

[k-bushi](#)

- [Top](#)
- [Archives](#)
- [Tags](#)
- [Categories](#)
- [About](#)

[k-bushi](#)

- [Top](#)
- [Archives](#)
- [Tags](#)
- [Categories](#)
- [About](#)

## MySQLでchar/varchar/text型のカラムで大文字/小文字を区別できるようにする

by k-bushi

2023-03-05

[技術](#)

### はじめに

MySQLを使用して文字列の検索を行ったときに、大文字でも小文字でも検索できるということに今更ながら気づいた。そのため、どういう仕組み/どういう理由で大文字/小文字を比較しているのかが気になり調査してみた。まずは、大文字/小文字を区別できるようにするためにどうするのかを記載する。

### 環境

- 1 Windows 10 Pro (操作PC)
- 2 Docker Desktop 4.17.0 (99724)
- 3 MySQL 8.0.32

### 準備

今回検証するにあたって、Docker を使用し、手で確かめられるようにした。下記リポジトリをクローンし、Docker のビルドを行うことで一緒に確かめられるので是非試してほしい。  
<https://github.com/katsuobushiFPGA/docker-mysql-binary-practice>

### 特定の型のカラムで大文字/小文字を区別できるようにする

char, varchar, text 型の文字列検索では大文字/小文字は区別しない。そのため、以下に解決方法できる方法を記載する。

#### SQLクエリにBINARY演算子を利用する

例えば、以下のようなテーブル構造とレコードがあったとする

テーブル構造

```
1 mysql> desc test_case_sensitive;
2 +-----+-----+-----+-----+-----+-----+
3 | Field      | Type          | Null | Key | Default | Extra |
4 +-----+-----+-----+-----+-----+-----+
5 | test_varchar | varchar(100) | YES  |     | NULL    |      |
6 | test_char   | char(100)    | YES  |     | NULL    |      |
7 | test_text   | text         | YES  |     | NULL    |      |
8 +-----+-----+-----+-----+-----+-----+
```

レコード

```
1 INSERT INTO `test_case_sensitive` VALUES (
2   'aaaAaaAaAaaAAaA',
3   'aaaAaaAaAaaAAaA',
4   'aaaAaaAaAaaAAaA'
5 );
```

```
1 mysql> select * from test_case_sensitive;
2 +-----+-----+-----+
3 | test_varchar | test_char | test_text |
4 +-----+-----+-----+
5 | aaaAaaAaAaaAAaA | aaaAaaAaAaaAAaA | aaaAaaAaAaaAAaA |
```

```

5 +-----+
6

```

このときに、いくつかの SELECT 文を実行し検索できるかを試してみる。

```

1 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'aaaaaaaaaaaaaa';
2 +-----+
3 | test_varchar | test_char | test_text |
4 +-----+
5 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
6 +-----+
7 1 row in set (0.00 sec)
8
9 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'aaaaaaaaaaaaaa';
10 +-----+
11 | test_varchar | test_char | test_text |
12 +-----+
13 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
14 +-----+
15 1 row in set (0.00 sec)
16
17 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'aaaaaaaaaaaaaa';
18 +-----+
19 | test_varchar | test_char | test_text |
20 +-----+
21 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
22 +-----+
23 1 row in set (0.00 sec)
24
25
26 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'AAAAAAAAAAAAAAAA';
27 +-----+
28 | test_varchar | test_char | test_text |
29 +-----+
30 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
31 +-----+
32 1 row in set (0.00 sec)
33
34 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'AAAAAAAAAAAAAAAA';
35 +-----+
36 | test_varchar | test_char | test_text |
37 +-----+
38 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
39 +-----+
40 1 row in set (0.01 sec)
41
42 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'AAAAAAAAAAAAAAAA';
43 +-----+
44 | test_varchar | test_char | test_text |
45 +-----+
46 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
47 +-----+
48 1 row in set (0.00 sec)

