MySQL®
Conference & Expo

Presented by

MySQL Conference & Expo

O'REILLY

Introduction to the Japanese Character Set

Yoshinori Matsunobu
Senior Consultant
MySQL AB

Speaker's Profile

- **Yoshinori Matsunobu**
- Senior Consultant, working at MySQL Japan
  - Performance Tuning
  - DBA
  - MySQL Cluster
  - HA/Scale-out Architecture Design/Implementation
  - Migration
  - i18n
- Authored four MySQL books in Japanese

Agenda

- The essences of the Japanese Character Set
  - Multi-byte character
  - A lot of character sets and encodings
  - Character code conversion

- Hot issues
  - UTF-8 : 4-byte characters
  - Shift_JIS : 0x5C escape problem
  - Full text search

What is multi-byte character?

- Two or more bytes per one character

  e.g. あ 0x82A0

- 1 byte = 8bit, $2^8 = 256$

- Is “256” enough to handle all characters (symbols) in your country ?

  Alphabet -- A-Z, a-z (26*2=52)
  Number -- 0-9 (10)
  Others -- (space, tab, semicolon, etc..)

ASCII

- 7 bit Encoding Scheme (0x00 - 0x7F)
- Most Significant Bit is always 0 (1byte = 1character)
- Control Character

  (0x00 – 0x1F, 0x20, 0x7F)

  Total 34
- Graphic Character

  (0x21 – 0x7E)

  Total 94

Lower 4bit \ Upper 3bit	0	1	2	3	4	5	6	7
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	EQT	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	DEL

Japanese Characters

- Hiragana (Over 50 characters)

  あいうえお　かきくけこ　さしすせそ　たちつてと　なにぬねの　…

- Katakana (Over 50 characters)

  アイウエオ　カキクケコ　サシスセソ　タチツテト　ナニヌネノ　…

  * Half-Width Katakana:

    ｱｲｳｴｵ　ｶｷｸｹｺ　ｻｼｽｾｿ　ﾀﾁﾂﾃﾄ　ﾅﾆﾇﾈﾉ　…

- Kanji (Over **6,000** characters)

  亜　哀　愛　悪　握　圧　扱　安　暗　案　…

1 byte(256) is not enough to handle Japanese characters.
-> multi-byte character was adopted
A set of these characters is called “Character Set”

Japanese Character Set

- Japan Industrial Standard (JIS) specifies Japanese Character Set
- Sometimes updated

```
JIS X 0208:1990  ->  JIS X 0208:1997
JIS X 0213:2000  ->  JIS X 0213:2004
```

- Vendor defined Japanese Character Set

```
NEC Kanji, IBM Kanji
```

- Why are there so many character sets?

  - There are too many characters (Kanji) in Japan. It is difficult to define cover area.

