# Sobre COLLATION y CHARSET en MySQL

Hace bastante tiempo que tenía ganas de abordar varias cosas de MySQL, pero una de las más importantes (a mi juicio) es lo relativo a los COLLATIONs y los CHARSETs que tanto dolor de cabeza le pueden dar a aquellos de habla hispana. ¿Quién no ha tenido problemas con los tildes (no confundir con los acentos) en alguna página?

Y aunque este tema ya fue tocado en un post previo, éste será un poco más específico y aclaratorio sólo sobre estas dos propiedades de cada tabla y/o campo.

## Introducción

Desde prácticamente los inicios de MySQL es que siempre he tenido la curiosidad con respecto a qué significan estas cosas raras: al principio todo funcionaba mágicamente usando latin1, el cual también era el predeterminado de MySQL (así como también Apache y PHP). Sin embargo, los problemas empezaron cuando el mundo decidió migrar hacia UTF-8, y más específicamente cuando Apache o PHP (una de las dos, no me acuerdo bien cuál) decidió que de ahora en adelante el charset predeterminado sería este mismo. Aunque siempre existía la opción de aplicar iconv(), estaba más que claro que era un poco contra-productivo tener que aplicar esta función a cualquier salida hecha por la base de datos, así que en algún momento de la vida empecé a ocupar UTF-8 también en la base de datos.

## Partiendo por lo básico

Una de las primeras cosas que se deben saber es el cómo un CHARSET u otro guarda la información. Para ser muy breves y concisos, utf8 ocupa hasta 3 bytes en cada caracter, mientras que latin1 siempre ocupará 1 solo byte.

Aunque ninguno de los dos campos es obligatorio al crear una tabla, se recomienda que sí se haga no por ustedes, sino que si algún día cambian la configuración, tendrán una mezcla de charsets en sus bases de datos, el cual puede ser un parto detectar y cambiar.
Con respecto a esto mismo, si colocan un CHARSET, MySQL le asignará de forma automática un COLLATION que está asociado al CHARSET. La lista de estos dos la pueden averiguar con el siguiente SQL:

```
1 -- Mostrar los CHARSETs instalados:
2 SHOW CHARACTER SET;
3 -- Mostrar COLLATIONS instalados:
4 SHOW COLLATION;
```

El primero debería entregarles un listado de todos los CHARSET instalados y también debería mostrarles el COLLATION asociado. También les muestra el número de bytes máximo que ocupa cada CHARSET.
El segundo comando muestra todos los COLLATIONs instalados.

Por último, cabe destacar que podemos asignar COLLATIONs y CHARSETs; en orden de prioridad; a nivel de:

- MySQL completo
- Bases de datos
- Tablas
- Campos

Eso significa que si se asigna un CHARSET y COLLATION a nivel de MySQL, todas las bases de datos heredarán esta propiedad. A su vez, todas las tablas heredarán las propiedades de la base de datos y asimismo, todos los campos heredarán las propiedades predeterminadas de las tablas. En cualquiera de estos pasos podemos sobre-escribir el predeterminado con nuestro propio CHARSET y COLLATION.

## ¿Y qué significan entonces?

En palabras muy breves, una de las principales diferencias es que CHARSET hace referencia a cómo MySQL guarda internamente el dato y COLLATION es una manera de decirle cómo debe comparar el texto y/o ordenarlo. Para explicar bien este punto apliquemos algunos ejemplos:

```
1  -- Creamos la tabla:
2  CREATE TABLE IF NOT EXISTS collationTests (
3      name01 CHAR(5) CHARSET utf8 COLLATE utf8_unicode_ci,
4      name02 CHAR(5) CHARSET latin1 COLLATE latin1_general_cs,
5      name03 CHAR(5) CHARSET ASCII COLLATE ascii_general_ci,
6      name04 CHAR(5) CHARSET utf8 COLLATE utf8_bin,
7      name05 CHAR(5) CHARSET latin1 COLLATE latin1_bin,
8      name06 CHAR(5) CHARSET ASCII COLLATE ascii_bin,
9  ) ENGINE=MyISAM;
10
11 -- Insertamos algunos datos:
12 INSERT INTO collationTests VALUES ('Ñandú','Ñandú','Nandu','Ñandú','Ñandú','Nandu');
```

Lo que hicimos arriba fue crear una tabla con diversos CHARSETs y luego insertamos una columna con el

Camilo Sperberg es Ingeniero Informático especializado en Linux y PHP. Éste es su blog oficial y aquí podrá encontrar mayor información.

zend CERTIFIED ENGINEER PHP 5.3

**Yo en Internet**

CHW.net
Twitter
Facebook
Zend Certified Engineer
Youtube
Google+
phpclasses.org
Github

**Más vistos**

¿Debería comprarme un Mac o un PC? - 31,879 vistas
Sobre COLLATION y CHARSET en MySQL - 31,537 vistas
Ventajas y desventajas de Mac OS Lion - 31,006 vistas
¿Problemas en los tildes o acentos? - 29,239 vistas
Por qué jQuery le devolvió la diversión a JavaScript - 27,793 vistas
Instalando CentOS 6 con PHP 5.4.17 y MySQL 5.5.31 - 27,429 vistas
Regiones, provincias y comunas de Chile en SQL - 24,419 vistas
¿Qué tengo que hacer para sacar la certificación en PHP? - 18,487 vistas
INSERT a partir de un SELECT, en una sola consulta - 18,321 vistas
Cómo ocupar ob_start(), ob_get_contents() y otros relacionados - 17,853 vistas

**Categorías**

Apple/Mac (16)
Bases de Datos (13)
Classes (7)

Lo que hicimos arriba fue crear una tabla con diversos CHARSETs y luego insertamos una columna con el mismo dato, a método de comparación. Enseguida, aplicamos algunos SELECTs que es donde empieza la diversión total.