```

上記のように、文字列の大文字/小文字を区別せずに検索していることがわかる。

ここで、BINARY 演算子をつけると結果が変わる。

下記のように、SELECT 文を流してみる。

```

1 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_varchar = 'aaaaaaaaaaaaaa';
2 Empty set, 1 warning (0.00 sec)
3
4 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_char = 'aaaaaaaaaaaaaa';
5 Empty set, 1 warning (0.00 sec)
6
7 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_text = 'aaaaaaaaaaaaaa';
8 Empty set, 1 warning (0.00 sec)
9
10 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_varchar = 'AAAAAAAAAAAAAAAA';
11 Empty set, 1 warning (0.00 sec)
12
13 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_char = 'AAAAAAAAAAAAAAAA';
14 Empty set, 1 warning (0.00 sec)
15
16 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_text = 'AAAAAAAAAAAAAAAA';
17 Empty set, 1 warning (0.00 sec)
18
19 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_varchar = 'aaaAaaAaaAaaA';
20 +-----+
21 | test_varchar | test_char | test_text |
22 +-----+
23 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
24 +-----+
25 1 row in set, 1 warning (0.00 sec)
26
27 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_char = 'aaaAaaAaaAaaA';
28 +-----+
29 | test_varchar | test_char | test_text |
30 +-----+
31 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
32 +-----+
33 1 row in set, 1 warning (0.00 sec)
34
35 mysql> SELECT * FROM test_case_sensitive WHERE BINARY test_text = 'aaaAaaAaaAaaA';
36 +-----+
37 | test_varchar | test_char | test_text |
38 +-----+
39 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
40 +-----+
41 1 row in set, 1 warning (0.00 sec)

```

クエリのWHERE句にBINARY 演算子をつけることで比較できることがわかる。

## カラムにBINARY属性をつける

```

1 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_varchar` varchar(100) binary DEFAULT NULL;
2 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_char` char(100) binary DEFAULT NULL;
3 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_text` text binary DEFAULT NULL;
4
5 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_text` text binary DEFAULT NULL;Query OK, 0 rows affected, 1 warning (0.17 sec)
6 Records: 0 Duplicates: 0 Warnings: 1
7
8 mysql> ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_varchar` char(100) binary DEFAULT NULL;
9 Query OK, 1 row affected, 1 warning (2.91 sec)
10 Records: 1 Duplicates: 0 Warnings: 1
11
12 mysql> ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_text` text binary DEFAULT NULL;
13 Query OK, 0 rows affected, 1 warning (0.62 sec)
14 Records: 0 Duplicates: 0 Warnings: 1
15
16 /*
17 戻すとき
18 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_varchar` varchar(100) DEFAULT NULL;
19 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_char` char(100) DEFAULT NULL;
20 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_text` text DEFAULT NULL;
21 */

```

この状態で再度チェックをしてみる。

```

1 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'aaaaaaaaaaaaaaaa';
2 Empty set (0.00 sec)
3
4 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'aaaaaaaaaaaaaaaa';
5 Empty set (0.00 sec)
6
7 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'aaaaaaaaaaaaaaaa';
8 Empty set (0.00 sec)
9
10 mysql>
11 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'AAAAAAAAAAAAAAAA';
12 Empty set (0.00 sec)
13
14 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'AAAAAAAAAAAAAAAA';
15 Empty set (0.00 sec)
16
17 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'AAAAAAAAAAAAAAAA';
18 Empty set (0.00 sec)
19
20 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'aaaAaaAaAaaAAa';
21 +-----+-----+-----+
22 | test_varchar | test_char | test_text |
23 +-----+-----+-----+
24 | aaaAaaAaAaaAAa | aaaAaaAaAaaAAa | aaaAaaAaAaaAAa |
25 +-----+-----+-----+
26 1 row in set (0.00 sec)
27
28 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'aaaAaaAaAaaAAa';
29 +-----+-----+-----+
30 | test_varchar | test_char | test_text |
31 +-----+-----+-----+
32 | aaaAaaAaAaaAAa | aaaAaaAaAaaAAa | aaaAaaAaAaaAAa |
33 +-----+-----+-----+
34 1 row in set (0.00 sec)
35
36 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'aaaAaaAaAaaAAa';
37 +-----+-----+-----+
38 | test_varchar | test_char | test_text |
39 +-----+-----+-----+
40 | aaaAaaAaAaaAAa | aaaAaaAaAaaAAa | aaaAaaAaAaaAAa |
41 +-----+-----+-----+
42 1 row in set (0.00 sec)

