# 字符集、对齐方式、统一编码

**章节目录**

[10.1 Character Sets and Collations in General](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-general)

[10.2 Character Sets and Collations in MySQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-mysql)

[10.2.1 Character Set Repertoire](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-repertoire)

[10.2.2 UTF-8 for Metadata](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-metadata)

[10.3 Specifying Character Sets and Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-syntax)

[10.3.1 Collation Naming Conventions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-names)

[10.3.2 Server Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-server)

[10.3.3 Database Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-database)

[10.3.4 Table Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-table)

[10.3.5 Column Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-column)

[10.3.6 Character String Literal Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-literal)

[10.3.7 The National Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-national)

[10.3.8 Character Set Introducers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer)

[10.3.9 Examples of Character Set and Collation Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-examples)

[10.3.10 Compatibility with Other DBMSs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-compatibility)

[10.4 Connection Character Sets and Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection)

[10.5 Configuring Application Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-applications)

[10.6 Error Message Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-errors)

[10.7 Column Character Set Conversion](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-conversion)

[10.8 Collation Issues](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collations)

[10.8.1 Using COLLATE in SQL Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collate)

[10.8.2 COLLATE Clause Precedence](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collate-precedence)

[10.8.3 Character Set and Collation Compatibility](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-compatibility)

[10.8.4 Collation Coercibility in Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility)

[10.8.5 The binary Collation Compared to \_bin Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations)

[10.8.6 Examples of the Effect of Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-effect)

[10.8.7 Using Collation in INFORMATION\_SCHEMA Searches](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-information-schema)

[10.9 Unicode Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode)

[10.9.1 The utf8mb4 Character Set (4-Byte UTF-8 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8mb4)

[10.9.2 The utf8mb3 Character Set (3-Byte UTF-8 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8mb3)

[10.9.3 The utf8 Character Set (Alias for utf8mb3)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8)

[10.9.4 The ucs2 Character Set (UCS-2 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-ucs2)

[10.9.5 The utf16 Character Set (UTF-16 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf16)

[10.9.6 The utf16le Character Set (UTF-16LE Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf16le)

[10.9.7 The utf32 Character Set (UTF-32 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf32)

[10.9.8 Converting Between 3-Byte and 4-Byte Unicode Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-conversion)

[10.10 Supported Character Sets and Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-charsets)

[10.10.1 Unicode Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets)

[10.10.2 West European Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-we-sets)

[10.10.3 Central European Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-ce-sets)

[10.10.4 South European and Middle East Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-se-me-sets)

[10.10.5 Baltic Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-baltic-sets)

[10.10.6 Cyrillic Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-cyrillic-sets)

[10.10.7 Asian Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-asian-sets)

[10.10.8 The Binary Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-set)

[10.11 Restrictions on Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-restrictions)

[10.12 Setting the Error Message Language](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#error-message-language)

[10.13 Adding a Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-character-set)

[10.13.1 Character Definition Arrays](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#character-arrays)

[10.13.2 String Collating Support for Complex Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#string-collating)

[10.13.3 Multi-Byte Character Support for Complex Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#multibyte-characters)

[10.14 Adding a Collation to a Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation)

[10.14.1 Collation Implementation Types](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-implementations)

[10.14.2 Choosing a Collation ID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-choosing-id)

[10.14.3 Adding a Simple Collation to an 8-Bit Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-simple-8bit)

[10.14.4 Adding a UCA Collation to a Unicode Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-unicode-uca)

[10.15 Character Set Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-configuration)

[10.16 MySQL Server Locale Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#locale-support)

MySQL includes character set support that enables you to store data using a variety of character sets and perform comparisons according to a variety of collations. The default MySQL server character set and collation are **utf8mb4** and **utf8mb4\_0900\_ai\_ci**, but you can specify character sets at the server, database, table, column, and string literal levels.

This chapter discusses the following topics:

What are character sets and collations?

The multiple-level default system for character set assignment.

Syntax for specifying character sets and collations.

Affected functions and operations.

Unicode support.

The character sets and collations that are available, with notes.

Selecting the language for error messages.

Selecting the locale for day and month names.

Character set issues affect not only data storage, but also communication between client programs and the MySQL server. If you want the client program to communicate with the server using a character set different from the default, you'll need to indicate which one. For example, to use the **utf8** Unicode character set, issue this statement after connecting to the server:

SET NAMES 'utf8';

For more information about configuring character sets for application use and character set-related issues in client/server communication, see [Section 10.5, “Configuring Application Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-applications), and [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).

## 10.1 Character Sets and Collations in General

A character set is a set of symbols and encodings. A collation is a set of rules for comparing characters in a character set. Let's make the distinction clear with an example of an imaginary character set.

Suppose that we have an alphabet with four letters: **A**, **B**, **a**, **b**. We give each letter a number: **A** = 0, **B** = 1, **a** = 2, **b** = 3. The letter **A** is a symbol, the number 0 is the encoding for **A**, and the combination of all four letters and their encodings is a character set.

Suppose that we want to compare two string values, **A** and **B**. The simplest way to do this is to look at the encodings: 0 for **A** and 1 for **B**. Because 0 is less than 1, we say **A** is less than **B**. What we've just done is apply a collation to our character set. The collation is a set of rules (only one rule in this case): “compare the encodings.” We call this simplest of all possible collations a binary collation.

But what if we want to say that the lowercase and uppercase letters are equivalent? Then we would have at least two rules: (1) treat the lowercase letters **a** and **b** as equivalent to **A** and **B**; (2) then compare the encodings. We call this a case-insensitive collation. It is a little more complex than a binary collation.

In real life, most character sets have many characters: not just **A** and **B** but whole alphabets, sometimes multiple alphabets or eastern writing systems with thousands of characters, along with many special symbols and punctuation marks. Also in real life, most collations have many rules, not just for whether to distinguish lettercase, but also for whether to distinguish accents (an “accent” is a mark attached to a character as in German **Ö**), and for multiple-character mappings (such as the rule that **Ö** = **OE** in one of the two German collations).

MySQL can do these things for you:

Store strings using a variety of character sets.

Compare strings using a variety of collations.

Mix strings with different character sets or collations in the same server, the same database, or even the same table.

Enable specification of character set and collation at any level.

To use these features effectively, you must know what character sets and collations are available, how to change the defaults, and how they affect the behavior of string operators and functions.

## 10.2 Character Sets and Collations in MySQL

[10.2.1 Character Set Repertoire](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-repertoire)

[10.2.2 UTF-8 for Metadata](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-metadata)

MySQL Server supports multiple character sets, including several Unicode character sets. To display the available character sets, use the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table or the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement. A partial listing follows. For more complete information, see [Section 10.10, “Supported Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-charsets).

mysql> **SHOW CHARACTER SET;**

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

| Charset | Description | Default collation | Maxlen |

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

| big5 | Big5 Traditional Chinese | big5\_chinese\_ci | 2 |

| binary | Binary pseudo charset | binary | 1 |

...

| latin1 | cp1252 West European | latin1\_swedish\_ci | 1 |

...

| ucs2 | UCS-2 Unicode | ucs2\_general\_ci | 2 |

...

| utf8 | UTF-8 Unicode | utf8\_general\_ci | 3 |

| utf8mb4 | UTF-8 Unicode | utf8mb4\_0900\_ai\_ci | 4 |

...

By default, the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement displays all available character sets. It takes an optional [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) or **WHERE** clause that indicates which character set names to match. The following example shows some of the Unicode character sets (those based on Unicode Transformation Format):

mysql> **SHOW CHARACTER SET LIKE 'utf%';**

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

| Charset | Description | Default collation | Maxlen |

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

| utf16 | UTF-16 Unicode | utf16\_general\_ci | 4 |

| utf16le | UTF-16LE Unicode | utf16le\_general\_ci | 4 |

| utf32 | UTF-32 Unicode | utf32\_general\_ci | 4 |

| utf8 | UTF-8 Unicode | utf8\_general\_ci | 3 |

| utf8mb4 | UTF-8 Unicode | utf8mb4\_0900\_ai\_ci | 4 |

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

A given character set always has at least one collation, and most character sets have several. To list the display collations for a character set, use the **INFORMATION\_SCHEMA** [**COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table or the [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) statement.

By default, the [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) statement displays all available collations. It takes an optional [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) or **WHERE** clause that indicates which collation names to display. For example, to see the collations for the default character set, **utf8mb4**, use this statement:

mysql> **SHOW COLLATION WHERE Charset = 'utf8mb4';**

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

| Collation | Charset | Id | Default | Compiled | Sortlen | Pad\_attribute |

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

| utf8mb4\_0900\_ai\_ci | utf8mb4 | 255 | Yes | Yes | 0 | NO PAD |

| utf8mb4\_0900\_as\_ci | utf8mb4 | 305 | | Yes | 0 | NO PAD |

| utf8mb4\_0900\_as\_cs | utf8mb4 | 278 | | Yes | 0 | NO PAD |

| utf8mb4\_0900\_bin | utf8mb4 | 309 | | Yes | 1 | NO PAD |

| utf8mb4\_bin | utf8mb4 | 46 | | Yes | 1 | PAD SPACE |

| utf8mb4\_croatian\_ci | utf8mb4 | 245 | | Yes | 8 | PAD SPACE |

| utf8mb4\_cs\_0900\_ai\_ci | utf8mb4 | 266 | | Yes | 0 | NO PAD |

| utf8mb4\_cs\_0900\_as\_cs | utf8mb4 | 289 | | Yes | 0 | NO PAD |

| utf8mb4\_czech\_ci | utf8mb4 | 234 | | Yes | 8 | PAD SPACE |

| utf8mb4\_danish\_ci | utf8mb4 | 235 | | Yes | 8 | PAD SPACE |

| utf8mb4\_da\_0900\_ai\_ci | utf8mb4 | 267 | | Yes | 0 | NO PAD |

| utf8mb4\_da\_0900\_as\_cs | utf8mb4 | 290 | | Yes | 0 | NO PAD |

| utf8mb4\_de\_pb\_0900\_ai\_ci | utf8mb4 | 256 | | Yes | 0 | NO PAD |

| utf8mb4\_de\_pb\_0900\_as\_cs | utf8mb4 | 279 | | Yes | 0 | NO PAD |

| utf8mb4\_eo\_0900\_ai\_ci | utf8mb4 | 273 | | Yes | 0 | NO PAD |

| utf8mb4\_eo\_0900\_as\_cs | utf8mb4 | 296 | | Yes | 0 | NO PAD |

| utf8mb4\_esperanto\_ci | utf8mb4 | 241 | | Yes | 8 | PAD SPACE |

| utf8mb4\_estonian\_ci | utf8mb4 | 230 | | Yes | 8 | PAD SPACE |

| utf8mb4\_es\_0900\_ai\_ci | utf8mb4 | 263 | | Yes | 0 | NO PAD |

| utf8mb4\_es\_0900\_as\_cs | utf8mb4 | 286 | | Yes | 0 | NO PAD |

| utf8mb4\_es\_trad\_0900\_ai\_ci | utf8mb4 | 270 | | Yes | 0 | NO PAD |

| utf8mb4\_es\_trad\_0900\_as\_cs | utf8mb4 | 293 | | Yes | 0 | NO PAD |

| utf8mb4\_et\_0900\_ai\_ci | utf8mb4 | 262 | | Yes | 0 | NO PAD |

| utf8mb4\_et\_0900\_as\_cs | utf8mb4 | 285 | | Yes | 0 | NO PAD |

| utf8mb4\_general\_ci | utf8mb4 | 45 | | Yes | 1 | PAD SPACE |

| utf8mb4\_german2\_ci | utf8mb4 | 244 | | Yes | 8 | PAD SPACE |

| utf8mb4\_hr\_0900\_ai\_ci | utf8mb4 | 275 | | Yes | 0 | NO PAD |

| utf8mb4\_hr\_0900\_as\_cs | utf8mb4 | 298 | | Yes | 0 | NO PAD |

| utf8mb4\_hungarian\_ci | utf8mb4 | 242 | | Yes | 8 | PAD SPACE |

| utf8mb4\_hu\_0900\_ai\_ci | utf8mb4 | 274 | | Yes | 0 | NO PAD |

| utf8mb4\_hu\_0900\_as\_cs | utf8mb4 | 297 | | Yes | 0 | NO PAD |

| utf8mb4\_icelandic\_ci | utf8mb4 | 225 | | Yes | 8 | PAD SPACE |

| utf8mb4\_is\_0900\_ai\_ci | utf8mb4 | 257 | | Yes | 0 | NO PAD |

| utf8mb4\_is\_0900\_as\_cs | utf8mb4 | 280 | | Yes | 0 | NO PAD |

| utf8mb4\_ja\_0900\_as\_cs | utf8mb4 | 303 | | Yes | 0 | NO PAD |

| utf8mb4\_ja\_0900\_as\_cs\_ks | utf8mb4 | 304 | | Yes | 24 | NO PAD |

| utf8mb4\_latvian\_ci | utf8mb4 | 226 | | Yes | 8 | PAD SPACE |

| utf8mb4\_la\_0900\_ai\_ci | utf8mb4 | 271 | | Yes | 0 | NO PAD |

| utf8mb4\_la\_0900\_as\_cs | utf8mb4 | 294 | | Yes | 0 | NO PAD |

| utf8mb4\_lithuanian\_ci | utf8mb4 | 236 | | Yes | 8 | PAD SPACE |

| utf8mb4\_lt\_0900\_ai\_ci | utf8mb4 | 268 | | Yes | 0 | NO PAD |

| utf8mb4\_lt\_0900\_as\_cs | utf8mb4 | 291 | | Yes | 0 | NO PAD |

| utf8mb4\_lv\_0900\_ai\_ci | utf8mb4 | 258 | | Yes | 0 | NO PAD |

| utf8mb4\_lv\_0900\_as\_cs | utf8mb4 | 281 | | Yes | 0 | NO PAD |

| utf8mb4\_persian\_ci | utf8mb4 | 240 | | Yes | 8 | PAD SPACE |

| utf8mb4\_pl\_0900\_ai\_ci | utf8mb4 | 261 | | Yes | 0 | NO PAD |

| utf8mb4\_pl\_0900\_as\_cs | utf8mb4 | 284 | | Yes | 0 | NO PAD |

| utf8mb4\_polish\_ci | utf8mb4 | 229 | | Yes | 8 | PAD SPACE |

| utf8mb4\_romanian\_ci | utf8mb4 | 227 | | Yes | 8 | PAD SPACE |

| utf8mb4\_roman\_ci | utf8mb4 | 239 | | Yes | 8 | PAD SPACE |

| utf8mb4\_ro\_0900\_ai\_ci | utf8mb4 | 259 | | Yes | 0 | NO PAD |

| utf8mb4\_ro\_0900\_as\_cs | utf8mb4 | 282 | | Yes | 0 | NO PAD |

| utf8mb4\_ru\_0900\_ai\_ci | utf8mb4 | 306 | | Yes | 0 | NO PAD |

| utf8mb4\_ru\_0900\_as\_cs | utf8mb4 | 307 | | Yes | 0 | NO PAD |

| utf8mb4\_sinhala\_ci | utf8mb4 | 243 | | Yes | 8 | PAD SPACE |

| utf8mb4\_sk\_0900\_ai\_ci | utf8mb4 | 269 | | Yes | 0 | NO PAD |

| utf8mb4\_sk\_0900\_as\_cs | utf8mb4 | 292 | | Yes | 0 | NO PAD |

| utf8mb4\_slovak\_ci | utf8mb4 | 237 | | Yes | 8 | PAD SPACE |

| utf8mb4\_slovenian\_ci | utf8mb4 | 228 | | Yes | 8 | PAD SPACE |

| utf8mb4\_sl\_0900\_ai\_ci | utf8mb4 | 260 | | Yes | 0 | NO PAD |

| utf8mb4\_sl\_0900\_as\_cs | utf8mb4 | 283 | | Yes | 0 | NO PAD |

| utf8mb4\_spanish2\_ci | utf8mb4 | 238 | | Yes | 8 | PAD SPACE |

| utf8mb4\_spanish\_ci | utf8mb4 | 231 | | Yes | 8 | PAD SPACE |

| utf8mb4\_sv\_0900\_ai\_ci | utf8mb4 | 264 | | Yes | 0 | NO PAD |

| utf8mb4\_sv\_0900\_as\_cs | utf8mb4 | 287 | | Yes | 0 | NO PAD |

| utf8mb4\_swedish\_ci | utf8mb4 | 232 | | Yes | 8 | PAD SPACE |

| utf8mb4\_tr\_0900\_ai\_ci | utf8mb4 | 265 | | Yes | 0 | NO PAD |

| utf8mb4\_tr\_0900\_as\_cs | utf8mb4 | 288 | | Yes | 0 | NO PAD |

| utf8mb4\_turkish\_ci | utf8mb4 | 233 | | Yes | 8 | PAD SPACE |

| utf8mb4\_unicode\_520\_ci | utf8mb4 | 246 | | Yes | 8 | PAD SPACE |

| utf8mb4\_unicode\_ci | utf8mb4 | 224 | | Yes | 8 | PAD SPACE |

| utf8mb4\_vietnamese\_ci | utf8mb4 | 247 | | Yes | 8 | PAD SPACE |

| utf8mb4\_vi\_0900\_ai\_ci | utf8mb4 | 277 | | Yes | 0 | NO PAD |

| utf8mb4\_vi\_0900\_as\_cs | utf8mb4 | 300 | | Yes | 0 | NO PAD |

| utf8mb4\_zh\_0900\_as\_cs | utf8mb4 | 308 | | Yes | 0 | NO PAD |

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

For more information about those collations, see [Section 10.10.1, “Unicode Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets).

Collations have these general characteristics:

Two different character sets cannot have the same collation.

Each character set has a default collation. For example, the default collations for **utf8mb4** and **latin1** are **utf8mb4\_0900\_ai\_ci** and **latin1\_swedish\_ci**, respectively. The **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table and the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement indicate the default collation for each character set. The **INFORMATION\_SCHEMA** [**COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table and the [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) statement have a column that indicates for each collation whether it is the default for its character set (**Yes** if so, empty if not).

Collation names start with the name of the character set with which they are associated, generally followed by one or more suffixes indicating other collation characteristics. For additional information about naming conventions, see [Section 10.3.1, “Collation Naming Conventions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-names).

When a character set has multiple collations, it might not be clear which collation is most suitable for a given application. To avoid choosing an inappropriate collation, perform some comparisons with representative data values to make sure that a given collation sorts values the way you expect.

### 10.2.1 Character Set Repertoire

The repertoire of a character set is the collection of characters in the set.

String expressions have a repertoire attribute, which can have two values:

**ASCII**: The expression can contain only ASCII characters; that is, characters in the Unicode range **U+0000** to **U+007F**.

**UNICODE**: The expression can contain characters in the Unicode range **U+0000** to **U+10FFFF**. This includes characters in the Basic Multilingual Plane (BMP) range (**U+0000** to **U+FFFF**) and supplementary characters outside the BMP range (**U+10000** to **U+10FFFF**).

The **ASCII** range is a subset of **UNICODE** range, so a string with **ASCII** repertoire can be converted safely without loss of information to the character set of any string with **UNICODE** repertoire. It can also be converted safely to any character set that is a superset of the **ascii** character set. (All MySQL character sets are supersets of **ascii** with the exception of **swe7**, which reuses some punctuation characters for Swedish accented characters.)

The use of repertoire enables character set conversion in expressions for many cases where MySQL would otherwise return an “illegal mix of collations” error when the rules for collation coercibility are insufficient to resolve ambiguities. (For information about coercibility, see [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility).)

The following discussion provides examples of expressions and their repertoires, and describes how the use of repertoire changes string expression evaluation:

The repertoire for a string constant depends on string content and may differ from the repertoire of the string character set. Consider these statements:

SET NAMES utf8mb4; SELECT 'abc';

SELECT \_utf8mb4'def';

Although the character set is **utf8mb4** in each of the preceding cases, the strings do not actually contain any characters outside the ASCII range, so their repertoire is **ASCII** rather than **UNICODE**.

A column having the **ascii** character set has **ASCII** repertoire because of its character set. In the following table, **c1** has **ASCII** repertoire:

CREATE TABLE t1 (c1 CHAR(1) CHARACTER SET ascii);

The following example illustrates how repertoire enables a result to be determined in a case where an error occurs without repertoire:

CREATE TABLE t1 (

c1 CHAR(1) CHARACTER SET latin1,

c2 CHAR(1) CHARACTER SET ascii

);

INSERT INTO t1 VALUES ('a','b');

SELECT CONCAT(c1,c2) FROM t1;

Without repertoire, this error occurs:

ERROR 1267 (HY000): Illegal mix of collations (latin1\_swedish\_ci,IMPLICIT)

and (ascii\_general\_ci,IMPLICIT) for operation 'concat'

Using repertoire, subset to superset (**ascii** to **latin1**) conversion can occur and a result is returned:

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

| CONCAT(c1,c2) |

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

| ab |

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

Functions with one string argument inherit the repertoire of their argument. The result of [**UPPER(\_utf8mb4'abc')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) has **ASCII** repertoire because its argument has **ASCII** repertoire. (Despite the **\_utf8mb4** introducer, the string **'abc'** contains no characters outside the ASCII range.)

For functions that return a string but do not have string arguments and use [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) as the result character set, the result repertoire is **ASCII** if [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) is **ascii**, and **UNICODE** otherwise:

FORMAT(***numeric\_column***, 4);

Use of repertoire changes how MySQL evaluates the following example:

SET NAMES ascii;

CREATE TABLE t1 (a INT, b VARCHAR(10) CHARACTER SET latin1);

INSERT INTO t1 VALUES (1,'b');

SELECT CONCAT(FORMAT(a, 4), b) FROM t1;

Without repertoire, this error occurs:

ERROR 1267 (HY000): Illegal mix of collations (ascii\_general\_ci,COERCIBLE)

and (latin1\_swedish\_ci,IMPLICIT) for operation 'concat'

With repertoire, a result is returned:

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

| CONCAT(FORMAT(a, 4), b) |

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

| 1.0000b |

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

Functions with two or more string arguments use the “widest” argument repertoire for the result repertoire, where **UNICODE** is wider than **ASCII**. Consider the following [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) calls:

CONCAT(\_ucs2 X'0041', \_ucs2 X'0042')

CONCAT(\_ucs2 X'0041', \_ucs2 X'00C2')

For the first call, the repertoire is **ASCII** because both arguments are within the ASCII range. For the second call, the repertoire is **UNICODE** because the second argument is outside the ASCII range.

The repertoire for function return values is determined based on the repertoire of only those arguments that affect the result's character set and collation.

IF(column1 < column2, 'smaller', 'greater')

The result repertoire is **ASCII** because the two string arguments (the second argument and the third argument) both have **ASCII** repertoire. The first argument does not matter for the result repertoire, even if the expression uses string values.

### 10.2.2 UTF-8 for Metadata

Metadata is “the data about the data.” Anything that describes the database—as opposed to being the contents of the database—is metadata. Thus column names, database names, user names, version names, and most of the string results from [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) are metadata. This is also true of the contents of tables in **INFORMATION\_SCHEMA** because those tables by definition contain information about database objects.

Representation of metadata must satisfy these requirements:

All metadata must be in the same character set. Otherwise, neither the [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statements nor [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements for tables in **INFORMATION\_SCHEMA** would work properly because different rows in the same column of the results of these operations would be in different character sets.

Metadata must include all characters in all languages. Otherwise, users would not be able to name columns and tables using their own languages.

To satisfy both requirements, MySQL stores metadata in a Unicode character set, namely UTF-8. This does not cause any disruption if you never use accented or non-Latin characters. But if you do, you should be aware that metadata is in UTF-8.

The metadata requirements mean that the return values of the [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user), [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), [**SESSION\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_session-user), [**SYSTEM\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_system-user), [**DATABASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database), and [**VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version) functions have the UTF-8 character set by default.

The server sets the [**character\_set\_system**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_system) system variable to the name of the metadata character set:

mysql> **SHOW VARIABLES LIKE 'character\_set\_system';**

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

| Variable\_name | Value |

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

| character\_set\_system | utf8 |

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

Storage of metadata using Unicode does not mean that the server returns headers of columns and the results of [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) functions in the [**character\_set\_system**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_system) character set by default. When you use **SELECT column1 FROM t**, the name **column1** itself is returned from the server to the client in the character set determined by the value of the [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) system variable, which has a default value of **utf8mb4**. If you want the server to pass metadata results back in a different character set, use the [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) statement to force the server to perform character set conversion. [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) sets the [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) and other related system variables. (See [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).) Alternatively, a client program can perform the conversion after receiving the result from the server. It is more efficient for the client to perform the conversion, but this option is not always available for all clients.

If [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) is set to **NULL**, no conversion is performed and the server returns metadata using its original character set (the set indicated by [**character\_set\_system**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_system)).

Error messages returned from the server to the client are converted to the client character set automatically, as with metadata.

If you are using (for example) the [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) function for comparison or assignment within a single statement, don't worry. MySQL performs some automatic conversion for you.

SELECT \* FROM t1 WHERE USER() = latin1\_column;

This works because the contents of **latin1\_column** are automatically converted to UTF-8 before the comparison.

INSERT INTO t1 (latin1\_column) SELECT USER();

This works because the contents of [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) are automatically converted to **latin1** before the assignment.

Although automatic conversion is not in the SQL standard, the standard does say that every character set is (in terms of supported characters) a “subset” of Unicode. Because it is a well-known principle that “what applies to a superset can apply to a subset,” we believe that a collation for Unicode can apply for comparisons with non-Unicode strings. For more information about coercion of strings, see [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility).

## 10.3 Specifying Character Sets and Collations

[10.3.1 Collation Naming Conventions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-names)

[10.3.2 Server Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-server)

[10.3.3 Database Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-database)

[10.3.4 Table Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-table)

[10.3.5 Column Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-column)

[10.3.6 Character String Literal Character Set and Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-literal)

[10.3.7 The National Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-national)

[10.3.8 Character Set Introducers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer)

[10.3.9 Examples of Character Set and Collation Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-examples)

[10.3.10 Compatibility with Other DBMSs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-compatibility)

There are default settings for character sets and collations at four levels: server, database, table, and column. The description in the following sections may appear complex, but it has been found in practice that multiple-level defaulting leads to natural and obvious results.

