fixed cost)
- (b) Conversion Cost Like many other instructions, the CPU time of these instructions increases in proportion to the string length



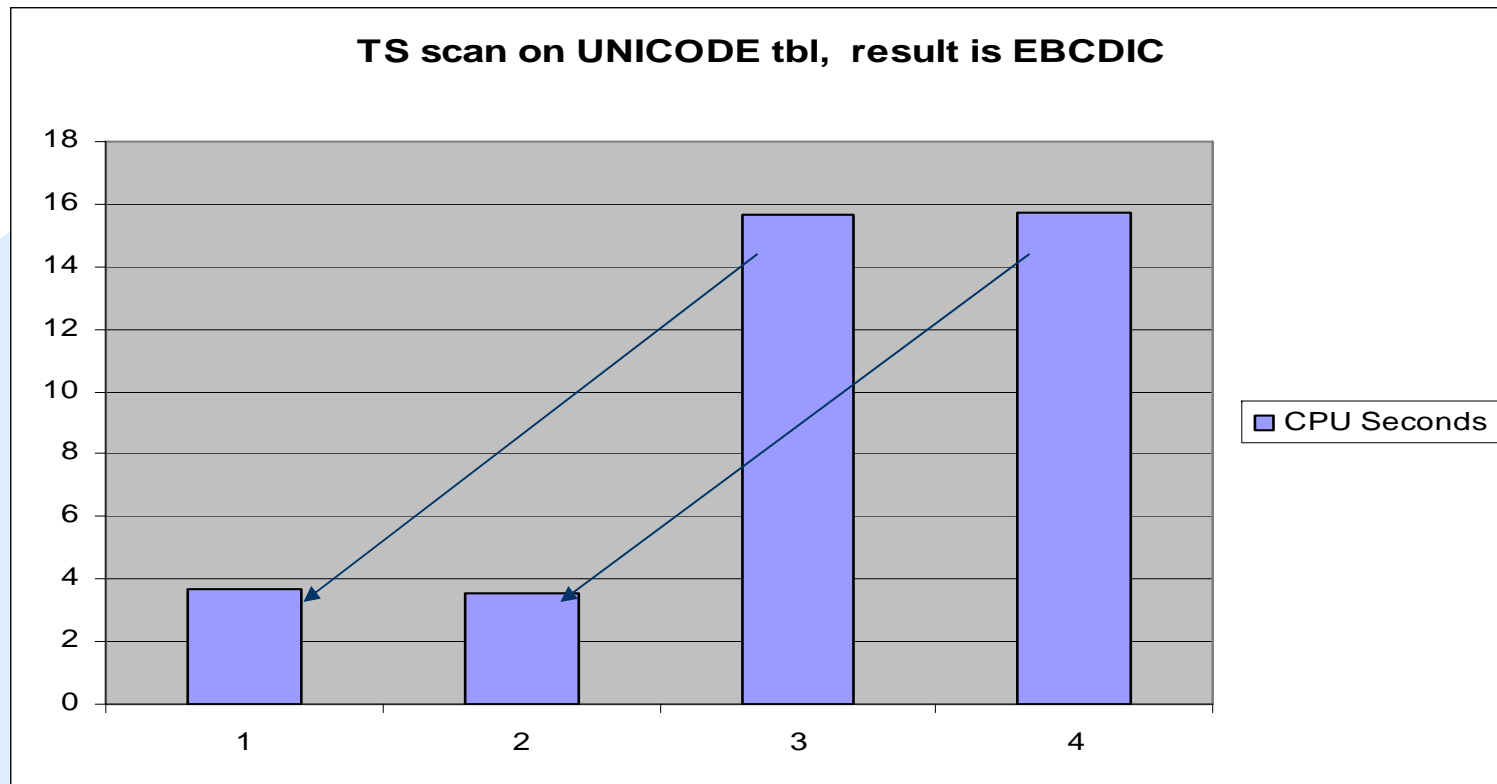
# z/OS 1.7 Unicode Conversion Services

## TS Scan in ..... Result in EBCDIC



# z/OS 1.7 Unicode Conversion Services

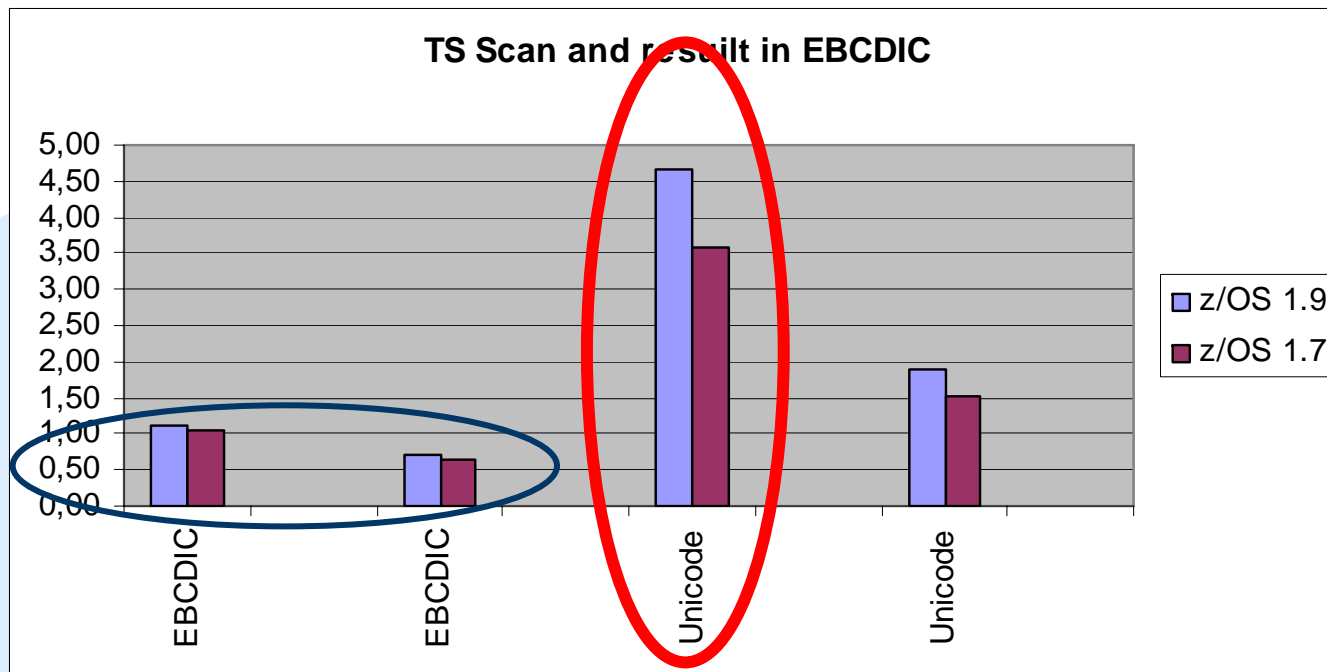
- APAR OA21903 : Unicode services (z/OS 1.7)



- Its is still +2 times more CPU vs the EBCDIC version!**

# z/OS 1.9 Unicode Conversion Services

- Then came z/OS 1.9 .....



DSNTIAUL job : unload 45120 row, with 66 Graphic columns, 6 date columns

- CPU Increase in conversion Services !!
  - AparFix CA23705 (APAR OA23705)



## ● z/OS 1.9 Unicode Conversion Services

- After OA23705
  - Better but still 6% increase
  - Found a 'problem'
    - with varying string sizes and seeing a difference in the amount of time it took to perform that conversion between z/OS 1.7 and 1.9
    - USerFix available, testing next week
  - Z/OS conversion services is now part of the z/OS performance benchmark



# z/OS Unicode Conversion Services

Some OEM Report .....

```
                ** PROGRAM SECTION USAGE SUMMARY **
```

MODULE NAME	SECTION NAME	16M <,>	SECT SIZE	FUNCTION	% CPU TIME	
					SOLO	TOTAL
SYSTEM	.DB2			DB2 SYSTEM SERVICES	62.78	64.44
SYSTEM	.SVC			SUPERVISOR CONTROL	2.22	2.22
SYSTEM	.XMEMORY			CROSS MEMORY	1.11	1.11
					-----	-----
SYSTEM	TOTALS			SYSTEM SERVICES	66.11	67.77
<b>CUNMUNI</b>		<b>&gt;</b>	<b>230392</b>		<b>31.67</b>	<b>32.22</b>
					-----	-----
PROGRAM	IKJEFT01	TOTALS			97.78	100.00