```

今度は BINARY 演算子を使用せずとも比較できていることがわかる。

## どうしてこのようになっているのか？

大文字小文字の区別をしない理由についてはSQL標準で決まっており、それに準拠しているからというのが理由となる。

※SQL標準についての中身は確認できなかった。(規格のドキュメントを購入しないといけなさそう)

## おまけ

今回使用したときの、各変数の文字コードセット↓

```

1 mysql> show variables like 'character*_set*%';
2 +-----+-----+
3 | Variable_name | Value |
4 +-----+-----+
5 | character_set_client | latin1 |
6 | character_set_connection | latin1 |
7 | character_set_database | utf8mb4 |
8 | character_set_filesystem | binary |
9 | character_set_results | latin1 |
10 | character_set_server | utf8mb4 |
11 | character_set_system | utf8mb3 |
12 +-----+-----+
13 7 rows in set (0.01 sec)

```

※ latin1 は日本語が扱えないので、character\_set\_client は utf8mb4 で良いかなと思いました。

今回は日本語を使用しないのでデフォルトの設定をそのまま使っています。

テーブルの確認

```

1 mysql> show table status from test;
2 +-----+-----+-----+-----+-----+-----+

```

```

3 | Name | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_incr
4 | Checksum | Create_options | Comment |
5 +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 | test_case_sensitive | InnoDB | 10 | Dynamic | 0 | 0 | 16384 | 0 | 0 | 0 | 0 | 1
7 +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
8 1 row in set (0.00 sec)

```

## テーブルの確認 BINARY付与前

```

1 mysql> show create table test_case_sensitive;
2 +-----+-----+
3 | Table | Create Table |
4 +-----+-----+
5 |
6 | test_case_sensitive | CREATE TABLE `test_case_sensitive` (
7 | `test_varchar` varchar(100) DEFAULT NULL,
8 | `test_char` char(100) DEFAULT NULL,
9 | `test_text` text
10 | ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
11 +-----+-----+
12 1 row in set (0.00 sec)

```

## テーブルの確認 BINARY付与後

```

1 mysql> show create table test_case_sensitive;
2 +-----+-----+
3 | Table | Create Table |
4 +-----+-----+
5 |
6 | test_case_sensitive | CREATE TABLE `test_case_sensitive` (
7 | `test_varchar` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
8 | `test_char` char(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
9 | `test_text` text CHARACTER SET utf8mb4 COLLATE utf8mb4_bin
10 | ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
11 +-----+-----+
12 1 row in set (0.01 sec)

```

各カラムに `_bin` の `COLLATE` (照合順序)ができています。さて、これについて参考にした MySQLの公式ページを見ると

たとえば、両方とも `utf8mb4` 文字セットを持つカラムと文字列を比較する場合は、`COLLATE` 演算子を使用して、いずれかのオペランドに `utf8mb4_0900_as_cs` 照合順序または `utf8mb4_bin` 照合順序を設定できます:

- 参考: <https://dev.mysql.com/doc/refman/8.0/ja/case-sensitivity.html>

ということであれば、`COLLATE` に `_cs` 付きの文字コードをセットすれば比較できるということか。

照合順序は、`_ci`, `_cs`, `_bin` といくつか種類があり、それぞれ 大文字小文字の区別が できない, できる, できる。

- `ci`: `case_insensitive` の略
- `cs`: `case_sensitive` の略
- `bin`: `binary` バイナリーの略