**CHARACTER SET** is used in clauses that specify a character set. **CHARSET** can be used as a synonym for **CHARACTER SET**.

Character set issues affect not only data storage, but also communication between client programs and the MySQL server. If you want the client program to communicate with the server using a character set different from the default, you'll need to indicate which one. For example, to use the **utf8mb4** Unicode character set, issue this statement after connecting to the server:

SET NAMES 'utf8mb4';

For more information about character set-related issues in client/server communication, see [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).

### 10.3.1 Collation Naming Conventions

MySQL collation names follow these conventions:

A collation name starts with the name of the character set with which it is associated, generally followed by one or more suffixes indicating other collation characteristics. For example, **utf8mb4\_0900\_ai\_ci** and **latin1\_swedish\_ci** are collations for the **utf8mb4** and **latin1** character sets, respectively. The **binary** character set has a single collation, also named **binary**, with no suffixes.

A language-specific collation includes a locale code or language name. For example, **utf8mb4\_tr\_0900\_ai\_ci** and **utf8mb4\_hu\_0900\_ai\_ci** sort characters for the **utf8mb4** character set using the rules of Turkish and Hungarian, respectively. **utf8mb4\_turkish\_ci** and **utf8mb4\_hungarian\_ci** are similar but based on a less recent version of the Unicode Collation Algorithm.

Collation suffixes indicate whether a collation is case-sensitive, accent-sensitive, or kana-sensitive (or some combination thereof), or binary. The following table shows the suffixes used to indicate these characteristics.

**Table 10.1 Collation Suffix Meanings**

| **Suffix** | **Meaning** |
| --- | --- |
| **\_ai** | Accent-insensitive |
| **\_as** | Accent-sensitive |
| **\_ci** | Case-insensitive |
| **\_cs** | Case-sensitive |
| **\_ks** | Kana-sensitive |
| **\_bin** | Binary |

For nonbinary collation names that do not specify accent sensitivity, it is determined by case sensitivity. If a collation name does not contain **\_ai** or **\_as**, **\_ci** in the name implies **\_ai** and **\_cs** in the name implies **\_as**. For example, **latin1\_general\_ci** is explicitly case-insensitive and implicitly accent-insensitive, **latin1\_general\_cs** is explicitly case-sensitive and implicitly accent-sensitive, and **utf8mb4\_0900\_ai\_ci** is explicitly case-insensitive and accent-insensitive.

For Japanese collations, the **\_ks** suffix indicates that a collation is kana-sensitive; that is, it distinguishes Katakana characters from Hiragana characters. Japanese collations without the **\_ks** suffix are not kana-sensitive and treat Katakana and Hiragana characters equal for sorting.

For the **binary** collation of the **binary** character set, comparisons are based on numeric byte values. For the **\_bin** collation of a nonbinary character set, comparisons are based on numeric character code values, which differ from byte values for multibyte characters. For information about the differences between the **binary** collation of the **binary** character set and the **\_bin** collations of nonbinary character sets, see [Section 10.8.5, “The binary Collation Compared to \_bin Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations).

Collation names for Unicode character sets may include a version number to indicate the version of the Unicode Collation Algorithm (UCA) on which the collation is based. UCA-based collations without a version number in the name use the version-4.0.0 UCA weight keys. For example:

**utf8mb4\_0900\_ai\_ci** is based on UCA 9.0.0 weight keys (<http://www.unicode.org/Public/UCA/9.0.0/allkeys.txt>).

**utf8mb4\_unicode\_520\_ci** is based on UCA 5.2.0 weight keys (<http://www.unicode.org/Public/UCA/5.2.0/allkeys.txt>).

**utf8mb4\_unicode\_ci** (with no version named) is based on UCA 4.0.0 weight keys (<http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt>).

For Unicode character sets, the ***xxx*\_general\_mysql500\_ci** collations preserve the pre-5.1.24 ordering of the original ***xxx*\_general\_ci** collations and permit upgrades for tables created before MySQL 5.1.24 (Bug #27877).

### 10.3.2 Server Character Set and Collation

MySQL Server has a server character set and a server collation. By default, these are **utf8mb4** and **utf8mb4\_0900\_ai\_ci**, but they can be set explicitly at server startup on the command line or in an option file and changed at runtime.

Initially, the server character set and collation depend on the options that you use when you start [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld). You can use [--character-set-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) for the character set. Along with it, you can add [--collation-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) for the collation. If you don't specify a character set, that is the same as saying [--character-set-server=utf8mb4](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server). If you specify only a character set (for example, **utf8mb4**) but not a collation, that is the same as saying [--character-set-server=utf8mb4](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) [--collation-server=utf8mb4\_0900\_ai\_ci](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) because **utf8mb4\_0900\_ai\_ci** is the default collation for **utf8mb4**. Therefore, the following three commands all have the same effect:

mysqld

mysqld --character-set-server=utf8mb4

mysqld --character-set-server=utf8mb4 \

--collation-server=utf8mb4\_0900\_ai\_ci

One way to change the settings is by recompiling. To change the default server character set and collation when building from sources, use the [DEFAULT\_CHARSET](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_default_charset) and [DEFAULT\_COLLATION](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_default_collation) options for **CMake**. For example:

cmake . -DDEFAULT\_CHARSET=latin1

Or:

cmake . -DDEFAULT\_CHARSET=latin1 \

-DDEFAULT\_COLLATION=latin1\_german1\_ci

Both [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) and **CMake** verify that the character set/collation combination is valid. If not, each program displays an error message and terminates.

The server character set and collation are used as default values if the database character set and collation are not specified in [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statements. They have no other purpose.

The current server character set and collation can be determined from the values of the [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) system variables. These variables can be changed at runtime.

### 10.3.3 Database Character Set and Collation

Every database has a database character set and a database collation. The [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) and [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) statements have optional clauses for specifying the database character set and collation:

CREATE DATABASE ***db\_name***

[[DEFAULT] CHARACTER SET ***charset\_name***]

[[DEFAULT] COLLATE ***collation\_name***]

ALTER DATABASE ***db\_name***

[[DEFAULT] CHARACTER SET ***charset\_name***]

[[DEFAULT] COLLATE ***collation\_name***]

The keyword **SCHEMA** can be used instead of **DATABASE**.

The **CHARACTER SET** and **COLLATE** clauses make it possible to create databases with different character sets and collations on the same MySQL server.

Database options are stored in the data dictionary and can be examined by checking the [**INFORMATION\_SCHEMA.SCHEMATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-table) table.

Example:

CREATE DATABASE ***db\_name*** CHARACTER SET latin1 COLLATE latin1\_swedish\_ci;

MySQL chooses the database character set and database collation in the following manner:

If both **CHARACTER SET *charset\_name*** and **COLLATE *collation\_name*** are specified, character set ***charset\_name*** and collation ***collation\_name*** are used.

If **CHARACTER SET *charset\_name*** is specified without **COLLATE**, character set ***charset\_name*** and its default collation are used. To see the default collation for each character set, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement or query the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table.

If **COLLATE *collation\_name*** is specified without **CHARACTER SET**, the character set associated with ***collation\_name*** and collation ***collation\_name*** are used.

Otherwise (neither **CHARACTER SET** nor **COLLATE** is specified), the server character set and server collation are used.

The character set and collation for the default database can be determined from the values of the [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) and [**collation\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_database) system variables. The server sets these variables whenever the default database changes. If there is no default database, the variables have the same value as the corresponding server-level system variables, [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server).

To see the default character set and collation for a given database, use these statements:

USE ***db\_name***;

SELECT @@character\_set\_database, @@collation\_database;

Alternatively, to display the values without changing the default database:

SELECT DEFAULT\_CHARACTER\_SET\_NAME, DEFAULT\_COLLATION\_NAME

FROM INFORMATION\_SCHEMA.SCHEMATA WHERE SCHEMA\_NAME = '***db\_name***';

The database character set and collation affect these aspects of server operation:

For [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements, the database character set and collation are used as default values for table definitions if the table character set and collation are not specified. To override this, provide explicit **CHARACTER SET** and **COLLATE** table options.

For [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements that include no **CHARACTER SET** clause, the server uses the character set indicated by the [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) system variable to interpret the information in the file. To override this, provide an explicit **CHARACTER SET** clause.

For stored routines (procedures and functions), the database character set and collation in effect at routine creation time are used as the character set and collation of character data parameters for which the declaration includes no **CHARACTER SET** or a **COLLATE** attribute. To override this, provide **CHARACTER SET** and **COLLATE** explicitly.

### 10.3.4 Table Character Set and Collation

Every table has a table character set and a table collation. The [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements have optional clauses for specifying the table character set and collation:

CREATE TABLE ***tbl\_name*** (***column\_list***)

[[DEFAULT] CHARACTER SET ***charset\_name***]

[COLLATE ***collation\_name***]]

ALTER TABLE ***tbl\_name***

[[DEFAULT] CHARACTER SET ***charset\_name***]

[COLLATE ***collation\_name***]

Example:

CREATE TABLE t1 ( ... )

CHARACTER SET latin1 COLLATE latin1\_danish\_ci;

MySQL chooses the table character set and collation in the following manner:

If both **CHARACTER SET *charset\_name*** and **COLLATE *collation\_name*** are specified, character set ***charset\_name*** and collation ***collation\_name*** are used.

If **CHARACTER SET *charset\_name*** is specified without **COLLATE**, character set ***charset\_name*** and its default collation are used. To see the default collation for each character set, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement or query the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table.

If **COLLATE *collation\_name*** is specified without **CHARACTER SET**, the character set associated with ***collation\_name*** and collation ***collation\_name*** are used.

Otherwise (neither **CHARACTER SET** nor **COLLATE** is specified), the database character set and collation are used.

The table character set and collation are used as default values for column definitions if the column character set and collation are not specified in individual column definitions. The table character set and collation are MySQL extensions; there are no such things in standard SQL.

### 10.3.5 Column Character Set and Collation

Every “character” column (that is, a column of type [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), a [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) type, or any synonym) has a column character set and a column collation. Column definition syntax for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) has optional clauses for specifying the column character set and collation:

***col\_name*** {CHAR | VARCHAR | TEXT} (***col\_length***)

[CHARACTER SET ***charset\_name***]

[COLLATE ***collation\_name***]

These clauses can also be used for [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) columns:

***col\_name*** {ENUM | SET} (***val\_list***)

[CHARACTER SET ***charset\_name***]

[COLLATE ***collation\_name***]

Examples:

CREATE TABLE t1

(

col1 VARCHAR(5)

CHARACTER SET latin1

COLLATE latin1\_german1\_ci

);

ALTER TABLE t1 MODIFY

col1 VARCHAR(5)

CHARACTER SET latin1

COLLATE latin1\_swedish\_ci;

MySQL chooses the column character set and collation in the following manner:

If both **CHARACTER SET *charset\_name*** and **COLLATE *collation\_name*** are specified, character set ***charset\_name*** and collation ***collation\_name*** are used.

CREATE TABLE t1

(

col1 CHAR(10) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci

) CHARACTER SET latin1 COLLATE latin1\_bin;

The character set and collation are specified for the column, so they are used. The column has character set **utf8** and collation **utf8\_unicode\_ci**.

If **CHARACTER SET *charset\_name*** is specified without **COLLATE**, character set ***charset\_name*** and its default collation are used.

CREATE TABLE t1

(

col1 CHAR(10) CHARACTER SET utf8

) CHARACTER SET latin1 COLLATE latin1\_bin;

The character set is specified for the column, but the collation is not. The column has character set **utf8** and the default collation for **utf8**, which is **utf8\_general\_ci**. To see the default collation for each character set, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement or query the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table.

If **COLLATE *collation\_name*** is specified without **CHARACTER SET**, the character set associated with ***collation\_name*** and collation ***collation\_name*** are used.

CREATE TABLE t1

(

col1 CHAR(10) COLLATE utf8\_polish\_ci

) CHARACTER SET latin1 COLLATE latin1\_bin;

The collation is specified for the column, but the character set is not. The column has collation **utf8\_polish\_ci** and the character set is the one associated with the collation, which is **utf8**.

Otherwise (neither **CHARACTER SET** nor **COLLATE** is specified), the table character set and collation are used.

CREATE TABLE t1

(

col1 CHAR(10)

) CHARACTER SET latin1 COLLATE latin1\_bin;

Neither the character set nor collation is specified for the column, so the table defaults are used. The column has character set **latin1** and collation **latin1\_bin**.

The **CHARACTER SET** and **COLLATE** clauses are standard SQL.

If you use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to convert a column from one character set to another, MySQL attempts to map the data values, but if the character sets are incompatible, there may be data loss.

### 10.3.6 Character String Literal Character Set and Collation

Every character string literal has a character set and a collation.

For the simple statement **SELECT '*string*'**, the string has the connection default character set and collation defined by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables.

A character string literal may have an optional character set introducer and **COLLATE** clause, to designate it as a string that uses a particular character set and collation:

[\_***charset\_name***]'***string***' [COLLATE ***collation\_name***]

The **\_*charset\_name*** expression is formally called an introducer. It tells the parser, “the string that follows uses character set ***charset\_name***.” An introducer does not change the string to the introducer character set like [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) would do. It does not change the string value, although padding may occur. The introducer is just a signal. See [Section 10.3.8, “Character Set Introducers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer).

Examples:

SELECT 'abc';

SELECT \_latin1'abc';

SELECT \_binary'abc';

SELECT \_utf8mb4'abc' COLLATE utf8mb4\_danish\_ci;

Character set introducers and the **COLLATE** clause are implemented according to standard SQL specifications.

MySQL determines the character set and collation of a character string literal in the following manner:

If both ***\_charset\_name*** and **COLLATE *collation\_name*** are specified, character set ***charset\_name*** and collation ***collation\_name*** are used. ***collation\_name*** must be a permitted collation for ***charset\_name***.

If ***\_charset\_name*** is specified but **COLLATE** is not specified, character set ***charset\_name*** and its default collation are used. To see the default collation for each character set, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement or query the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table.

If ***\_charset\_name*** is not specified but **COLLATE *collation\_name*** is specified, the connection default character set given by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) system variable and collation ***collation\_name*** are used. ***collation\_name*** must be a permitted collation for the connection default character set.

Otherwise (neither ***\_charset\_name*** nor **COLLATE *collation\_name*** is specified), the connection default character set and collation given by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables are used.

Examples:

A nonbinary string with **latin1** character set and **latin1\_german1\_ci** collation:

SELECT \_latin1'Müller' COLLATE latin1\_german1\_ci;

A nonbinary string with **utf8mb4** character set and its default collation (that is, **utf8mb4\_0900\_ai\_ci**):

SELECT \_utf8mb4'Müller';

A binary string with **binary** character set and its default collation (that is, **binary**):

SELECT \_binary'Müller';

A nonbinary string with the connection default character set and **utf8mb4\_0900\_ai\_ci** collation (fails if the connection character set is not **utf8mb4**):

SELECT 'Müller' COLLATE utf8mb4\_0900\_ai\_ci;

A string with the connection default character set and collation:

SELECT 'Müller';

An introducer indicates the character set for the following string, but does not change how the parser performs escape processing within the string. Escapes are always interpreted by the parser according to the character set given by [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection).

The following examples show that escape processing occurs using [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) even in the presence of an introducer. The examples use [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) (which changes [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection), as discussed in [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection)), and display the resulting strings using the [**HEX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_hex) function so that the exact string contents can be seen.

Example 1:

mysql> **SET NAMES latin1;**

mysql> **SELECT HEX('à\n'), HEX(\_sjis'à\n');**

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

| HEX('à\n') | HEX(\_sjis'à\n') |

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

| E00A | E00A |

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

Here, **à** (hexadecimal value **E0**) is followed by **\n**, the escape sequence for newline. The escape sequence is interpreted using the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) value of **latin1** to produce a literal newline (hexadecimal value **0A**). This happens even for the second string. That is, the **\_sjis** introducer does not affect the parser's escape processing.

Example 2:

mysql> **SET NAMES sjis;**

mysql> **SELECT HEX('à\n'), HEX(\_latin1'à\n');**

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

| HEX('à\n') | HEX(\_latin1'à\n') |

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

| E05C6E | E05C6E |

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

Here, [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) is **sjis**, a character set in which the sequence of **à** followed by **\** (hexadecimal values **05** and **5C**) is a valid multibyte character. Hence, the first two bytes of the string are interpreted as a single **sjis** character, and the **\** is not interpreted as an escape character. The following **n** (hexadecimal value **6E**) is not interpreted as part of an escape sequence. This is true even for the second string; the **\_latin1** introducer does not affect escape processing.

### 10.3.7 The National Character Set

Standard SQL defines [**NCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**NATIONAL CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) as a way to indicate that a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column should use some predefined character set. MySQL uses **utf8** as this predefined character set. For example, these data type declarations are equivalent:

CHAR(10) CHARACTER SET utf8

NATIONAL CHARACTER(10)

NCHAR(10)

As are these:

VARCHAR(10) CHARACTER SET utf8

NATIONAL VARCHAR(10)

NVARCHAR(10)

NCHAR VARCHAR(10)

NATIONAL CHARACTER VARYING(10)

NATIONAL CHAR VARYING(10)

You can use **N'*literal*'** (or **n'*literal*'**) to create a string in the national character set. These statements are equivalent:

SELECT N'some text';

SELECT n'some text';

SELECT \_utf8'some text';

### 10.3.8 Character Set Introducers

A character string literal, hexadecimal literal, or bit-value literal may have an optional character set introducer and **COLLATE** clause, to designate it as a string that uses a particular character set and collation:

[\_***charset\_name***] ***literal*** [COLLATE ***collation\_name***]

The **\_*charset\_name*** expression is formally called an introducer. It tells the parser, “the string that follows uses character set ***charset\_name***.” An introducer does not change the string to the introducer character set like [**CONVERT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert) would do. It does not change the string value, although padding may occur. The introducer is just a signal.

For character string literals, space between the introducer and the string is permitted but optional.

For character set literals, an introducer indicates the character set for the following string, but does not change how the parser performs escape processing within the string. Escapes are always interpreted by the parser according to the character set given by [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection). For additional discussion and examples, see [Section 10.3.6, “Character String Literal Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-literal).

Examples:

SELECT 'abc';

SELECT \_latin1'abc';

SELECT \_binary'abc';

SELECT \_utf8mb4'abc' COLLATE utf8mb4\_danish\_ci;

SELECT \_latin1 X'4D7953514C';

SELECT \_utf8mb4 0x4D7953514C COLLATE utf8mb4\_danish\_ci;

SELECT \_latin1 b'1000001';

SELECT \_utf8mb4 0b1000001 COLLATE utf8mb4\_danish\_ci;

Character set introducers and the **COLLATE** clause are implemented according to standard SQL specifications.

Character string literals can be designated as binary strings by using the **\_binary** introducer. Hexadecimal literals and bit-value literals are binary strings by default, so **\_binary** is permitted, but normally unnecessary. **\_binary** may be useful to preserve a hexadecimal or bit literal as a binary string in contexts for which the literal is otherwise treated as a number. For example, bit operations permit numeric or binary string arguments in MySQL 8.0 and higher, but treat hexadecimal and bit literals as numbers by default. To explicitly specify binary string context for such literals, use a **\_binary** introducer for at least one of the arguments:

mysql> **SET @v1 = X'000D' | X'0BC0';**

mysql> **SET @v2 = \_binary X'000D' | X'0BC0';**

mysql> **SELECT HEX(@v1), HEX(@v2);**

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

| HEX(@v1) | HEX(@v2) |

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

| BCD | 0BCD |

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

The displayed result appears similar for both bit operations, but the result without **\_binary** is a **BIGINT** value, whereas the result with **\_binary** is a binary string. Due to the difference in result types, the displayed values differ: High-order 0 digits are not displayed for the numeric result.

MySQL determines the character set and collation of a character string literal, hexadecimal literal, or bit-value literal in the following manner:

If both ***\_charset\_name*** and **COLLATE *collation\_name*** are specified, character set ***charset\_name*** and collation ***collation\_name*** are used. ***collation\_name*** must be a permitted collation for ***charset\_name***.

If ***\_charset\_name*** is specified but **COLLATE** is not specified, character set ***charset\_name*** and its default collation are used. To see the default collation for each character set, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement or query the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table.

If ***\_charset\_name*** is not specified but **COLLATE *collation\_name*** is specified:

For a character string literal, the connection default character set given by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) system variable and collation ***collation\_name*** are used. ***collation\_name*** must be a permitted collation for the connection default character set.

For a hexadecimal literal or bit-value literal, the only permitted collation is **binary** because these types of literals are binary strings by default.

Otherwise (neither ***\_charset\_name*** nor **COLLATE *collation\_name*** is specified):

For a character string literal, the connection default character set and collation given by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables are used.

For a hexadecimal literal or bit-value literal, the character set and collation are **binary**.

Examples:

Nonbinary strings with **latin1** character set and **latin1\_german1\_ci** collation:

SELECT \_latin1'Müller' COLLATE latin1\_german1\_ci;

SELECT \_latin1 X'0A0D' COLLATE latin1\_german1\_ci;

SELECT \_latin1 b'0110' COLLATE latin1\_german1\_ci;

Nonbinary strings with **utf8mb4** character set and its default collation (that is, **utf8mb4\_0900\_ai\_ci**):

SELECT \_utf8mb4'Müller';

SELECT \_utf8mb4 X'0A0D';

SELECT \_utf8mb4 b'0110';

Binary strings with **binary** character set and its default collation (that is, **binary**):

SELECT \_binary'Müller';

SELECT X'0A0D';

SELECT b'0110';

The hexadecimal literal and bit-value literal need no introducer because they are binary strings by default.

A nonbinary string with the connection default character set and **utf8mb4\_0900\_ai\_ci** collation (fails if the connection character set is not **utf8mb4**):

SELECT 'Müller' COLLATE utf8mb4\_0900\_ai\_ci;

This construction (**COLLATE** only) does not work for hexadecimal literals or bit literals because their character set is **binary** no matter the connection character set, and **binary** is not compatible with the **utf8mb4\_0900\_ai\_ci** collation. The only permitted **COLLATE** clause in the absence of an introducer is **COLLATE binary**.

A string with the connection default character set and collation:

SELECT 'Müller';

### 10.3.9 Examples of Character Set and Collation Assignment

The following examples show how MySQL determines default character set and collation values.

***Example 1: Table and Column Definition***

CREATE TABLE t1

(

c1 CHAR(10) CHARACTER SET latin1 COLLATE latin1\_german1\_ci

) DEFAULT CHARACTER SET latin2 COLLATE latin2\_bin;

Here we have a column with a **latin1** character set and a **latin1\_german1\_ci** collation. The definition is explicit, so that is straightforward. Notice that there is no problem with storing a **latin1** column in a **latin2** table.

***Example 2: Table and Column Definition***

CREATE TABLE t1

(

c1 CHAR(10) CHARACTER SET latin1

) DEFAULT CHARACTER SET latin1 COLLATE latin1\_danish\_ci;

This time we have a column with a **latin1** character set and a default collation. Although it might seem natural, the default collation is not taken from the table level. Instead, because the default collation for **latin1** is always **latin1\_swedish\_ci**, column **c1** has a collation of **latin1\_swedish\_ci** (not **latin1\_danish\_ci**).

***Example 3: Table and Column Definition***

CREATE TABLE t1

(

c1 CHAR(10)

) DEFAULT CHARACTER SET latin1 COLLATE latin1\_danish\_ci;

We have a column with a default character set and a default collation. In this circumstance, MySQL checks the table level to determine the column character set and collation. Consequently, the character set for column **c1** is **latin1** and its collation is **latin1\_danish\_ci**.

***Example 4: Database, Table, and Column Definition***

CREATE DATABASE d1

DEFAULT CHARACTER SET latin2 COLLATE latin2\_czech\_ci;

USE d1;

CREATE TABLE t1

(

c1 CHAR(10)

);

We create a column without specifying its character set and collation. We're also not specifying a character set and a collation at the table level. In this circumstance, MySQL checks the database level to determine the table settings, which thereafter become the column settings.) Consequently, the character set for column **c1** is **latin2** and its collation is **latin2\_czech\_ci**.

### 10.3.10 Compatibility with Other DBMSs

For MaxDB compatibility these two statements are the same:

CREATE TABLE t1 (f1 CHAR(***N***) UNICODE);

CREATE TABLE t1 (f1 CHAR(***N***) CHARACTER SET ucs2);

## 10.4 Connection Character Sets and Collations

A “connection” is what a client program makes when it connects to the server, to begin a session within which it interacts with the server. The client sends SQL statements, such as queries, over the session connection. The server sends responses, such as result sets or error messages, over the connection back to the client.

[Connection Character Set and Collation System Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-system-variables)

[Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset)

[Client Program Connection Character Set Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-client-configuration)

[SQL Statements for Connection Character Set Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-sql-statements)

[Connection Character Set Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-error-handling)

### Connection Character Set and Collation System Variables

Several character set and collation system variables relate to a client's interaction with the server. Some of these have been mentioned in earlier sections:

The [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) system variables indicate the server character set and collation. See [Section 10.3.2, “Server Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-server).

The [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) and [**collation\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_database) system variables indicate the character set and collation of the default database. See [Section 10.3.3, “Database Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-database).

Additional character set and collation system variables are involved in handling traffic for the connection between a client and the server. Every client has session-specific connection-related character set and collation system variables. These session system variable values are initialized at connect time, but can be changed within the session.

Several questions about character set and collation handling for client connections can be answered in terms of system variables:

What character set are statements in when they leave the client?

The server takes the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable to be the character set in which statements are sent by the client.

**Note**

Some character sets cannot be used as the client character set. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

What character set should the server translate statements to after receiving them?

To determine this, the server uses the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables:

The server converts statements sent by the client from [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) to [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection). Exception: For string literals that have an introducer such as **\_utf8mb4** or **\_latin2**, the introducer determines the character set. See [Section 10.3.8, “Character Set Introducers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer).