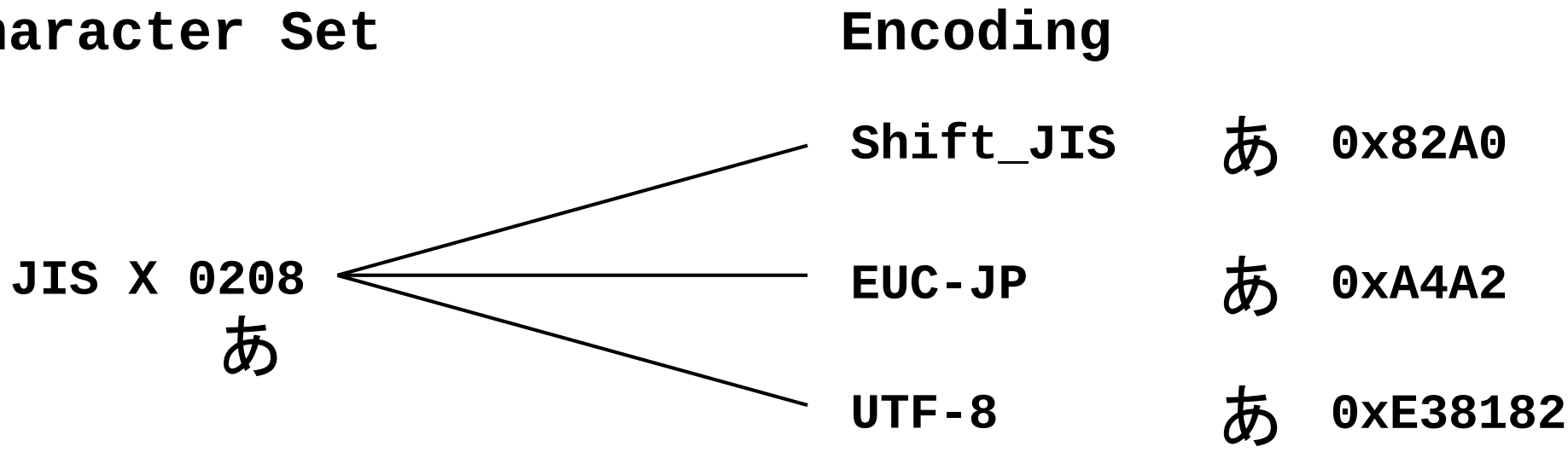
    JIS X 0208 is subset of them. NEC/IBM Kanji supplements JIS X 0208.

  - The number of symbols is increasing

    e.g. Cellular phone specific characters

    Some advanced author sometimes create new symbols

Character Set and Encoding



- Character Set and Encoding are different meanings, but usually being used without distinction
- There are several encodings (Shift_JIS,EUC-JP,UTF-8,etc..)

  Shift_JIS is the most widely used encoding now.
  Gradually moving to Unicode (UTF-8)
- Each code mapping is different from each other

Size of Japanese Characters

- Shift_JIS
  - All ASCII characters and Half-width katakana are 1 byte
  - The others are 2 bytes
- EUC-JP
  - All ASCII characters are 1 byte
  - Most of Japanese characters are 2 bytes
  - The rest are 3 bytes
- UTF-8
  - All ASCII characters are 1 byte
  - Most of Japanese characters are 3 bytes
  - The rest of Japanese characters are 4 bytes

```
*This is one of the reason that Japanese people
 do not want to use UTF-8.
```

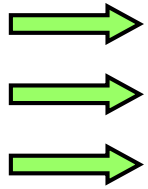
Character Set and Encoding (2)

Character Set		Shift_JIS encoding	EUC-JP encoding	Unicode encoding
JIS X 0208:1997	⇒	Shift_JIS	EUC-JP	UTF-8
JIS X 0208:1997 + NEC/IBM Kanji	⇒	CP932,Windows-31J	EUC-JP-Open	UTF-8
JIS X 0213:2004	⇒	Shift_JIS-2004	EUC-JIS-2004	UTF-8

- There are several character sets
- That's why there are too many encodings, which make us confused

Supported Encodings in MySQL

Character Set
JIS X 0208:1997
JIS X 0208:1997 + NEC/IBM Kanji
JIS X 0213:2004



Shift_JIS encoding	EUC-JP encoding	Unicode encoding
sjis	ujis	utf8
cp932	eucjpms	utf8

	Shift_JIS		EUC-JP		Unicode	
	sjis	cp932	ujis	eucjpms	utf8	ucs2
4.0	✔		✔			
4.1	✔	✔	✔		✔	✔
5.0	✔	✔	✔	✔	✔	✔

Example

```
$ mysql --default-character-set=cp932

mysql> create table t1 (c1 varchar(100)) charset cp932;
Query OK, 0 rows affected (0.08 sec)

mysql> insert into t1 values('あいうえ');
Query OK, 1 row affected (0.14 sec)

mysql> select c1, char_length(c1), length(c1) from t1;
+----------+------------------+------------+
| c1       | char_length(c1) | length(c1) |
+----------+------------------+------------+
| あいうえ |                4 |          8 |
+----------+------------------+------------+
1 row in set (0.00 sec)
```

