

Motores de bases de datos MySQL

Afortunadamente para nosotros, *MySQL* nos permite trabajar con diferentes motores de almacenamiento, entre los que destacan *MyISAM* e *InnoDB*.

Un motor de almacenamiento, es el encargado de almacenar, gestionar y recuperar toda la información de una tabla.

Para que conozcamos qué motores de almacenamiento podemos utilizar, basta con ejecutar la siguiente sentencia.

SHOW ENGINES;

Obtendremos un listado parecido al siguiente, que vamos a explicar a continuación.

- **InnoDB** es el motor de almacenamiento más utilizado con soporte para transacciones. (Las bases de datos transaccionales están diseñadas para cumplir los requisitos de ACID->Atomicidad, Consistencia, Aislamiento, Durabilidad. De esta forma, las operaciones de escritura en la base de datos se ejecutan correctamente o fallan en conjunto, manteniendo un alto nivel de integridad de datos al escribirlos en la base de datos). Admite bloqueo de nivel de fila, recuperación de fallos y control de concurrencia. Oracle recomienda usar InnoDB para tablas, excepto para casos de uso especializados.
- **MyISAM** es el motor de almacenamiento original. Es un motor de almacenamiento rápido. No admite transacciones. MyISAM proporciona bloqueo a nivel de tabla. Se utiliza principalmente en Web y almacenamiento de datos.
- El motor de almacenamiento **Memory** crea tablas en la memoria (de ahí el nombre). Es el motor más rápido. No admite transacciones. El motor de almacenamiento de memoria es ideal para crear tablas temporales o búsquedas rápidas. Los datos se pierden cuando se reinicia la base de datos.
- **CSV** almacena datos en archivos CSV (comma separated values). Proporciona una gran flexibilidad porque los datos en este formato se integran fácilmente en otras aplicaciones.
- **Merge** opera en tablas **MyISAM** subyacentes. Las tablas merge ayudan a administrar grandes volúmenes de datos con mayor facilidad. Agrupa

lógicamente una serie de tablas MyISAM idénticas y las referencia como un solo objeto.

- El motor de almacenamiento **Archive** (de archivos) está optimizado para la inserción de alta velocidad. Comprime los datos a medida que se insertan. No admite transacciones. Es ideal para almacenar y recuperar grandes cantidades de datos históricos archivados a los que rara vez se hace referencia.
- El motor de almacenamiento **Blackhole**, acepta pero no almacena datos. Las recuperaciones siempre devuelven un conjunto vacío. La funcionalidad está en el diseño de bases de datos distribuidas donde los datos se replican automáticamente, pero no se almacenan localmente. Este motor de almacenamiento se puede utilizar para realizar pruebas de rendimiento entre otras.
- El motor de almacenamiento **Federated** (federado) ofrece la capacidad de separar los servidores MySQL para crear una base de datos lógica a partir de muchos servidores físicos. Las consultas en el servidor local se ejecutan automáticamente en las tablas remotas (federadas). No se almacenan datos en las tablas locales. Es bueno para entornos distribuidos

Ahora nos centraremos en explicar los dos motores de almacenamiento más populares *MyISAM* e *InnoDB*.

MyISAM es el motor por defecto de *MySQL*. Una de las principales ventajas de este motor es la velocidad en el momento de recuperar información. *MyISAM* es una excelente opción cuando las sentencias predominantes en nuestra aplicación sean de consultas. Esta es una de las razones por las cuales *MyISAM* es tan popular en aplicaciones web.

La principal desventaja de *MyISAM* recae en que no se comprueba la integridad referencial de los datos. Se gana tiempo en la inserción, sí, pero perdemos confiabilidad en los datos.

Por otro lado, tenemos el motor de almacenamiento *InnoDB*. La principal ventaja de este motor recae en la seguridad de las operaciones. *InnoDB* permite la ejecución de transacciones, esto nos garantiza que los datos persisten de forma correcta y si existe algún error podamos revertir todos los cambios realizados.