[**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) is important for comparisons of literal strings. For comparisons of strings with column values, [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) does not matter because columns have their own collation, which has a higher collation precedence (see [Section 10.8.4, “Collation Coercibility in Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility)).

What character set should the server translate query results to before shipping them back to the client?

The [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) system variable indicates the character set in which the server returns query results to the client. This includes result data such as column values, result metadata such as column names, and error messages.

To tell the server to perform no conversion of result sets or error messages, set [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) to **NULL** or **binary**:

SET character\_set\_results = NULL;

SET character\_set\_results = binary;

For more information about character sets and error messages, see [Section 10.6, “Error Message Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-errors).

To see the values of the character set and collation system variables that apply to the current session, use this statement:

SELECT \* FROM performance\_schema.session\_variables

WHERE VARIABLE\_NAME IN (

'character\_set\_client', 'character\_set\_connection',

'character\_set\_results', 'collation\_connection'

) ORDER BY VARIABLE\_NAME;

The following simpler statements also display the connection variables, but include other related variables as well. They can be useful to see all character set and collation system variables:

SHOW SESSION VARIABLES LIKE 'character\\_set\\_%';

SHOW SESSION VARIABLES LIKE 'collation\\_%';

Clients can fine-tune the settings for these variables, or depend on the defaults (in which case, you can skip the rest of this section). If you do not use the defaults, you must change the character settings for each connection to the server.

### Impermissible Client Character Sets

The [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable cannot be set to certain character sets:

ucs2

utf16

utf16le

utf32

Attempting to use any of those character sets as the client character set produces an error:

mysql> **SET character\_set\_client = 'ucs2';**

ERROR 1231 (42000): Variable 'character\_set\_client'

can't be set to the value of 'ucs2'

The same error occurs if any of those character sets are used in the following contexts, all of which result in an attempt to set [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) to the named character set:

The [--default-character-set=***charset\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) command option used by MySQL client programs such as [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) and [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin).

The [**SET NAMES '*charset\_name*'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) statement.

The [**SET CHARACTER SET '*charset\_name*'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) statement.

### Client Program Connection Character Set Configuration

When a client connects to the server, it indicates which character set it wants to use for communication with the server. (Actually, the client indicates the default collation for that character set, from which the server can determine the character set.) The server uses this information to set the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client), [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results), [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) system variables to the character set, and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) to the character set default collation. In effect, the server performs the equivalent of a [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) operation.

If the server does not support the requested character set or collation, it falls back to using the server character set and collation to configure the connection. For additional detail about this fallback behavior, see [Connection Character Set Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-error-handling).

The [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql), [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin), [**mysqlcheck**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlcheck), [**mysqlimport**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlimport), and [**mysqlshow**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow) client programs determine the default character set to use as follows:

In the absence of other information, each client uses the compiled-in default character set, usually **utf8mb4**.

Each client can autodetect which character set to use based on the operating system setting, such as the value of the **LANG** or **LC\_ALL** locale environment variable on Unix systems or the code page setting on Windows systems. For systems on which the locale is available from the OS, the client uses it to set the default character set rather than using the compiled-in default. For example, setting **LANG** to **ru\_RU.KOI8-R** causes the **koi8r** character set to be used. Thus, users can configure the locale in their environment for use by MySQL clients.

The OS character set is mapped to the closest MySQL character set if there is no exact match. If the client does not support the matching character set, it uses the compiled-in default. For example, **utf8** and **utf-8** map to **utf8mb4**, and **ucs2** is not supported as a connection character set, so it maps to the compiled-in default.

C applications can use character set autodetection based on the OS setting by invoking [**mysql\_options()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) as follows before connecting to the server:

mysql\_options(mysql,

MYSQL\_SET\_CHARSET\_NAME,

MYSQL\_AUTODETECT\_CHARSET\_NAME);

Each client supports a [--default-character-set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) option, which enables users to specify the character set explicitly to override whatever default the client otherwise determines.

**Note**

Some character sets cannot be used as the client character set. Attempting to use them with [--default-character-set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) produces an error. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

With the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, to use a character set different from the default, you could explicitly execute a [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) statement every time you connect to the server (see [Client Program Connection Character Set Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-client-configuration)). To accomplish the same result more easily, specify the character set in your option file. For example, the following option file setting changes the three connection-related character set system variables set to **koi8r** each time you invoke [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql):

[mysql]

default-character-set=koi8r

If you are using the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client with auto-reconnect enabled (which is not recommended), it is preferable to use the **charset** command rather than [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names). For example:

mysql> **charset koi8r**

Charset changed

The **charset** command issues a [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) statement, and also changes the default character set that [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) uses when it reconnects after the connection has dropped.

When configuration client programs, you must also consider the environment within which they execute. See [Section 10.5, “Configuring Application Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-applications).

### SQL Statements for Connection Character Set Configuration

After a connection has been established, clients can change the character set and collation system variables for the current session. These variables can be changed individually using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-statement) statements, but two more convenient statements affect the connection-related character set sytem variables as a group:

**SET NAMES '*charset\_name*' [COLLATE '*collation\_name*']**

[**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) indicates what character set the client uses to send SQL statements to the server. Thus, [**SET NAMES 'cp1251'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) tells the server, “future incoming messages from this client are in character set **cp1251**.” It also specifies the character set that the server should use for sending results back to the client. (For example, it indicates what character set to use for column values if you use a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement that produces a result set.)

A [**SET NAMES '*charset\_name*'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) statement is equivalent to these three statements:

SET character\_set\_client = ***charset\_name***;

SET character\_set\_results = ***charset\_name***;

SET character\_set\_connection = ***charset\_name***;

Setting [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) to ***charset\_name*** also implicitly sets [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) to the default collation for ***charset\_name***. It is unnecessary to set that collation explicitly. To specify a particular collation to use for [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection), add a **COLLATE** clause:

SET NAMES '***charset\_name***' COLLATE '***collation\_name***'

**SET CHARACTER SET '*charset\_name***'

[**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) is similar to [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) but sets [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) to [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) and [**collation\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_database) (which, as mentioned previously, indicate the character set and collation of the default database).

A [**SET CHARACTER SET *charset\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) statement is equivalent to these three statements:

SET character\_set\_client = ***charset\_name***;

SET character\_set\_results = ***charset\_name***;

SET collation\_connection = @@collation\_database;

Setting [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) also implicitly sets [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) to the character set associated with the collation (equivalent to executing **SET character\_set\_connection = @@character\_set\_database**). It is unnecessary to set [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) explicitly.

**Note**

Some character sets cannot be used as the client character set. Attempting to use them with [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) or [**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) produces an error. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

Example: Suppose that **column1** is defined as **CHAR(5) CHARACTER SET latin2**. If you do not say [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) or [**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set), then for **SELECT column1 FROM t**, the server sends back all the values for **column1** using the character set that the client specified when it connected. On the other hand, if you say **SET NAMES 'latin1'** or **SET CHARACTER SET 'latin1'** before issuing the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, the server converts the **latin2** values to **latin1** just before sending results back. Conversion may be lossy for characters that are not in both character sets.

### Connection Character Set Error Handling

Attempts to use an inappropriate connection character set or collation can produce an error, or cause the server to fall back to its default character set and collation for a given connection. This section describes problems that can occur when configuring the connection character set. These problems can occur when establishing a connection or when changing the character set within an established connection.

[Connect-Time Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-error-handling-connect-time)

[Runtime Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-error-handling-runtime)

#### Connect-Time Error Handling

Some character sets cannot be used as the client character set; see [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset). If you specify a character set that is valid but not permitted as a client character set, the server returns an error:

shell> **mysql --default-character-set=ucs2**

ERROR 1231 (42000): Variable 'character\_set\_client' can't be set to

the value of 'ucs2'

If you specify a character set that the client does not recognize, it produces an error:

shell> **mysql --default-character-set=bogus**

mysql: Character set 'bogus' is not a compiled character set and is

not specified in the '/usr/local/mysql/share/charsets/Index.xml' file

ERROR 2019 (HY000): Can't initialize character set bogus

(path: /usr/local/mysql/share/charsets/)

If you specify a character set that the client recognizes but the server does not, the server falls back to its default character set and collation. Suppose that the server is configured to use **latin1** and **latin1\_swedish\_ci** as its defaults, and that it does not recognize **gb18030** as a valid character set. A client that specifies [--default-character-set=gb18030](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) is able to connect to the server, but the resulting character set is not what the client wants:

mysql> **SHOW SESSION VARIABLES LIKE 'character\\_set\\_%';**

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

| Variable\_name | Value |

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

| character\_set\_client | latin1 |

| character\_set\_connection | latin1 |

...

| character\_set\_results | latin1 |

...

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

mysql> **SHOW SESSION VARIABLES LIKE 'collation\_connection';**

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

| Variable\_name | Value |

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

| collation\_connection | latin1\_swedish\_ci |

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

You can see that the connection system variables have been set to reflect a character set and collation of **latin1** and **latin1\_swedish\_ci**. This occurs because the server cannot satisfy the client character set request and falls back to its defaults.

In this case, the client cannot use the character set that it wants because the server does not support it. The client must either be willing to use a different character set, or connect to a different server that supports the desired character set.

The same problem occurs in a more subtle context: When the client tells the server to use a character set that the server recognizes, but the default collation for that character set on the client side is not known on the server side. This occurs, for example, when a MySQL 8.0 client wants to connect to a MySQL 5.7 server using **utf8mb4** as the client character set. A client that specifies [--default-character-set=utf8mb4](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) is able to connect to the server. However, as in the previous example, the server falls back to its default character set and collation, not what the client requested:

mysql> **SHOW SESSION VARIABLES LIKE 'character\\_set\\_%';**

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

| Variable\_name | Value |

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

| character\_set\_client | latin1 |

| character\_set\_connection | latin1 |

...

| character\_set\_results | latin1 |

...

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

mysql> **SHOW SESSION VARIABLES LIKE 'collation\_connection';**

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

| Variable\_name | Value |

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

| collation\_connection | latin1\_swedish\_ci |

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

Why does this occur? After all, **utf8mb4** is known to the 8.0 client and the 5.7 server, so both of them recognize it. To understand this behavior, it is necessary to understand that when the client tells the server which character set it wants to use, it really tells the server the default collation for that character set. Therefore, the aforementioned behavior occurs due to a combination of factors:

The default collation for **utf8mb4** differs between MySQL 5.7 and 8.0 (**utf8mb4\_general\_ci** for 5.7, **utf8mb4\_0900\_ai\_ci** for 8.0).

When the 8.0 client requests a character set of **utf8mb4**, what it sends to the server is the default 8.0 **utf8mb4** collation; that is, the **utf8mb4\_0900\_ai\_ci**.

**utf8mb4\_0900\_ai\_ci** is implemented only as of MySQL 8.0, so the 5.7 server does not recognize it.

Because the 5.7 server does not recognize **utf8mb4\_0900\_ai\_ci**, it cannot satisfy the client character set request, and falls back to its default character set and collation (**latin1** and **latin1\_swedish\_ci**).

In this case, the client can still use **utf8mb4** by issuing a **SET NAMES 'utf8mb4'** statement after connecting. The resulting collation is the 5.7 default **utf8mb4** collation; that is, **utf8mb4\_general\_ci**. If the client additionally wants a collation of **utf8mb4\_0900\_ai\_ci**, it cannot achieve that because the server does not recognize that collation. The client must either be willing to use a different **utf8mb4** collation, or connect to a server from MySQL 8.0 or higher.

#### Runtime Error Handling

Within an established connection, the client can request a change of connection character set and collation with [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) or [**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set).

Some character sets cannot be used as the client character set; see [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset). If you specify a character set that is valid but not permitted as a client character set, the server returns an error:

mysql> **SET NAMES 'ucs2';**

ERROR 1231 (42000): Variable 'character\_set\_client' can't be set to

the value of 'ucs2'

If the server does not recognize the character set (or the collation), it produces an error:

mysql> **SET NAMES 'bogus';**

ERROR 1115 (42000): Unknown character set: 'bogus'

mysql> **SET NAMES 'utf8mb4' COLLATE 'bogus';**

ERROR 1273 (HY000): Unknown collation: 'bogus'

Tip

A client that wants to verify whether its requested character set was honored by the server can execute the following statement after connecting and checking that the result is the expected character set:

SELECT @@character\_set\_client;

## 10.5 Configuring Application Character Set and Collation

For applications that store data using the default MySQL character set and collation (**utf8mb4**, **utf8mb4\_0900\_ai\_ci**), no special configuration should be needed. If applications require data storage using a different character set or collation, you can configure character set information several ways:

Specify character settings per database. For example, applications that use one database might use the default of **utf8mb4**, whereas applications that use another database might use **sjis**.

Specify character settings at server startup. This causes the server to use the given settings for all applications that do not make other arrangements.

Specify character settings at configuration time, if you build MySQL from source. This causes the server to use the given settings as the defaults for all applications, without having to specify them at server startup.

When different applications require different character settings, the per-database technique provides a good deal of flexibility. If most or all applications use the same character set, specifying character settings at server startup or configuration time may be most convenient.

For the per-database or server-startup techniques, the settings control the character set for data storage. Applications must also tell the server which character set to use for client/server communications, as described in the following instructions.

The examples shown here assume use of the **latin1** character set and **latin1\_swedish\_ci** collation in particular contexts as an alternative to the defaults of **utf8mb4** and **utf8mb4\_0900\_ai\_ci**.

**Specify character settings per database.** To create a database such that its tables use a given default character set and collation for data storage, use a [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statement like this:

CREATE DATABASE mydb

CHARACTER SET latin1

COLLATE latin1\_swedish\_ci;

Tables created in the database use **latin1** and **latin1\_swedish\_ci** by default for any character columns.

Applications that use the database should also configure their connection to the server each time they connect. This can be done by executing a **SET NAMES 'latin1'** statement after connecting. The statement can be used regardless of connection method (the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, PHP scripts, and so forth).

In some cases, it may be possible to configure the connection to use the desired character set some other way. For example, to connect using [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql), you can specify the [--default-character-set=latin1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) command-line option to achieve the same effect as **SET NAMES 'latin1'**.

For more information about configuring client connections, see [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).

**Note**

If you use [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) to change the database default character set or collation, existing stored routines in the database that use those defaults must be dropped and recreated so that they use the new defaults. (In a stored routine, variables with character data types use the database defaults if the character set or collation are not specified explicitly. See [Section 13.1.17, “CREATE PROCEDURE and CREATE FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).)

**Specify character settings at server startup.** To select a character set and collation at server startup, use the [--character-set-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [--collation-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) options. For example, to specify the options in an option file, include these lines:

[mysqld]

character-set-server=latin1

collation-server=latin1\_swedish\_ci

These settings apply server-wide and apply as the defaults for databases created by any application, and for tables created in those databases.

It is still necessary for applications to configure their connection using [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) or equivalent after they connect, as described previously. You might be tempted to start the server with the [--init\_connect="SET NAMES 'latin1'"](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_connect) option to cause [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) to be executed automatically for each client that connects. However, this may yield inconsistent results because the [**init\_connect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_connect) value is not executed for users who have the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

**Specify character settings at MySQL configuration time.** To select a character set and collation if you configure and build MySQL from source, use the [DEFAULT\_CHARSET](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_default_charset) and [DEFAULT\_COLLATION](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_default_collation) **CMake** options:

cmake . -DDEFAULT\_CHARSET=latin1 \

-DDEFAULT\_COLLATION=latin1\_swedish\_ci

The resulting server uses **latin1** and **latin1\_swedish\_ci** as the default for databases and tables and for client connections. It is unnecessary to use [--character-set-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [--collation-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) to specify those defaults at server startup. It is also unnecessary for applications to configure their connection using [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) or equivalent after they connect to the server.

Regardless of how you configure the MySQL character set for application use, you must also consider the environment within which those applications execute. For example, if you intend to send statements using UTF-8 text taken from a file that you create in an editor, you should edit the file with the locale of your environment set to UTF-8 so that the file encoding is correct and so that the operating system handles it correctly. If you use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client from within a terminal window, the window must be configured to use UTF-8 or characters may not display properly. For a script that executes in a Web environment, the script must handle character encoding properly for its interaction with the MySQL server, and it must generate pages that correctly indicate the encoding so that browsers know how to display the content of the pages. For example, you can include this **<meta>** tag within your **<head>** element:

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

## 10.6 Error Message Character Set

This section describes how the MySQL server uses character sets for constructing error messages. For information about the language of error messages (rather than the character set), see [Section 10.12, “Setting the Error Message Language”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#error-message-language). For general information about configuring error logging, see [Section 5.4.2, “The Error Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log).

[Character Set for Error Message Construction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-errors-construction)

[Character Set for Error Message Disposition](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-errors-disposition)

### Character Set for Error Message Construction

The server constructs error messages as follows:

The message template uses UTF-8 (**utf8mb3**).

Parameters in the message template are replaced with values that apply to a specific error occurrence:

Identifiers such as table or column names use UTF-8 internally so they are copied as is.

Character (nonbinary) string values are converted from their character set to UTF-8.

Binary string values are copied as is for bytes in the range **0x20** to **0x7E**, and using **\x** hexadecimal encoding for bytes outside that range. For example, if a duplicate-key error occurs for an attempt to insert **0x41CF9F** into a [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) unique column, the resulting error message uses UTF-8 with some bytes hexadecimal encoded:

Duplicate entry 'A\xCF\x9F' for key 1

### Character Set for Error Message Disposition

An error message, once constructed, can be written by the server to the error log or sent to clients:

If the server writes the error message to the error log, it writes it in UTF-8, as constructed, without conversion to another character set.

If the server sends the error message to a client program, the server converts it from UTF-8 to the character set specified by the [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) system variable. If [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) has a value of **NULL** or **binary**, no conversion occurs. No conversion occurs if the variable value is **utf8mb3** or **utf8mb4**, either, because those character sets have a repertoire that includes all UTF-8 characters used in message construction.

If characters cannot be represented in [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results), some encoding may occur during the conversion. The encoding uses Unicode code point values:

Characters in the Basic Multilingual Plane (BMP) range (**0x0000** to **0xFFFF**) are written using **\*nnnn*** notation.

Characters outside the BMP range (**0x10000** to **0x10FFFF**) are written using **\+*nnnnnn*** notation.

Clients can set [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) to control the character set in which they receive error messages. The variable can be set directly, or indirectly by means such as [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names). For more information about [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results), see [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).

## 10.7 Column Character Set Conversion

To convert a binary or nonbinary string column to use a particular character set, use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). For successful conversion to occur, one of the following conditions must apply:

If the column has a binary data type ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)), all the values that it contains must be encoded using a single character set (the character set you're converting the column to). If you use a binary column to store information in multiple character sets, MySQL has no way to know which values use which character set and cannot convert the data properly.

If the column has a nonbinary data type ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)), its contents should be encoded in the column character set, not some other character set. If the contents are encoded in a different character set, you can convert the column to use a binary data type first, and then to a nonbinary column with the desired character set.

Suppose that a table **t** has a binary column named **col1** defined as **VARBINARY(50)**. Assuming that the information in the column is encoded using a single character set, you can convert it to a nonbinary column that has that character set. For example, if **col1** contains binary data representing characters in the **greek** character set, you can convert it as follows:

ALTER TABLE t MODIFY col1 VARCHAR(50) CHARACTER SET greek;

If your original column has a type of **BINARY(50)**, you could convert it to **CHAR(50)**, but the resulting values are padded with **0x00** bytes at the end, which may be undesirable. To remove these bytes, use the [**TRIM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_trim) function:

UPDATE t SET col1 = TRIM(TRAILING 0x00 FROM col1);

Suppose that table **t** has a nonbinary column named **col1** defined as **CHAR(50) CHARACTER SET latin1** but you want to convert it to use **utf8** so that you can store values from many languages. The following statement accomplishes this:

ALTER TABLE t MODIFY col1 CHAR(50) CHARACTER SET utf8;

Conversion may be lossy if the column contains characters that are not in both character sets.

A special case occurs if you have old tables from before MySQL 4.1 where a nonbinary column contains values that actually are encoded in a character set different from the server's default character set. For example, an application might have stored **sjis** values in a column, even though MySQL's default character set was different. It is possible to convert the column to use the proper character set but an additional step is required. Suppose that the server's default character set was **latin1** and **col1** is defined as **CHAR(50)** but its contents are **sjis** values. The first step is to convert the column to a binary data type, which removes the existing character set information without performing any character conversion:

ALTER TABLE t MODIFY col1 BLOB;

The next step is to convert the column to a nonbinary data type with the proper character set:

ALTER TABLE t MODIFY col1 CHAR(50) CHARACTER SET sjis;

This procedure requires that the table not have been modified already with statements such as [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) after an upgrade to MySQL 4.1 or higher. In that case, MySQL would store new values in the column using **latin1**, and the column would contain a mix of **sjis** and **latin1** values and cannot be converted properly.

If you specified attributes when creating a column initially, you should also specify them when altering the table with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). For example, if you specified **NOT NULL** and an explicit **DEFAULT** value, you should also provide them in the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. Otherwise, the resulting column definition does not include those attributes.

To convert all character columns in a table, the **ALTER TABLE ... CONVERT TO CHARACTER SET *charset*** statement may be useful. See [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

## 10.8 Collation Issues

[10.8.1 Using COLLATE in SQL Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collate)

[10.8.2 COLLATE Clause Precedence](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collate-precedence)

[10.8.3 Character Set and Collation Compatibility](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-compatibility)

[10.8.4 Collation Coercibility in Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-coercibility)

[10.8.5 The binary Collation Compared to \_bin Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations)

[10.8.6 Examples of the Effect of Collation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-effect)

[10.8.7 Using Collation in INFORMATION\_SCHEMA Searches](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-information-schema)

The following sections discuss various aspects of character set collations.

### 10.8.1 Using COLLATE in SQL Statements

With the **COLLATE** clause, you can override whatever the default collation is for a comparison. **COLLATE** may be used in various parts of SQL statements. Here are some examples:

With **ORDER BY**:

SELECT k

FROM t1

ORDER BY k COLLATE latin1\_german2\_ci;

With **AS**:

SELECT k COLLATE latin1\_german2\_ci AS k1

FROM t1

ORDER BY k1;

With **GROUP BY**:

SELECT k

FROM t1

GROUP BY k COLLATE latin1\_german2\_ci;

With aggregate functions:

SELECT MAX(k COLLATE latin1\_german2\_ci)

FROM t1;

With **DISTINCT**:

SELECT DISTINCT k COLLATE latin1\_german2\_ci

FROM t1;

With **WHERE**:

SELECT \*

FROM t1

WHERE \_latin1 'Müller' COLLATE latin1\_german2\_ci = k;

SELECT \*

FROM t1

WHERE k LIKE \_latin1 'Müller' COLLATE latin1\_german2\_ci;

With **HAVING**:

SELECT k

FROM t1

GROUP BY k

HAVING k = \_latin1 'Müller' COLLATE latin1\_german2\_ci;

### 10.8.2 COLLATE Clause Precedence

The **COLLATE** clause has high precedence (higher than [**||**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or)), so the following two expressions are equivalent:

x || y COLLATE z

x || (y COLLATE z)

### 10.8.3 Character Set and Collation Compatibility

Each character set has one or more collations, but each collation is associated with one and only one character set. Therefore, the following statement causes an error message because the **latin2\_bin** collation is not legal with the **latin1** character set:

mysql> **SELECT \_latin1 'x' COLLATE latin2\_bin;**

ERROR 1253 (42000): COLLATION 'latin2\_bin' is not valid

for CHARACTER SET 'latin1'

### 10.8.4 Collation Coercibility in Expressions

In the great majority of statements, it is obvious what collation MySQL uses to resolve a comparison operation. For example, in the following cases, it should be clear that the collation is the collation of column **x**:

SELECT x FROM T ORDER BY x;

SELECT x FROM T WHERE x = x;

SELECT DISTINCT x FROM T;

However, with multiple operands, there can be ambiguity. For example, this statement performs a comparison between the column **x** and the string literal **'Y'**:

SELECT x FROM T WHERE x = 'Y';

If **x** and **'Y'** have the same collation, there is no ambiguity about the collation to use for the comparison. But if they have different collations, should the comparison use the collation of **x**, or of **'Y'**? Both **x** and **'Y'** have collations, so which collation takes precedence?

A mix of collations may also occur in contexts other than comparison. For example, a multiple-argument concatenation operation such as [**CONCAT(x,'Y')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) combines its arguments to produce a single string. What collation should the result have?

To resolve questions like these, MySQL checks whether the collation of one item can be coerced to the collation of the other. MySQL assigns coercibility values as follows:

An explicit **COLLATE** clause has a coercibility of 0 (not coercible at all).

The concatenation of two strings with different collations has a coercibility of 1.

The collation of a column or a stored routine parameter or local variable has a coercibility of 2.

A “system constant” (the string returned by functions such as [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) or [**VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version)) has a coercibility of 3.

The collation of a literal has a coercibility of 4.

The collation of a numeric or temporal value has a coercibility of 5.

**NULL** or an expression that is derived from **NULL** has a coercibility of 6.

MySQL uses coercibility values with the following rules to resolve ambiguities:

Use the collation with the lowest coercibility value.

If both sides have the same coercibility, then:

If both sides are Unicode, or both sides are not Unicode, it is an error.

If one of the sides has a Unicode character set, and another side has a non-Unicode character set, the side with Unicode character set wins, and automatic character set conversion is applied to the non-Unicode side. For example, the following statement does not return an error:

SELECT CONCAT(utf8mb4\_column, latin1\_column) FROM t1;

It returns a result that has a character set of **utf8mb4** and the same collation as **utf8mb4\_column**. Values of **latin1\_column** are automatically converted to **utf8mb4** before concatenating.

For an operation with operands from the same character set but that mix a **\_bin** collation and a **\_ci** or **\_cs** collation, the **\_bin** collation is used. This is similar to how operations that mix nonbinary and binary strings evaluate the operands as binary strings, applied to collations rather than data types.