Hiragana

Failed example

```
$ mysql

mysql> create table t1 (c1 varchar(100)) charset cp932;
Query OK, 0 rows affected (0.08 sec)
mysql> insert into t1 values('あいうえ');
Query OK, 1 row affected, 1 warning (0.08 sec)

mysql> show warnings;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1265 | Data truncated for column 'c1' at row 1 |
+---------+------+----------------------------------------+

mysql> select c1, char_length(c1), length(c1) from t1;
+----------+-----------------+------------+
| c1       | char_length(c1) | length(c1) |
+----------+-----------------+------------+
| ???????? |               8 |          8 |
+----------+-----------------+------------+
```

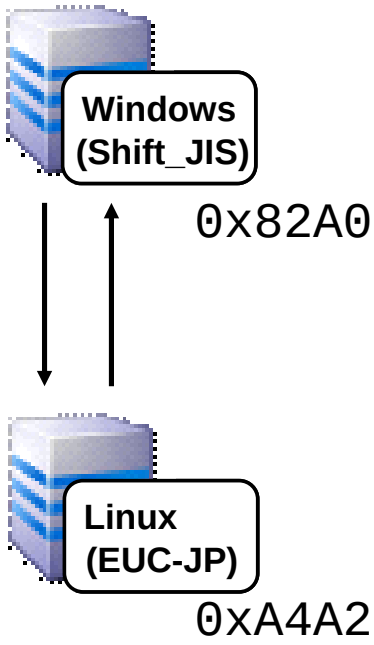
Well known as “mojibake”

Character code conversion

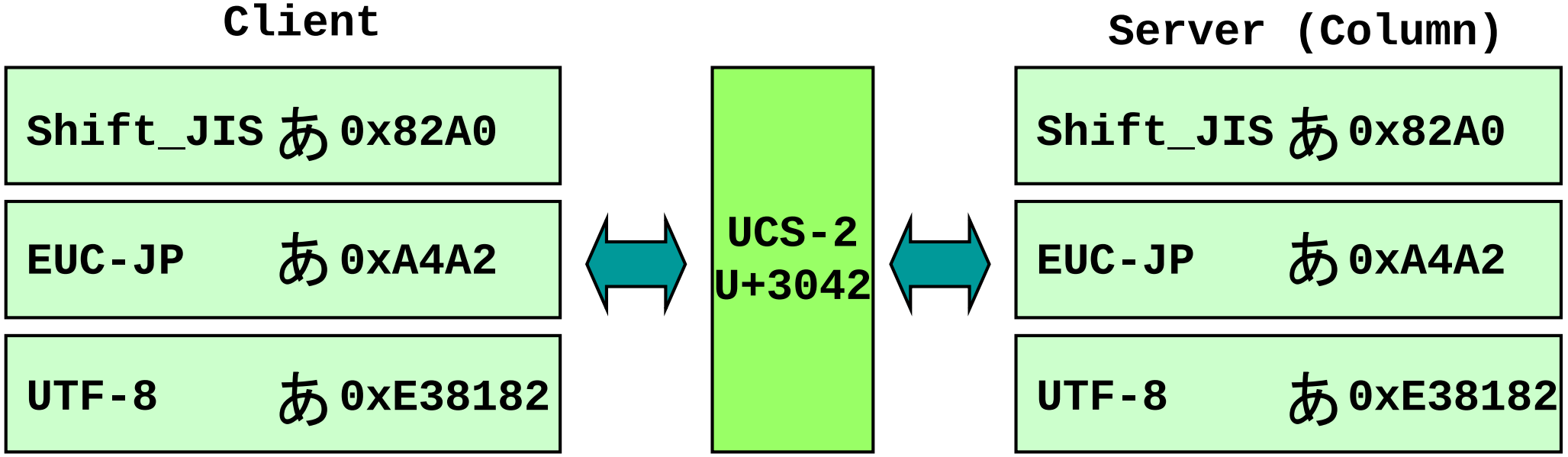
Shift_JIS	あ	0x82A0
EUC-JP	あ	0xA4A2
UTF-8	あ	0xE38182

