Character Sets and Unicode in Firebird

Stefan Heymann

www.destructor.de
stefan@destructor.de

This session will show you how non-ASCII text and text from different languages and scripts can be handled with Firebird. It will introduce you to the Firebird world of Character Sets, Collations, Unicode, case insensitive searching and Client character sets.

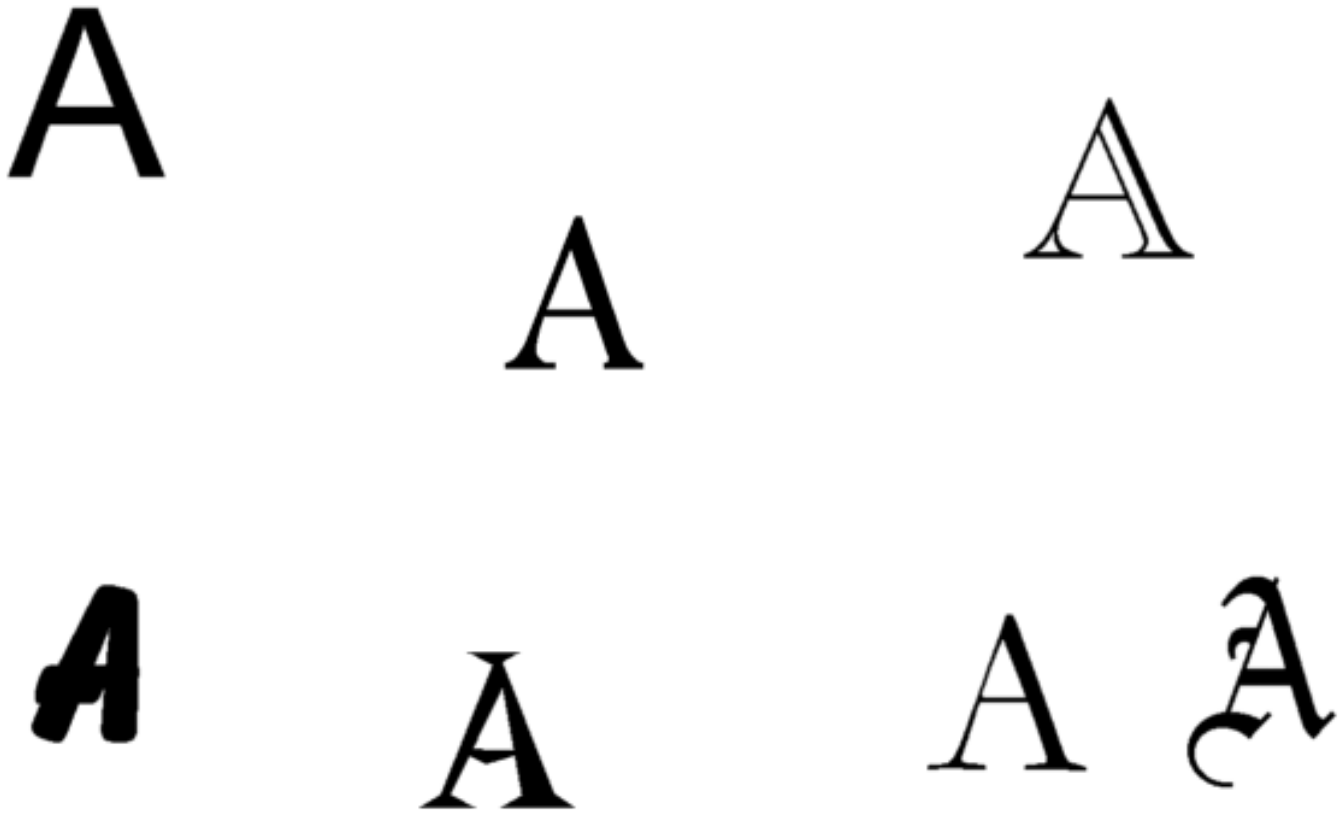
We will take a look at how international text influences the design of your database and the client programming. The session will also discuss the various new international features of Firebird 2.0.