Although automatic conversion is not in the SQL standard, the standard does say that every character set is (in terms of supported characters) a “subset” of Unicode. Because it is a well-known principle that “what applies to a superset can apply to a subset,” we believe that a collation for Unicode can apply for comparisons with non-Unicode strings. More generally, MySQL uses the concept of character set repertoire, which can sometimes be used to determine subset relationships among character sets and enable conversion of operands in operations that would otherwise produce an error. See [Section 10.2.1, “Character Set Repertoire”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-repertoire).

The following table illustrates some applications of the preceding rules.

| **Comparison** | **Collation Used** |
| --- | --- |
| **column1 = 'A'** | Use collation of **column1** |
| **column1 = 'A' COLLATE x** | Use collation of **'A' COLLATE x** |
| **column1 COLLATE x = 'A' COLLATE y** | Error |

To determine the coercibility of a string expression, use the [**COERCIBILITY()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coercibility) function (see [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions)):

mysql> **SELECT COERCIBILITY(\_utf8'A' COLLATE utf8\_bin);**

-> 0

mysql> **SELECT COERCIBILITY(VERSION());**

-> 3

mysql> **SELECT COERCIBILITY('A');**

-> 4

mysql> **SELECT COERCIBILITY(1000);**

-> 5

mysql> **SELECT COERCIBILITY(NULL);**

-> 6

For implicit conversion of a numeric or temporal value to a string, such as occurs for the argument **1** in the expression [**CONCAT(1, 'abc')**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat), the result is a character (nonbinary) string that has a character set and collation determined by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) and [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variables. See [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion).

### 10.8.5 The binary Collation Compared to \_bin Collations

This section describes how the **binary** collation for binary strings compares to **\_bin** collations for nonbinary strings.

Binary strings (as stored using the [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) data types) have a character set and collation named **binary**. Binary strings are sequences of bytes and the numeric values of those bytes determine comparison and sort order. See [Section 10.10.8, “The Binary Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-set).

Nonbinary strings (as stored using the [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) data types) have a character set and collation other than **binary**. A given nonbinary character set can have several collations, each of which defines a particular comparison and sort order for the characters in the set. For most character sets, one of these is the binary collation, indicated by a **\_bin** suffix in the collation name. For example, the binary collation for **utf8** and **latin1** is named **utf8\_bin** and **latin1\_bin**, respectively. **utf8mb4** is an exception that has two binary collations, **utf8mb4\_bin** and **utf8mb4\_0900\_bin**; see [Section 10.10.1, “Unicode Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets).

The **binary** collation differs from **\_bin** collations in several respects, discussed in the following sections:

[The Unit for Comparison and Sorting](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-comparison-units)

[Character Set Conversion](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-charset-conversion)

[Lettercase Conversion](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-lettercase-conversion)

[Trailing Space Handling in Comparisons](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-trailing-space-comparisons)

[Trailing Space Handling for Inserts and Retrievals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-trailing-space-inserts-retrievals)

#### The Unit for Comparison and Sorting

Binary strings are sequences of bytes. For the **binary** collation, comparison and sorting are based on numeric byte values. Nonbinary strings are sequences of characters, which might be multibyte. Collations for nonbinary strings define an ordering of the character values for comparison and sorting. For **\_bin** collations, this ordering is based on numeric character code values, which is similar to ordering for binary strings except that character code values might be multibyte.

#### Character Set Conversion

A nonbinary string has a character set and is automatically converted to another character set in many cases, even when the string has a **\_bin** collation:

When assigning column values to another column that has a different character set:

UPDATE t1 SET utf8\_bin\_column=latin1\_column;

INSERT INTO t1 (latin1\_column) SELECT utf8\_bin\_column FROM t2;

When assigning column values for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) using a string literal:

SET NAMES latin1;

INSERT INTO t1 (utf8\_bin\_column) VALUES ('string-in-latin1');

When sending results from the server to a client:

SET NAMES latin1;

SELECT utf8\_bin\_column FROM t2;

For binary string columns, no conversion occurs. For cases similar to those preceding, the string value is copied byte-wise.

#### Lettercase Conversion

Collations for nonbinary character sets provide information about lettercase of characters, so characters in a nonbinary string can be converted from one lettercase to another, even for **\_bin** collations that ignore lettercase for ordering:

mysql> **SET NAMES utf8mb4 COLLATE utf8mb4\_bin;**

mysql> **SELECT LOWER('aA'), UPPER('zZ');**

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

| LOWER('aA') | UPPER('zZ') |

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

| aa | ZZ |

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

The concept of lettercase does not apply to bytes in a binary string. To perform lettercase conversion, the string must first be converted to a nonbinary string using a character set appropriate for the data stored in the string:

mysql> **SET NAMES binary;**

mysql> **SELECT LOWER('aA'), LOWER(CONVERT('aA' USING utf8mb4));**

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

| LOWER('aA') | LOWER(CONVERT('aA' USING utf8mb4)) |

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

| aA | aa |

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

#### Trailing Space Handling in Comparisons

MySQL collations have a pad attribute, which has a value of **PAD SPACE** or **NO PAD**:

Most MySQL collations have a pad attribute of **PAD SPACE**.

The Unicode collations based on UCA 9.0.0 and higher have a pad attribute of **NO PAD**; see [Section 10.10.1, “Unicode Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets).

For nonbinary strings (**CHAR**, **VARCHAR**, and **TEXT** values), the string collation pad attribute determines treatment in comparisons of trailing spaces at the end of strings:

For **PAD SPACE** collations, trailing spaces are insignificant in comparisons; strings are compared without regard to trailing spaces.

**NO PAD** collations treat trailing spaces as significant in comparisons, like any other character.

The differing behaviors can be demonstrated using the two **utf8mb4** binary collations, one of which is **PAD SPACE**, the other of which is **NO PAD**. The example also shows how to use the **INFORMATION\_SCHEMA** [**COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table to determine the pad attribute for collations.

mysql> **SELECT COLLATION\_NAME, PAD\_ATTRIBUTE**

**FROM INFORMATION\_SCHEMA.COLLATIONS**

**WHERE COLLATION\_NAME LIKE 'utf8mb4%bin';**

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

| COLLATION\_NAME | PAD\_ATTRIBUTE |

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

| utf8mb4\_bin | PAD SPACE |

| utf8mb4\_0900\_bin | NO PAD |

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

mysql> **SET NAMES utf8mb4 COLLATE utf8mb4\_bin;**

mysql> **SELECT 'a ' = 'a';**

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

| 'a ' = 'a' |

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

| 1 |

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

mysql> **SET NAMES utf8mb4 COLLATE utf8mb4\_0900\_bin;**

mysql> **SELECT 'a ' = 'a';**

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

| 'a ' = 'a' |

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

| 0 |

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

**Note**

“Comparison” in this context does not include the [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) pattern-matching operator, for which trailing spaces are significant, regardless of collation.

For binary strings (**BINARY**, **VARBINARY**, and **BLOB** values), all bytes are significant in comparisons, including trailing spaces:

mysql> **SET NAMES binary;**

mysql> **SELECT 'a ' = 'a';**

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

| 'a ' = 'a' |

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

| 0 |

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

#### Trailing Space Handling for Inserts and Retrievals

**CHAR(*N*)** columns store nonbinary strings ***N*** characters long. For inserts, values shorter than ***N*** characters are extended with spaces. For retrievals, trailing spaces are removed.

**BINARY(*N*)** columns store binary strings ***N*** bytes long. For inserts, values shorter than ***N*** bytes are extended with **0x00** bytes. For retrievals, nothing is removed; a value of the declared length is always returned.

mysql> **CREATE TABLE t1 (**

**a CHAR(10) CHARACTER SET utf8 COLLATE utf8\_bin,**

**b BINARY(10)**

**);**

mysql> **INSERT INTO t1 VALUES ('x','x');**

mysql> **INSERT INTO t1 VALUES ('x ','x ');**

mysql> **SELECT a, b, HEX(a), HEX(b) FROM t1;**

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

| a | b | HEX(a) | HEX(b) |

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

| x | x | 78 | 78000000000000000000 |

| x | x | 78 | 78200000000000000000 |

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

### 10.8.6 Examples of the Effect of Collation

***Example 1: Sorting German Umlauts***

Suppose that column **X** in table **T** has these **latin1** column values:

Muffler

Müller

MX Systems

MySQL

Suppose also that the column values are retrieved using the following statement:

SELECT X FROM T ORDER BY X COLLATE ***collation\_name***;

The following table shows the resulting order of the values if we use **ORDER BY** with different collations.

| **latin1\_swedish\_ci** | **latin1\_german1\_ci** | **latin1\_german2\_ci** |
| --- | --- | --- |
| **Muffler** | Muffler | Müller |
| **MX Systems** | Müller | Muffler |
| **Müller** | MX Systems | MX Systems |
| **MySQL** | MySQL | MySQL |

The character that causes the different sort orders in this example is the U with two dots over it (**ü**), which the Germans call “U-umlaut.”

The first column shows the result of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) using the Swedish/Finnish collating rule, which says that U-umlaut sorts with Y.

The second column shows the result of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) using the German DIN-1 rule, which says that U-umlaut sorts with U.

The third column shows the result of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) using the German DIN-2 rule, which says that U-umlaut sorts with UE.

***Example 2: Searching for German Umlauts***

Suppose that you have three tables that differ only by the character set and collation used:

mysql> **SET NAMES utf8;**

mysql> **CREATE TABLE german1 (**

**c CHAR(10)**

**) CHARACTER SET latin1 COLLATE latin1\_german1\_ci;**

mysql> **CREATE TABLE german2 (**

**c CHAR(10)**

**) CHARACTER SET latin1 COLLATE latin1\_german2\_ci;**

mysql> **CREATE TABLE germanutf8 (**

**c CHAR(10)**

**) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci;**

Each table contains two records:

mysql> **INSERT INTO german1 VALUES ('Bar'), ('Bär');**

mysql> **INSERT INTO german2 VALUES ('Bar'), ('Bär');**

mysql> **INSERT INTO germanutf8 VALUES ('Bar'), ('Bär');**

Two of the above collations have an **A = Ä** equality, and one has no such equality (**latin1\_german2\_ci**). For that reason, you'll get these results in comparisons:

mysql> **SELECT \* FROM german1 WHERE c = 'Bär';**

+------+

| c |

+------+

| Bar |

| Bär |

+------+

mysql> **SELECT \* FROM german2 WHERE c = 'Bär';**

+------+

| c |

+------+

| Bär |

+------+

mysql> **SELECT \* FROM germanutf8 WHERE c = 'Bär';**

+------+

| c |

+------+

| Bar |

| Bär |

+------+

This is not a bug but rather a consequence of the sorting properties of **latin1\_german1\_ci** and **utf8\_unicode\_ci** (the sorting shown is done according to the German DIN 5007 standard).

### 10.8.7 Using Collation in INFORMATION\_SCHEMA Searches

String columns in **INFORMATION\_SCHEMA** tables have a collation of **utf8\_general\_ci**, which is case-insensitive. However, for values that correspond to objects that are represented in the file system, such as databases and tables, searches in **INFORMATION\_SCHEMA** string columns can be case-sensitive or case-insensitive, depending on the characteristics of the underlying file system and the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable setting. For example, searches may be case-sensitive if the file system is case-sensitive. This section describes this behavior and how to modify it if necessary.

Suppose that a query searches the **SCHEMATA.SCHEMA\_NAME** column for the **test** database. On Linux, file systems are case-sensitive, so comparisons of **SCHEMATA.SCHEMA\_NAME** with **'test'** match, but comparisons with **'TEST'** do not:

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME = 'test';**

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

| SCHEMA\_NAME |

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

| test |

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

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME = 'TEST';**

Empty set (0.00 sec)

These results occur with the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable set to 0. A [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting of 1 or 2 causes the second query to return the same (nonempty) result as the first query.

**Note**

It is prohibited to start the server with a [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting that is different from the setting used when the server was initialized.

On Windows or macOS, file systems are not case-sensitive, so comparisons match both **'test'** and **'TEST'**:

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME = 'test';**

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

| SCHEMA\_NAME |

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

| test |

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

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME = 'TEST';**

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

| SCHEMA\_NAME |

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

| TEST |

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

The value of [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) makes no difference in this context.

The preceding behavior occurs because the **utf8\_general\_ci** collation is not used for **INFORMATION\_SCHEMA** queries when searching for values that correspond to objects represented in the file system.

If the result of a string operation on an **INFORMATION\_SCHEMA** column differs from expectations, a workaround is to use an explicit **COLLATE** clause to force a suitable collation (see [Section 10.8.1, “Using COLLATE in SQL Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collate)). For example, to perform a case-insensitive search, use **COLLATE** with the **INFORMATION\_SCHEMA** column name:

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME COLLATE utf8\_general\_ci = 'test';**

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

| SCHEMA\_NAME |

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

| test |

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

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME COLLATE utf8\_general\_ci = 'TEST';**

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

| SCHEMA\_NAME |

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

| test |

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

You can also use the [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) or [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) function:

WHERE UPPER(SCHEMA\_NAME) = 'TEST'

WHERE LOWER(SCHEMA\_NAME) = 'test'

Although a case-insensitive comparison can be performed even on platforms with case-sensitive file systems, as just shown, it is not necessarily always the right thing to do. On such platforms, it is possible to have multiple objects with names that differ only in lettercase. For example, tables named **city**, **CITY**, and **City** can all exist simultaneously. Consider whether a search should match all such names or just one and write queries accordingly. The first of the following comparisons (with **utf8\_bin**) is case-sensitive; the others are not:

WHERE TABLE\_NAME COLLATE utf8\_bin = 'City'

WHERE TABLE\_NAME COLLATE utf8\_general\_ci = 'city'

WHERE UPPER(TABLE\_NAME) = 'CITY'

WHERE LOWER(TABLE\_NAME) = 'city'

Searches in **INFORMATION\_SCHEMA** string columns for values that refer to **INFORMATION\_SCHEMA** itself do use the **utf8\_general\_ci** collation because **INFORMATION\_SCHEMA** is a “virtual” database not represented in the file system. For example, comparisons with **SCHEMATA.SCHEMA\_NAME** match **'information\_schema'** or **'INFORMATION\_SCHEMA'** regardless of platform:

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME = 'information\_schema';**

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

| SCHEMA\_NAME |

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

| information\_schema |

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

mysql> **SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE SCHEMA\_NAME = 'INFORMATION\_SCHEMA';**

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

| SCHEMA\_NAME |

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

| information\_schema |

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

## 10.9 Unicode Support

[10.9.1 The utf8mb4 Character Set (4-Byte UTF-8 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8mb4)

[10.9.2 The utf8mb3 Character Set (3-Byte UTF-8 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8mb3)

[10.9.3 The utf8 Character Set (Alias for utf8mb3)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8)

[10.9.4 The ucs2 Character Set (UCS-2 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-ucs2)

[10.9.5 The utf16 Character Set (UTF-16 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf16)

[10.9.6 The utf16le Character Set (UTF-16LE Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf16le)

[10.9.7 The utf32 Character Set (UTF-32 Unicode Encoding)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf32)

[10.9.8 Converting Between 3-Byte and 4-Byte Unicode Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-conversion)

The Unicode Standard includes characters from the Basic Multilingual Plane (BMP) and supplementary characters that lie outside the BMP. This section describes support for Unicode in MySQL. For information about the Unicode Standard itself, visit the [Unicode Consortium website](http://www.unicode.org/).

BMP characters have these characteristics:

Their code point values are between 0 and 65535 (or **U+0000** and **U+FFFF**).

They can be encoded in a variable-length encoding using 8, 16, or 24 bits (1 to 3 bytes).

They can be encoded in a fixed-length encoding using 16 bits (2 bytes).

They are sufficient for almost all characters in major languages.

Supplementary characters lie outside the BMP:

Their code point values are between **U+10000** and **U+10FFFF**).

Unicode support for supplementary characters requires character sets that have a range outside BMP characters and therefore take more space than BMP characters (up to 4 bytes per character).

The UTF-8 (Unicode Transformation Format with 8-bit units) method for encoding Unicode data is implemented according to RFC 3629, which describes encoding sequences that take from one to four bytes. The idea of UTF-8 is that various Unicode characters are encoded using byte sequences of different lengths:

Basic Latin letters, digits, and punctuation signs use one byte.

Most European and Middle East script letters fit into a 2-byte sequence: extended Latin letters (with tilde, macron, acute, grave and other accents), Cyrillic, Greek, Armenian, Hebrew, Arabic, Syriac, and others.

Korean, Chinese, and Japanese ideographs use 3-byte or 4-byte sequences.

MySQL supports these Unicode character sets:

**utf8mb4**: A UTF-8 encoding of the Unicode character set using one to four bytes per character.

**utf8mb3**: A UTF-8 encoding of the Unicode character set using one to three bytes per character.

**utf8**: An alias for **utf8mb3**.

**ucs2**: The UCS-2 encoding of the Unicode character set using two bytes per character.

**utf16**: The UTF-16 encoding for the Unicode character set using two or four bytes per character. Like **ucs2** but with an extension for supplementary characters.

**utf16le**: The UTF-16LE encoding for the Unicode character set. Like **utf16** but little-endian rather than big-endian.

**utf32**: The UTF-32 encoding for the Unicode character set using four bytes per character.

**Note**

The **utf8mb3** character set is deprecated and you should expect it to be removed in a future MySQL release. Please use **utf8mb4** instead. Although **utf8** is currently an alias for **utf8mb3**, at some point **utf8** is expected to become a reference to **utf8mb4**. To avoid ambiguity about the meaning of **utf8**, consider specifying **utf8mb4** explicitly for character set references instead of **utf8**.

[Table 10.2, “Unicode Character Set General Characteristics”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-charset-characteristics), summarizes the general characteristics of Unicode character sets supported by MySQL.

**Table 10.2 Unicode Character Set General Characteristics**

| **Character Set** | **Supported Characters** | **Required Storage Per Character** |
| --- | --- | --- |
| **utf8mb3, utf8** | BMP only | 1, 2, or 3 bytes |
| **ucs2** | BMP only | 2 bytes |
| **utf8mb4** | BMP and supplementary | 1, 2, 3, or 4 bytes |
| **utf16** | BMP and supplementary | 2 or 4 bytes |
| **utf16le** | BMP and supplementary | 2 or 4 bytes |
| **utf32** | BMP and supplementary | 4 bytes |

Characters outside the BMP compare as REPLACEMENT CHARACTER and convert to **'?'** when converted to a Unicode character set that supports only BMP characters (**utf8mb3** or **ucs2**).

If you use character sets that support supplementary characters and thus are “wider” than the BMP-only **utf8mb3** and **ucs2** character sets, there are potential incompatibility issues for your applications; see [Section 10.9.8, “Converting Between 3-Byte and 4-Byte Unicode Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-conversion). That section also describes how to convert tables from the (3-byte) **utf8mb3** to the (4-byte) **utf8mb4**, and what constraints may apply in doing so.

A similar set of collations is available for most Unicode character sets. For example, each has a Danish collation, the names of which are **utf8mb4\_danish\_ci**, **utf8mb3\_danish\_ci**, **utf8\_danish\_ci**, **ucs2\_danish\_ci**, **utf16\_danish\_ci**, and **utf32\_danish\_ci**. The exception is **utf16le**, which has only two collations. For information about Unicode collations and their differentiating properties, including collation properties for supplementary characters, see [Section 10.10.1, “Unicode Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets).

The MySQL implementation of UCS-2, UTF-16, and UTF-32 stores characters in big-endian byte order and does not use a byte order mark (BOM) at the beginning of values. Other database systems might use little-endian byte order or a BOM. In such cases, conversion of values needs to be performed when transferring data between those systems and MySQL. The implementation of UTF-16LE is little-endian.

MySQL uses no BOM for UTF-8 values.

Client applications that communicate with the server using Unicode should set the client character set accordingly (for example, by issuing a **SET NAMES 'utf8mb4'** statement). Some character sets cannot be used as the client character set. Attempting to use them with [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) or [**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) produces an error. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

The following sections provide additional detail on the Unicode character sets in MySQL.

### 10.9.1 The utf8mb4 Character Set (4-Byte UTF-8 Unicode Encoding)

The **utfmb4** character set has these characteristics:

Supports BMP and supplementary characters.

Requires a maximum of four bytes per multibyte character.

**utf8mb4** contrasts with the **utf8mb3** character set, which supports only BMP characters and uses a maximum of three bytes per character:

For a BMP character, **utf8mb4** and **utf8mb3** have identical storage characteristics: same code values, same encoding, same length.

For a supplementary character, **utf8mb4** requires four bytes to store it, whereas **utf8mb3** cannot store the character at all. When converting **utf8mb3** columns to **utf8mb4**, you need not worry about converting supplementary characters because there are none.

**utf8mb4** is a superset of **utf8mb3**, so for an operation such as the following concatenation, the result has character set **utf8mb4** and the collation of **utf8mb4\_col**:

SELECT CONCAT(utf8mb3\_col, utf8mb4\_col);

Similarly, the following comparison in the **WHERE** clause works according to the collation of **utf8mb4\_col**:

SELECT \* FROM utf8mb3\_tbl, utf8mb4\_tbl

WHERE utf8mb3\_tbl.utf8mb3\_col = utf8mb4\_tbl.utf8mb4\_col;

For information about data type storage as it relates to multibyte character sets, see [String Type Storage Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-types-storage-reqs-strings).

### 10.9.2 The utf8mb3 Character Set (3-Byte UTF-8 Unicode Encoding)

The **utf8mb3** character set has these characteristics:

Supports BMP characters only (no support for supplementary characters)

Requires a maximum of three bytes per multibyte character.

Applications that use UTF-8 data but require supplementary character support should use **utf8mb4** rather than **utf8mb3** (see [Section 10.9.1, “The utf8mb4 Character Set (4-Byte UTF-8 Unicode Encoding)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8mb4)).

Exactly the same set of characters is available in **utf8mb3** and **ucs2**. That is, they have the same [repertoire](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_repertoire).

**utf8** is an alias for **utf8mb3**; the character limit is implicit, rather than explicit in the name.

**Note**

The **utf8mb3** character set is deprecated and you should expect it to be removed in a future MySQL release. Please use **utf8mb4** instead. Although **utf8** is currently an alias for **utf8mb3**, at some point **utf8** is expected to become a reference to **utf8mb4**. To avoid ambiguity about the meaning of **utf8**, consider specifying **utf8mb4** explicitly for character set references instead of **utf8**.

**utf8mb3** can be used in **CHARACTER SET** clauses, and **utf8mb3\_*collation\_substring*** in **COLLATE** clauses, where ***collation\_substring*** is **bin**, **czech\_ci**, **danish\_ci**, **esperanto\_ci**, **estonian\_ci**, and so forth. For example:

CREATE TABLE t (s1 CHAR(1) CHARACTER SET utf8mb3;

SELECT \* FROM t WHERE s1 COLLATE utf8mb3\_general\_ci = 'x';

DECLARE x VARCHAR(5) CHARACTER SET utf8mb3 COLLATE utf8mb3\_danish\_ci;

SELECT CAST('a' AS CHAR CHARACTER SET utf8) COLLATE utf8\_czech\_ci;

MySQL immediately converts instances of **utf8mb3** in statements to **utf8**, so in statements such as **SHOW CREATE TABLE** or **SELECT CHARACTER\_SET\_NAME FROM INFORMATION\_SCHEMA.COLUMNS** or **SELECT COLLATION\_NAME FROM INFORMATION\_SCHEMA.COLUMNS**, users see the name **utf8** or **utf8\_*collation\_substring***.

**utf8mb3** is also valid in contexts other than **CHARACTER SET** clauses. For example:

mysqld --character-set-server=utf8mb3

SET NAMES 'utf8mb3'; /\* and other SET statements that have similar effect \*/

SELECT \_utf8mb3 'a';

For information about data type storage as it relates to multibyte character sets, see [String Type Storage Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-types-storage-reqs-strings).

### 10.9.3 The utf8 Character Set (Alias for utf8mb3)

**utf8** is an alias for the **utf8mb3** character set. For more information, see [Section 10.9.2, “The utf8mb3 Character Set (3-Byte UTF-8 Unicode Encoding)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-utf8mb3).

**Note**

The **utf8mb3** character set is deprecated and you should expect it to be removed in a future MySQL release. Please use **utf8mb4** instead. Although **utf8** is currently an alias for **utf8mb3**, at some point **utf8** is expected to become a reference to **utf8mb4**. To avoid ambiguity about the meaning of **utf8**, consider specifying **utf8mb4** explicitly for character set references instead of **utf8**.

### 10.9.4 The ucs2 Character Set (UCS-2 Unicode Encoding)

In UCS-2, every character is represented by a 2-byte Unicode code with the most significant byte first. For example: **LATIN CAPITAL LETTER A** has the code **0x0041** and it is stored as a 2-byte sequence: **0x00 0x41**. **CYRILLIC SMALL LETTER YERU** (Unicode **0x044B**) is stored as a 2-byte sequence: **0x04 0x4B**. For Unicode characters and their codes, please refer to the [Unicode Consortium website](http://www.unicode.org/).

The **ucs2** character set has these characteristics:

Supports BMP characters only (no support for supplementary characters)

Uses a fixed-length 16-bit encoding and requires two bytes per character.

### 10.9.5 The utf16 Character Set (UTF-16 Unicode Encoding)

The **utf16** character set is the **ucs2** character set with an extension that enables encoding of supplementary characters:

For a BMP character, **utf16** and **ucs2** have identical storage characteristics: same code values, same encoding, same length.

For a supplementary character, **utf16** has a special sequence for representing the character using 32 bits. This is called the “surrogate” mechanism: For a number greater than **0xffff**, take 10 bits and add them to **0xd800** and put them in the first 16-bit word, take 10 more bits and add them to **0xdc00** and put them in the next 16-bit word. Consequently, all supplementary characters require 32 bits, where the first 16 bits are a number between **0xd800** and **0xdbff**, and the last 16 bits are a number between **0xdc00** and **0xdfff**. Examples are in Section [15.5 Surrogates Area](http://www.unicode.org/versions/Unicode4.0.0/ch15.pdf) of the Unicode 4.0 document.

