

Practical character sets

In MySQL, on the web, and everywhere

Domas Mituzas
MySQL @ Sun Microsystems
Wikimedia Foundation

It seems simple

- a b c d e f
- a ą b c Ć d e ę è f
- а б ц д е ф
- ƒ Ƴ Ƨ Ƶ Ɓ Ƭ
- ... --- ...

It is not

- Linguists are not computer scientists
- Linguists hate computers
- Computers hate linguists
- Languages evolved before computers
- They continue evolving faster than computers

Character sets

- a=1, b=2, c=3, d=4, e=5, ae=?
- A=11, B=12, C=13, D=14, E=15
- upper()? lower()?
- ascii → latin1 → iso8859 → unicode
- Arabic, Cyrillic, Hebrew, Japanese, Chinese (multiple), ...
- Unicode! \o/

First problems

- No universal order
- Different upper() and lower() functions
- BMP, does not include gothic or some asian scripts
- Even Unicode has multiple encodings

Collations to the rescue

- Define sorting of strings
- Define comparison of strings
- Define upper/lower case rules
- Define case sensitivity (or lack of it)
- Sometimes non-intuitive
- Sometimes very language, region or use specific (phonebook vs dictionary orders)
- Deal with digraphs :'-('

charsets in MySQL!

- pre 4.1 - serverwide 8-bit collation map
- 4.1 - extensive character set and collation support, multibyte, per-column/client, et al
- 5.0 - minor fixes, lots of work elsewhere :)
- 5.1 - space efficient multibyte sorting, xml user collations
- 6.0 - 4 byte utf8, more than BMP

IMPORTANT!

- `character_set_client` - what server receives
- `character_set_results` - what server sends
- column character set - how data is stored
 - derived from table character set
 - derived from database character set
 - derived from server character set

Consistency

- It is mandatory to have right definitions
- Otherwise data is mistreated, corrupted or just stored inefficiently
- It may work nice but be corrupted at backup

Ä™ Ä—Ä

- This is how server sees utf8 data stream passed as latin I
- This is how correctly configured clients will see the data passed by misconfigured clients
- Fix: Convert to binary, convert to utf8

??????????


- This is how misconfigured clients will see correct utf8 data
- Though data passed by misconfigured clients may be seen properly

Switching off

- VARBINARY, BINARY, BLOB, VARCHAR BINARY, .. field types
- set names binary;
- set character_set_...=null;
- BINARY modifier:
 - SELECT BINARY ...
 - ..WHERE BINARY x = ...

Performance

- Fast: BINARY comparison
- OK: Same character set
 - utf8_general_ci faster than utf8_unicode_ci
- Penalty: Character set conversion (storage/ client charset mismatch)



need numbers!
and mysqldump
comment

Unicode [in]Security

- Homographic attacks
 - <https://paypal.com>, Verisign certified!
 - admin
- Control sequences (rtl, etc)
- Noncritical, but annoying
- Needs remedies in application

Control characters

- tfe! ot thgir
- Language tags
- Annotation, context, additional whitespace, ..

Digraphs

- ae=ae, ss=s, ll=l,
- Two letters can make a digraph, but do they?
 - Only dictionary can tell.
 - Defines position in search tree
- Efficient prefix search impossible
- Performance sacrificed to maintain consistency
- Stay with utf8_general_ci?

Collations

- Most are good enough
- Some needs not satisfied
 - Per-language accent rules differ
 - No accent-sensitive case-insensitive collation
 - Sometimes multiple collations have to be merged to satisfy special site needs

Normalization

- $\text{¨} + u = \text{ü} ?$
- Four normalization forms
- Software is different
- <http://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/includes/normal/>

Hacking the collation

- 4.0 - trivial mapping files
- 4.1, 5.0 - multibyte - edit and recompile
- 5.1 - xml based configuration files

Latin I

- Is actually Windows-1252, not ISO8859-1
- Uses display characters instead of control characters
- ISO8859_1 in Connector/J is silently mapped to CP1252

Utf8

- 3-byte-characters only
- Gothic, cherokee, some asian scripts not included
- Truncates strings at unknown characters in 5.0
- Undefined behavior before

UCS2

- Can be used only for storage in 5.0
 - Won't work in 'SET NAMES'
- Can be used as client character set in 5.1
- More efficient storage for very-international sites (3 byte utf8 chars become 2 byte)

Performance revisited

- Use single-byte string functions if possible
 - Detect multibyte characters with regexp and 'upgrade' code

Browsers

- URLs may come in utf8
- May not... need to guess charsets
- Windows default charsets differ..
- Old or esoteric browsers may corrupt Unicode data

Questions?

- domas at #mysql at freenode
- <http://dammit.lt/>
- slides: <http://dammit.lt/uc/>