Topics

- Characters
- Character Sets
- Unicode
- Firebird
- Examples

Characters

Glyphs vs. Characters



Latin uppercase A

Glyph, Character, Character Set

- A Glyph is something you can see with your eyes
- A Character is an abstract concept
- Rendering of characters as Glyphs is the job of the rendering machine (Postscript, GDI, TrueType, Web Browser, etc.)
- We mostly care for processing the characters
- A Character Set gives all characters a number:

  Uppercase A = 65
  Uppercase B = 66
  etc.

Glyphs

- Not all languages display glyphs as a string of left-to-right, contiguous rectangles
- Right-to-left (Arabic, Hebrew), top-to-bottom (Japanese, Chinese)
- Several characters can „melt" into one glyph

Keystrokes	ل	ا	لا		غ	غ
Input characters	ل	ا	ل	ا	غ	غ
Encoded characters (byte values in hex)	0644	0627	0644	0627	0639	0639
Display	لاغغ					

Character Sets

ASCII: The Mother of Character Sets

- American Standard Code for Information Interchange: ASCII, ISO 646
- 7 bits, characters ranging from 0 to 127 (00..7F)
- 32 invisible control characters (NUL, TAB, CR, LF, FF, BEL, ESC, ...)
- A..Z, a..z, Digits 0..9, Punctuation (;.-?)
- Optimized for English
- Only Latin characters, no accents, no umlauts
- MIME code: US-ASCII

ASCII for Europe

- Language specific characters get a new assignment
- [=Ä \=Ö]=Ü
- Problem: Printer and screen must have same setting
- Impossible to mix French and German in one text:

  „Amélie knackt gerne die Kruste von Crème Brulé mit dem Löffel“
- Died together with DOS

Use the 8th Bit

- 128 new characters: 0 to 255 (00..FF)
- A lot of 8-Bit character sets have evolved
  - ISO 8859-x = ASCII + 160..255
  - ISO 8859-1 = Latin-1
    (Western European languages)
  - Windows 1252 = ISO 8859-1 + 128..159
  - etc.

ISO 8859

- Characters 0..127 identical to ASCII
- 128..159 undefined control characters
- 160..255 individually assigned

ISO 8859-1 / Latin-1

A0	A1 ¡	A2 ¢	A3 £	A4 ¤	A5 ¥	A6 ¦	A7 §	A8 ¨	A9 ©	AA ª	AB «	AC ¬	AD	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 ´	B5 µ	B6 ¶	B7 ·	B8 ¸	B9 ¹	BA º	BB »	BC ¼	BD ½	BE ¾	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

Afrikaans, Albanese, Basque, Danish, German, **English**, Faroese, Finnish, French, Icelandic, Italian, Catalan, Dutch, Norwegian, Portuguese, Rhaeto-Romance, Scottish Gaelic, Schwedish, Spanish, Suaheli

Large parts of the world – wide-spread use

ISO 8859-2 / Latin-2

A0	A1 Ą	A2 ˘	A3 Ł	A4 ¤	A5 Ľ	A6 Ś	A7 §	A8 ¨	A9 Š	AA Ş	AB Ť	AC Ź	AD -	AE Ž	AF Ż
B0 °	B1 ą	B2 ˛	B3 ł	B4 ´	B5 ľ	B6 ś	B7 ˇ	B8 ¸	B9 š	BA ş	BB ť	BC ź	BD ˝	BE ž	BF ż
C0 Ŕ	C1 Á	C2 Â	C3 Ă	C4 Ä	C5 Ĺ	C6 Ć	C7 Ç	C8 Č	C9 É	CA Ę	CB Ë	CC Ě	CD Í	CE Î	CF Ď
D0 Đ	D1 Ń	D2 Ň	D3 Ó	D4 Ô	D5 Ő	D6 Ö	D7 ×	D8 Ř	D9 Ů	DA Ú	DB Ű	DC Ü	DD Ý	DE Ţ	DF ß
E0 ŕ	E1 á	E2 â	E3 ă	E4 ä	E5 ĺ	E6 ć	E7 ç	E8 č	E9 é	EA ę	EB ë	EC ě	ED í	EE î	EF ď
F0 đ	F1 ń	F2 ň	F3 ó	F4 ô	F5 ő	F6 ö	F7 ÷	F8 ř	F9 ů	FA ú	FB ű	FC ü	FD ý	FE ţ	FF ˙

Central and Eastern Europe (Czech, Polish, etc.)