# Conversions Requested by DB2

```
EXEC SQL DECLARE PX01 CURSOR
FOR SELECT KLN_NR -- graphic(7)
FROM UNICODE_TABLE_UTF16
WHERE KLN_NR > :KLNR-START -- Pic X(7)
END-EXEC
```

```
EXEC SQL FETCH PX01
INTO
:KLNR-FETCH -- Pic x(7)
END-EXEC
```

SQLCODE -330 rc16 and runtime  
Must be defined as Pic N(7)

If not correctly defined, CONVERSIONS !

```
EXEC SQL DECLARE PX01 CURSOR
FOR SELECT
U.KLN_NR -- graphic(7)
FROM UNICODE_TABLE_UTF16 U, EBCDIC_TABLE E
WHERE U.KLN_NR = E.KLN_NR
AND U.KLN_NR > :KLNR-START -- Pic n(7)
```

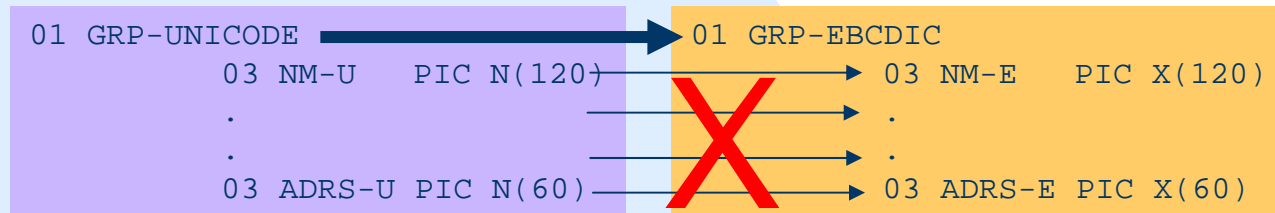
```
EXEC SQL FETCH PX01
INTO
:KLNR-FETCH -- Pic N(7)
END-EXEC
```

- For **each** row of the outer table, conversion of the column needs to be done
- The more columns are joined the higher the conversion cost will be
- Cost for unnecessary join criteria !

CONVERSIONS !

## ● Conversions Requested by DB2

- Test the **performance** of Unicode Conversion services after z/OS maintenance or Upgrade !
- Conversions are CPU expensive in DB2, when possible do it in your application



## ● Date/Time fields and Unicode

- In a UNICODE table, DB2 represents Date/Time columns as **UTF-8**

- DCLGEN generates PIC X(10) for these fields

- MR open to generate N(10)

- By choosing UTF-16 in the application, we will have a conversions cost

**Select a.Uni\_date\_end**

**From Unicode\_table**

**Where a.Uni\_date\_start = :hv**

In application defined as Pic N(10)

- V8 Bind Out

- From internal format to UTF-8 date/time/timestamp value was 1.05X

- From internal format to UTF-16 date/time/timestamp value was 2X

- V9 Bind Out

- From internal format to UTF-16 date/time/timestamp reduced to 1.10x

- DB2 is doing the CU12, rather than using Conversion Services



## ● Date/Time fields and Unicode

- Unload – Load utility only supports date/time and timestamps in UTF-8
- “Upcoming” V9 implementations **(MR0727073120-accepted)**:
  - Unload utility :
    - Unload date/time and timestamps to a UTF-16
  - Load Utility :
    - Load date/time and timestamps in UTF-16

# ● Date/Time fields and Unicode

- Any Local Date(dd/mm/yyyy) – Local Time installations ?
  - DSNXVDTA ASCII exit (must exist)
  - DSNXVDTU Unicode exit (must exist)
  - DSNXVDTX EBCDIC exit

* * Check day value <b>EBCDIC</b> *		* * Check day value <b>UNICODE</b> *	
LA	3,LDAG	LA	3,LDAG
<del>LA</del>	<del>9,L'LDAG</del>	<del>LA</del>	<del>9,L'LDAG</del>
DAGFM	CLI 0(3),C'0'	DAGFM	CLI 0(3),X'30' Is 0 for Unicode
BE	FORMATR	BE	FORMATR
CLI	0(3),C'9'	CLI	0(3),X'39' Is 9 for Unicode
BH	FORMATR	BH	FORMATR
LA	3,1(3)	LA	3,1(3)
<del>DCT</del>	<del>9,DAGFM</del>	<del>DCT</del>	<del>9,DAGFM</del>
SLASH1	CLI LSLASH1,C'/'	SLASH1	CLI LSLASH1,X'2F' Is a / in Unicode
BNE	FORMATR	BNE	FORMATR