Because **utf16** supports surrogates and **ucs2** does not, there is a validity check that applies only in **utf16**: You cannot insert a top surrogate without a bottom surrogate, or vice versa. For example:

INSERT INTO t (ucs2\_column) VALUES (0xd800); /\* legal \*/

INSERT INTO t (utf16\_column)VALUES (0xd800); /\* illegal \*/

There is no validity check for characters that are technically valid but are not true Unicode (that is, characters that Unicode considers to be “unassigned code points” or “private use” characters or even “illegals” like **0xffff**). For example, since **U+F8FF** is the Apple Logo, this is legal:

INSERT INTO t (utf16\_column)VALUES (0xf8ff); /\* legal \*/

Such characters cannot be expected to mean the same thing to everyone.

Because MySQL must allow for the worst case (that one character requires four bytes) the maximum length of a **utf16** column or index is only half of the maximum length for a **ucs2** column or index. For example, the maximum length of a **MEMORY** table index key is 3072 bytes, so these statements create tables with the longest permitted indexes for **ucs2** and **utf16** columns:

CREATE TABLE tf (s1 VARCHAR(1536) CHARACTER SET ucs2) ENGINE=MEMORY;

CREATE INDEX i ON tf (s1);

CREATE TABLE tg (s1 VARCHAR(768) CHARACTER SET utf16) ENGINE=MEMORY;

CREATE INDEX i ON tg (s1);

### 10.9.6 The utf16le Character Set (UTF-16LE Unicode Encoding)

This is the same as **utf16** but is little-endian rather than big-endian.

### 10.9.7 The utf32 Character Set (UTF-32 Unicode Encoding)

The **utf32** character set is fixed length (like **ucs2** and unlike **utf16**). **utf32** uses 32 bits for every character, unlike **ucs2** (which uses 16 bits for every character), and unlike **utf16** (which uses 16 bits for some characters and 32 bits for others).

**utf32** takes twice as much space as **ucs2** and more space than **utf16**, but **utf32** has the same advantage as **ucs2** that it is predictable for storage: The required number of bytes for **utf32** equals the number of characters times 4. Also, unlike **utf16**, there are no tricks for encoding in **utf32**, so the stored value equals the code value.

To demonstrate how the latter advantage is useful, here is an example that shows how to determine a **utf8mb4** value given the **utf32** code value:

/\* Assume code value = 100cc LINEAR B WHEELED CHARIOT \*/

CREATE TABLE tmp (utf32\_col CHAR(1) CHARACTER SET utf32,

utf8mb4\_col CHAR(1) CHARACTER SET utf8mb4);

INSERT INTO tmp VALUES (0x000100cc,NULL);

UPDATE tmp SET utf8mb4\_col = utf32\_col;

SELECT HEX(utf32\_col),HEX(utf8mb4\_col) FROM tmp;

MySQL is very forgiving about additions of unassigned Unicode characters or private-use-area characters. There is in fact only one validity check for **utf32**: No code value may be greater than **0x10ffff**. For example, this is illegal:

INSERT INTO t (utf32\_column) VALUES (0x110000); /\* illegal \*/

### 10.9.8 Converting Between 3-Byte and 4-Byte Unicode Character Sets

This section describes issues that you may face when converting character data between the **utf8mb3** and **utf8mb4** character sets.

**Note**

This discussion focuses primarily on converting between **utf8mb3** and **utf8mb4**, but similar principles apply to converting between the **ucs2** character set and character sets such as **utf16** or **utf32**.

The **utf8mb3** and **utf8mb4** character sets differ as follows:

**utf8mb3** supports only characters in the Basic Multilingual Plane (BMP). **utf8mb4** additionally supports supplementary characters that lie outside the BMP.

**utf8mb3** uses a maximum of three bytes per character. **utf8mb4** uses a maximum of four bytes per character.

**Note**

This discussion refers to the **utf8mb3** and **utf8mb4** character set names to be explicit about referring to 3-byte and 4-byte UTF-8 character set data. The exception is that in table definitions, **utf8** is used because MySQL converts instances of **utf8mb3** specified in such definitions to **utf8**, which is an alias for **utf8mb3**.

One advantage of converting from **utf8mb3** to **utf8mb4** is that this enables applications to use supplementary characters. One tradeoff is that this may increase data storage space requirements.

In terms of table content, conversion from **utf8mb3** to **utf8mb4** presents no problems:

For a BMP character, **utf8mb4** and **utf8mb3** have identical storage characteristics: same code values, same encoding, same length.

For a supplementary character, **utf8mb4** requires four bytes to store it, whereas **utf8mb3** cannot store the character at all. When converting **utf8mb3** columns to **utf8mb4**, you need not worry about converting supplementary characters because there are none.

In terms of table structure, these are the primary potential incompatibilities:

For the variable-length character data types ([**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) and the [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types), the maximum permitted length in characters is less for **utf8mb4** columns than for **utf8mb3** columns.

For all character data types ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and the [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types), the maximum number of characters that can be indexed is less for **utf8mb4** columns than for **utf8mb3** columns.

Consequently, to convert tables from **utf8mb3** to **utf8mb4**, it may be necessary to change some column or index definitions.

Tables can be converted from **utf8mb3** to **utf8mb4** by using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Suppose that a table has this definition:

CREATE TABLE t1 (

col1 CHAR(10) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci NOT NULL,

col2 CHAR(10) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL

) CHARACTER SET utf8;

The following statement converts **t1** to use **utf8mb4**:

ALTER TABLE t1

DEFAULT CHARACTER SET utf8mb4,

MODIFY col1 CHAR(10)

CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

MODIFY col2 CHAR(10)

CHARACTER SET utf8mb4 COLLATE utf8mb4\_bin NOT NULL;

The catch when converting from **utf8mb3** to **utf8mb4** is that the maximum length of a column or index key is unchanged in terms of bytes. Therefore, it is smaller in terms of characters because the maximum length of a character is four bytes instead of three. For the [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) data types, watch for these issues when converting your MySQL tables:

Check all definitions of **utf8mb3** columns and make sure they do not exceed the maximum length for the storage engine.

Check all indexes on **utf8mb3** columns and make sure they do not exceed the maximum length for the storage engine. Sometimes the maximum can change due to storage engine enhancements.

If the preceding conditions apply, you must either reduce the defined length of columns or indexes, or continue to use **utf8mb3** rather than **utf8mb4**.

Here are some examples where structural changes may be needed:

A [**TINYTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column can hold up to 255 bytes, so it can hold up to 85 3-byte or 63 4-byte characters. Suppose that you have a [**TINYTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column that uses **utf8mb3** but must be able to contain more than 63 characters. You cannot convert it to **utf8mb4** unless you also change the data type to a longer type such as [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob).

Similarly, a very long [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column may need to be changed to one of the longer [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types if you want to convert it from **utf8mb3** to **utf8mb4**.

**InnoDB** has a maximum index length of 767 bytes for tables that use [**COMPACT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compact_row_format) or [**REDUNDANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redundant_row_format) row format, so for **utf8mb3** or **utf8mb4** columns, you can index a maximum of 255 or 191 characters, respectively. If you currently have **utf8mb3** columns with indexes longer than 191 characters, you must index a smaller number of characters.

In an **InnoDB** table that uses [**COMPACT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compact_row_format) or [**REDUNDANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redundant_row_format) row format, these column and index definitions are legal:

col1 VARCHAR(500) CHARACTER SET utf8, INDEX (col1(255))

To use **utf8mb4** instead, the index must be smaller:

col1 VARCHAR(500) CHARACTER SET utf8mb4, INDEX (col1(191))

**Note**

For **InnoDB** tables that use [**COMPRESSED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compressed_row_format) or [**DYNAMIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dynamic_row_format) row format, [index key prefixes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_index_prefix) longer than 767 bytes (up to 3072 bytes) are permitted. Tables created with these row formats enable you to index a maximum of 1024 or 768 characters for **utf8mb3** or **utf8mb4** columns, respectively. For related information, see [Section 15.22, “InnoDB Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-limits), and [DYNAMIC Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-dynamic).

The preceding types of changes are most likely to be required only if you have very long columns or indexes. Otherwise, you should be able to convert your tables from **utf8mb3** to **utf8mb4** without problems, using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) as described previously.

The following items summarize other potential incompatibilities:

**SET NAMES 'utf8mb4'** causes use of the 4-byte character set for connection character sets. As long as no 4-byte characters are sent from the server, there should be no problems. Otherwise, applications that expect to receive a maximum of three bytes per character may have problems. Conversely, applications that expect to send 4-byte characters must ensure that the server understands them.

For replication, if character sets that support supplementary characters are to be used on the source, all replicas must understand them as well.

Also, keep in mind the general principle that if a table has different definitions on the source and replica, this can lead to unexpected results. For example, the differences in maximum index key length make it risky to use **utf8mb3** on the source and **utf8mb4** on the replica.

If you have converted to **utf8mb4**, **utf16**, **utf16le**, or **utf32**, and then decide to convert back to **utf8mb3** or **ucs2** (for example, to downgrade to an older version of MySQL), these considerations apply:

**utf8mb3** and **ucs2** data should present no problems.

The server must be recent enough to recognize definitions referring to the character set from which you are converting.

For object definitions that refer to the **utf8mb4** character set, you can dump them with [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) prior to downgrading, edit the dump file to change instances of **utf8mb4** to **utf8**, and reload the file in the older server, as long as there are no 4-byte characters in the data. The older server sees **utf8** in the dump file object definitions and create new objects that use the (3-byte) **utf8** character set.

## 10.10 Supported Character Sets and Collations

[10.10.1 Unicode Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets)

[10.10.2 West European Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-we-sets)

[10.10.3 Central European Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-ce-sets)

[10.10.4 South European and Middle East Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-se-me-sets)

[10.10.5 Baltic Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-baltic-sets)

[10.10.6 Cyrillic Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-cyrillic-sets)

[10.10.7 Asian Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-asian-sets)

[10.10.8 The Binary Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-set)

This section indicates which character sets MySQL supports. There is one subsection for each group of related character sets. For each character set, the permissible collations are listed.

To list the available character sets and their default collations, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement or query the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table. For example:

mysql> **SHOW CHARACTER SET;**

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

| Charset | Description | Default collation | Maxlen |

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

| armscii8 | ARMSCII-8 Armenian | armscii8\_general\_ci | 1 |

| ascii | US ASCII | ascii\_general\_ci | 1 |

| big5 | Big5 Traditional Chinese | big5\_chinese\_ci | 2 |

| binary | Binary pseudo charset | binary | 1 |

| cp1250 | Windows Central European | cp1250\_general\_ci | 1 |

| cp1251 | Windows Cyrillic | cp1251\_general\_ci | 1 |

| cp1256 | Windows Arabic | cp1256\_general\_ci | 1 |

| cp1257 | Windows Baltic | cp1257\_general\_ci | 1 |

| cp850 | DOS West European | cp850\_general\_ci | 1 |

| cp852 | DOS Central European | cp852\_general\_ci | 1 |

| cp866 | DOS Russian | cp866\_general\_ci | 1 |

| cp932 | SJIS for Windows Japanese | cp932\_japanese\_ci | 2 |

| dec8 | DEC West European | dec8\_swedish\_ci | 1 |

| eucjpms | UJIS for Windows Japanese | eucjpms\_japanese\_ci | 3 |

| euckr | EUC-KR Korean | euckr\_korean\_ci | 2 |

| gb18030 | China National Standard GB18030 | gb18030\_chinese\_ci | 4 |

| gb2312 | GB2312 Simplified Chinese | gb2312\_chinese\_ci | 2 |

| gbk | GBK Simplified Chinese | gbk\_chinese\_ci | 2 |

| geostd8 | GEOSTD8 Georgian | geostd8\_general\_ci | 1 |

| greek | ISO 8859-7 Greek | greek\_general\_ci | 1 |

| hebrew | ISO 8859-8 Hebrew | hebrew\_general\_ci | 1 |

| hp8 | HP West European | hp8\_english\_ci | 1 |

| keybcs2 | DOS Kamenicky Czech-Slovak | keybcs2\_general\_ci | 1 |

| koi8r | KOI8-R Relcom Russian | koi8r\_general\_ci | 1 |

| koi8u | KOI8-U Ukrainian | koi8u\_general\_ci | 1 |

| latin1 | cp1252 West European | latin1\_swedish\_ci | 1 |

| latin2 | ISO 8859-2 Central European | latin2\_general\_ci | 1 |

| latin5 | ISO 8859-9 Turkish | latin5\_turkish\_ci | 1 |

| latin7 | ISO 8859-13 Baltic | latin7\_general\_ci | 1 |

| macce | Mac Central European | macce\_general\_ci | 1 |

| macroman | Mac West European | macroman\_general\_ci | 1 |

| sjis | Shift-JIS Japanese | sjis\_japanese\_ci | 2 |

| swe7 | 7bit Swedish | swe7\_swedish\_ci | 1 |

| tis620 | TIS620 Thai | tis620\_thai\_ci | 1 |

| ucs2 | UCS-2 Unicode | ucs2\_general\_ci | 2 |

| ujis | EUC-JP Japanese | ujis\_japanese\_ci | 3 |

| utf16 | UTF-16 Unicode | utf16\_general\_ci | 4 |

| utf16le | UTF-16LE Unicode | utf16le\_general\_ci | 4 |

| utf32 | UTF-32 Unicode | utf32\_general\_ci | 4 |

| utf8 | UTF-8 Unicode | utf8\_general\_ci | 3 |

| utf8mb4 | UTF-8 Unicode | utf8mb4\_0900\_ai\_ci | 4 |

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

In cases where a character set has multiple collations, it might not be clear which collation is most suitable for a given application. To avoid choosing the wrong collation, it can be helpful to perform some comparisons with representative data values to make sure that a given collation sorts values the way you expect.

### 10.10.1 Unicode Character Sets

This section describes the collations available for Unicode character sets and their differentiating properties. For general information about Unicode, see [Section 10.9, “Unicode Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode).

MySQL supports multiple Unicode character sets:

**utf8mb4**: A UTF-8 encoding of the Unicode character set using one to four bytes per character.

**utf8mb3**: A UTF-8 encoding of the Unicode character set using one to three bytes per character.

**utf8**: An alias for **utf8mb3**.

**ucs2**: The UCS-2 encoding of the Unicode character set using two bytes per character.

**utf16**: The UTF-16 encoding for the Unicode character set using two or four bytes per character. Like **ucs2** but with an extension for supplementary characters.

**utf16le**: The UTF-16LE encoding for the Unicode character set. Like **utf16** but little-endian rather than big-endian.

**utf32**: The UTF-32 encoding for the Unicode character set using four bytes per character.

**Note**

The **utf8mb3** character set is deprecated and you should expect it to be removed in a future MySQL release. Please use **utf8mb4** instead. Although **utf8** is currently an alias for **utf8mb3**, at some point **utf8** is expected to become a reference to **utf8mb4**. To avoid ambiguity about the meaning of **utf8**, consider specifying **utf8mb4** explicitly for character set references instead of **utf8**.

**utf8mb4**, **utf16**, **utf16le**, and **utf32** support Basic Multilingual Plane (BMP) characters and supplementary characters that lie outside the BMP. **utf8** and **ucs2** support only BMP characters.

Most Unicode character sets have a general collation (indicated by **\_general** in the name or by the absence of a language specifier), a binary collation (indicated by **\_bin** in the name), and several language-specific collations (indicated by language specifiers). For example, for **utf8mb4**, **utf8mb4\_general\_ci** and **utf8mb4\_bin** are its general and binary collations, and **utf8mb4\_danish\_ci** is one of its language-specific collations.

Most character sets have a single binary collation. **utf8mb4** is an exception that has two: **utf8mb4\_bin** and (as of MySQL 8.0.17) **utf8mb4\_0900\_bin**. These two binary collations have the same sort order but are distinguished by their pad attribute and collating weight characteristics. See [Collation Pad Attributes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-pad-attributes), and [Character Collating Weights](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-collating-weights).

Collation support for **utf16le** is limited. The only collations available are **utf16le\_general\_ci** and **utf16le\_bin**. These are similar to **utf16\_general\_ci** and **utf16\_bin**.

[Unicode Collation Algorithm (UCA) Versions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-uca)

[Collation Pad Attributes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-pad-attributes)

[Language-Specific Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-language-specific-collations)

[\_general\_ci Versus \_unicode\_ci Collations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-general-versus-unicode)

[Character Collating Weights](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-collating-weights)

[Miscellaneous Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-unicode-sets-miscellaneous)

#### Unicode Collation Algorithm (UCA) Versions

MySQL implements the ***xxx*\_unicode\_ci** collations according to the Unicode Collation Algorithm (UCA) described at <http://www.unicode.org/reports/tr10/>. The collation uses the version-4.0.0 UCA weight keys: <http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt>. The ***xxx*\_unicode\_ci** collations have only partial support for the Unicode Collation Algorithm. Some characters are not supported, and combining marks are not fully supported. This affects primarily Vietnamese, Yoruba, and some smaller languages such as Navajo. A combined character is considered different from the same character written with a single unicode character in string comparisons, and the two characters are considered to have a different length (for example, as returned by the [**CHAR\_LENGTH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_char-length) function or in result set metadata).

Unicode collations based on UCA versions higher than 4.0.0 include the version in the collation name. Examples:

**utf8mb4\_unicode\_520\_ci** is based on UCA 5.2.0 weight keys (<http://www.unicode.org/Public/UCA/5.2.0/allkeys.txt>),

**utf8mb4\_0900\_ai\_ci** is based on UCA 9.0.0 weight keys (<http://www.unicode.org/Public/UCA/9.0.0/allkeys.txt>).

The [**LOWER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_lower) and [**UPPER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_upper) functions perform case folding according to the collation of their argument. A character that has uppercase and lowercase versions only in a Unicode version higher than 4.0.0 is converted by these functions only if the argument collation uses a high enough UCA version.

#### Collation Pad Attributes

Collations based on UCA 9.0.0 and higher are faster than collations based on UCA versions prior to 9.0.0. They also have a pad attribute of **NO PAD**, in contrast to **PAD SPACE** as used in collations based on UCA versions prior to 9.0.0. For comparison of nonbinary strings, **NO PAD** collations treat spaces at the end of strings like any other character (see [Trailing Space Handling in Comparisons](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations-trailing-space-comparisons)).

To determine the pad attribute for a collation, use the **INFORMATION\_SCHEMA** [**COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table, which has a **PAD\_ATTRIBUTE** column. For example:

mysql> **SELECT COLLATION\_NAME, PAD\_ATTRIBUTE**

**FROM INFORMATION\_SCHEMA.COLLATIONS**

**WHERE CHARACTER\_SET\_NAME = 'utf8mb4';**

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

| COLLATION\_NAME | PAD\_ATTRIBUTE |

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

| utf8mb4\_general\_ci | PAD SPACE |

| utf8mb4\_bin | PAD SPACE |

| utf8mb4\_unicode\_ci | PAD SPACE |

| utf8mb4\_icelandic\_ci | PAD SPACE |

...

| utf8mb4\_0900\_ai\_ci | NO PAD |

| utf8mb4\_de\_pb\_0900\_ai\_ci | NO PAD |

| utf8mb4\_is\_0900\_ai\_ci | NO PAD |

...

| utf8mb4\_ja\_0900\_as\_cs | NO PAD |

| utf8mb4\_ja\_0900\_as\_cs\_ks | NO PAD |

| utf8mb4\_0900\_as\_ci | NO PAD |

| utf8mb4\_ru\_0900\_ai\_ci | NO PAD |

| utf8mb4\_ru\_0900\_as\_cs | NO PAD |

| utf8mb4\_zh\_0900\_as\_cs | NO PAD |

| utf8mb4\_0900\_bin | NO PAD |

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

Comparison of nonbinary string values (**CHAR**, **VARCHAR**, and **TEXT**) that have a **NO PAD** collation differ from other collations with respect to trailing spaces. For example, **'a'** and **'a '** compare as different strings, not the same string. This can be seen using the binary collations for **utf8mb4**. The pad attribute for **utf8mb4\_bin** is **PAD SPACE**, whereas for **utf8mb4\_0900\_bin** it is **NO PAD**. Consequently, operations involving **utf8mb4\_0900\_bin** do not add trailing spaces, and comparisons involving strings with trailing spaces may differ for the two collations:

mysql> **CREATE TABLE t1 (c CHAR(10) COLLATE utf8mb4\_bin);**

Query OK, 0 rows affected (0.03 sec)

mysql> **INSERT INTO t1 VALUES('a');**

Query OK, 1 row affected (0.01 sec)

mysql> **SELECT \* FROM t1 WHERE c = 'a ';**

+------+

| c |

+------+

| a |

+------+

1 row in set (0.00 sec)

mysql> **ALTER TABLE t1 MODIFY c CHAR(10) COLLATE utf8mb4\_0900\_bin;**

Query OK, 0 rows affected (0.02 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **SELECT \* FROM t1 WHERE c = 'a ';**

Empty set (0.00 sec)

#### Language-Specific Collations

MySQL implements language-specific Unicode collations if the ordering based only on the Unicode Collation Algorithm (UCA) does not work well for a language. Language-specific collations are UCA-based, with additional language tailoring rules. Examples of such rules appear later in this section. For questions about particular language orderings, [unicode.org](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\unicode.org) provides Common Locale Data Repository (CLDR) collation charts at <http://www.unicode.org/cldr/charts/30/collation/index.html>.

For example, the nonlanguage-specific **utf8mb4\_0900\_ai\_ci** and language-specific **utf8mb4\_*LOCALE*\_0900\_ai\_ci** Unicode collations each have these characteristics:

The collation is based on UCA 9.0.0 and CLDR v30, is accent-insensitive and case-insensitive. These characteristics are indicated by **\_0900**, **\_ai**, and **\_ci** in the collation name. Exception: **utf8mb4\_la\_0900\_ai\_ci** is not based on CLDR because Classical Latin is not defined in CLDR.

The collation works for all characters in the range [U+0, U+10FFFF].

If the collation is not language specific, it sorts all characters, including supplementary characters, in default order (described following). If the collation is language specific, it sorts characters of the language correctly according to language-specific rules, and characters not in the language in default order.

By default, the collation sorts characters having a code point listed in the DUCET table (Default Unicode Collation Element Table) according to the weight value assigned in the table. The collation sorts characters not having a code point listed in the DUCET table using their implicit weight value, which is constructed according to the UCA.

For non-language-specific collations, characters in contraction sequences are treated as separate characters. For language-specific collations, contractions might change character sorting order.

A collation name that includes a locale code or language name shown in the following table is a language-specific collation. Unicode character sets may include collations for one or more of these languages.

**Table 10.3 Unicode Collation Language Specifiers**

| **Language** | **Language Specifier** |
| --- | --- |
| Chinese | **zh** |
| Classical Latin | **la** or **roman** |
| Croatian | **hr** or **croatian** |
| Czech | **cs** or **czech** |
| Danish | **da** or **danish** |
| Esperanto | **eo** or **esperanto** |
| Estonian | **et** or **estonian** |
| German phone book order | **de\_pb** or **german2** |
| Hungarian | **hu** or **hungarian** |
| Icelandic | **is** or **icelandic** |
| Japanese | **ja** |
| Latvian | **lv** or **latvian** |
| Lithuanian | **lt** or **lithuanian** |
| Persian | **persian** |
| Polish | **pl** or **polish** |
| Romanian | **ro** or **romanian** |
| Russian | **ru** |
| Sinhala | **sinhala** |
| Slovak | **sk** or **slovak** |
| Slovenian | **sl** or **slovenian** |
| Modern Spanish | **es** or **spanish** |
| Traditional Spanish | **es\_trad** or **spanish2** |
| Swedish | **sv** or **swedish** |
| Turkish | **tr** or **turkish** |
| Vietnamese | **vi** or **vietnamese** |

Croatian collations are tailored for these Croatian letters: **Č**, **Ć**, **Dž**, **Đ**, **Lj**, **Nj**, **Š**, **Ž**.

Danish collations may also be used for Norwegian.

For Japanese, the **utf8mb4** character set includes **utf8mb4\_ja\_0900\_as\_cs** and **utf8mb4\_ja\_0900\_as\_cs\_ks** collations. Both collations are accent-sensitive and case-sensitive. **utf8mb4\_ja\_0900\_as\_cs\_ks** is also kana-sensitive and distinguishes Katakana characters from Hiragana characters, whereas **utf8mb4\_ja\_0900\_as\_cs** treats Katakana and Hiragana characters as equal for sorting. Applications that require a Japanese collation but not kana sensitivity may use **utf8mb4\_ja\_0900\_as\_cs** for better sort performance. **utf8mb4\_ja\_0900\_as\_cs** uses three weight levels for sorting; **utf8mb4\_ja\_0900\_as\_cs\_ks** uses four.

For Classical Latin collations that are accent-insensitive, **I** and **J** compare as equal, and **U** and **V** compare as equal. **I** and **J**, and **U** and **V** compare as equal on the base letter level. In other words, **J** is regarded as an accented **I**, and **U** is regarded as an accented **V**.

Spanish collations are available for modern and traditional Spanish. For both, **ñ** (n-tilde) is a separate letter between **n** and **o**. In addition, for traditional Spanish, **ch** is a separate letter between **c** and **d**, and **ll** is a separate letter between **l** and **m**.

Traditional Spanish collations may also be used for Asturian and Galician.

Swedish collations include Swedish rules. For example, in Swedish, the following relationship holds, which is not something expected by a German or French speaker:

Ü = Y < Ö

#### \_general\_ci Versus \_unicode\_ci Collations

For any Unicode character set, operations performed using the ***xxx*\_general\_ci** collation are faster than those for the ***xxx*\_unicode\_ci** collation. For example, comparisons for the **utf8\_general\_ci** collation are faster, but slightly less correct, than comparisons for **utf8\_unicode\_ci**. The reason is that **utf8\_unicode\_ci** supports mappings such as expansions; that is, when one character compares as equal to combinations of other characters. For example, **ß** is equal to **ss** in German and some other languages. **utf8\_unicode\_ci** also supports contractions and ignorable characters. **utf8\_general\_ci** is a legacy collation that does not support expansions, contractions, or ignorable characters. It can make only one-to-one comparisons between characters.