ISO 8859-4 / Latin-4

A0	A1 Ą	A2 ĸ	A3 Ŗ	A4 ¤	A5 Ĩ	A6 Ļ	A7 §	A8 ¨	A9 Š	AA Ē	AB Ģ	AC Ŧ	AD -	AE Ž	AF ¯
B0 °	B1 ą	B2 ˛	B3 ŗ	B4 ´	B5 ĩ	B6 ļ	B7 ˇ	B8 ¸	B9 š	BA ē	BB ģ	BC ŧ	BD Ŋ	BE ž	BF ŋ
C0 Ā	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Į	C8 Č	C9 É	CA Ę	CB Ë	CC Ė	CD Í	CE Î	CF Ī
D0 Đ	D1 Ņ	D2 Ō	D3 Ķ	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ų	DA Ú	DB Û	DC Ü	DD Ũ	DE Ū	DF ß
E0 ā	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 į	E8 č	E9 é	EA ę	EB ë	EC ė	ED í	EE î	EF ī
F0 đ	F1 ņ	F2 ō	F3 ķ	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ų	FA ú	FB û	FC ü	FD ũ	FE ū	FF ˙

Northern Europe, Baltic, Greenlanic, Sami

ISO 8859-5 / Cyrillic

A0	A1 Ё	A2 Ђ	A3 Ѓ	A4 Є	A5 Ѕ	A6 І	A7 Ї	A8 Ј	A9 Љ	AA Њ	AB Ћ	AC Ќ	AD	AE Ў	AF Џ
B0 А	B1 Б	B2 В	B3 Г	B4 Д	B5 Е	B6 Ж	B7 З	B8 И	B9 Й	BA К	BB Л	BC М	BD Н	BE О	BF П
C0 Р	C1 С	C2 Т	C3 У	C4 Ф	C5 Х	C6 Ц	C7 Ч	C8 Ш	C9 Щ	CA Ъ	CB Ы	CC Ь	CD Э	CE Ю	CF Я
D0 а	D1 б	D2 в	D3 г	D4 д	D5 е	D6 ж	D7 з	D8 и	D9 й	DA к	DB л	DC м	DD н	DE о	DF п
E0 р	E1 с	E2 т	E3 у	E4 ф	E5 х	E6 ц	E7 ч	E8 ш	E9 щ	EA ъ	EB ы	EC ь	ED э	EE ю	EF я
F0 №	F1 ё	F2 ђ	F3 ѓ	F4 є	F5 ѕ	F6 і	F7 ї	F8 ј	F9 љ	FA њ	FB ћ	FC ќ	FD §	FE ў	FF џ

Cyrillic (Russia, Ukraine, etc.)
More important: KOI8-R (Russian), KOI8-U (Ukrainian)

ISO 8859-9 / Latin-5

A0	A1 ¡	A2 ¢	A3 £	A4 ¤	A5 ¥	A6 ¦	A7 §	A8 ¨	A9 ©	AA ª	AB «	AC ¬	AD	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 ´	B5 µ	B6 ¶	B7 ·	B8 ¸	B9 ¹	BA º	BB »	BC ¼	BD ½	BE ¾	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ğ	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD İ	DE Ş	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ğ	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ı	FE ş	FF ÿ

Turkish

Windows Character Sets

- Congruent to ISO-8859
- Additional assignment of characters **128..159** with visible (non-control) characters
  - Hyphens n length – and m length — (vs. Dash -)
  - Typographic „quotation marks“, etc.
- Windows character sets officially registered at IANA

windows-1252

- Congruent to ISO 8859-1 (Western European languages, including English)
- Additional characters in the 128..159 range:

€ ‚ ƒ „ … † ‡ ˆ ‰ Š ‹ Œ Ž
‘ ’ “ ” • – — ˜ ™ š › œ ž Ÿ

Euro sign € since 2000 (Windows 1252-2000)

Multi-Byte Character Sets MBCS

- Multiple Bytes per Character
- Eastern Asian Languages (CJK)
- String Length <> Length of character chain
- Making extraction of sub-strings more difficult
- Firebird functions:
  - **BIT_LENGTH**: length of a string in bits (!)
  - **CHAR_LENGTH/CHARACTER_LENGTH**: length of a string in characters
  - **OCTET_LENGTH**: length of a string in bytes

Unicode

Why Unicode?