Lo primero que cabe destacar es que el CHARSET ASCII no permite ingresar otra cosa que no esté en la tabla ASCII predeterminada, así que la letra "Ñ" y aquellas letras con tilde quedan absolutamente descartadas.

```
SELECT LENGTH(name01) AS bL01, CHAR_LENGTH(name01) AS cL01 FROM collationTests;
SELECT LENGTH(name02) AS bL02, CHAR_LENGTH(name02) AS cL02 FROM collationTests;
SELECT LENGTH(name03) AS bL03, CHAR_LENGTH(name03) AS cL03 FROM collationTests;
```

La primera serie de consultas que vamos a realizar tiene que ver con el tamaño (en bytes) de cada campo relevante y su tamaño en número de caracteres. Hago esta aclaración ya que ambas funciones tienen esa distinción.

El resultado de esto es:

```
bL01    cL01
7       5
bL02    cL02
5       5
bL03    cL03
5       5
```

Como vemos, en UTF-8 se guardaron 7 bytes de información (un byte extra en la letra "Ñ", otro byte extra en la letra "ú") pero el largo de cada cadena en cada caso es de 5 caracteres.

Aplicamos algunas consultas para ver las diferencias entre COLLATIONs:

```
SELECT * FROM collationTests WHERE name01 LIKE 'N%';
SELECT * FROM collationTests WHERE name01 LIKE 'ñ%';
SELECT * FROM collationTests WHERE name01 LIKE 'Ñ%';
SELECT * FROM collationTests WHERE name04 LIKE 'N%';
SELECT * FROM collationTests WHERE name04 LIKE 'ñ%';
SELECT * FROM collationTests WHERE name04 LIKE 'Ñ%';
```

Las primeras 3 (`CHARSET utf8 COLLATE utf8_general_ci`) devolverán un registro cada uno, mientras que de las últimas 3 (`CHARSET utf8 COLLATE utf8_bin`) sólo el último devolverá un resultado positivo.
Esto se debe a que la columna "name01" tiene COLLATION utf8-general-ci, que, entre otras cosas, considera como sinónimo la letra "N" y "Ñ", y además es case-insensitive (`utf8-general-ci`). Esto también se aplica a los tildes, de forma que si buscamos por ñandu (sin tilde) el resultado entregado será el mismo que si buscamos por "Ñandú". De igual forma, "ÑaÑdU" igual entregará un resultado positivo para "Ñandú".
En las últimas 3 sin embargo, si no buscamos **exactamente** por lo ingresado en la base de datos no se devolverá ningún registro, así que cualquier cosa que no sea un match exacto de "Ñandú" simplemente se descartará.

## Ordenando resultados

Al principio del punto anterior dije que los COLLATION también servían para cambiar la manera en que ordenamos la información. Si nos fijamos en los COLLATION presentes para UTF-8, podemos apreciar de que existen 2 que llaman la atención: utf8_spanish_ci y utf8_spanish2_ci.

La diferencia entre ambos es que la primera se ocupa para español moderno mientras que la segunda se aplica a español tradicional.
De esta forma, podemos ver que ambos incorporan el uso de la "Ñ" como una letra entre la "N" y la "O", de forma que si tenemos los siguientes registros:

```
Nicolás
Ñandú
Operación
```

Ambos COLLATION ordenarán esos 2 registros de esa forma. Sin embargo, si tenemos los siguientes registros:

```
Cruzada
Baño
Carlos
Dedo
Chile
```

Una COLLATION utf8_spanish_ci ordenará los registros de una forma mientras que utf8_spanish2_ci la ordenará de otra, debido a que el español tradicional considera "ch" como una letra entre la "C" y la "D". Asimismo, considera el uso de la letra "LL" como una letra entre la "L" y "M".

Si quieren una prueba de concepto, ejecuten lo siguiente:

```
CREATE TABLE IF NOT EXISTS spanishCollation (
    name01 VARCHAR(15) CHARSET utf8 COLLATE utf8_spanish_ci,
    name02 VARCHAR(15) CHARSET utf8 COLLATE utf8_spanish2_ci
) ENGINE=MyISAM;

INSERT INTO spanishCollation VALUES ('Baño', 'Baño'),('Carlos', 'Carlos'),('Cruzada', 'Cruzada'),('Chile',
'Chile'),('Llorar', 'Llorar'),('Lámina', 'Lámina'),('Loreto', 'Loreto'),('Dedo', 'Dedo');

SELECT * FROM spanishCollation ORDER BY name01;
SELECT * FROM spanishCollation ORDER BY name02;
```

Al ejecutar las consultas se podrá ver claramente la diferencia entre ambas formas de ordenar los resultados.