- Japanese characters have different code point for each encoding
- Sometimes code conversion between different encodings is needed

  e.g : 0x82A0(Windows Shift_JIS) <-> 0xA4A2(Linux EUC-JP)



MySQL Code conversion algorithm



- UCS-2 facilitates conversion from one encoding to another
- MySQL has code conversion mapping to/from UCS-2 (See strings/ctype-cp932.c for example)
- If client encoding and server encoding are the same, code conversion doesn't occur
- If conversion fails, the character is converted to "?"

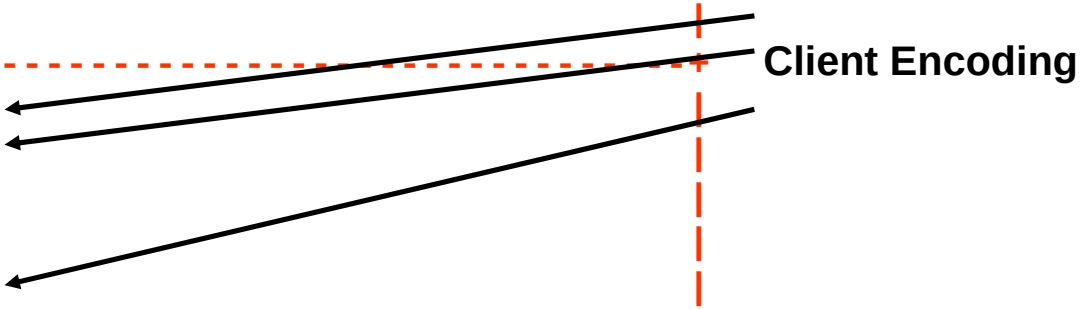
Failed character conversion

```
$ mysql
mysql> insert into t1 values('あいうえ');
Query OK, 1 row affected, 1 warning (0.08 sec)

mysql> show warnings;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1265 | Data truncated for column 'c1' at row 1 |
+---------+------+----------------------------------------+

mysql> select c1, char_length(c1), length(c1) from t1;
+----------+-----------------+------------+
| c1       | char_length(c1) | length(c1) |
+----------+-----------------+------------+
| ???????? |               8 |          8 |
+----------+-----------------+------------+
```

- Since client encoding is not specified, default MySQL encoding “latin1” is used
- latin1 doesn’t support Japanese Characters. Changing client encoding is needed.
- my.cnf parameter “skip-character-set-client-handshake” helps
  (client encoding is set to the same value of “character-set-server”)

How to check client encoding

```
mysql> SHOW VARIABLES LIKE 'char%';
+--------------------------+----------------------------------------+
| Variable_name            | Value                                  |
+--------------------------+----------------------------------------+
| character_set_client     | latin1                                 |
| character_set_connection | latin1                                 |
| character_set_database   | cp932                                  |
| character_set_filesystem | binary                                 |
| character_set_results    | latin1                                 |
| character_set_server     | cp932                                  |
| character_set_system     | utf8                                   |
| character_sets_dir       | D:¥mysql-5.0.38-win32¥share¥charsets¥ |
+--------------------------+----------------------------------------+
```