- One single Character Set for all languages/scripts
- No code overlaps
- Hardware and OS independant
- Standardisation ISO **10**646 (vs. ASCII = ISO 646)

Unicode

- Started with 16 Bits/Character, now 32 Bits/Char
- Ability to code 1,114,112 characters
- Currently only a fraction is used
- Current version: 5.0.0
- Defines Characters, *not* Glyphs
- Practically equal to ISO/IEC 10646

Character Definition

- Unicode defines a numerical *Code Point* (scalar value) and an *Identifier* for every character

0041 LATIN CAPITAL LETTER A
00E4 LATIN SMALL LETTER A WITH DIAERESIS
0391 GREEK CAPITAL LETTER ALPHA
05D0 HEBREW LETTER ALEF
0950 DEVANAGARI OM
1D56C MATHEMATICAL BOLD FRAKTUR CAPITAL A

Unicode Code Points

- Codespace: 0..10FFFF
- Usual Notation is hexadecimal with preceding U+ and at least 4 digits
- U+0020
- U+0041
- U+1D56C

Unicode Character Names

- Consisting of the uppercase letters A..Z, digits 0..9, hyphen (-) and whitespace.
- BYZANTINE MUSICAL SYMBOL LEIMMA ENOS CHRONOU
- DESERET CAPITAL LETTER OW
- BRAILLE PATTERN DOTS-1245

Unicode Coding

- Storage of Code Points in memory
- 32 Bits/Character easy but too fat
- There are various codings around:
  - 8-Bit (UTF-8)
  - 16-Bit (UCS-2, UTF-16)
  - 32-Bit (UCS-4, UTF-32)
  - PunyCode for international Domain names
  - UTF-7, UTF-1, etc.

UCS-2

- 16 Bits per Character
- Code Area: 0000..FFFF
  = Basic Multilingual Plane (BMP)
- Since Unicode 3.1 not all characters can be coded
- Replaced by UTF-16
- „Unicode" often used as a synonym for UCS-2
  (which is wrong and can lead to misunderstanding)

UTF-16

- 16 Bits per Character
- All characters > FFFF must be coded as a „Surrogate Pair“ and occupy 2 contiguous 16-Bit words
- Complete Code Space can be coded
- Difficult to calculate string length or substrings

Endianness

- Problem with UCS-2, UTF-16: Low/High-Byte ordering
- Differentiate in UTF-16BE and UTF-16LE in metadata
- Byte Order Mark BOM U+FEFF
- U+FEFF set at the very beginning of each text
- U+FFFE is (and will be) an invalid code point

UCS-2 vs. UTF-16

- UTF-16 ist backwards compatible
- UCS-2 often synonym for „Unicode“
- WideString, wchar_t
- NT3, NT4: UCS-2
- Since Windows 2000: UTF-16

UTF-8

- Coding as 8-Bit strings
- US-ASCII characters untouched, all others occupy 2 to 4 contiguous bytes
- Complete codespace can be coded
- Advantage: „Latin" texts quite compact and readable in unaware editors
- Problem: string length, substrings, etc.

UTF-8 coding

- US-ASCII characters untouched, Bit 7 reset `0xxxxxxx`
- All others are sequences of bytes with Bit 7 set `1xxxxxxx`
- First Byte: As many leading bits set as length of sequence: `110xxxxx`
- Following Bytes: Bit 7 set, Bit 6 reset: `10xxxxxx`
- Recognize the type of byte:
  - Complete character: 0xxxxxxx
  - Part of a sequence: 1xxxxxxx
  - Sequence head: 11xxxxxx
  - Sequence tail: 10xxxxxx

UTF-8 coding

- Bits after the type bits remain for coding the Coding Point

ä – LATIN SMALL LETTER A WITH DIAERESIS

$00E4_{16} = 11100100_2$

00000 – 00007F: 0xxxxxxx
00080 – 0007FF: 110xxxxx 10xxxxxx
00800 – 00FFFF: 1110xxxx 10xxxxxx 10xxxxxx
10000 – 1FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

```
110xxxxx 10xxxxxx
   ooo11   100100
-------- --------
11000011 10100100 bin
C3       A4       hex
195      164      dez
Ã        ¤        Latin-1
```

UTF-8 with ISO 8859-1 Rendering

JÃ¼rgen Klinsmann
(German Soccer Legend)

UCS-4, UTF-32

- 32 Bit per Character
- Every code point as one word in memory
- Fat but handy (1 word = 1 character)
- Endianness issues (UTF-32BE, UTF-32LE, BOM)
- Complete codespace can be coded
- No practical differences between UCS-4 and UTF-32

What is Plain Text?