※ここでは、大文字/小文字にフォーカスしているので、上記しか記載していないが他にもっと区別されるものはある。

`_cs` 付きの `COLLATE` でやってみる。

## COLLATE を `_cs` にして再度やってみる。🌀

まずは、指定できる `COLLATE` を見てみる。

```

1 mysql> show collation like 'utf8mb4%';
2 +-----+-----+-----+-----+-----+-----+-----+-----+
3 | Collation | Charset | Id | Default | Compiled | Sortlen | Pad_attribute |
4 +-----+-----+-----+-----+-----+-----+-----+-----+
5 | utf8mb4_0900_ai_ci | utf8mb4 | 255 | Yes | Yes | 0 | NO PAD |
6 | utf8mb4_0900_as_ci | utf8mb4 | 305 | Yes | Yes | 0 | NO PAD |
7 | utf8mb4_0900_as_cs | utf8mb4 | 278 | Yes | Yes | 0 | NO PAD |
8 | utf8mb4_0900_bin | utf8mb4 | 309 | Yes | Yes | 1 | NO PAD |
9 | utf8mb4_bg_0900_ai_ci | utf8mb4 | 318 | Yes | Yes | 0 | NO PAD |
10 | utf8mb4_bg_0900_as_cs | utf8mb4 | 319 | Yes | Yes | 0 | NO PAD |
11 | utf8mb4_bin | utf8mb4 | 46 | Yes | Yes | 1 | PAD SPACE |
12 | utf8mb4_bs_0900_ai_ci | utf8mb4 | 316 | Yes | Yes | 0 | NO PAD |
13 | utf8mb4_bs_0900_as_cs | utf8mb4 | 317 | Yes | Yes | 0 | NO PAD |
14 | utf8mb4_croatian_ci | utf8mb4 | 245 | Yes | Yes | 8 | PAD SPACE |
15 | utf8mb4_cs_0900_ai_ci | utf8mb4 | 266 | Yes | Yes | 0 | NO PAD |
16 | utf8mb4_cs_0900_as_cs | utf8mb4 | 289 | Yes | Yes | 0 | NO PAD |
17 | utf8mb4_czech_ci | utf8mb4 | 234 | Yes | Yes | 8 | PAD SPACE |
18 | utf8mb4_danish_ci | utf8mb4 | 235 | Yes | Yes | 8 | PAD SPACE |
19 | utf8mb4_da_0900_ai_ci | utf8mb4 | 267 | Yes | Yes | 0 | NO PAD |
20 | utf8mb4_da_0900_as_cs | utf8mb4 | 290 | Yes | Yes | 0 | NO PAD |
21 | utf8mb4_de_pb_0900_ai_ci | utf8mb4 | 256 | Yes | Yes | 0 | NO PAD |
22 | utf8mb4_de_pb_0900_as_cs | utf8mb4 | 279 | Yes | Yes | 0 | NO PAD |
23 | utf8mb4_eo_0900_ai_ci | utf8mb4 | 273 | Yes | Yes | 0 | NO PAD |
24 | utf8mb4_eo_0900_as_cs | utf8mb4 | 296 | Yes | Yes | 0 | NO PAD |
25 | utf8mb4_esperanto_ci | utf8mb4 | 241 | Yes | Yes | 8 | PAD SPACE |
26 | utf8mb4_estonian_ci | utf8mb4 | 230 | Yes | Yes | 8 | PAD SPACE |
27 | utf8mb4_es_0900_ai_ci | utf8mb4 | 263 | Yes | Yes | 0 | NO PAD |
28 | utf8mb4_es_0900_as_cs | utf8mb4 | 286 | Yes | Yes | 0 | NO PAD |
29 | utf8mb4_es_trad_0900_ai_ci | utf8mb4 | 270 | Yes | Yes | 0 | NO PAD |
30 | utf8mb4_es_trad_0900_as_cs | utf8mb4 | 293 | Yes | Yes | 0 | NO PAD |
31 | utf8mb4_et_0900_ai_ci | utf8mb4 | 262 | Yes | Yes | 0 | NO PAD |
32 | utf8mb4_et_0900_as_cs | utf8mb4 | 285 | Yes | Yes | 0 | NO PAD |
33 | utf8mb4_general_ci | utf8mb4 | 45 | Yes | Yes | 1 | PAD SPACE |
34 | utf8mb4_german2_ci | utf8mb4 | 244 | Yes | Yes | 8 | PAD SPACE |
35 | utf8mb4_gl_0900_ai_ci | utf8mb4 | 320 | Yes | Yes | 0 | NO PAD |
36 | utf8mb4_gl_0900_as_cs | utf8mb4 | 321 | Yes | Yes | 0 | NO PAD |
37 | utf8mb4_hr_0900_ai_ci | utf8mb4 | 275 | Yes | Yes | 0 | NO PAD |
38 | utf8mb4_hr_0900_as_cs | utf8mb4 | 298 | Yes | Yes | 0 | NO PAD |