## Standards

- COBOL (and PL/I) is UTF-16 oriented, PIC N()
- DB2 is (re) designed for UTF-8
  - SQL Parsing is done in UTF-8
  - Catalog/Directory in UTF-8
  - DBRM's in UTF-8
  - Date and Time columns are defined in UTF-8
  - Columns can be defined in UTF-8 or UTF-16
- z/OS & HW conversion instruction set is both UTF-8 and UTF-16 oriented



# UTF-16 and DASD Space

	EB_TBL	UNI_TBL	
CARD .....	3985333	3985333	
NPAGES .....	36645	43533	18,8%
PCTPAGES .....	94	94	
KEYCOLUMNS .....	4	4	
RECLENGTH .....	51	90	
SPACEF .....	155520	184320	18,5%
AVGROWLEN .....	26	31	19,2%

Note: Tablespace is COMPRESSED  
One table  
All columns graphic or Date

	EB_IDX1	UN_IDX1	
FIRSTKEYCARD ....	440094	440094	
FULLKEYCARD ....	3985333	3985333	
NLEAF .....	39556	70100	77,2%
NLEVELS .....	3	3	
CLUSTERRATIO ....	100	100	
SPACEF .....	173520	291600	68,0%
AVGKEYLEN .....	22	44	100,0%

	EB_IDX2	UN_IDX2	
FIRSTKEYCARD ....	2540258	2540258	
FULLKEYCARD ....	3985333	3985333	
NLEAF .....	44780	79707	78,0%
NLEVELS .....	4	4	
CLUSTERRATIO ....	68	68	
SPACEF .....	173520	360000	107,5%
AVGKEYLEN .....	22	44	100,0%

	EB_IDX3	UN_IDX3	
FIRSTKEYCARD ....	1016	1016	
FULLKEYCARD ....	595289	595289	
NLEAF .....	11215	13808	23,1%
NLEVELS .....	3	3	
CLUSTERRATIO ....	99	99	
SPACEF .....	50400	57600	14,3%
AVGKEYLEN .....	11	22	100,0%

Index levels 'might' increase !

# UTF-16 and DASD Space

- Less impact on Random I/O applications
- Bigger impact for Sequential applications (batch jobs)
  - To consider the use of 8K, 16K, 32K bufferpools
    - Data ONLY, indexes always in 4K bufferpools
    - More rows per page, but MAX is still 255 rows per page

	4k	8k	
CARD .....	3985333	3985333	
NPAGES .....	43533	22186	-49,0%
PCTPAGES .....	94	94	
KEYCOLUMNS .....	4	4	
RECLENGTH .....	90	90	
SPACEF .....	184320	188640	2,3%
AVGROWLEN .....	31	31	0,0%

- Use NOT PADDED if you have indexes on variable length columns !
- Does V9 with index compression help ?
  - Index can be defined in a larger BP size with compression
  - Only in the storage area
  - Index pages are expanded in the bufferpool

# UNICODE UTF-8 & Unload Utility

```
select hex(unicol)
from unicode_table_UTF8
WHERE unicol = '0'
```

```
-----+-----+-----
30
30
30
30
30
```

```
select hex(unicol)
from unicode_table_UTF8
WHERE unicol = X'30'
```

```
-----+-----+-----
30
30
30
30
30
```

```
UNLOAD DATA
FROM TABLE UNICODE_TABLE_UTF8
HEADER NONE
WHEN (unicol1 = '0')
SHRLEVEL CHANGE ISOLATION UR
```

HIGHEST RETURN CODE=0

```
UNLOAD DATA
FROM TABLE UNICODE_TABLE_UTF8
HEADER NONE
WHEN (unicol1 = X'30')
SHRLEVEL CHANGE ISOLATION UR
```

HIGHEST RETURN CODE=0

Column unicol defined as char(1) -> UTF8

# UNICODE UTF-16 & UNLOAD Utility

UX :Hexadecimal Unicode string UTF16 only

```
select hex(unicol)
from unicode_table_UTF16
WHERE unicol = '0'
```