- For every string, for every text (file, e-mail, attachment, download, etc) the encoding must be known.
- Plain text can begin with a BOM

> There Ain't No Such Thing As Plain Text.
> -- Joel Spolsky

Transcoding / Transliteration

- Transform from special character sets to Unicode and back
- e.g. Windows-1252 –> Unicode –> ISO 8859-1
- Translation tables: www.unicode.org
- Characters can get lost (ك becomes ?)
- Characters can get transformed (ç becomes c)

Sorting

- Sorting rules also apply for searching
- There are cultural differences
  - treat ä like a
  - treat ä like ae
  - treat ä as a seperate character after z
- Unicode defines algorithms and delivers tables for sorting

Case Mappings

- Not available in every language
- Not always reversible
- Turkish: ı –> I, i –> İ English: i –> I
- Not necessarily 1:1 ß –> SS
- Case Mappings are language dependant

Comparisons, Sorting

- Case Insensitive

  Firebird = FIREBIRD ?
  river = RiVeR ?
  Fluß = FLUSS ?

  a B c <--> B a c

- Accent Insensitive

  Amélie = Amelie ?

  a é i o ù <--> a i o é ù

Firebird and Character Sets

CHAR / VARCHAR Fields

- Every CHAR or VARCHAR column has a character set applied by Firebird
  (Remember? „There is no such thing as plain text“)
- This character set will be used for storage
- The character set can be defined when declaring the column:

```
create table persons (
  pers_id integer not null primary key,
  last_name  varchar (50) character set iso8859_1,
  first_name varchar (50) character set iso8859_1
);
```

Default Character Set

- A default character set for all string columns can be defined together with CREATE DATABASE:

```
create database 'employee.gdb'
  default character set ISO8859_1;
```

- You can override the default character set:

```
create table persons (
  pers_id integer not null primary key,
  last_name  varchar (50),
  first_name varchar (50),
  czech_name varchar (50) character set iso8859_2
);
```

Text Blobs

- Character Sets also apply to text blobs

```
create table persons (
  pers_id integer not null primary key,
  last_name  varchar (50),
  first_name varchar (50),
  resume blob sub_type text
);
```

Client Character Set

- The Client application defines a character set for its connection
- All strings will be transliterated to/from this Client Character Set
- Firebird 1.5: „Unable to transliterate between character sets“
- Firebird 2.0: ServerCS <–> Unicode <–> ClientCS

How to define the Client Character Set

- IB Objects:

```
IB_Connection1.CharSet := 'ISO8859_1';
```

- IBX:

```
IbDatabase1.Params.Add ('lc_ctype=ISO8859_1');
```

- PHP:

```
$db = ibase_connect ($Name, $Usr, $Pwd, "ISO8895_1");
```

Collations

- Define sort ordering for ORDER BY
- Define casing rules for UPPER(), LOWER()

```
SELECT *
FROM ...
WHERE UPPER (NAME COLLATE DE_DE) = :SEARCHNAME
ORDER BY LASTNAME COLLATE FR_CA
```

- Collations defined per Character Set
- Examples:

```
ISO8859_1: DE_DE, DU_NL, FR_FR, FR_CA, PT_PT, PT_BR
WIN1250: WIN1250, BS_BA, WIN_CZ, WIN_CZ_CI_AI
```

Defining the Collation for a Column

- There is no such thing as a default collation
- You can define the standard collation for a column:

```
create table persons (
  pers_id integer not null primary key,
  last_name  varchar (50) collate de_de,
  first_name varchar (50) collate de_de,

);
```

UPPER(), LOWER()

- Firebird 1.0/1.5: UPPER() only works correctly if there is a collation defined for the parameter field.

  Without collation no uppercasing of letters outside the a..z range.