To further illustrate, the following equalities hold in both **utf8\_general\_ci** and **utf8\_unicode\_ci** (for the effect of this in comparisons or searches, see [Section 10.8.6, “Examples of the Effect of Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-effect)):

Ä = A

Ö = O

Ü = U

A difference between the collations is that this is true for **utf8\_general\_ci**:

ß = s

Whereas this is true for **utf8\_unicode\_ci**, which supports the German DIN-1 ordering (also known as dictionary order):

ß = ss

MySQL implements **utf8** language-specific collations if the ordering with **utf8\_unicode\_ci** does not work well for a language. For example, **utf8\_unicode\_ci** works fine for German dictionary order and French, so there is no need to create special **utf8** collations.

**utf8\_general\_ci** also is satisfactory for both German and French, except that **ß** is equal to **s**, and not to **ss**. If this is acceptable for your application, you should use **utf8\_general\_ci** because it is faster. If this is not acceptable (for example, if you require German dictionary order), use **utf8\_unicode\_ci** because it is more accurate.

If you require German DIN-2 (phone book) ordering, use the **utf8\_german2\_ci** collation, which compares the following sets of characters equal:

Ä = Æ = AE

Ö = Œ = OE

Ü = UE

ß = ss

**utf8\_german2\_ci** is similar to **latin1\_german2\_ci**, but the latter does not compare **Æ** equal to **AE** or **Œ** equal to **OE**. There is no **utf8\_german\_ci** corresponding to **latin1\_german\_ci** for German dictionary order because **utf8\_general\_ci** suffices.

#### Character Collating Weights

A character's collating weight is determined as follows:

For all Unicode collations except the **\_bin** (binary) collations, MySQL performs a table lookup to find a character's collating weight.

For **\_bin** collations except **utf8mb4\_0900\_bin**, the weight is based on the code point, possibly with leading zero bytes added.

For **utf8mb4\_0900\_bin**, the weight is the **utf8mb4** encoding bytes. The sort order is the same as for **utf8mb4\_bin**, but much faster.

Collating weights can be displayed using the [**WEIGHT\_STRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) function. (See [Section 12.8, “String Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#string-functions).) If a collation uses a weight lookup table, but a character is not in the table (for example, because it is a “new” character), collating weight determination becomes more complex:

For BMP characters in general collations (***xxx*\_general\_ci**), the weight is the code point.

For BMP characters in UCA collations (for example, ***xxx*\_unicode\_ci** and language-specific collations), the following algorithm applies:

if (code >= 0x3400 && code <= 0x4DB5)

base= 0xFB80; /\* CJK Ideograph Extension \*/

else if (code >= 0x4E00 && code <= 0x9FA5)

base= 0xFB40; /\* CJK Ideograph \*/

else

base= 0xFBC0; /\* All other characters \*/

aaaa= base + (code >> 15);

bbbb= (code & 0x7FFF) | 0x8000;

The result is a sequence of two collating elements, **aaaa** followed by **bbbb**. For example:

mysql> **SELECT HEX(WEIGHT\_STRING(\_ucs2 0x04CF COLLATE ucs2\_unicode\_ci));**

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

| HEX(WEIGHT\_STRING(\_ucs2 0x04CF COLLATE ucs2\_unicode\_ci)) |

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

| FBC084CF |

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

Thus, **U+04cf CYRILLIC SMALL LETTER PALOCHKA** is, with all UCA 4.0.0 collations, greater than **U+04c0 CYRILLIC LETTER PALOCHKA**. With UCA 5.2.0 collations, all palochkas sort together.

For supplementary characters in general collations, the weight is the weight for **0xfffd REPLACEMENT CHARACTER**. For supplementary characters in UCA 4.0.0 collations, their collating weight is **0xfffd**. That is, to MySQL, all supplementary characters are equal to each other, and greater than almost all BMP characters.

An example with Deseret characters and **COUNT(DISTINCT)**:

CREATE TABLE t (s1 VARCHAR(5) CHARACTER SET utf32 COLLATE utf32\_unicode\_ci);

INSERT INTO t VALUES (0xfffd); /\* REPLACEMENT CHARACTER \*/

INSERT INTO t VALUES (0x010412); /\* DESERET CAPITAL LETTER BEE \*/

INSERT INTO t VALUES (0x010413); /\* DESERET CAPITAL LETTER TEE \*/

SELECT COUNT(DISTINCT s1) FROM t;

The result is 2 because in the MySQL ***xxx*\_unicode\_ci** collations, the replacement character has a weight of **0x0dc6**, whereas Deseret Bee and Deseret Tee both have a weight of **0xfffd**. (Were the **utf32\_general\_ci** collation used instead, the result is 1 because all three characters have a weight of **0xfffd** in that collation.)

An example with cuneiform characters and [**WEIGHT\_STRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string):

/\*

The four characters in the INSERT string are

00000041 # LATIN CAPITAL LETTER A

0001218F # CUNEIFORM SIGN KAB

000121A7 # CUNEIFORM SIGN KISH

00000042 # LATIN CAPITAL LETTER B

\*/

CREATE TABLE t (s1 CHAR(4) CHARACTER SET utf32 COLLATE utf32\_unicode\_ci);

INSERT INTO t VALUES (0x000000410001218f000121a700000042);

SELECT HEX(WEIGHT\_STRING(s1)) FROM t;

The result is:

0E33 FFFD FFFD 0E4A

**0E33** and **0E4A** are primary weights as in [UCA 4.0.0](ftp://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt). **FFFD** is the weight for KAB and also for KISH.

The rule that all supplementary characters are equal to each other is nonoptimal but is not expected to cause trouble. These characters are very rare, so it is very rare that a multi-character string consists entirely of supplementary characters. In Japan, since the supplementary characters are obscure Kanji ideographs, the typical user does not care what order they are in, anyway. If you really want rows sorted by the MySQL rule and secondarily by code point value, it is easy:

ORDER BY s1 COLLATE utf32\_unicode\_ci, s1 COLLATE utf32\_bin

For supplementary characters based on UCA versions higher than 4.0.0 (for example, ***xxx*\_unicode\_520\_ci**), supplementary characters do not necessarily all have the same collating weight. Some have explicit weights from the UCA allkeys.txt file. Others have weights calculated from this algorithm:

aaaa= base + (code >> 15);

bbbb= (code & 0x7FFF) | 0x8000;

There is a difference between “ordering by the character's code value” and “ordering by the character's binary representation,” a difference that appears only with **utf16\_bin**, because of surrogates.

Suppose that **utf16\_bin** (the binary collation for **utf16**) was a binary comparison “byte by byte” rather than “character by character.” If that were so, the order of characters in **utf16\_bin** would differ from the order in **utf8\_bin**. For example, the following chart shows two rare characters. The first character is in the range **E000**-**FFFF**, so it is greater than a surrogate but less than a supplementary. The second character is a supplementary.

Code point Character utf8 utf16

---------- --------- ---- -----

0FF9D HALFWIDTH KATAKANA LETTER N EF BE 9D FF 9D

10384 UGARITIC LETTER DELTA F0 90 8E 84 D8 00 DF 84

The two characters in the chart are in order by code point value because **0xff9d** < **0x10384**. And they are in order by **utf8** value because **0xef** < **0xf0**. But they are not in order by **utf16** value, if we use byte-by-byte comparison, because **0xff** > **0xd8**.

So MySQL's **utf16\_bin** collation is not “byte by byte.” It is “by code point.” When MySQL sees a supplementary-character encoding in **utf16**, it converts to the character's code-point value, and then compares. Therefore, **utf8\_bin** and **utf16\_bin** are the same ordering. This is consistent with the SQL:2008 standard requirement for a UCS\_BASIC collation: “UCS\_BASIC is a collation in which the ordering is determined entirely by the Unicode scalar values of the characters in the strings being sorted. It is applicable to the UCS character repertoire. Since every character repertoire is a subset of the UCS repertoire, the UCS\_BASIC collation is potentially applicable to every character set. NOTE 11: The Unicode scalar value of a character is its code point treated as an unsigned integer.”

If the character set is **ucs2**, comparison is byte-by-byte, but **ucs2** strings should not contain surrogates, anyway.

#### Miscellaneous Information

The ***xxx*\_general\_mysql500\_ci** collations preserve the pre-5.1.24 ordering of the original ***xxx*\_general\_ci** collations and permit upgrades for tables created before MySQL 5.1.24 (Bug #27877).

### 10.10.2 West European Character Sets

Western European character sets cover most West European languages, such as French, Spanish, Catalan, Basque, Portuguese, Italian, Albanian, Dutch, German, Danish, Swedish, Norwegian, Finnish, Faroese, Icelandic, Irish, Scottish, and English.

**ascii** (US ASCII) collations:

**ascii\_bin**

**ascii\_general\_ci** (default)

**cp850** (DOS West European) collations:

**cp850\_bin**

**cp850\_general\_ci** (default)

**dec8** (DEC Western European) collations:

**dec8\_bin**

**dec8\_swedish\_ci** (default)

**hp8** (HP Western European) collations:

**hp8\_bin**

**hp8\_english\_ci** (default)

**latin1** (cp1252 West European) collations:

**latin1\_bin**

**latin1\_danish\_ci**

**latin1\_general\_ci**

**latin1\_general\_cs**

**latin1\_german1\_ci**

**latin1\_german2\_ci**

**latin1\_spanish\_ci**

**latin1\_swedish\_ci** (default)

MySQL's **latin1** is the same as the Windows **cp1252** character set. This means it is the same as the official **ISO 8859-1** or IANA (Internet Assigned Numbers Authority) **latin1**, except that IANA **latin1** treats the code points between **0x80** and **0x9f** as “undefined,” whereas **cp1252**, and therefore MySQL's **latin1**, assign characters for those positions. For example, **0x80** is the Euro sign. For the “undefined” entries in **cp1252**, MySQL translates **0x81** to Unicode **0x0081**, **0x8d** to **0x008d**, **0x8f** to **0x008f**, **0x90** to **0x0090**, and **0x9d** to **0x009d**.

The **latin1\_swedish\_ci** collation is the default that probably is used by the majority of MySQL customers. Although it is frequently said that it is based on the Swedish/Finnish collation rules, there are Swedes and Finns who disagree with this statement.

The **latin1\_german1\_ci** and **latin1\_german2\_ci** collations are based on the DIN-1 and DIN-2 standards, where DIN stands for *Deutsches Institut für Normung* (the German equivalent of ANSI). DIN-1 is called the “dictionary collation” and DIN-2 is called the “phone book collation.” For an example of the effect this has in comparisons or when doing searches, see [Section 10.8.6, “Examples of the Effect of Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-effect).

**latin1\_german1\_ci** (dictionary) rules:

Ä = A

Ö = O

Ü = U

ß = s

**latin1\_german2\_ci** (phone-book) rules:

Ä = AE

Ö = OE

Ü = UE

ß = ss

In the **latin1\_spanish\_ci** collation, **ñ** (n-tilde) is a separate letter between **n** and **o**.

**macroman** (Mac West European) collations:

**macroman\_bin**

**macroman\_general\_ci** (default)

**swe7** (7bit Swedish) collations:

**swe7\_bin**

**swe7\_swedish\_ci** (default)

### 10.10.3 Central European Character Sets

MySQL provides some support for character sets used in the Czech Republic, Slovakia, Hungary, Romania, Slovenia, Croatia, Poland, and Serbia (Latin).

**cp1250** (Windows Central European) collations:

**cp1250\_bin**

**cp1250\_croatian\_ci**

**cp1250\_czech\_cs**

**cp1250\_general\_ci** (default)

**cp1250\_polish\_ci**

**cp852** (DOS Central European) collations:

**cp852\_bin**

**cp852\_general\_ci** (default)

**keybcs2** (DOS Kamenicky Czech-Slovak) collations:

**keybcs2\_bin**

**keybcs2\_general\_ci** (default)

**latin2** (ISO 8859-2 Central European) collations:

**latin2\_bin**

**latin2\_croatian\_ci**

**latin2\_czech\_cs**

**latin2\_general\_ci** (default)

**latin2\_hungarian\_ci**

**macce** (Mac Central European) collations:

**macce\_bin**

**macce\_general\_ci** (default)

### 10.10.4 South European and Middle East Character Sets

South European and Middle Eastern character sets supported by MySQL include Armenian, Arabic, Georgian, Greek, Hebrew, and Turkish.

**armscii8** (ARMSCII-8 Armenian) collations:

**armscii8\_bin**

**armscii8\_general\_ci** (default)

**cp1256** (Windows Arabic) collations:

**cp1256\_bin**

**cp1256\_general\_ci** (default)

**geostd8** (GEOSTD8 Georgian) collations:

**geostd8\_bin**

**geostd8\_general\_ci** (default)

**greek** (ISO 8859-7 Greek) collations:

**greek\_bin**

**greek\_general\_ci** (default)

**hebrew** (ISO 8859-8 Hebrew) collations:

**hebrew\_bin**

**hebrew\_general\_ci** (default)

**latin5** (ISO 8859-9 Turkish) collations:

**latin5\_bin**

**latin5\_turkish\_ci** (default)

### 10.10.5 Baltic Character Sets

The Baltic character sets cover Estonian, Latvian, and Lithuanian languages.

**cp1257** (Windows Baltic) collations:

**cp1257\_bin**

**cp1257\_general\_ci** (default)

**cp1257\_lithuanian\_ci**

**latin7** (ISO 8859-13 Baltic) collations:

**latin7\_bin**

**latin7\_estonian\_cs**

**latin7\_general\_ci** (default)

**latin7\_general\_cs**

### 10.10.6 Cyrillic Character Sets

The Cyrillic character sets and collations are for use with Belarusian, Bulgarian, Russian, Ukrainian, and Serbian (Cyrillic) languages.

**cp1251** (Windows Cyrillic) collations:

**cp1251\_bin**

**cp1251\_bulgarian\_ci**

**cp1251\_general\_ci** (default)

**cp1251\_general\_cs**

**cp1251\_ukrainian\_ci**

**cp866** (DOS Russian) collations:

**cp866\_bin**

**cp866\_general\_ci** (default)

**koi8r** (KOI8-R Relcom Russian) collations:

**koi8r\_bin**

**koi8r\_general\_ci** (default)

**koi8u** (KOI8-U Ukrainian) collations:

**koi8u\_bin**

**koi8u\_general\_ci** (default)

### 10.10.7 Asian Character Sets

[10.10.7.1 The cp932 Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-cp932)

[10.10.7.2 The gb18030 Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-gb18030)

The Asian character sets that we support include Chinese, Japanese, Korean, and Thai. These can be complicated. For example, the Chinese sets must allow for thousands of different characters. See [Section 10.10.7.1, “The cp932 Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-cp932), for additional information about the **cp932** and **sjis** character sets. See [Section 10.10.7.2, “The gb18030 Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-gb18030), for additional information about character set support for the Chinese National Standard GB 18030.

For answers to some common questions and problems relating support for Asian character sets in MySQL, see [Section A.11, “MySQL 8.0 FAQ: MySQL Chinese, Japanese, and Korean Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\faqs.html#faqs-cjk).

**big5** (Big5 Traditional Chinese) collations:

**big5\_bin**

**big5\_chinese\_ci** (default)

[**cp932**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-cp932) (SJIS for Windows Japanese) collations:

**cp932\_bin**

**cp932\_japanese\_ci** (default)

**eucjpms** (UJIS for Windows Japanese) collations:

**eucjpms\_bin**

**eucjpms\_japanese\_ci** (default)

**euckr** (EUC-KR Korean) collations:

**euckr\_bin**

**euckr\_korean\_ci** (default)

**gb2312** (GB2312 Simplified Chinese) collations:

**gb2312\_bin**

**gb2312\_chinese\_ci** (default)

**gbk** (GBK Simplified Chinese) collations:

**gbk\_bin**

**gbk\_chinese\_ci** (default)

[**gb18030**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-gb18030) (China National Standard GB18030) collations:

**gb18030\_bin**

**gb18030\_chinese\_ci** (default)

**gb18030\_unicode\_520\_ci**

**sjis** (Shift-JIS Japanese) collations:

**sjis\_bin**

**sjis\_japanese\_ci** (default)

**tis620** (TIS620 Thai) collations:

**tis620\_bin**

**tis620\_thai\_ci** (default)

**ujis** (EUC-JP Japanese) collations:

**ujis\_bin**

**ujis\_japanese\_ci** (default)

The **big5\_chinese\_ci** collation sorts on number of strokes.

#### 10.10.7.1 The cp932 Character Set

***Why is cp932 needed?***

In MySQL, the **sjis** character set corresponds to the **Shift\_JIS** character set defined by IANA, which supports JIS X0201 and JIS X0208 characters. (See <http://www.iana.org/assignments/character-sets>.)

However, the meaning of “SHIFT JIS” as a descriptive term has become very vague and it often includes the extensions to **Shift\_JIS** that are defined by various vendors.

For example, “SHIFT JIS” used in Japanese Windows environments is a Microsoft extension of **Shift\_JIS** and its exact name is **Microsoft Windows Codepage : 932** or **cp932**. In addition to the characters supported by **Shift\_JIS**, **cp932** supports extension characters such as NEC special characters, NEC selected—IBM extended characters, and IBM selected characters.

Many Japanese users have experienced problems using these extension characters. These problems stem from the following factors:

MySQL automatically converts character sets.

Character sets are converted using Unicode (**ucs2**).

The **sjis** character set does not support the conversion of these extension characters.

There are several conversion rules from so-called “SHIFT JIS” to Unicode, and some characters are converted to Unicode differently depending on the conversion rule. MySQL supports only one of these rules (described later).

The MySQL **cp932** character set is designed to solve these problems.

Because MySQL supports character set conversion, it is important to separate IANA **Shift\_JIS** and **cp932** into two different character sets because they provide different conversion rules.

***How does cp932 differ from sjis?***

The **cp932** character set differs from **sjis** in the following ways:

**cp932** supports NEC special characters, NEC selected—IBM extended characters, and IBM selected characters.

Some **cp932** characters have two different code points, both of which convert to the same Unicode code point. When converting from Unicode back to **cp932**, one of the code points must be selected. For this “round trip conversion,” the rule recommended by Microsoft is used. (See <http://support.microsoft.com/kb/170559/EN-US/>.)

The conversion rule works like this:

If the character is in both JIS X 0208 and NEC special characters, use the code point of JIS X 0208.

If the character is in both NEC special characters and IBM selected characters, use the code point of NEC special characters.

If the character is in both IBM selected characters and NEC selected—IBM extended characters, use the code point of IBM extended characters.

The table shown at <https://msdn.microsoft.com/en-us/goglobal/cc305152.aspx> provides information about the Unicode values of **cp932** characters. For **cp932** table entries with characters under which a four-digit number appears, the number represents the corresponding Unicode (**ucs2**) encoding. For table entries with an underlined two-digit value appears, there is a range of **cp932** character values that begin with those two digits. Clicking such a table entry takes you to a page that displays the Unicode value for each of the **cp932** characters that begin with those digits.

The following links are of special interest. They correspond to the encodings for the following sets of characters:

NEC special characters (lead byte **0x87**):

<https://msdn.microsoft.com/en-us/goglobal/gg674964>

NEC selected—IBM extended characters (lead byte **0xED** and **0xEE**):

<https://msdn.microsoft.com/en-us/goglobal/gg671837>

<https://msdn.microsoft.com/en-us/goglobal/gg671838>

IBM selected characters (lead byte **0xFA**, **0xFB**, **0xFC**):

<https://msdn.microsoft.com/en-us/goglobal/gg671839>

<https://msdn.microsoft.com/en-us/goglobal/gg671840>

<https://msdn.microsoft.com/en-us/goglobal/gg671841>

**cp932** supports conversion of user-defined characters in combination with **eucjpms**, and solves the problems with **sjis**/**ujis** conversion. For details, please refer to <http://www.sljfaq.org/afaq/encodings.html>.

For some characters, conversion to and from **ucs2** is different for **sjis** and **cp932**. The following tables illustrate these differences.

Conversion to **ucs2**:

| **sjis/cp932 Value** | **sjis -> ucs2 Conversion** | **cp932 -> ucs2 Conversion** |
| --- | --- | --- |
| **5C** | 005C | 005C |
| **7E** | 007E | 007E |
| **815C** | 2015 | 2015 |
| **815F** | 005C | FF3C |
| **8160** | 301C | FF5E |
| **8161** | 2016 | 2225 |
| **817C** | 2212 | FF0D |
| **8191** | 00A2 | FFE0 |
| **8192** | 00A3 | FFE1 |
| **81CA** | 00AC | FFE2 |

Conversion from **ucs2**:

| **ucs2 value** | **ucs2 -> sjis Conversion** | **ucs2 -> cp932 Conversion** |
| --- | --- | --- |
| **005C** | 815F | 5C |
| **007E** | 7E | 7E |
| **00A2** | 8191 | 3F |
| **00A3** | 8192 | 3F |
| **00AC** | 81CA | 3F |
| **2015** | 815C | 815C |
| **2016** | 8161 | 3F |
| **2212** | 817C | 3F |
| **2225** | 3F | 8161 |
| **301C** | 8160 | 3F |
| **FF0D** | 3F | 817C |
| **FF3C** | 3F | 815F |
| **FF5E** | 3F | 8160 |
| **FFE0** | 3F | 8191 |
| **FFE1** | 3F | 8192 |
| **FFE2** | 3F | 81CA |

Users of any Japanese character sets should be aware that using [--character-set-client-handshake](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_character-set-client-handshake) (or [--skip-character-set-client-handshake](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_character-set-client-handshake)) has an important effect. See [Section 5.1.7, “Server Command Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-options).

#### 10.10.7.2 The gb18030 Character Set

In MySQL, the **gb18030** character set corresponds to the “Chinese National Standard GB 18030-2005: Information technology — Chinese coded character set”, which is the official character set of the People's Republic of China (PRC).

##### Characteristics of the MySQL gb18030 Character Set

Supports all code points defined by the GB 18030-2005 standard. Unassigned code points in the ranges (GB+8431A439, GB+90308130) and (GB+E3329A36, GB+EF39EF39) are treated as '**?**' (0x3F). Conversion of unassigned code points return '**?**'.

Supports UPPER and LOWER conversion for all GB18030 code points. Case folding defined by Unicode is also supported (based on CaseFolding-6.3.0.txt).

Supports Conversion of data to and from other character sets.

Supports SQL statements such as [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names).

Supports comparison between **gb18030** strings, and between **gb18030** strings and strings of other character sets. There is a conversion if strings have different character sets. Comparisons that include or ignore trailing spaces are also supported.

The private use area (U+E000, U+F8FF) in Unicode is mapped to **gb18030**.

There is no mapping between (U+D800, U+DFFF) and GB18030. Attempted conversion of code points in this range returns '**?**'.

If an incoming sequence is illegal, an error or warning is returned. If an illegal sequence is used in **CONVERT()**, an error is returned. Otherwise, a warning is returned.

For consistency with **utf8** and **utf8mb4**, UPPER is not supported for ligatures.

Searches for ligatures also match uppercase ligatures when using the **gb18030\_unicode\_520\_ci** collation.

If a character has more than one uppercase character, the chosen uppercase character is the one whose lowercase is the character itself.

The minimum multibyte length is 1 and the maximum is 4. The character set determines the length of a sequence using the first 1 or 2 bytes.

##### Supported Collations

**gb18030\_bin**: A binary collation.

**gb18030\_chinese\_ci**: The default collation, which supports Pinyin. Sorting of non-Chinese characters is based on the order of the original sort key. The original sort key is **GB(UPPER(ch))** if **UPPER(ch)** exists. Otherwise, the original sort key is **GB(ch)**. Chinese characters are sorted according to the Pinyin collation defined in the Unicode Common Locale Data Repository (CLDR 24). Non-Chinese characters are sorted before Chinese characters with the exception of **GB+FE39FE39**, which is the code point maximum.

**gb18030\_unicode\_520\_ci**: A Unicode collation. Use this collation if you need to ensure that ligatures are sorted correctly.

### 10.10.8 The Binary Character Set

The **binary** character set is the character set for binary strings, which are sequences of bytes. The **binary** character set has one collation, also named **binary**. Comparison and sorting are based on numeric byte values, rather than on numeric character code values (which for multibyte characters differ from numeric byte values). For information about the differences between the **binary** collation of the **binary** character set and the **\_bin** collations of nonbinary character sets, see [Section 10.8.5, “The binary Collation Compared to \_bin Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-binary-collations).

For the **binary** character set, the concepts of lettercase and accent equivalence do not apply:

For single-byte characters stored as binary strings, character and byte boundaries are the same, so lettercase and accent differences are significant in comparisons. That is, the **binary** collation is case-sensitive and accent-sensitive.

mysql> **SET NAMES 'binary';**

mysql> **SELECT CHARSET('abc'), COLLATION('abc');**

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

| CHARSET('abc') | COLLATION('abc') |

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

| binary | binary |

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

mysql> **SELECT 'abc' = 'ABC', 'a' = 'ä';**

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

| 'abc' = 'ABC' | 'a' = 'ä' |

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

| 0 | 0 |

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

For multibyte characters stored as binary strings, character and byte boundaries differ. Character boundaries are lost, so comparisons that depend on them are not meaningful.

To perform lettercase conversion of a binary string, first convert it to a nonbinary string using a character set appropriate for the data stored in the string:

mysql> **SET @str = BINARY 'New York';**

mysql> **SELECT LOWER(@str), LOWER(CONVERT(@str USING utf8mb4));**

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

| LOWER(@str) | LOWER(CONVERT(@str USING utf8mb4)) |

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

| New York | new york |

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

To convert a string expression to a binary string, these constructs are equivalent:

BINARY ***expr***

CAST(***expr*** AS BINARY)

CONVERT(***expr*** USING BINARY)

If a value is a character string literal, the **\_binary** introducer may be used to designate it as a binary string. For example:

\_binary 'a'

The **\_binary** introducer is permitted for hexadecimal literals and bit-value literals as well, but unnecessary; such literals are binary strings by default.

