



Unicode : comportements implicites dans les bases de données

BreizhCTF 2k25

Alexandre ZANNI (@noraj)

15/03/2025

Dans les SGBD, des collisions Unicode peuvent se produire lorsque certains mécanismes implicites sont exécutés par la BDD sans que le développeur de l'app. ne s'en rendent compte.

- une **normalisation** est appliquée à certaines **colonnes** sur certains **types**,
- certains **types** de données sont associés à certains **encodages**, de sorte qu'un algorithme d'**ajustement optimal** peut être appliqué lors de la conversion d'un **espace** plus **large** (ex : Unicode UTF-8) vers un **espace** plus **étroit** (ex : ASCII, LATIN-1),
- une **séquence d'assemblage** (ex : insensible à la casse) est appliquée à une **colonne**,

- un **jeu de caractères** ou un **encodage** est appliqué à la **BDD** ou à une **colonne**,
- une **conversion de type** se produit lorsque différents types de **colonnes** sont **comparés**,
- les **fonctions** et **opérateurs** de **chaînes de caractères** pourraient ne pas être **compatibles** avec l'**Unicode**.

Testé avec l'image docker MariaDB `mariadb:11.6.2-noble`, Ruby 3.4.1, `ruby-mysql` gem 4.2.0.

Encodage par défaut

BDD, tables et colonnes utilisent UTF-8 (**utf8mb4**) par défaut.

```
MariaDB [unicode]> SELECT @@character_set_database, @@collation_database;
+-----+-----+
| @@character_set_database | @@collation_database |
+-----+-----+
| utf8mb4                  | utf8mb4_uca1400_ai_ci |
+-----+-----+
```

```
MariaDB [unicode]> SHOW TABLE STATUS where name like 'test_unicode';
```

Name	Engine	Version	...	Collation	...
test_unicode	InnoDB	10	...	utf8mb4_uca1400_ai_ci	...

```
MariaDB [unicode]> SHOW FULL COLUMNS FROM test_unicode;
```

Field	Type	Collation	...
id	int(11)	NULL	...
nom	varchar(100)	utf8mb4_uca1400_ai_ci	...
prenom	varchar(100)	utf8mb4_uca1400_ai_ci	...

...

En 2012, l'implémentation UTF-8 dans MySQL s'appelait `utf8` (ajd. `utf8mb3`), ne gérait que des **car. de 1 à 3 octets** (au lieu de 4 octets max pour de l'UTF-8 valide), donc car. Unicode de 4 octets était tronqué ainsi que le reste de la chaîne => **problèmes de sécu.**

Mais, ça ne se produisait que quand **mode strict** désactivé (activé par défaut), car il empêche l'insertion de valeurs invalides. Mais Wordpress désactivait le mode strict à l'époque => XSS.

Corrigé => même quand mode strict désactivé, valeurs invalides remplacées par `?` au lieu de tronquer la chaîne.

Pas grand-chose à voir avec Unicode, car la troncature des car. invalides se produisait avec presque tous les encodages. Mais `utf8mb3` n'aurait jamais dû exister.

Conversion de type

Pas de problème de **conversion de type**. Ex. si une colonne utilise `ascii` et qu'on essaye d'insérer des car. Unicode => erreur (mode strict) ou remplacés par `?` (mode strict désactivé) comme vu précédemment.

```
MariaDB [unicode2]> INSERT INTO test_unicode VALUES (1, 'scriptalert(document.cookie)/script', ...);  
ERROR 1366 (22007): Incorrect string value: '\xEF\xBC\x9Cscr...' for column `unicode2`.`test_unicode`.`nom` at row 1
```

Séquence d'assemblage

Séquence d'assemblage (*collating sequence*) utilisé pour les comparaisons.

Suffixe	Signification
_ai	Accent-insensitive
_as	Accent-sensitive
_ci	Case-insensitive
_cs	Case-sensitive
_ks	Kana-sensitive
_bin	Binary

```
MariaDB [unicode2]> SHOW COLLATION WHERE `Default` = 'Yes';
```

Collation	Charset	Id	Default	Compiled	Sortlen
big5_chinese_ci	big5	1	Yes	Yes	1
...					
latin1_swedish_ci	latin1	8	Yes	Yes	1
latin2_general_ci	latin2	9	Yes	Yes	1
swe7_swedish_ci	swe7	10	Yes	Yes	1
ascii_general_ci	ascii	11	Yes	Yes	1
...					
latin7_general_ci	latin7	41	Yes	Yes	1
cp1251_general_ci	cp1251	51	Yes	Yes	1
utf16le_general_ci	utf16le	56	Yes	Yes	1
cp1256_general_ci	cp1256	57	Yes	Yes	1
cp1257_general_ci	cp1257	59	Yes	Yes	1
binary	binary	63	Yes	Yes	1
...					