How to check table/column encoding

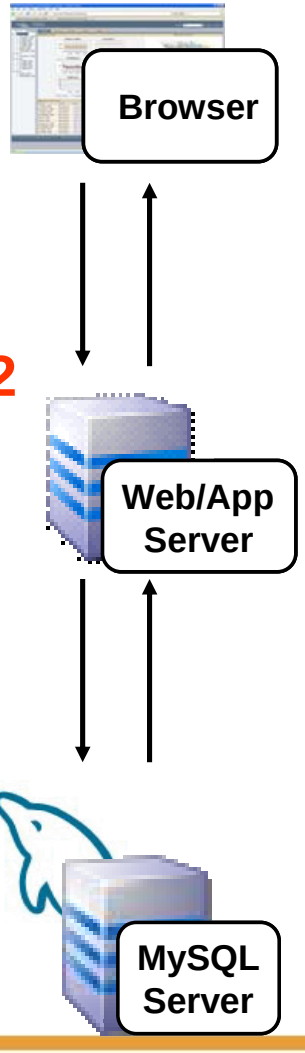
```
mysql> SHOW CREATE TABLE t1¥G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `c1` varchar(12) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=cp932
1 row in set (0.00 sec)
```

```
mysql> SELECT column_name, character_set_name, collation_name FROM
information_schema.columns WHERE table_name='t1';
+-------------+--------------------+-------------------+
| column_name | character_set_name | collation_name    |
+-------------+--------------------+-------------------+
| c1          | cp932              | cp932_japanese_ci |
+-------------+--------------------+-------------------+
1 row in set (0.02 sec)
```

Relationship with Application Layer

1. Read HTTP Parameter
HttpServletRequest #setCharacterEncoding ("Windows-31J")

Windows-31J -> UCS-2



2. Pass to Database Driver
characterEncoding=Windows-31J Statement#setString()

UCS-2 -> Windows-31J

3. Store into MySQL (Conversion if needed)

4. Get from Database Driver
ResultSet#getString()

Windows-31J ->UCS-2

5. Return HTML stream
contentType="text/html; charset=Windows-31J"

UCS-2 -> Windows-31J

Hot Issue in Japan

- The essences of the Japanese Character Set
  - Multi-byte character
  - A lot of character sets and encodings
  - Character code conversion

- Hot issues
  - UTF-8 : 4-byte characters
  - Shift_JIS : 0x5C escape problem
  - Full text search

Unicode

- Intended to support worldwide characters

- Fixed Length

  UCS-2, UCS-4

- Variable Length

  UTF-16, UTF-8

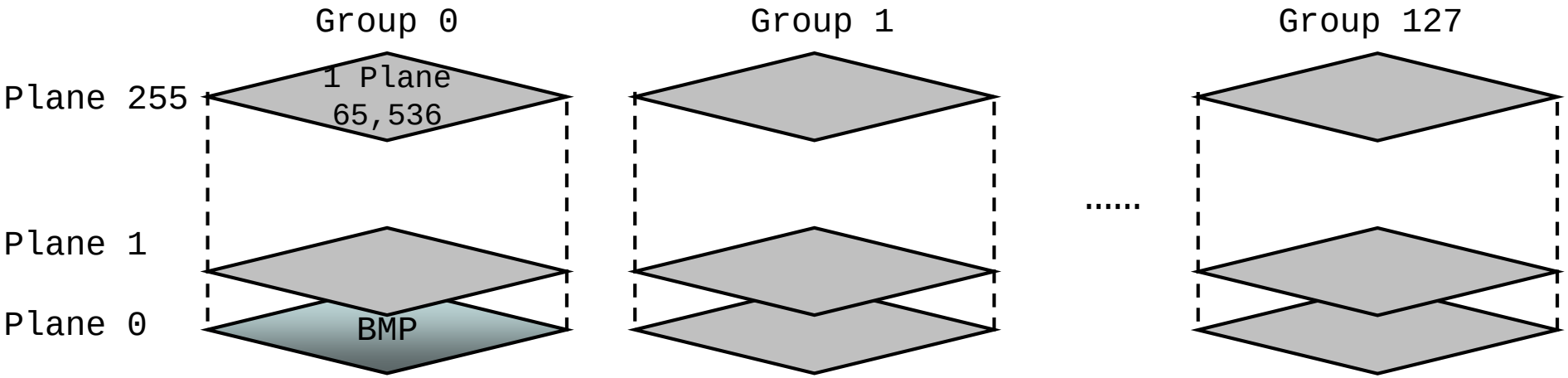
UCS-2 and UCS-4

- UCS-2

  2-byte Fixed Length. 2^{16}= 65,536 characters

- UCS-4

  4-byte Fixed Length. $2^{31} \fallingdotseq$ 2 billion characters



UCS-2 supports only Plane 0, Group 0(BMP)
UCS-4 supports 256 Planes and 128 Groups.
$2^{16} * 2^{8}(256) * 2^{7}(128) = 2^{31}$