Algo interesante a mencionar sobre *InnoDB* es que este motor realiza un bloqueo total sobre una tabla, cuando es ejecutada una de las siguientes sentencias.

- Select
- Insert
- Update
- Delete

Si deseamos trabajar con transacciones y la integridad de los datos es crucial, nuestra mejor opción será *InnoDB*. Por otro lado, si lo que deseamos es una mayor rapidez al obtener información, será necesario utilizar *MyISAM*.

Si deseamos conocer qué motor de almacenamiento utiliza una tabla en particular, podemos hacerlo ejecutando la siguiente sentencia.

`SHOW create table 'nombrede latabla';`

Si deseamos crear una tabla utilizando un motor en particular, debemos seguir la siguiente estructura.

```
CREATE TABLE tabla1 (id int, value int) ENGINE=INNODB;
```

```
CREATE TABLE tabla2 (id int, value int) ENGINE=MYISAM;
```

Si queremos utilizar el motor por defecto (innodb en el caso de mysql 8), no necesitamos indicar nada en la creación de tabla.

```
CREATE TABLE tabla_default (id int, value int);
```

Charset y collation

Conceptos

El charset es un conjunto de símbolos y codificaciones, es decir, la forma en que la base de datos guarda internamente *los datos*. Mientras que el collation es el conjunto de reglas que se aplican para comparar caracteres en un charset, es decir, indica a la base de datos como debe comparar *los datos*.

Cómo conocer el charset y collation de una base de datos.

Para ello usaremos la siguiente consulta

```
SELECT
  schema_name AS 'database',
  default_character_set_name AS 'charset',
  default_collation_name AS 'collation'
FROM
  information_schema.SCHEMATA
```

WHERE

```
schema_name = "XXXXX";
```

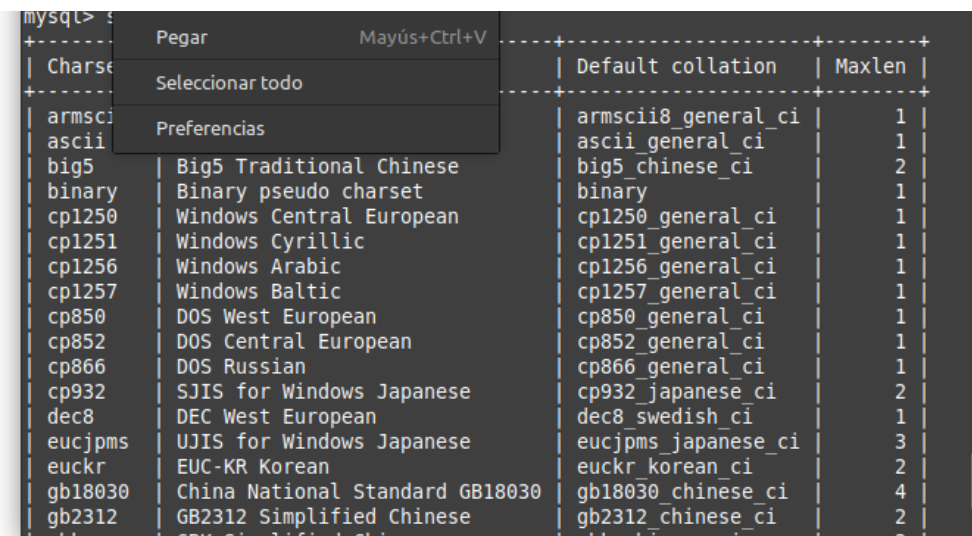
Donde XXXXX será sustituido por el nombre de la base de datos que queramos usar.