```

39	utf8mb4_hungarian_ci	utf8mb4	242	Yes	8	PAD SPACE
40	utf8mb4_hu_0900_ai_ci	utf8mb4	274	Yes	0	NO PAD
41	utf8mb4_hu_0900_as_cs	utf8mb4	297	Yes	0	NO PAD
42	utf8mb4_ice_0900_ai_ci	utf8mb4	225	Yes	8	PAD SPACE
43	utf8mb4_is_0900_ai_ci	utf8mb4	257	Yes	0	NO PAD
44	utf8mb4_is_0900_as_cs	utf8mb4	280	Yes	0	NO PAD
45	utf8mb4_ja_0900_as_cs	utf8mb4	303	Yes	0	NO PAD
46	utf8mb4_ja_0900_as_cs_ks	utf8mb4	304	Yes	24	NO PAD
47	utf8mb4_latvian_ci	utf8mb4	226	Yes	8	PAD SPACE
48	utf8mb4_la_0900_ai_ci	utf8mb4	271	Yes	0	NO PAD
49	utf8mb4_la_0900_as_cs	utf8mb4	294	Yes	0	NO PAD
50	utf8mb4_lithuanian_ci	utf8mb4	236	Yes	8	PAD SPACE
51	utf8mb4_lt_0900_ai_ci	utf8mb4	268	Yes	0	NO PAD
52	utf8mb4_lt_0900_as_cs	utf8mb4	291	Yes	0	NO PAD
53	utf8mb4_lv_0900_ai_ci	utf8mb4	258	Yes	0	NO PAD
54	utf8mb4_lv_0900_as_cs	utf8mb4	281	Yes	0	NO PAD
55	utf8mb4_mn_cyrl_0900_ai_ci	utf8mb4	322	Yes	0	NO PAD
56	utf8mb4_mn_cyrl_0900_as_cs	utf8mb4	323	Yes	0	NO PAD
57	utf8mb4_nb_0900_ai_ci	utf8mb4	310	Yes	0	NO PAD
58	utf8mb4_nb_0900_as_cs	utf8mb4	311	Yes	0	NO PAD
59	utf8mb4_nn_0900_ai_ci	utf8mb4	312	Yes	0	NO PAD
60	utf8mb4_nn_0900_as_cs	utf8mb4	313	Yes	0	NO PAD
61	utf8mb4_persian_ci	utf8mb4	240	Yes	8	PAD SPACE
62	utf8mb4_pl_0900_ai_ci	utf8mb4	261	Yes	0	NO PAD
63	utf8mb4_pl_0900_as_cs	utf8mb4	284	Yes	0	NO PAD
64	utf8mb4_polish_ci	utf8mb4	229	Yes	8	PAD SPACE
65	utf8mb4_romanian_ci	utf8mb4	227	Yes	8	PAD SPACE
66	utf8mb4_roman_ci	utf8mb4	239	Yes	8	PAD SPACE
67	utf8mb4_ro_0900_ai_ci	utf8mb4	259	Yes	0	NO PAD
68	utf8mb4_ro_0900_as_cs	utf8mb4	282	Yes	0	NO PAD
69	utf8mb4_ru_0900_ai_ci	utf8mb4	306	Yes	0	NO PAD
70	utf8mb4_ru_0900_as_cs	utf8mb4	307	Yes	0	NO PAD
71	utf8mb4_sinhala_ci	utf8mb4	243	Yes	8	PAD SPACE
72	utf8mb4_sk_0900_ai_ci	utf8mb4	269	Yes	0	NO PAD
73	utf8mb4_sk_0900_as_cs	utf8mb4	292	Yes	0	NO PAD
74	utf8mb4_slovak_ci	utf8mb4	237	Yes	8	PAD SPACE
75	utf8mb4_slovenian_ci	utf8mb4	228	Yes	8	PAD SPACE
76	utf8mb4_sl_0900_ai_ci	utf8mb4	260	Yes	0	NO PAD
77	utf8mb4_sl_0900_as_cs	utf8mb4	283	Yes	0	NO PAD
78	utf8mb4_spanish2_ci	utf8mb4	238	Yes	8	PAD SPACE
79	utf8mb4_spanish_ci	utf8mb4	231	Yes	8	PAD SPACE
80	utf8mb4_sr_latn_0900_ai_ci	utf8mb4	314	Yes	0	NO PAD
81	utf8mb4_sr_latn_0900_as_cs	utf8mb4	315	Yes	0	NO PAD
82	utf8mb4_sv_0900_ai_ci	utf8mb4	264	Yes	0	NO PAD
83	utf8mb4_sv_0900_as_cs	utf8mb4	287	Yes	0	NO PAD
84	utf8mb4_swedish_ci	utf8mb4	232	Yes	8	PAD SPACE
85	utf8mb4_tr_0900_ai_ci	utf8mb4	265	Yes	0	NO PAD
86	utf8mb4_tr_0900_as_cs	utf8mb4	288	Yes	0	NO PAD
87	utf8mb4_turkish_ci	utf8mb4	233	Yes	8	PAD SPACE
88	utf8mb4_unicode_520_ci	utf8mb4	246	Yes	8	PAD SPACE
89	utf8mb4_unicode_ci	utf8mb4	224	Yes	8	PAD SPACE
90	utf8mb4_vietnamese_ci	utf8mb4	247	Yes	8	PAD SPACE
91	utf8mb4_vi_0900_ai_ci	utf8mb4	277	Yes	0	NO PAD
92	utf8mb4_vi_0900_as_cs	utf8mb4	300	Yes	0	NO PAD
93	utf8mb4_zh_0900_as_cs	utf8mb4	308	Yes	0	NO PAD
94						
95	89 rows in set (0.01 sec)					