(BMP = Basic Multilingual Plane)

UCS-2 Overflow

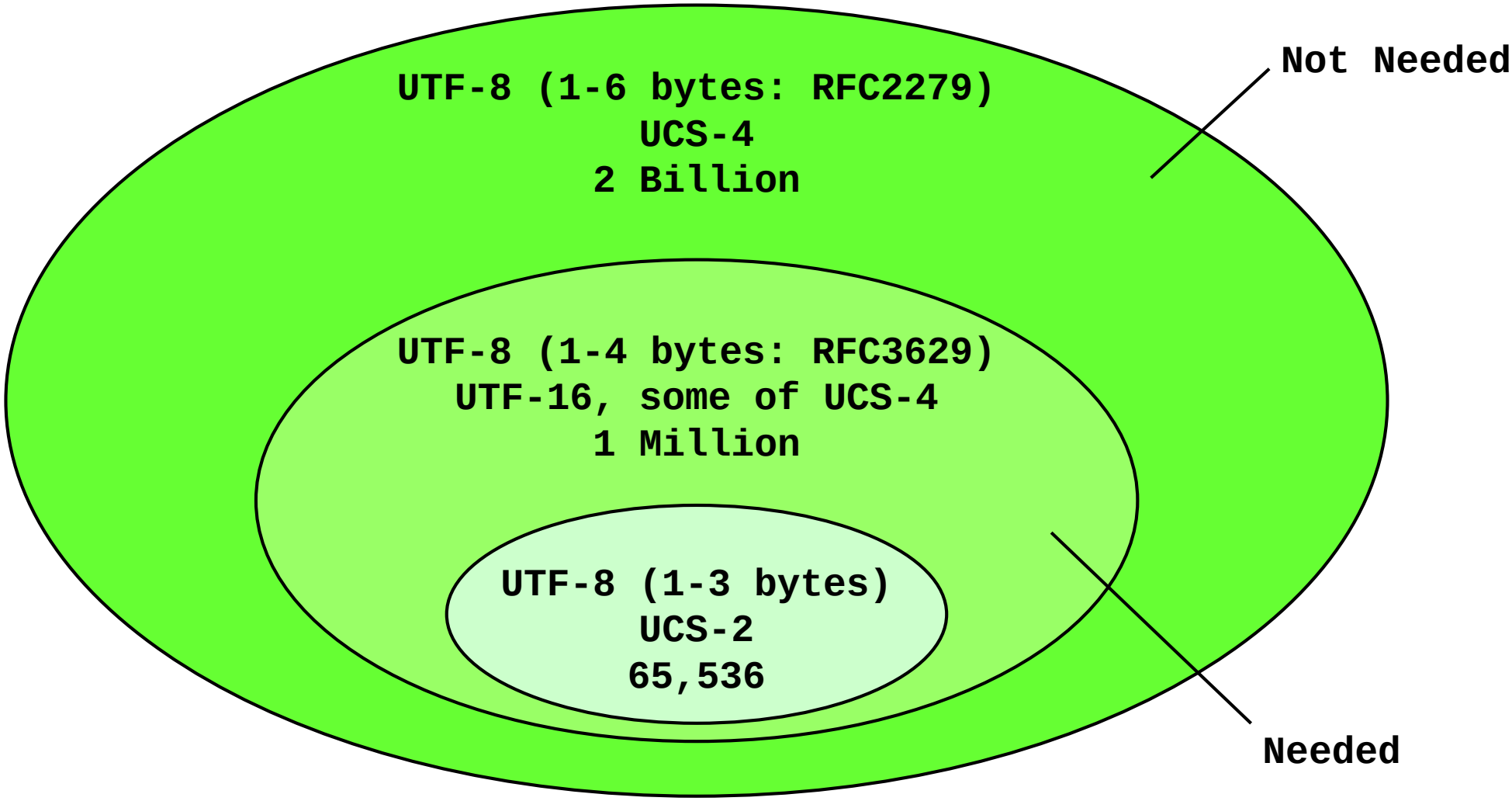
- Most of characters are covered by UCS-2.
- But some Japanese characters (some of JIS X 0213:2004) are not covered by UCS-2.
- Windows Vista supports JIS X 0213:2004 as standard character set in Japan.
- JIS X 0213:2004 is available even for Windows XP users if they applied Service Pack (KB927489) .