Toutes les séquences d'assemblage associés aux encodages par défaut sont insensibles à la casse.

Attentions aux comparaisons via SQL donc.

Que ce passe t-il avec l'encodage (`utf8mb4`) et la séquence d'assemblage (`utf8mb4_uca1400_ai_ci`) par défaut ?

Pour l'expression `LIKE`, `1` = `TRUE` et `0` = `FALSE`.

```
MariaDB [unicode]> SELECT 'abc' LIKE 'ABC';
+-----+
| 'abc' LIKE 'ABC' |
+-----+
|                  1 |
+-----+
```

Pour la fonction `STRCMP()`, `0` = chaînes de car. identiques, `-1` = 1ere chaine plus petit et `1` sinon.

```
MariaDB [unicode]> SELECT STRCMP('ABC', 'abc');
+-----+
| STRCMP('ABC', 'abc') |
+-----+
|                      0 |
+-----+
```

Collisions ?

```
MariaDB [unicode]> SELECT 'ß' LIKE 'SS';
```

```
+-----+  
| 'ß' LIKE 'SS' |  
+-----+  
|                0 |  
+-----+
```

```
MariaDB [unicode]> SELECT 'SS' LIKE 'ß';
```

```
+-----+  
| 'SS' LIKE 'ß' |  
+-----+  
|                0 |  
+-----+
```

LIKE comportement ❌, collision ❌

```
MariaDB [unicode]> SELECT STRCMP('ß', 'ss');
```

```
+-----+  
| STRCMP('ß', 'ss') |  
+-----+  
|                  0 |  
+-----+
```

```
MariaDB [unicode]> SELECT STRCMP('ß', 'SS');
```

```
+-----+  
| STRCMP('ß', 'SS') |  
+-----+  
|                  0 |  
+-----+
```

STRCMP() comportement , collision  (mais pas pour tout)

Pour les noms de seq. d'assemblage non binaires qui ne précisent pas la sensibilité aux accents, celle-ci est déterminée par la sensibilité à la casse. Si un nom de seq. d'assemblage ne contient pas `_ai` ou `_as`, `_ci` dans le nom implique `_ai` et `_cs` dans le nom implique `_as`. Ex : `latin1_general_ci` est explicitement insensible à la casse et implicitement insensible aux accents, `latin1_general_cs` est explicitement sensible à la casse et implicitement sensible aux accents, et `utf8mb4_0900_ai_ci` est explicitement insensible à la casse et aux accents.

```
SELECT STRCMP('ı', 'I');
```

Sequence d'assemblage	Collision
utf8mb4_uca1400_ai_ci	✗
utf8mb4_general_ci	✓
utf8mb4_unicode_ci	✗
utf8mb4_general1400_as_ci	✓

Logique ? Version UCA (Unicode Collation Algorithm) ?

Côté app, le dev vérifie à l'inscription qu'une adresse courriel n'est pas déjà utilisée.

Avec comparaison sensible à la casse / binaire.

```
melißa-admin@yopmail.com != melissa-admin@yopmail.com
```

Côté BDD, le dev ne se doute pas que MySQL / Mariadb n'est pas sensible à la casse et qu'il y a des risques de collision.

```
meLißa-admin@yopmail.com == melissa-admin@yopmail.com
```

```
[u]> SELECT * FROM test_unicode WHERE STRCMP(courriel, 'melissa-admin@yopmail.com') = 0;
```

id	prenom	courriel
1	Melissa	melissa-admin@yopmail.com
2	Hacker	melissa-admin@yopmail.com

```
[u]> SELECT * FROM test_unicode WHERE STRCMP(courriel, _utf8mb4 'melissa-admin@yopmail.com' COLLATE utf8mb4_bin) = 0;
```

id	prenom	courriel
1	Melissa	melissa-admin@yopmail.com

- Forcer le max de chose explicitement (ex : séq. d'assemblage)
- Traiter le max de chose (ex : comparaison) côté applicatif plutôt que côté BDD (rêq. SQL)
- S'assurer d'avoir même encodage et séq. d'assemblage côté app. et BDD
- Normalisation Unicode avant envoi en BDD



<https://www.linkedin.com/company/synacktiv>



<https://x.com/synacktiv>



<https://synacktiv.com>