Control de versiones (5)
CSS (5)
i18n/L10n (9)
Javascript/jQuery (8)
Linux (18)
Magento (2)
Mi mundo en noticias (7)
Mundo Web (28)
Pensamientos Personales (36)
PHP (43)

**Calendario de Posts**

AGOSTO 2012

| L | M | Mi | J | V | S | D |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | | |

« Jun Sep »

## Conclusiones

Aunque parece ser un tema menor, la verdad es que hay harto que pensar y decidir detrás de los CHARSETs y COLLATIONs. Por el lado del CHARSET, aunque parezca como mejor opción utilizar siempre UTF8, hay casos en que resulta totalmente inútil: si queremos guardar una cadena cuyo valor sabemos que no contendrá valores distintos del alfabeto, es mejor ocupar ASCII y de esa forma estar seguros que no se pueden ingresar caracteres inválidos. Por otro lado, si estamos seguros de que nuestra aplicación nunca jamás tendrá otro idioma que no sea español, también podemos utilizar latin1, aunque hay que tener cuidado en hacer los ajustes necesarios en todos aquellos lados donde podría haber una influencia de otro set de caracteres.

El tema del COLLATION sí es un poco más extenso ya que dependerá mucho del cómo se efectúen las búsquedas y también el idioma en que estemos trabajando. MySQL asigna utf8_general_ci como COLLATION predeterminado de UTF-8, pero éste presenta algunos problemas en hebreo y en algunas localizaciones de idiomas cirílicos; principalmente bielorruso, macedonio, serbio y ucraniano; así que para estos casos resulta mejor ocupar utf8_unicode_ci, la que tampoco está exenta de polémica ya que es más lento que utf8_general_ci.

Como gran conclusión: traten de usar UTF-8 como CHARSET donde sea posible, y si quieren tener velocidad vayan por utf8_general_ci como COLLATION pero si quieren tener certeza de que todo está bien ordenado y que sea compatible con (casi) todos los idiomas del mundo ocupen utf8_unicode_ci.

Fuentes:
code.openark.org
dev MySQL



**Sobre Camilo Sperberg**

Es Ingeniero Informático especializado en Linux y PHP (Es la primera persona en certificarse en PHP5.3 en Chile). En su tiempo libre le gusta estudiar nuevas técnicas de programación y escribir. Además, es amigo de todo ser viviente y cree que la tecnocracia es la mejor forma de politica.

Archivado en: Bases de Datos, i18n/L10n, Mundo Web, 31,537 vistas Deja un comentario

Comentarios (29) Trackbars (1) ( suscribirse a los comentarios de esta entrada )

**iri**
Mayo 25th, 2014 - 08:51

grax x la info me ayudo bastante

( RESPONDER )

**Gustavo**
Junio 8th, 2014 - 19:37

Camilo,
Gracias por explicar este tema. Interesante articulo.
Saludos.

( RESPONDER )

**Carlitox**
Junio 10th, 2014 - 12:08

Excelente aporte! Yo estoy con un problema, como vivo en Brasil desarrollo sistemas en portugues y a la hora de almacenar determinados valores en la base de datos o bien recuperarlos me encuentro con varios dolores de cabeza ya que si bien en portugues no se usa la ñ la acentuación de las palabras es bastante compleja (ç, ê, ã, õ, etc.) Que CHARSTE y/o COLLATION deberia utilizar en esots casos? Desde ya muchisimas gracias por la respuesta.

( RESPONDER )

**unreal4u**
Junio 11th, 2014 - 17:36

Charset siempre UTF-8. COLLATION sugiero utf8_general_ci siempre y cuando no trabajes con idiomas cirílicos y otros (lee las conclusiones para mayor información al respecto). De lo contrario, ocupa utf8_unicode_ci.

Saludos.

**pablo durón**
Julio 17th, 2014 - 17:17

el problema que tengo al migrar hacía UTF8 (utf8_general_ci) es que al realizar selects para recuperar por ejemplo “josé”, no me regresa valores porque en la tabla el valor del campo es “JOSÃ‰”, ¿alguna sugerencia para lograr buscar por “josé” y que me regrese todas las ocurrencias con “JOSÃ‰”?

Saludos desde México…

( RESPONDER )

**Raúl Olivares G.**
Octubre 5th, 2014 - 21:58

Saludos desde México Pablo, usa lo siguiente:

SELECT * FROM `usuarios` WHERE `nombre` = 'josé' COLLATE utf8_bin

Raúl Olivares G.

« Comentarios Anteriores

## Deja un comentario

Nombre (requerido)

Email (no será publicado) (requerido)

Página Web

Enviar

« ¿Qué tengo que hacer para sacar la certificación en PHP?

La odisea de instalar los developer tools en Lion »



Ir al inicio ↑