UCS-2 doesn't meet our needs !

UTF-8 and UTF-16

- Variable length encoding of UCS-2 and UCS-4
- UTF-16
  - 2-byte or 4-byte length
  - All UCS-2 characters are mapped to 2 bytes
  - Not all UCS-4 characters are supported (1 Million, supposed to be fine)
  - Supported UCS-4 characters are mapped to 4 bytes
- UTF-8
  - There are some specifications/implementations. RFC3629(4bytes) is the latest.
  - From 1 byte to 6 bytes (RFC 2279) Fully compliant with UCS-4
  - From 1 byte to 4 bytes (RFC 3629) Fully compliant with UTF-16
  - From 1 byte to 3 bytes Full compliant with UCS-2

Unicode coverage area



MySQL Unicode Implementation

- Internally handles all characters as UCS-2.
- UCS-4 is not supported.
- UCS-2 for client encoding is not supported.
- UTF-8 support is up to 3 bytes. 4-byte UTF-8 is not supported now.
- Currently being discussed to support in future builds

***For a long time and many platforms**
(e.g. J2SE <= 1.4), UTF-8 support only 3-byte length
(only UCS-2. So, not just MySQL!) ☺

Example of 4-byte UTF-8 problem

```
$ mysql --default-character-set=utf8

mysql> CREATE TABLE t1 (c1 VARCHAR(30)) CHARSET=utf8;
Query OK, 0 rows affected (0.09 sec)

#'ab' + 4-byte UTF-8 + 'cdef'
mysql> INSERT INTO t1 VALUES(0x6162F0A0808B63646566);
Query OK, 1 row affected, 1 warning (0.05 sec)

mysql> SELECT c1, HEX(c1) FROM t1;
+------+---------+
| c1   | HEX(c1) |
+------+---------+
| ab   | 6162    |
+------+---------+
1 row in set (0.00 sec)
```

- Invalid character(4-byte UTF-8) is truncated.
- Even valid characters after invalid character are also truncated.

Possible workarounds(1)

- Using VARBINARY/BLOB types
  - Can store any binary data
  - Always case-sensitive
  - FULLTEXT index is not supported
  - Application code modification might be needed

    e.g. *resultSet.getString(“string_column”)*

    -> new String(resultSet.getBytes("blob_column"),"UTF-8")

  - A specific configuration parameter might be introduced to Connector/J and Connector/.NET in the near future builds

Possible workarounds(2)

- Using UCS-2 for column encoding

```
$ mysql --default-character-set=utf8
mysql> CREATE TABLE t1 (c1 VARCHAR(30)) CHARSET=ucs2;
Query OK, 0 rows affected (0.09 sec)

#'ab' + 4-byte UTF-8 + 'cdef'
mysql> INSERT INTO t1 VALUES(_utf8 0x6162F0A0808B63646566);
Query OK, 1 row affected, 1 warning (0.05 sec)

mysql> SELECT c1, HEX(c1) FROM t1;
+------------+--------------------------------------+
| c1         | HEX(c1)                              |
+------------+--------------------------------------+
| ab????cdef | 00610062003F003F003F003F006300640065006600 |
+------------+--------------------------------------+
2 rows in set (0.03 sec)
```