-----+-----+-----

0030  
0030  
0030  
0030  
0030

```
select hex(unicol)
from unicode_table_UTF16
WHERE unicol = X'0030'
```

-----+-----+-----

DSNE610I NUMBER OF ROWS  
DISPLAYED IS 0

```
select hex(unicol)
from unicode_table_UTF16
WHERE unicol = UX'0030'
```

-----+-----+-----

0030  
0030  
0030  
0030  
0030

```
UNLOAD DATA
FROM TABLE UNICODE_TABLE_UTF16
HEADER NONE
WHEN (unicol = '0')
SHRLEVEL CHANGE ISOLATION UR
```

```
UNLOAD DATA
FROM TABLE UNICODE_TABLE_UTF16
HEADER NONE
WHEN (unicol = X'0030')
SHRLEVEL CHANGE ISOLATION UR
```

```
UNLOAD DATA
FROM TABLE UNICODE_TABLE_UTF16
HEADER NONE
WHEN (unicol = UX'0030')
SHRLEVEL CHANGE ISOLATION UR
```

INVALID OPERAND '0' FOR KEYWORD 'unicol'

MR0228083923 opened, and accepted

UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0

Column unicol defined as graphic(1) -> UTF16

INVALID OPERAND ' ' FOR KEYWORD 'unicol'

# UNICODE UTF-8 & UNLOAD Utility

## Convert Unicode UTF-8 to EBCDIC

```
UNLOAD DATA  
FROM TABLE unicode_table_UTF8
```

```
UNLDDN UNLFILE EBCDIC CCSID(,500,)
```

Specifies that all output data of the character type is to be in .....

Specifies up to three coded character set identifiers (SBCS, MBCS, DBCS) that are to be used for the data of character type in the output records, including data that is unloaded in the external character formats

**UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0**

Be aware of possible EBCDIC substitution characters (**X'3F'**) in the output file



# UNICODE UTF-16 & UNLOAD Utility

## Convert Unicode UTF16 to EBCDIC

```
UNLOAD DATA  
FROM TABLE unicode_table_UTF16
```

```
UNLDDN UNLFILE EBCDIC CCSID(,,500)
```

Specifies that all output data of the character type is to be in .....

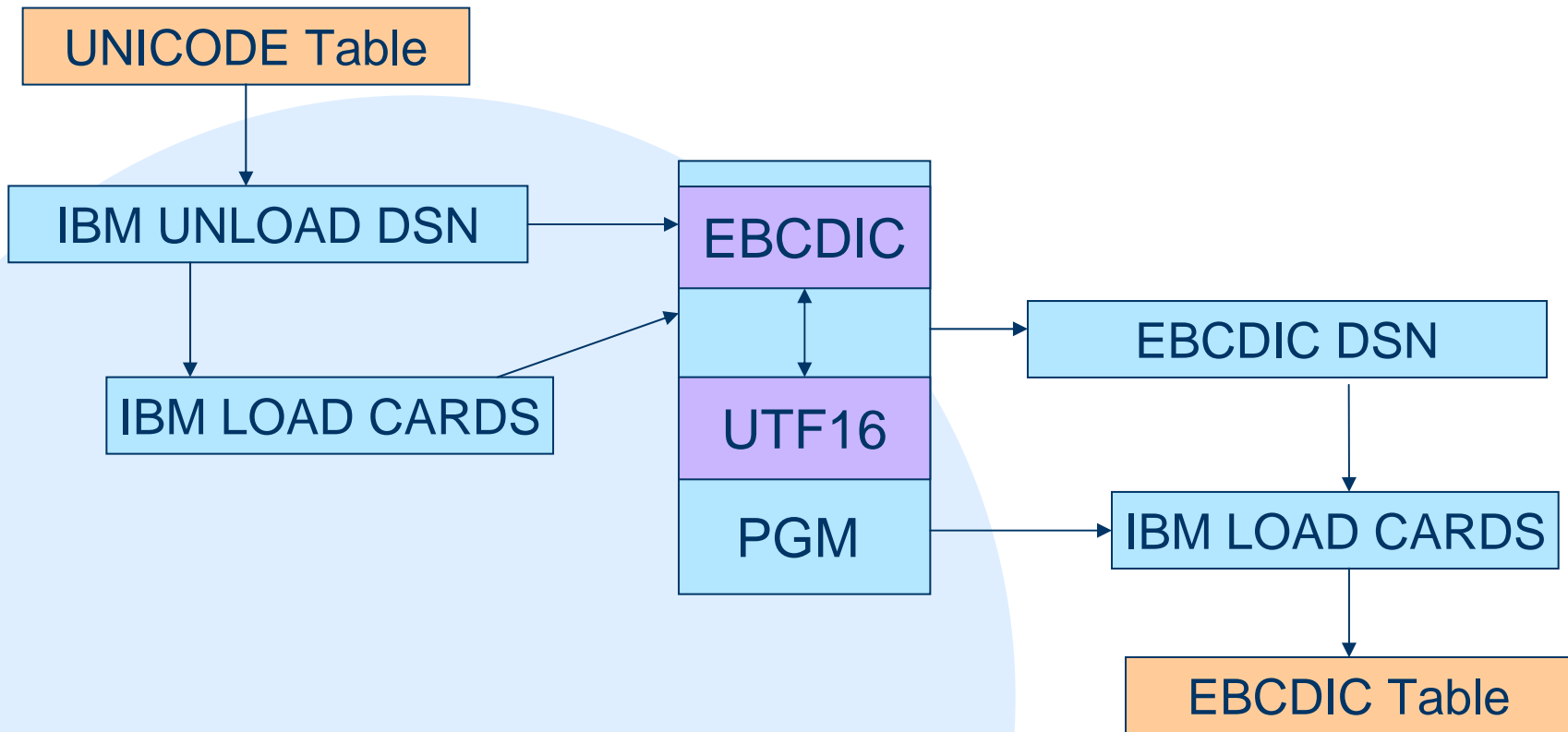
Specifies up to three coded character set identifiers (SBCS, MBCS, DBCS) that are to be used for the data of character type in the output records, including data that is unloaded in the external character formats