- ai: AccessInsensitive の略 濁音、半濁音を区別しない
- as: AccessSensitive の略 濁音、半濁音を区別する 他、国のコードがあるので、その国の言語用のセットとなっている

さて、ここではテーブルが utf8mb4\_0900\_ai\_ci となっているので、 utf8mb4\_0900\_as\_cs を選択する。

```

1 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_varchar` varchar(100) COLLATE `utf8mb4_0900_as_cs` DEFAULT NULL;
2 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_char` char(100) COLLATE `utf8mb4_0900_as_cs` DEFAULT NULL;
3 ALTER TABLE `test_case_sensitive` MODIFY COLUMN `test_text` text COLLATE `utf8mb4_0900_as_cs` DEFAULT NULL;
4
5 mysql> show create table test_case_sensitive;
6 +-----+-----+
7 | Table | Create Table |
8 +-----+-----+
9
10 | test_case_sensitive | CREATE TABLE `test_case_sensitive` (
11 | `test_varchar` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_as_cs DEFAULT NULL,
12 | `test_char` char(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_as_cs DEFAULT NULL,
13 | `test_text` text CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_as_cs
14 | ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
15 +-----+-----+
16 1 row in set (0.00 sec)
17
18 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'aaaaaaaaaaaaa';
19 Empty set (0.00 sec)
20
21 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'aaaaaaaaaaaaa';
22 Empty set (0.00 sec)
23
24 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'aaaaaaaaaaaaa';
25 Empty set (0.00 sec)
26
27 mysql>
28 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'AAAAAAAAAAAAA';
29 Empty set (0.00 sec)
30
31 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'AAAAAAAAAAAAA';
32 Empty set (0.00 sec)
33
34 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'AAAAAAAAAAAAA';
35 Empty set (0.00 sec)
36
37 mysql>
38 mysql>
39 mysql> SELECT * FROM test_case_sensitive WHERE test_varchar = 'aaaAaaAaaAAaA';
40 +-----+-----+-----+-----+
41 | test_varchar | test_char | test_text | |
42 +-----+-----+-----+-----+
43 | aaaAaaAaaAAaA | aaaAaaAaaAAaA | aaaAaaAaaAAaA | |
44 +-----+-----+-----+-----+
45 1 row in set (0.00 sec)
46
47 mysql> SELECT * FROM test_case_sensitive WHERE test_char = 'aaaAaaAaaAAaA';
48 +-----+-----+-----+-----+
49 | test_varchar | test_char | test_text | |
50 +-----+-----+-----+-----+
51 | aaaAaaAaaAAaA | aaaAaaAaaAAaA | aaaAaaAaaAAaA | |
52 +-----+-----+-----+-----+