- Better than truncated

- Every character (even ASCII character) consumes 2 bytes

Possible workarounds(3)

- Stop using Unicode,
  then use Shift_JIS(cp932) or EUC-JP(eucjpms)

  - All Japanese characters are stored/retrieved successfully

  - Code conversion of JIS X 0213:2004 characters
    is not currently supported

Shift_JIS

- The most widely used character encoding in Japan
- 1 or 2 byte encoding
  - All ASCII characters and Half-width katakana are 1 byte
  - The rest are 2 bytes

```
If the first byte value is:
0x00 – 0x7F  -> 1-byte characters
0xA0 – 0xDF  -> 1-byte characters
The rest     -> 2-byte characters
```

- 2nd byte might be in ASCII graphic code area (0x40 - 0x7E)

e.g ソ -- 0x835C

0x5C Escape problem

- What is 0x5C ? --> escape sequence (¥ : backslash in the US)
- Some Shift_JIS characters contain 0x5C in 2nd byte.

  ―ソ Ыl Ⅸ 噂 浬 欺 圭 構 蚕 十 申 曾 箪 貼 能 表 暴 予 禄
  兔 喀 媾 彌 拿 杤 歔 濬 畚 秉 綵 臀 藹 觸 軆 鐔 饅 鷭 偬 瑯 繽 狀

  ソ -- 0x835C

- Escape Rules in MySQL

```
mysql> SELECT 'AAA ¥n BBB'
AS c1;
+-----------+
| c1        |
+-----------+
| AAA
 BBB |
+-----------+
```

0x5C6E -> 0x0A

```
mysql> SELECT '¥100 JPY';
+---------+
| 100 JPY |
+---------+
| 100 JPY |
+---------+
mysql> SELECT '¥¥100 JPY';
+----------+
| ¥100 JPY |
+----------+
| ¥100 JPY |
+----------+
```

Single 0x5C -> removed

0x5C5C -> 0x5C

Character Corruption Example

```
$ mysql

mysql> create table t1 (c1 varchar(30));
Query OK, 0 rows affected (0.09 sec)

mysql> insert into t1 values('ソabc');
Query OK, 1 row affected (0.08 sec)

mysql> select c1, hex(c1) from t1;
+------+----------+
| c1   | hex(c1)  |
+------+----------+
| ヂbc | 83616263 |
+------+----------+
1 row in set (0.02 sec)
```

0x835C + 616263

0x83 + 616263
(single 0x5C is truncated)

Conversion logic needs to pay special care to Shift_JIS encoding in order not to truncate 0x5C in 2nd byte.
(MySQL does support this for sjis/cp932 client encoding)

Full text search in Japanese

- Native MySQL doesn't support full text search in Japanese

  - Korean and Chinese are the same (Known as CJK issue)

- Japanese words are not delimited by space

```
English:
MySQL, the most popular Open Source SQL database management
system, is developed, distributed, and supported by MySQL AB.

Japanese:
MySQLは最も人気のあるオープンソースのDBMSで、
MySQL ABによって開発、配布、サポートが行なわれています。
```

Full text search in Japanese (general solutions)

- Dictionary based indexing
  - Dividing words by pre-installed dictionary
- N-Gram indexing
  - Dividing words by N letters (N=1,2,3..)
- Implemented for MySQL by one of our partners
  - MySQL + Senna
  - Officially supported by Sumisho Computer Systems

Conclusion

- Character Set and Encoding
  - There are many character sets in Japan

    JIS X 0208, Vendor Defined Kanji (NEC/IBM Kanji), JIS X 0213
  - There are many encodings in Japan

    Shift_JIS(sjis,cp932), EUC-JP(ujis, eucjpms), Unicode(utf8)
- 4-Byte UTF-8 support is needed
  - Some Japanese characters are not covered by UCS-2.
- Shift_JIS is dangerous, but widely used
  - 0x5C problem
  - Widely used for historical reasons

Thanks for coming!

- Contact

  ymatsunobu@mysql.com