For more information about introducers, see [Section 10.3.8, “Character Set Introducers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-introducer).

## 10.11 Restrictions on Character Sets

Identifiers are stored in **mysql** database tables (**user**, **db**, and so forth) using **utf8**, but identifiers can contain only characters in the Basic Multilingual Plane (BMP). Supplementary characters are not permitted in identifiers.

The **ucs2**, **utf16**, **utf16le**, and **utf32** character sets have the following restrictions:

None of them can be used as the client character set. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

It is currently not possible to use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) to load data files that use these character sets.

**FULLTEXT** indexes cannot be created on a column that uses any of these character sets. However, you can perform **IN BOOLEAN MODE** searches on the column without an index.

The [**REGEXP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) and [**RLIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_regexp) operators work in byte-wise fashion, so they are not multibyte safe and may produce unexpected results with multibyte character sets. In addition, these operators compare characters by their byte values and accented characters may not compare as equal even if a given collation treats them as equal.

## 10.12 Setting the Error Message Language

By default, [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) produces error messages in English, but they can be displayed instead in any of several other languages: Czech, Danish, Dutch, Estonian, French, German, Greek, Hungarian, Italian, Japanese, Korean, Norwegian, Norwegian-ny, Polish, Portuguese, Romanian, Russian, Slovak, Spanish, or Swedish. This applies to messages the server writes to the error log and sends to clients.

To select the language in which the server writes error messages, follow the instructions in this section. For information about changing the character set for error messages (rather than the language), see [Section 10.6, “Error Message Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-errors). For general information about configuring error logging, see [Section 5.4.2, “The Error Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log).

The server searches for the error message file using these rules:

It looks for the file in a directory constructed from two system variable values, [**lc\_messages\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages_dir) and [**lc\_messages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages), with the latter converted to a language name. Suppose that you start the server using this command:

mysqld --lc\_messages\_dir=/usr/share/mysql --lc\_messages=fr\_FR

In this case, [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) maps the locale **fr\_FR** to the language **french** and looks for the error file in the /usr/share/mysql/french directory.

By default, the language files are located in the share/mysql/***LANGUAGE*** directory under the MySQL base directory.

If the message file cannot be found in the directory constructed as just described, the server ignores the [**lc\_messages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages) value and uses only the [**lc\_messages\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages_dir) value as the location in which to look.

If the server cannot find the configured message file, it writes a message to the error log to indicate the problem and defaults to built-in English messages.

The [**lc\_messages\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages_dir) system variable can be set only at server startup and has only a global read-only value at runtime. [**lc\_messages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages) can be set at server startup and has global and session values that can be modified at runtime. Thus, the error message language can be changed while the server is running, and each client can have its own error message language by setting its session [**lc\_messages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_messages) value to the desired locale name. For example, if the server is using the **fr\_FR** locale for error messages, a client can execute this statement to receive error messages in English:

SET lc\_messages = 'en\_US';

## 10.13 Adding a Character Set

[10.13.1 Character Definition Arrays](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#character-arrays)

[10.13.2 String Collating Support for Complex Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#string-collating)

[10.13.3 Multi-Byte Character Support for Complex Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#multibyte-characters)

This section discusses the procedure for adding a character set to MySQL. The proper procedure depends on whether the character set is simple or complex:

If the character set does not need special string collating routines for sorting and does not need multibyte character support, it is simple.

If the character set needs either of those features, it is complex.

For example, **greek** and **swe7** are simple character sets, whereas **big5** and **czech** are complex character sets.

To use the following instructions, you must have a MySQL source distribution. In the instructions, ***MYSET*** represents the name of the character set that you want to add.

Add a **<charset>** element for ***MYSET*** to the sql/share/charsets/Index.xml file. Use the existing contents in the file as a guide to adding new contents. A partial listing for the **latin1** **<charset>** element follows:

<charset name="latin1">

<family>Western</family>

<description>cp1252 West European</description>

...

<collation name="latin1\_swedish\_ci" id="8" order="Finnish, Swedish">

<flag>primary</flag>

<flag>compiled</flag>

</collation>

<collation name="latin1\_danish\_ci" id="15" order="Danish"/>

...

<collation name="latin1\_bin" id="47" order="Binary">

<flag>binary</flag>

<flag>compiled</flag>

</collation>

...

</charset>

The **<charset>** element must list all the collations for the character set. These must include at least a binary collation and a default (primary) collation. The default collation is often named using a suffix of **general\_ci** (general, case-insensitive). It is possible for the binary collation to be the default collation, but usually they are different. The default collation should have a **primary** flag. The binary collation should have a **binary** flag.

You must assign a unique ID number to each collation. The range of IDs from 1024 to 2047 is reserved for user-defined collations. To find the maximum of the currently used collation IDs, use this query:

SELECT MAX(ID) FROM INFORMATION\_SCHEMA.COLLATIONS;

This step depends on whether you are adding a simple or complex character set. A simple character set requires only a configuration file, whereas a complex character set requires C source file that defines collation functions, multibyte functions, or both.

For a simple character set, create a configuration file, ***MYSET***.xml, that describes the character set properties. Create this file in the sql/share/charsets directory. You can use a copy of latin1.xml as the basis for this file. The syntax for the file is very simple:

Comments are written as ordinary XML comments (**<!-- *text* -->**).

Words within **<map>** array elements are separated by arbitrary amounts of whitespace.

Each word within **<map>** array elements must be a number in hexadecimal format.

The **<map>** array element for the **<ctype>** element has 257 words. The other **<map>** array elements after that have 256 words. See [Section 10.13.1, “Character Definition Arrays”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#character-arrays).

For each collation listed in the **<charset>** element for the character set in Index.xml, ***MYSET***.xml must contain a **<collation>** element that defines the character ordering.

For a complex character set, create a C source file that describes the character set properties and defines the support routines necessary to properly perform operations on the character set:

Create the file ctype-***MYSET***.c in the strings directory. Look at one of the existing ctype-\*.c files (such as ctype-big5.c) to see what needs to be defined. The arrays in your file must have names like **ctype\_*MYSET***, **to\_lower\_*MYSET***, and so on. These correspond to the arrays for a simple character set. See [Section 10.13.1, “Character Definition Arrays”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#character-arrays).

For each **<collation>** element listed in the **<charset>** element for the character set in Index.xml, the ctype-***MYSET***.c file must provide an implementation of the collation.

If the character set requires string collating functions, see [Section 10.13.2, “String Collating Support for Complex Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#string-collating).

If the character set requires multibyte character support, see [Section 10.13.3, “Multi-Byte Character Support for Complex Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#multibyte-characters).

Modify the configuration information. Use the existing configuration information as a guide to adding information for ***MYSYS***. The example here assumes that the character set has default and binary collations, but more lines are needed if ***MYSET*** has additional collations.

Edit mysys/charset-def.c, and “register” the collations for the new character set.

Add these lines to the “declaration” section:

#ifdef HAVE\_CHARSET\_***MYSET***

extern CHARSET\_INFO my\_charset\_***MYSET***\_general\_ci;

extern CHARSET\_INFO my\_charset\_***MYSET***\_bin;

#endif

Add these lines to the “registration” section:

#ifdef HAVE\_CHARSET\_***MYSET***

add\_compiled\_collation(&my\_charset\_***MYSET***\_general\_ci);

add\_compiled\_collation(&my\_charset\_***MYSET***\_bin);

#endif

If the character set uses ctype-***MYSET***.c, edit strings/CMakeLists.txt and add ctype-***MYSET***.c to the definition of the **STRINGS\_SOURCES** variable.

Edit cmake/character\_sets.cmake:

Add ***MYSET*** to the value of with **CHARSETS\_AVAILABLE** in alphabetic order.

Add ***MYSET*** to the value of **CHARSETS\_COMPLEX** in alphabetic order. This is needed even for simple character sets, so that **CMake** can recognize [-DDEFAULT\_CHARSET=***MYSET***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_default_charset).

Reconfigure, recompile, and test.

### 10.13.1 Character Definition Arrays

Each simple character set has a configuration file located in the sql/share/charsets directory. For a character set named ***MYSYS***, the file is named ***MYSET***.xml. It uses **<map>** array elements to list character set properties. **<map>** elements appear within these elements:

**<ctype>** defines attributes for each character.

**<lower>** and **<upper>** list the lowercase and uppercase characters.

**<unicode>** maps 8-bit character values to Unicode values.

**<collation>** elements indicate character ordering for comparison and sorting, one element per collation. Binary collations need no **<map>** element because the character codes themselves provide the ordering.

For a complex character set as implemented in a ctype-***MYSET***.c file in the strings directory, there are corresponding arrays: **ctype\_*MYSET*[]**, **to\_lower\_*MYSET*[]**, and so forth. Not every complex character set has all of the arrays. See also the existing ctype-\*.c files for examples. See the CHARSET\_INFO.txt file in the strings directory for additional information.

Most of the arrays are indexed by character value and have 256 elements. The **<ctype>** array is indexed by character value + 1 and has 257 elements. This is a legacy convention for handling **EOF**.

**<ctype>** array elements are bit values. Each element describes the attributes of a single character in the character set. Each attribute is associated with a bitmask, as defined in include/m\_ctype.h:

#define \_MY\_U 01 /\* Upper case \*/

#define \_MY\_L 02 /\* Lower case \*/

#define \_MY\_NMR 04 /\* Numeral (digit) \*/

#define \_MY\_SPC 010 /\* Spacing character \*/

#define \_MY\_PNT 020 /\* Punctuation \*/

#define \_MY\_CTR 040 /\* Control character \*/

#define \_MY\_B 0100 /\* Blank \*/

#define \_MY\_X 0200 /\* heXadecimal digit \*/

The **<ctype>** value for a given character should be the union of the applicable bitmask values that describe the character. For example, **'A'** is an uppercase character (**\_MY\_U**) as well as a hexadecimal digit (**\_MY\_X**), so its **ctype** value should be defined like this:

ctype['A'+1] = \_MY\_U | \_MY\_X = 01 | 0200 = 0201

The bitmask values in m\_ctype.h are octal values, but the elements of the **<ctype>** array in ***MYSET***.xml should be written as hexadecimal values.

The **<lower>** and **<upper>** arrays hold the lowercase and uppercase characters corresponding to each member of the character set. For example:

lower['A'] should contain 'a'

upper['a'] should contain 'A'

Each **<collation>** array indicates how characters should be ordered for comparison and sorting purposes. MySQL sorts characters based on the values of this information. In some cases, this is the same as the **<upper>** array, which means that sorting is case-insensitive. For more complicated sorting rules (for complex character sets), see the discussion of string collating in [Section 10.13.2, “String Collating Support for Complex Character Sets”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#string-collating).

### 10.13.2 String Collating Support for Complex Character Sets

For a simple character set named ***MYSET***, sorting rules are specified in the ***MYSET***.xml configuration file using **<map>** array elements within **<collation>** elements. If the sorting rules for your language are too complex to be handled with simple arrays, you must define string collating functions in the ctype-***MYSET***.c source file in the strings directory.

The existing character sets provide the best documentation and examples to show how these functions are implemented. Look at the ctype-\*.c files in the strings directory, such as the files for the **big5**, **czech**, **gbk**, **sjis**, and **tis160** character sets. Take a look at the **MY\_COLLATION\_HANDLER** structures to see how they are used. See also the CHARSET\_INFO.txt file in the strings directory for additional information.

### 10.13.3 Multi-Byte Character Support for Complex Character Sets

If you want to add support for a new character set named ***MYSET*** that includes multibyte characters, you must use multibyte character functions in the ctype-***MYSET***.c source file in the strings directory.

The existing character sets provide the best documentation and examples to show how these functions are implemented. Look at the ctype-\*.c files in the strings directory, such as the files for the **euc\_kr**, **gb2312**, **gbk**, **sjis**, and **ujis** character sets. Take a look at the **MY\_CHARSET\_HANDLER** structures to see how they are used. See also the CHARSET\_INFO.txt file in the strings directory for additional information.

## 10.14 Adding a Collation to a Character Set

[10.14.1 Collation Implementation Types](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-implementations)

[10.14.2 Choosing a Collation ID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-choosing-id)

[10.14.3 Adding a Simple Collation to an 8-Bit Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-simple-8bit)

[10.14.4 Adding a UCA Collation to a Unicode Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-unicode-uca)

A collation is a set of rules that defines how to compare and sort character strings. Each collation in MySQL belongs to a single character set. Every character set has at least one collation, and most have two or more collations.

A collation orders characters based on weights. Each character in a character set maps to a weight. Characters with equal weights compare as equal, and characters with unequal weights compare according to the relative magnitude of their weights.

The [**WEIGHT\_STRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_weight-string) function can be used to see the weights for the characters in a string. The value that it returns to indicate weights is a binary string, so it is convenient to use **HEX(WEIGHT\_STRING(*str*))** to display the weights in printable form. The following example shows that weights do not differ for lettercase for the letters in **'AaBb'** if it is a nonbinary case-insensitive string, but do differ if it is a binary string:

mysql> **SELECT HEX(WEIGHT\_STRING('AaBb' COLLATE latin1\_swedish\_ci));**

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

| HEX(WEIGHT\_STRING('AaBb' COLLATE latin1\_swedish\_ci)) |

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

| 41414242 |

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

mysql> **SELECT HEX(WEIGHT\_STRING(BINARY 'AaBb'));**

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

| HEX(WEIGHT\_STRING(BINARY 'AaBb')) |

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

| 41614262 |

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

MySQL supports several collation implementations, as discussed in [Section 10.14.1, “Collation Implementation Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-collation-implementations). Some of these can be added to MySQL without recompiling:

Simple collations for 8-bit character sets.

UCA-based collations for Unicode character sets.

Binary (***xxx*\_bin**) collations.

The following sections describe how to add user-defined collations of the first two types to existing character sets. All existing character sets already have a binary collation, so there is no need here to describe how to add one.

**Warning**

Redefining built-in collations is not supported and may result in unexpected server behavior.

Summary of the procedure for adding a new user-defined collation:

Choose a collation ID.

Add configuration information that names the collation and describes the character-ordering rules.

Restart the server.

Verify that the server recognizes the collation.

The instructions here cover only user-defined collations that can be added without recompiling MySQL. To add a collation that does require recompiling (as implemented by means of functions in a C source file), use the instructions in [Section 10.13, “Adding a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-character-set). However, instead of adding all the information required for a complete character set, just modify the appropriate files for an existing character set. That is, based on what is already present for the character set's current collations, add data structures, functions, and configuration information for the new collation.

**Note**

If you modify an existing user-defined collation, that may affect the ordering of rows for indexes on columns that use the collation. In this case, rebuild any such indexes to avoid problems such as incorrect query results. See [Section 2.11.13, “Rebuilding or Repairing Tables or Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#rebuilding-tables).

### Additional Resources

Example showing how to add a collation for full-text searches: [Section 12.10.7, “Adding a User-Defined Collation for Full-Text Indexing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#full-text-adding-collation)

The Unicode Collation Algorithm (UCA) specification: <http://www.unicode.org/reports/tr10/>

The Locale Data Markup Language (LDML) specification: <http://www.unicode.org/reports/tr35/>

### 10.14.1 Collation Implementation Types

MySQL implements several types of collations:

***Simple collations for 8-bit character sets***

This kind of collation is implemented using an array of 256 weights that defines a one-to-one mapping from character codes to weights. **latin1\_swedish\_ci** is an example. It is a case-insensitive collation, so the uppercase and lowercase versions of a character have the same weights and they compare as equal.

mysql> **SET NAMES 'latin1' COLLATE 'latin1\_swedish\_ci';**

Query OK, 0 rows affected (0.01 sec)

mysql> **SELECT HEX(WEIGHT\_STRING('a')), HEX(WEIGHT\_STRING('A'));**

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

| HEX(WEIGHT\_STRING('a')) | HEX(WEIGHT\_STRING('A')) |

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

| 41 | 41 |

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

1 row in set (0.01 sec)

mysql> **SELECT 'a' = 'A';**

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

| 'a' = 'A' |

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

| 1 |

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

1 row in set (0.12 sec)

For implementation instructions, see [Section 10.14.3, “Adding a Simple Collation to an 8-Bit Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-simple-8bit).

***Complex collations for 8-bit character sets***

This kind of collation is implemented using functions in a C source file that define how to order characters, as described in [Section 10.13, “Adding a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-character-set).

***Collations for non-Unicode multibyte character sets***

For this type of collation, 8-bit (single-byte) and multibyte characters are handled differently. For 8-bit characters, character codes map to weights in case-insensitive fashion. (For example, the single-byte characters **'a'** and **'A'** both have a weight of **0x41**.) For multibyte characters, there are two types of relationship between character codes and weights:

Weights equal character codes. **sjis\_japanese\_ci** is an example of this kind of collation. The multibyte character **'ぢ'** has a character code of **0x82C0**, and the weight is also **0x82C0**.

mysql> **CREATE TABLE t1**

**(c1 VARCHAR(2) CHARACTER SET sjis COLLATE sjis\_japanese\_ci);**

Query OK, 0 rows affected (0.01 sec)

mysql> **INSERT INTO t1 VALUES ('a'),('A'),(0x82C0);**

Query OK, 3 rows affected (0.00 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT c1, HEX(c1), HEX(WEIGHT\_STRING(c1)) FROM t1;**

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

| c1 | HEX(c1) | HEX(WEIGHT\_STRING(c1)) |

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

| a | 61 | 41 |

| A | 41 | 41 |

| ぢ | 82C0 | 82C0 |

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

3 rows in set (0.00 sec)

Character codes map one-to-one to weights, but a code is not necessarily equal to the weight. **gbk\_chinese\_ci** is an example of this kind of collation. The multibyte character **'膰'** has a character code of **0x81B0** but a weight of **0xC286**.

mysql> **CREATE TABLE t1**

**(c1 VARCHAR(2) CHARACTER SET gbk COLLATE gbk\_chinese\_ci);**

Query OK, 0 rows affected (0.33 sec)

mysql> **INSERT INTO t1 VALUES ('a'),('A'),(0x81B0);**

Query OK, 3 rows affected (0.00 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> **SELECT c1, HEX(c1), HEX(WEIGHT\_STRING(c1)) FROM t1;**

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

| c1 | HEX(c1) | HEX(WEIGHT\_STRING(c1)) |

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

| a | 61 | 41 |

| A | 41 | 41 |

| 膰 | 81B0 | C286 |

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

3 rows in set (0.00 sec)

For implementation instructions, see [Section 10.13, “Adding a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-character-set).

***Collations for Unicode multibyte character sets***

Some of these collations are based on the Unicode Collation Algorithm (UCA), others are not.

Non-UCA collations have a one-to-one mapping from character code to weight. In MySQL, such collations are case-insensitive and accent-insensitive. **utf8\_general\_ci** is an example: **'a'**, **'A'**, **'À'**, and **'á'** each have different character codes but all have a weight of **0x0041** and compare as equal.

mysql> **SET NAMES 'utf8' COLLATE 'utf8\_general\_ci';**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE TABLE t1**

**(c1 CHAR(1) CHARACTER SET UTF8 COLLATE utf8\_general\_ci);**

Query OK, 0 rows affected (0.01 sec)

mysql> **INSERT INTO t1 VALUES ('a'),('A'),('À'),('á');**

Query OK, 4 rows affected (0.00 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> **SELECT c1, HEX(c1), HEX(WEIGHT\_STRING(c1)) FROM t1;**

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

| c1 | HEX(c1) | HEX(WEIGHT\_STRING(c1)) |

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

| a | 61 | 0041 |

| A | 41 | 0041 |

| À | C380 | 0041 |

| á | C3A1 | 0041 |

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

4 rows in set (0.00 sec)

UCA-based collations in MySQL have these properties:

If a character has weights, each weight uses 2 bytes (16 bits).

A character may have zero weights (or an empty weight). In this case, the character is ignorable. Example: "U+0000 NULL" does not have a weight and is ignorable.

A character may have one weight. Example: **'a'** has a weight of **0x0E33**.

mysql> **SET NAMES 'utf8' COLLATE 'utf8\_unicode\_ci';**

Query OK, 0 rows affected (0.05 sec)

mysql> **SELECT HEX('a'), HEX(WEIGHT\_STRING('a'));**

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

| HEX('a') | HEX(WEIGHT\_STRING('a')) |

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

| 61 | 0E33 |

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

1 row in set (0.02 sec)

A character may have many weights. This is an expansion. Example: The German letter **'ß'** (SZ ligature, or SHARP S) has a weight of **0x0FEA0FEA**.

mysql> **SET NAMES 'utf8' COLLATE 'utf8\_unicode\_ci';**

Query OK, 0 rows affected (0.11 sec)

mysql> **SELECT HEX('ß'), HEX(WEIGHT\_STRING('ß'));**

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

| HEX('ß') | HEX(WEIGHT\_STRING('ß')) |

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

| C39F | 0FEA0FEA |

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

1 row in set (0.00 sec)

Many characters may have one weight. This is a contraction. Example: **'ch'** is a single letter in Czech and has a weight of **0x0EE2**.

mysql> **SET NAMES 'utf8' COLLATE 'utf8\_czech\_ci';**

Query OK, 0 rows affected (0.09 sec)

mysql> **SELECT HEX('ch'), HEX(WEIGHT\_STRING('ch'));**

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

| HEX('ch') | HEX(WEIGHT\_STRING('ch')) |

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

| 6368 | 0EE2 |

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

1 row in set (0.00 sec)

A many-characters-to-many-weights mapping is also possible (this is contraction with expansion), but is not supported by MySQL.

For implementation instructions, for a non-UCA collation, see [Section 10.13, “Adding a Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-character-set). For a UCA collation, see [Section 10.14.4, “Adding a UCA Collation to a Unicode Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-unicode-uca).

***Miscellaneous collations***

There are also a few collations that do not fall into any of the previous categories.

### 10.14.2 Choosing a Collation ID

Each collation must have a unique ID. To add a collation, you must choose an ID value that is not currently used. MySQL supports two-byte collation IDs. The range of IDs from 1024 to 2047 is reserved for user-defined collations.

The collation ID that you choose appears in these contexts:

The **ID** column of the [**INFORMATION\_SCHEMA.COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table.

The **Id** column of [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) output.

The **charsetnr** member of the **MYSQL\_FIELD** C API data structure.

The **number** member of the **MY\_CHARSET\_INFO** data structure returned by the [**mysql\_get\_character\_set\_info()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-get-character-set-info.html) C API function.

To determine the largest currently used ID, issue the following statement:

mysql> **SELECT MAX(ID) FROM INFORMATION\_SCHEMA.COLLATIONS;**

+---------+

| MAX(ID) |

+---------+

| 247 |

+---------+

To display a list of all currently used IDs, issue this statement:

mysql> **SELECT ID FROM INFORMATION\_SCHEMA.COLLATIONS ORDER BY ID;**

+-----+

| ID |

+-----+

| 1 |

| 2 |

| ... |

| 52 |

| 53 |

| 57 |

| 58 |

| ... |

| 98 |

| 99 |

| 128 |

| 129 |

| ... |

| 247 |

+-----+

**Warning**

Before upgrading, you should save the configuration files that you change. If you upgrade in place, the process replaces the modified files.

### 10.14.3 Adding a Simple Collation to an 8-Bit Character Set

This section describes how to add a simple collation for an 8-bit character set by writing the **<collation>** elements associated with a **<charset>** character set description in the MySQL Index.xml file. The procedure described here does not require recompiling MySQL. The example adds a collation named **latin1\_test\_ci** to the **latin1** character set.

Choose a collation ID, as shown in [Section 10.14.2, “Choosing a Collation ID”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-choosing-id). The following steps use an ID of 1024.

Modify the **Index.xml** and **latin1.xml** configuration files. These files are located in the directory named by the [**character\_sets\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_sets_dir) system variable. You can check the variable value as follows, although the path name might be different on your system:

mysql> **SHOW VARIABLES LIKE 'character\_sets\_dir';**

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

| Variable\_name | Value |

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

| character\_sets\_dir | /user/local/mysql/share/mysql/charsets/ |

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

Choose a name for the collation and list it in the Index.xml file. Find the **<charset>** element for the character set to which the collation is being added, and add a **<collation>** element that indicates the collation name and ID, to associate the name with the ID. For example:

<charset name="latin1">

...

<collation name="latin1\_test\_ci" id="1024"/>

...

</charset>

In the latin1.xml configuration file, add a **<collation>** element that names the collation and that contains a **<map>** element that defines a character code-to-weight mapping table for character codes 0 to 255. Each value within the **<map>** element must be a number in hexadecimal format.

<collation name="latin1\_test\_ci">

<map>

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F

30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F

40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F

60 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

50 51 52 53 54 55 56 57 58 59 5A 7B 7C 7D 7E 7F

80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF

B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF

41 41 41 41 5B 5D 5B 43 45 45 45 45 49 49 49 49

44 4E 4F 4F 4F 4F 5C D7 5C 55 55 55 59 59 DE DF

41 41 41 41 5B 5D 5B 43 45 45 45 45 49 49 49 49

44 4E 4F 4F 4F 4F 5C F7 5C 55 55 55 59 59 DE FF

</map>

</collation>

Restart the server and use this statement to verify that the collation is present:

mysql> **SHOW COLLATION WHERE Collation = 'latin1\_test\_ci';**

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

| Collation | Charset | Id | Default | Compiled | Sortlen |

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

| latin1\_test\_ci | latin1 | 1024 | | | 1 |

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

### 10.14.4 Adding a UCA Collation to a Unicode Character Set

[10.14.4.1 Defining a UCA Collation Using LDML Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#ldml-collation-example)

[10.14.4.2 LDML Syntax Supported in MySQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#ldml-rules)

[10.14.4.3 Diagnostics During Index.xml Parsing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#collation-diagnostics)

This section describes how to add a UCA collation for a Unicode character set by writing the **<collation>** element within a **<charset>** character set description in the MySQL Index.xml file. The procedure described here does not require recompiling MySQL. It uses a subset of the Locale Data Markup Language (LDML) specification, which is available at <http://www.unicode.org/reports/tr35/>. With this method, you need not define the entire collation. Instead, you begin with an existing “base” collation and describe the new collation in terms of how it differs from the base collation. The following table lists the base collations of the Unicode character sets for which UCA collations can be defined. It is not possible to create user-defined UCA collations for **utf16le**; there is no **utf16le\_unicode\_ci** collation that would serve as the basis for such collations.