- Firebird 2.0: UPPER() will return uppercased characters for all characters
- Firebird 2.0: New LOWER() function

Searching (Case insensitive)

- WHERE LIKE, WHERE STARTING WITH, WHERE =

```
select * from persons
where upper (last_name) like '%TARK%'

select * from persons
where upper (last_name) starting with 'STARK'

select * from persons
where upper (last_name collate de_de) like '%Ä%'
```

Indexed Case Insensitive Searches

- Firebird 1.5: Shadow Field and Trigger

```
create table persons (
  name       varchar (50) collate de_de,
  name_upper varchar (50) collate de_de);

create index idx_person_name on persons (name_upper);

create or replace trigger biu_persons for persons
before insert or update as
begin
  new.name_upper = upper (new.name);
end;

select * from persons
where name_upper = 'LÖW'
```

Indexed Case Insensitive Searches

- Firebird 2.0: Expression Index

```
create table persons (
  name varchar (50) collate de_de,
  city varchar (50));

create index idx_person_name on persons
  computed by ( upper (name) );

select * from persons
where upper (name) = 'FIREBIRD'


create index idx_person_city on persons
  computed by ( upper (city collate de_de) );

select * from persons
where upper (city collate de_de) = 'MÜNCHEN'
```

Firebird and Unicode

- Unicode internally stored as UTF-8
- Character Set **UNICODE_FSS**: Up until Firebird 1.5: an old version of UTF8 that accepts malformed strings and does not enforce correct maximum string length.
- Character Set **UTF8**: Firebird 2.0: Replacement for UNICODE_FSS
- Unicode used to transliterate between character sets
- Unicode collation implemented for comparisons and casings

Unicode Collations for UTF8

- UCS_BASIC: sorts in Unicode Code Point order
- UNICODE: uses the Unicode Collation Algorithm

```
select * from t order by c collate ucs_basic;
A B a b á

select * from t order by c collate unicode;
a A á b B
```

Examples

- LanguageMix.htm
- CharsetDemo.fdb
- CharsetDemo.dpr
- charsetdemo.php

Special Character Sets

- **NONE:** No character set applied. With this character set setting, Firebird is unable to perform conversion operations like UPPER() correctly on anything other than the standard 26 latin letters.
- **OCTETS:** Same as NONE. Cannot be used as client connection character set. Space character is #x00
- **ASCII:** US-ASCII
- **UNICODE_FSS:** an old version of UTF8 that accepts malformed strings and does not enforce correct maximum string length. Replaced by **UTF8** in FB2.0.

Character Sets and Collations

- **ISO8859_*x*** (e.g. ISO8895_1, ISO8859_2)
  Collations: DE_DE, FR_FR, CS_CZ, etc.
- **WIN125*x*** (e.g. WIN1252, WIN1250)
  Collations: WIN1252, WIN_PTBR, PXW_CSY, etc.
- **DOS*xxx*** (e.g. DOS850, DOS852)
  Collations: DB_DEU850, PDOX_CSY

- Complete List:
  www.destructor.de/firebird/charsets.htm

Other Character Sets

- BIG_5: Chinese
- CYRL, KOI8-R, KOI8-U: Russian, Ukrainian
- KSC_5601: Korean
- SJIS_0208: Japanese
- EUCJ_0208: Japanese
- GB_2312: Chinese

Which one to select?

- DOS, dBASE and Paradox only for legacy support
- WINxxx is extension of corresponding ISOxxx, but may lead to problems on non-Windows systems.
- ISOxxx is missing a few characters of WINxxx (e.g. typographic dash signs) –> prepare to handle this
- If you expect to store various languages, go for Unicode UTF-8 but use Firebird 2.0 then.

Links

- Unicode Consortium, ISO 10646
  Book: The Unicode Standard, Version 5.0
  ISBN 0-321-48091-0
  www.unicode.org
- Troy Wolbrink: Tnt Delphi Free Unicode Controls
  www.tntware.com
- My Firebird Website and Conference Blog/Gallery
  www.destructor.de/firebird
- Papers of this talk
  www.destructor.de/talks

Questions?

Stefan Heymann

www.destructor.de
stefan@destructor.de

Thank You!
Danke! Merci! Grazie!
Gracias! Obrigado!
Děkuji! Paldies! Hvala!
СПАСИБО

Stefan Heymann

www.destructor.de
stefan@destructor.de