```

```

53 1 row in set (0.00 sec)
54
55 mysql> SELECT * FROM test_case_sensitive WHERE test_text = 'aaaAaaAaaAaaA';
56
57 +-----+-----+-----+
58 | test_varchar | test_char | test_text |
59 +-----+-----+-----+
59 | aaaAaaAaaAaaA | aaaAaaAaaAaaA | aaaAaaAaaAaaA |
60 +-----+-----+-----+
61 1 row in set (0.00 sec)

```

おお〜いけてますね。

※全部のカラムで変更したいのであれば、テーブルの COLLATE を変えるほうが良いですね。

## COLLATE の \_cs, \_bin どちらがいいのか

ケースバイケースではありますが、個人的には \_bin はバイナリ比較し、すべて区別できるので良いのではないかと思います。

※なにか問題が発生した際に、この可能性をつぶせるというのはでかいですね。

\_cs のメリットは何なんだろう…。わからない…。おそらくパフォーマンスの部分なのだろうか。

と思ったけど、chatGPT君に聞いてみたらこうだった。

cs と bin の照合順序を比較した場合、一般的には bin の方が cs よりもパフォーマンスが良いとされています。

cs は文字列の比較時に大文字と小文字を区別するため、文字列の比較には多少のオーバーヘッドが発生します。一方、bin はバイト単位で比較するため、cs よりも高速に文字列の比較ができます。

ただし、照合順序のパフォーマンスは、使用するデータベースやデータベースの設定、クエリの複雑さなどによって異なります。そのため、アプリケーションの要件に応じて、適切な照合順序を選択する必要があります。性能の観点から照合順序を選択する場合は、実際に性能テストを実施して、適切な照合順序を決定することが重要です。

結論からすると要件によるという感じですね。

私は、utf8mb4\_bin を使います。

## 参考

- What is difference between binary and non-binary string data types?  
<https://www.mysqlfaq.net/mysql-faqs/Data-Types/What-is-difference-between-binary-and-non-binary-string-data-types>
- MySQL - 10.3.5 カラム文字セットおよび照合順序  
<https://dev.mysql.com/doc/refman/8.0/ja/charset-column.html>
- MySQL - 10.8.5 バイナリ照合順序と\_bin 照合順序  
<https://dev.mysql.com/doc/refman/8.0/ja/charset-binary-collations.html>
- MySQL - B.3.4.1 文字列検索での大文字/小文字の区別  
<https://dev.mysql.com/doc/refman/8.0/ja/case-sensitivity.html>
- MySQLのencodingをutf8からutf8mb4に変更して寿司ビール問題に対応する  
[https://techracho.bpsinc.jp/hachi8833/2020\\_11\\_26/25044#mysqlutf8mb4](https://techracho.bpsinc.jp/hachi8833/2020_11_26/25044#mysqlutf8mb4)
- 寿司ビール問題④ 初心者→中級者へのSTEP20/25  
<https://qiita.com/kamohicokamo/items/3cc05f63a90148525caf>
- Collation(デフォルト照合順序, 大文字と小文字, 半角と全角)  
<https://www.wakuwakubank.com/posts/797-mysql-collation/>
- MariaDB(MySQL)の照合順序の話  
[https://blog.ver001.com/mariadb\\_collation/](https://blog.ver001.com/mariadb_collation/)

## 最後に

データベースに関して多少なりとも知識はあったが、それはSQLの構文とかそういうレベルの話で、一步踏み込んだ部分には何もわからないことがわかった。

このあたり知識をつけていきたいなあ。

参考にさせていただいたサイト様は大変勉強になりました、ありがとうございます。m(\_ \_)m

[MySQL DB RDBMS](#)

[◀ barcodeを使ってバーコードを生成する 前へ ChatGPT APIが提供されたので使ってみる 次へ ▶](#)

目次

- [はじめに](#)
- [環境](#)
- [準備](#)
- [特定の型のカラムで大文字/小文字を区別できるようにする](#)

- [SQLクエリにBINARY演算子を利用する](#)
- [カラムにBINARY属性をつける](#)
- [どうしてこのようになっているのか?](#)
- [おまけ](#)
- [COLLATE を `\_cs` にして再度やってみる。](#)
  - [COLLATE の `\_cs`, `\_bin` どちらがいいのか](#)
- [参考](#)
- [最後に](#)



Powered by [Hugo](#) | Theme - [Jane](#) © 2020 - 2023 ♥k-bushi