**Table 10.4 MySQL Character Sets Available for User-Defined UCA Collations**

| **Character Set** | **Base Collation** |
| --- | --- |
| **utf8** | **utf8\_unicode\_ci** |
| **ucs2** | **ucs2\_unicode\_ci** |
| **utf16** | **utf16\_unicode\_ci** |
| **utf32** | **utf32\_unicode\_ci** |

The following sections show how to add a collation that is defined using LDML syntax, and provide a summary of LDML rules supported in MySQL.

#### 10.14.4.1 Defining a UCA Collation Using LDML Syntax

To add a UCA collation for a Unicode character set without recompiling MySQL, use the following procedure. If you are unfamiliar with the LDML rules used to describe the collation's sort characteristics, see [Section 10.14.4.2, “LDML Syntax Supported in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#ldml-rules).

The example adds a collation named **utf8\_phone\_ci** to the **utf8** character set. The collation is designed for a scenario involving a Web application for which users post their names and phone numbers. Phone numbers can be given in very different formats:

+7-12345-67

+7-12-345-67

+7 12 345 67

+7 (12) 345 67

+71234567

The problem raised by dealing with these kinds of values is that the varying permissible formats make searching for a specific phone number very difficult. The solution is to define a new collation that reorders punctuation characters, making them ignorable.

Choose a collation ID, as shown in [Section 10.14.2, “Choosing a Collation ID”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-choosing-id). The following steps use an ID of 1029.

To modify the **Index.xml** configuration file. This file is located in the directory named by the [**character\_sets\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_sets_dir) system variable. You can check the variable value as follows, although the path name might be different on your system:

mysql> **SHOW VARIABLES LIKE 'character\_sets\_dir';**

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

| Variable\_name | Value |

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

| character\_sets\_dir | /user/local/mysql/share/mysql/charsets/ |

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

Choose a name for the collation and list it in the Index.xml file. In addition, you'll need to provide the collation ordering rules. Find the **<charset>** element for the character set to which the collation is being added, and add a **<collation>** element that indicates the collation name and ID, to associate the name with the ID. Within the **<collation>** element, provide a **<rules>** element containing the ordering rules:

<charset name="utf8">

...

<collation name="utf8\_phone\_ci" id="1029">

<rules>

<reset>\u0000</reset>

<i>\u0020</i> <!-- space -->

<i>\u0028</i> <!-- left parenthesis -->

<i>\u0029</i> <!-- right parenthesis -->

<i>\u002B</i> <!-- plus -->

<i>\u002D</i> <!-- hyphen -->

</rules>

</collation>

...

</charset>

If you want a similar collation for other Unicode character sets, add other **<collation>** elements. For example, to define **ucs2\_phone\_ci**, add a **<collation>** element to the **<charset name="ucs2">** element. Remember that each collation must have its own unique ID.

Restart the server and use this statement to verify that the collation is present:

mysql> **SHOW COLLATION WHERE Collation = 'utf8\_phone\_ci';**

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

| Collation | Charset | Id | Default | Compiled | Sortlen |

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

| utf8\_phone\_ci | utf8 | 1029 | | | 8 |

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

Now test the collation to make sure that it has the desired properties.

Create a table containing some sample phone numbers using the new collation:

mysql> **CREATE TABLE phonebook (**

**name VARCHAR(64),**

**phone VARCHAR(64) CHARACTER SET utf8 COLLATE utf8\_phone\_ci**

**);**

Query OK, 0 rows affected (0.09 sec)

mysql> **INSERT INTO phonebook VALUES ('Svoj','+7 912 800 80 02');**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO phonebook VALUES ('Hf','+7 (912) 800 80 04');**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO phonebook VALUES ('Bar','+7-912-800-80-01');**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO phonebook VALUES ('Ramil','(7912) 800 80 03');**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO phonebook VALUES ('Sanja','+380 (912) 8008005');**

Query OK, 1 row affected (0.00 sec)

Run some queries to see whether the ignored punctuation characters are in fact ignored for comparison and sorting:

mysql> **SELECT \* FROM phonebook ORDER BY phone;**

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

| name | phone |

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

| Sanja | +380 (912) 8008005 |

| Bar | +7-912-800-80-01 |

| Svoj | +7 912 800 80 02 |

| Ramil | (7912) 800 80 03 |

| Hf | +7 (912) 800 80 04 |

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

5 rows in set (0.00 sec)

mysql> **SELECT \* FROM phonebook WHERE phone='+7(912)800-80-01';**

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

| name | phone |

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

| Bar | +7-912-800-80-01 |

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

1 row in set (0.00 sec)

mysql> **SELECT \* FROM phonebook WHERE phone='79128008001';**

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

| name | phone |

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

| Bar | +7-912-800-80-01 |

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

1 row in set (0.00 sec)

mysql> **SELECT \* FROM phonebook WHERE phone='7 9 1 2 8 0 0 8 0 0 1';**

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

| name | phone |

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

| Bar | +7-912-800-80-01 |

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

1 row in set (0.00 sec)

#### 10.14.4.2 LDML Syntax Supported in MySQL

This section describes the LDML syntax that MySQL recognizes. This is a subset of the syntax described in the LDML specification available at <http://www.unicode.org/reports/tr35/>, which should be consulted for further information. MySQL recognizes a large enough subset of the syntax that, in many cases, it is possible to download a collation definition from the Unicode Common Locale Data Repository and paste the relevant part (that is, the part between the **<rules>** and **</rules>** tags) into the MySQL Index.xml file. The rules described here are all supported except that character sorting occurs only at the primary level. Rules that specify differences at secondary or higher sort levels are recognized (and thus can be included in collation definitions) but are treated as equality at the primary level.

The MySQL server generates diagnostics when it finds problems while parsing the Index.xml file. See [Section 10.14.4.3, “Diagnostics During Index.xml Parsing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#collation-diagnostics).

***Character Representation***

Characters named in LDML rules can be written literally or in **\u*nnnn*** format, where ***nnnn*** is the hexadecimal Unicode code point value. For example, **A** and **á** can be written literally or as **\u0041** and **\u00E1**. Within hexadecimal values, the digits **A** through **F** are not case-sensitive; **\u00E1** and **\u00e1** are equivalent. For UCA 4.0.0 collations, hexadecimal notation can be used only for characters in the Basic Multilingual Plane, not for characters outside the BMP range of **0000** to **FFFF**. For UCA 5.2.0 collations, hexadecimal notation can be used for any character.

The Index.xml file itself should be written using UTF-8 encoding.

***Syntax Rules***

LDML has reset rules and shift rules to specify character ordering. Orderings are given as a set of rules that begin with a reset rule that establishes an anchor point, followed by shift rules that indicate how characters sort relative to the anchor point.

A **<reset>** rule does not specify any ordering in and of itself. Instead, it “resets” the ordering for subsequent shift rules to cause them to be taken in relation to a given character. Either of the following rules resets subsequent shift rules to be taken in relation to the letter **'A'**:

<reset>A</reset>

<reset>\u0041</reset>

The **<p>**, **<s>**, and **<t>** shift rules define primary, secondary, and tertiary differences of a character from another character:

Use primary differences to distinguish separate letters.

Use secondary differences to distinguish accent variations.

Use tertiary differences to distinguish lettercase variations.

Either of these rules specifies a primary shift rule for the **'G'** character:

<p>G</p>

<p>\u0047</p>

The **<i>** shift rule indicates that one character sorts identically to another. The following rules cause **'b'** to sort the same as **'a'**:

<reset>a</reset>

<i>b</i>

Abbreviated shift syntax specifies multiple shift rules using a single pair of tags. The following table shows the correspondence between abbreviated syntax rules and the equivalent nonabbreviated rules.

**Table 10.5 Abbreviated Shift Syntax**

| **Abbreviated Syntax** | **Nonabbreviated Syntax** |
| --- | --- |
| **<pc>xyz</pc>** | **<p>x</p><p>y</p><p>z</p>** |
| **<sc>xyz</sc>** | **<s>x</s><s>y</s><s>z</s>** |
| **<tc>xyz</tc>** | **<t>x</t><t>y</t><t>z</t>** |
| **<ic>xyz</ic>** | **<i>x</i><i>y</i><i>z</i>** |

An expansion is a reset rule that establishes an anchor point for a multiple-character sequence. MySQL supports expansions 2 to 6 characters long. The following rules put **'z'** greater at the primary level than the sequence of three characters **'abc'**:

<reset>abc</reset>

<p>z</p>

A contraction is a shift rule that sorts a multiple-character sequence. MySQL supports contractions 2 to 6 characters long. The following rules put the sequence of three characters **'xyz'** greater at the primary level than **'a'**:

<reset>a</reset>

<p>xyz</p>

Long expansions and long contractions can be used together. These rules put the sequence of three characters **'xyz'** greater at the primary level than the sequence of three characters **'abc'**:

<reset>abc</reset>

<p>xyz</p>

Normal expansion syntax uses **<x>** plus **<extend>** elements to specify an expansion. The following rules put the character **'k'** greater at the secondary level than the sequence **'ch'**. That is, **'k'** behaves as if it expands to a character after **'c'** followed by **'h'**:

<reset>c</reset>

<x><s>k</s><extend>h</extend></x>

This syntax permits long sequences. These rules sort the sequence **'ccs'** greater at the tertiary level than the sequence **'cscs'**:

<reset>cs</reset>

<x><t>ccs</t><extend>cs</extend></x>

The LDML specification describes normal expansion syntax as “tricky.” See that specification for details.

Previous context syntax uses **<x>** plus **<context>** elements to specify that the context before a character affects how it sorts. The following rules put **'-'** greater at the secondary level than **'a'**, but only when **'-'** occurs after **'b'**:

<reset>a</reset>

<x><context>b</context><s>-</s></x>

Previous context syntax can include the **<extend>** element. These rules put **'def'** greater at the primary level than **'aghi'**, but only when **'def'** comes after **'abc'**:

<reset>a</reset>

<x><context>abc</context><p>def</p><extend>ghi</extend></x>

Reset rules permit a **before** attribute. Normally, shift rules after a reset rule indicate characters that sort after the reset character. Shift rules after a reset rule that has the **before** attribute indicate characters that sort before the reset character. The following rules put the character **'b'** immediately before **'a'** at the primary level:

<reset before="primary">a</reset>

<p>b</p>

Permissible **before** attribute values specify the sort level by name or the equivalent numeric value:

<reset before="primary">

<reset before="1">

<reset before="secondary">

<reset before="2">

<reset before="tertiary">

<reset before="3">

A reset rule can name a logical reset position rather than a literal character:

<first\_tertiary\_ignorable/>

<last\_tertiary\_ignorable/>

<first\_secondary\_ignorable/>

<last\_secondary\_ignorable/>

<first\_primary\_ignorable/>

<last\_primary\_ignorable/>

<first\_variable/>

<last\_variable/>

<first\_non\_ignorable/>

<last\_non\_ignorable/>

<first\_trailing/>

<last\_trailing/>

These rules put **'z'** greater at the primary level than nonignorable characters that have a Default Unicode Collation Element Table (DUCET) entry and that are not CJK:

<reset><last\_non\_ignorable/></reset>

<p>z</p>

Logical positions have the code points shown in the following table.

**Table 10.6 Logical Reset Position Code Points**

| **Logical Position** | **Unicode 4.0.0 Code Point** | **Unicode 5.2.0 Code Point** |
| --- | --- | --- |
| **<first\_non\_ignorable/>** | U+02D0 | U+02D0 |
| **<last\_non\_ignorable/>** | U+A48C | U+1342E |
| **<first\_primary\_ignorable/>** | U+0332 | U+0332 |
| **<last\_primary\_ignorable/>** | U+20EA | U+101FD |
| **<first\_secondary\_ignorable/>** | U+0000 | U+0000 |
| **<last\_secondary\_ignorable/>** | U+FE73 | U+FE73 |
| **<first\_tertiary\_ignorable/>** | U+0000 | U+0000 |
| **<last\_tertiary\_ignorable/>** | U+FE73 | U+FE73 |
| **<first\_trailing/>** | U+0000 | U+0000 |
| **<last\_trailing/>** | U+0000 | U+0000 |
| **<first\_variable/>** | U+0009 | U+0009 |
| **<last\_variable/>** | U+2183 | U+1D371 |

The **<collation>** element permits a **shift-after-method** attribute that affects character weight calculation for shift rules. The attribute has these permitted values:

**simple**: Calculate character weights as for reset rules that do not have a **before** attribute. This is the default if the attribute is not given.

**expand**: Use expansions for shifts after reset rules.

Suppose that **'0'** and **'1'** have weights of **0E29** and **0E2A** and we want to put all basic Latin letters between **'0'** and **'1'**:

<reset>0</reset>

<pc>abcdefghijklmnopqrstuvwxyz</pc>

For simple shift mode, weights are calculated as follows:

'a' has weight 0E29+1

'b' has weight 0E29+2

'c' has weight 0E29+3

...

However, there are not enough vacant positions to put 26 characters between **'0'** and **'1'**. The result is that digits and letters are intermixed.

To solve this, use **shift-after-method="expand"**. Then weights are calculated like this:

'a' has weight [0E29][233D+1]

'b' has weight [0E29][233D+2]

'c' has weight [0E29][233D+3]

...

**233D** is the UCA 4.0.0 weight for character **0xA48C**, which is the last nonignorable character (a sort of the greatest character in the collation, excluding CJK). UCA 5.2.0 is similar but uses **3ACA**, for character **0x1342E**.

***MySQL-Specific LDML Extensions***

An extension to LDML rules permits the **<collation>** element to include an optional **version** attribute in **<collation>** tags to indicate the UCA version on which the collation is based. If the **version** attribute is omitted, its default value is **4.0.0**. For example, this specification indicates a collation that is based on UCA 5.2.0:

<collation id="***nnn***" name="utf8\_***xxx***\_ci" version="5.2.0">

...

</collation>

#### 10.14.4.3 Diagnostics During Index.xml Parsing

The MySQL server generates diagnostics when it finds problems while parsing the Index.xml file:

Unknown tags are written to the error log. For example, the following message results if a collation definition contains a **<aaa>** tag:

[Warning] Buffered warning: Unknown LDML tag:

'charsets/charset/collation/rules/aaa'

If collation initialization is not possible, the server reports an “Unknown collation” error, and also generates warnings explaining the problems, such as in the previous example. In other cases, when a collation description is generally correct but contains some unknown tags, the collation is initialized and is available for use. The unknown parts are ignored, but a warning is generated in the error log.

Problems with collations generate warnings that clients can display with [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings). Suppose that a reset rule contains an expansion longer than the maximum supported length of 6 characters:

<reset>abcdefghi</reset>

<i>x</i>

An attempt to use the collation produces warnings:

mysql> **SELECT \_utf8'test' COLLATE utf8\_test\_ci;**

ERROR 1273 (HY000): Unknown collation: 'utf8\_test\_ci'

mysql> **SHOW WARNINGS;**

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

| Level | Code | Message |

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

| Error | 1273 | Unknown collation: 'utf8\_test\_ci' |

| Warning | 1273 | Expansion is too long at 'abcdefghi=x' |

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

## 10.15 Character Set Configuration

The MySQL server has a compiled-in default character set and collation. To change these defaults, use the [--character-set-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) and [--collation-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server) options when you start the server. See [Section 5.1.7, “Server Command Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-options). The collation must be a legal collation for the default character set. To determine which collations are available for each character set, use the [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) statement or query the **INFORMATION\_SCHEMA** [**COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table.

If you try to use a character set that is not compiled into your binary, you might run into the following problems:

If your program uses an incorrect path to determine where the character sets are stored (which is typically the share/mysql/charsets or share/charsets directory under the MySQL installation directory), this can be fixed by using the --character-sets-dir option when you run the program. For example, to specify a directory to be used by MySQL client programs, list it in the **[client]** group of your option file. The examples given here show what the setting might look like for Unix or Windows, respectively:

[client]

character-sets-dir=/usr/local/mysql/share/mysql/charsets

[client]

character-sets-dir="C:/Program Files/MySQL/MySQL Server 8.0/share/charsets"

If the character set is a complex character set that cannot be loaded dynamically, you must recompile the program with support for the character set.

For Unicode character sets, you can define collations without recompiling by using LDML notation. See [Section 10.14.4, “Adding a UCA Collation to a Unicode Character Set”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#adding-collation-unicode-uca).

If the character set is a dynamic character set, but you do not have a configuration file for it, you should install the configuration file for the character set from a new MySQL distribution.

If your character set index file (Index.xml) does not contain the name for the character set, your program displays an error message:

Character set '***charset\_name***' is not a compiled character set and is not

specified in the '/usr/share/mysql/charsets/Index.xml' file

To solve this problem, you should either get a new index file or manually add the name of any missing character sets to the current file.

You can force client programs to use specific character set as follows:

[client]

default-character-set=***charset\_name***

This is normally unnecessary. However, when [**character\_set\_system**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_system) differs from [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) or [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client), and you input characters manually (as database object identifiers, column values, or both), these may be displayed incorrectly in output from the client or the output itself may be formatted incorrectly. In such cases, starting the mysql client with [--default-character-set=***system\_character\_set***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set)—that is, setting the client character set to match the system character set—should fix the problem.

## 10.16 MySQL Server Locale Support

The locale indicated by the [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) system variable controls the language used to display day and month names and abbreviations. This variable affects the output from the [**DATE\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_date-format), [**DAYNAME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_dayname), and [**MONTHNAME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_monthname) functions.

[**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) does not affect the [**STR\_TO\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_str-to-date) or [**GET\_FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-format) function.

The [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) value does not affect the result from [**FORMAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_format), but this function takes an optional third parameter that enables a locale to be specified to be used for the result number's decimal point, thousands separator, and grouping between separators. Permissible locale values are the same as the legal values for the [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) system variable.

Locale names have language and region subtags listed by IANA (<http://www.iana.org/assignments/language-subtag-registry>) such as **'ja\_JP'** or **'pt\_BR'**. The default value is **'en\_US'** regardless of your system's locale setting, but you can set the value at server startup, or set the **GLOBAL** value at runtime if you have privileges sufficient to set global system variables; see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges). Any client can examine the value of [**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) or set its **SESSION** value to affect the locale for its own connection.

mysql> **SET NAMES 'utf8';**

Query OK, 0 rows affected (0.09 sec)

mysql> **SELECT @@lc\_time\_names;**

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

| @@lc\_time\_names |

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

| en\_US |

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

1 row in set (0.00 sec)

mysql> **SELECT DAYNAME('2010-01-01'), MONTHNAME('2010-01-01');**

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

| DAYNAME('2010-01-01') | MONTHNAME('2010-01-01') |

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

| Friday | January |

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

1 row in set (0.00 sec)

mysql> **SELECT DATE\_FORMAT('2010-01-01','%W %a %M %b');**

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

| DATE\_FORMAT('2010-01-01','%W %a %M %b') |

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

| Friday Fri January Jan |

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

1 row in set (0.00 sec)

mysql> **SET lc\_time\_names = 'es\_MX';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @@lc\_time\_names;**

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

| @@lc\_time\_names |

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

| es\_MX |

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

1 row in set (0.00 sec)

mysql> **SELECT DAYNAME('2010-01-01'), MONTHNAME('2010-01-01');**

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

| DAYNAME('2010-01-01') | MONTHNAME('2010-01-01') |

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

| viernes | enero |

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

1 row in set (0.00 sec)

mysql> **SELECT DATE\_FORMAT('2010-01-01','%W %a %M %b');**

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

| DATE\_FORMAT('2010-01-01','%W %a %M %b') |

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

| viernes vie enero ene |

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

1 row in set (0.00 sec)

The day or month name for each of the affected functions is converted from **utf8** to the character set indicated by the [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) system variable.

[**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names) may be set to any of the following locale values. The set of locales supported by MySQL may differ from those supported by your operating system.

| **Locale Value** | **Meaning** |
| --- | --- |
| **ar\_AE** | Arabic - United Arab Emirates |
| **ar\_BH** | Arabic - Bahrain |
| **ar\_DZ** | Arabic - Algeria |
| **ar\_EG** | Arabic - Egypt |
| **ar\_IN** | Arabic - India |
| **ar\_IQ** | Arabic - Iraq |
| **ar\_JO** | Arabic - Jordan |
| **ar\_KW** | Arabic - Kuwait |
| **ar\_LB** | Arabic - Lebanon |
| **ar\_LY** | Arabic - Libya |
| **ar\_MA** | Arabic - Morocco |
| **ar\_OM** | Arabic - Oman |
| **ar\_QA** | Arabic - Qatar |
| **ar\_SA** | Arabic - Saudi Arabia |
| **ar\_SD** | Arabic - Sudan |
| **ar\_SY** | Arabic - Syria |
| **ar\_TN** | Arabic - Tunisia |
| **ar\_YE** | Arabic - Yemen |
| **be\_BY** | Belarusian - Belarus |
| **bg\_BG** | Bulgarian - Bulgaria |
| **ca\_ES** | Catalan - Spain |
| **cs\_CZ** | Czech - Czech Republic |
| **da\_DK** | Danish - Denmark |
| **de\_AT** | German - Austria |
| **de\_BE** | German - Belgium |
| **de\_CH** | German - Switzerland |
| **de\_DE** | German - Germany |
| **de\_LU** | German - Luxembourg |
| **el\_GR** | Greek - Greece |
| **en\_AU** | English - Australia |
| **en\_CA** | English - Canada |
| **en\_GB** | English - United Kingdom |
| **en\_IN** | English - India |
| **en\_NZ** | English - New Zealand |
| **en\_PH** | English - Philippines |
| **en\_US** | English - United States |
| **en\_ZA** | English - South Africa |
| **en\_ZW** | English - Zimbabwe |
| **es\_AR** | Spanish - Argentina |
| **es\_BO** | Spanish - Bolivia |
| **es\_CL** | Spanish - Chile |
| **es\_CO** | Spanish - Colombia |
| **es\_CR** | Spanish - Costa Rica |
| **es\_DO** | Spanish - Dominican Republic |
| **es\_EC** | Spanish - Ecuador |
| **es\_ES** | Spanish - Spain |
| **es\_GT** | Spanish - Guatemala |
| **es\_HN** | Spanish - Honduras |
| **es\_MX** | Spanish - Mexico |
| **es\_NI** | Spanish - Nicaragua |
| **es\_PA** | Spanish - Panama |
| **es\_PE** | Spanish - Peru |
| **es\_PR** | Spanish - Puerto Rico |
| **es\_PY** | Spanish - Paraguay |
| **es\_SV** | Spanish - El Salvador |
| **es\_US** | Spanish - United States |
| **es\_UY** | Spanish - Uruguay |
| **es\_VE** | Spanish - Venezuela |
| **et\_EE** | Estonian - Estonia |
| **eu\_ES** | Basque - Spain |
| **fi\_FI** | Finnish - Finland |
| **fo\_FO** | Faroese - Faroe Islands |
| **fr\_BE** | French - Belgium |
| **fr\_CA** | French - Canada |
| **fr\_CH** | French - Switzerland |
| **fr\_FR** | French - France |
| **fr\_LU** | French - Luxembourg |
| **gl\_ES** | Galician - Spain |
| **gu\_IN** | Gujarati - India |
| **he\_IL** | Hebrew - Israel |
| **hi\_IN** | Hindi - India |
| **hr\_HR** | Croatian - Croatia |
| **hu\_HU** | Hungarian - Hungary |
| **id\_ID** | Indonesian - Indonesia |
| **is\_IS** | Icelandic - Iceland |
| **it\_CH** | Italian - Switzerland |
| **it\_IT** | Italian - Italy |
| **ja\_JP** | Japanese - Japan |
| **ko\_KR** | Korean - Republic of Korea |
| **lt\_LT** | Lithuanian - Lithuania |
| **lv\_LV** | Latvian - Latvia |
| **mk\_MK** | Macedonian - North Macedonia |
| **mn\_MN** | Mongolia - Mongolian |
| **ms\_MY** | Malay - Malaysia |
| **nb\_NO** | Norwegian(Bokmål) - Norway |
| **nl\_BE** | Dutch - Belgium |
| **nl\_NL** | Dutch - The Netherlands |
| **no\_NO** | Norwegian - Norway |
| **pl\_PL** | Polish - Poland |
| **pt\_BR** | Portugese - Brazil |
| **pt\_PT** | Portugese - Portugal |
| **rm\_CH** | Romansh - Switzerland |
| **ro\_RO** | Romanian - Romania |
| **ru\_RU** | Russian - Russia |
| **ru\_UA** | Russian - Ukraine |
| **sk\_SK** | Slovak - Slovakia |
| **sl\_SI** | Slovenian - Slovenia |
| **sq\_AL** | Albanian - Albania |
| **sr\_RS** | Serbian - Serbia |
| **sv\_FI** | Swedish - Finland |
| **sv\_SE** | Swedish - Sweden |
| **ta\_IN** | Tamil - India |
| **te\_IN** | Telugu - India |
| **th\_TH** | Thai - Thailand |
| **tr\_TR** | Turkish - Turkey |
| **uk\_UA** | Ukrainian - Ukraine |
| **ur\_PK** | Urdu - Pakistan |
| **vi\_VN** | Vietnamese - Vietnam |
| **zh\_CN** | Chinese - China |
| **zh\_HK** | Chinese - Hong Kong |
| **zh\_TW** | Chinese - Taiwan |

|  |  |  |
| --- | --- | --- |
| [Prev](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html) | Up | [Next](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html) |
| Chapter 9 Language Structure | [Home](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\index.html) | Chapter 11 Data Types |