**ERROR IN CCSID TRANSLATION FOR "DBCS PAD CHAR", FROM CCSID 1200 TO 500**

IBM Response -> HPUUnload supports this (\$).....MR1218066147

# UNICODE & DB2 UTILITIES

- Our solution for the UNICODE (UTF-16) <-> EBCDIC data conversions :



- DSNTIAUL will cause to much conversion overhead....

## Nice to know

- REXX does not support UTF-16
- No RI possible between a Unicode table and an EBCDIC table

# To summarize our UTF-16 experiences

- Unicode overhead :
  - CPU increase :
    - Eliminate conversions as much as possible.....
    - Improve date/time/timestamp to UTF16 conversion in V9
    - V9 end this year, Q2 2009 production
  - Index Space increase :
- DB2 does not support UTF-8 and UTF-16 equally
  - From a SQL point of view it does
  - From a UNLOAD/LOAD view it does NOT
- More usage of BP8K and BP16K bufferpools
  - Need more memory
- Majority of Unicode problems, performance, .....
  - RAD/EGL, not generation/supporting the correct COBOL
  - Cobol compiler
  - LE runtime environment

## Take your time

- Implementing Unicode is not a 'normal' project
- Multiple skills are needed (Architect, Application, Development Tools, Application Design, Database Administration, z/OS Technology, DB2 Technology,...)
- It is worth a thorough study
  - A learning curve of a couple of years is not unusual
- The difficulty is when introducing Unicode in an existing application portfolio
- Take your time for the first implementation projects
  - Building applications will take more time (+50% or more?)
  - Spend attention to testing (+100%?)
  - Try as good as possible to limit the impact on other applications

## Maturity of technology

- Unicode support is still in evolution
  - Unicode is not new on mainframe (e.g. DB2 has introduced mixed data since V2.3)
  - Unicode in development tools doesn't exist so long
  - Still functional and performance enhancements
  - OEM vendors 'may' not support it fully
    - How can I browse/edit unicode UTF16 data through ISPF ?
    - What about output archiving tools ?
    - Can debugger tools and file browsing tools show the representable UTF-16 characters ?
- Mainframe perception is different compared to Open systems
  - Mainframe has a longer tradition of measuring resource usage on an application level
  - Mainframe hardware runs at a higher CPU busy %



# Thank You

KBC ICT Services

Jan.Tielemans@kbc.be