Algunos Collation y Charset incluidos en mysql

Con la sentencia

```
Show Character set;
```

Podremos ver ambas cosas en pantalla listadas en formato tabla. Obtendremos una salida parecida a la que sigue



Charset	Default collation	Maxlen
armscii8	armscii8_general_ci	1
ascii	ascii_general_ci	1
big5	big5_chinese_ci	2
binary	binary	1
cp1250	cp1250_general_ci	1
cp1251	cp1251_general_ci	1
cp1256	cp1256_general_ci	1
cp1257	cp1257_general_ci	1
cp850	cp850_general_ci	1
cp852	cp852_general_ci	1
cp866	cp866_general_ci	1
cp932	cp932_japanese_ci	2
dec8	dec8_swedish_ci	1
eucjpms	eucjpms_japanese_ci	3
euckr	euckr_korean_ci	2
gb18030	gb18030_chinese_ci	4
gb2312	gb2312_chinese_ci	2

En el caso del idioma español, se suele emplear como charset 'latin1' y como collation 'latin1_spanish_ci'

Modificación de tablas

```
ALTER TABLE tbl_name
```

```
[alter_option [, alter_option] ...]
```

```
alter_option: {
```

```
  table_options
```

```
| ADD [COLUMN] col_name column_definition
```

```
  [FIRST | AFTER col_name]
```

```
| ADD [COLUMN] (col_name column_definition,...)
```

```
| ADD {INDEX | KEY} [index_name]
```

```
  [index_type] (key_part,...) [index_option] ...
```

```
| ADD {FULLTEXT | SPATIAL} {INDEX | KEY} [index_name]
```

```
  (key_part,...) [index_option] ...
```

```
| ADD [CONSTRAINT [symbol]] PRIMARY KEY
```

```
  [index_type] (key_part,...)
```

```

    [index_option] ...
| ADD [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
    [index_name] [index_type] (key_part,...)
    [index_option] ...
| ADD [CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (col_name,...)
    reference_definition
| ADD [CONSTRAINT [symbol]] CHECK (expr) [[NOT] ENFORCED]
| DROP {CHECK | CONSTRAINT} symbol
| ALTER {CHECK | CONSTRAINT} symbol [[NOT] ENFORCED]
| ALGORITHM [=] {DEFAULT | INSTANT | INPLACE | COPY}
| ALTER [COLUMN] col_name {
    SET DEFAULT {literal | (expr)}
    | SET {VISIBLE | INVISIBLE}
    | DROP DEFAULT
}
| ALTER INDEX index_name {VISIBLE | INVISIBLE}
| CHANGE [COLUMN] old_col_name new_col_name column_definition
    [FIRST | AFTER col_name]
| [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
| CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
| {DISABLE | ENABLE} KEYS
| {DISCARD | IMPORT} TABLESPACE
| DROP [COLUMN] col_name
| DROP {INDEX | KEY} index_name
| DROP PRIMARY KEY
| DROP FOREIGN KEY fk_symbol
| FORCE
| LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
| MODIFY [COLUMN] col_name column_definition
    [FIRST | AFTER col_name]
| ORDER BY col_name [, col_name] ...
| RENAME COLUMN old_col_name TO new_col_name
| RENAME {INDEX | KEY} old_index_name TO new_index_name
| RENAME [TO | AS] new_tbl_name
| {WITHOUT | WITH} VALIDATION
}

```

Ejemplos:

Cambiar un tipo de datos

```
mysql> describe prueba3;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| a     | int           | NO   | PRI | NULL    |       |
| b     | varchar(8)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,01 sec)

mysql> ALTER TABLE prueba3 modify b int;
Query OK, 0 rows affected (0,05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe prueba3;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| a     | int  | NO   | PRI | NULL    |       |
| b     | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,00 sec)

mysql>
```

Renombrar un campo

```
mysql> describe prueba3;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| a     | int  | NO   | PRI | NULL    |       |
| b     | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,00 sec)

mysql> alter table prueba3 rename column b to edad;
Query OK, 0 rows affected (0,03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe prueba3;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| a     | int  | NO   | PRI | NULL    |       |
| edad  | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,00 sec)
```

Añadir restricción

```
mysql> alter table prueba3 add constraint mayordieciocho check (edad>18);
Query OK, 0 rows affected (0,06 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> show create table prueba3;
```

```
+-----+-----+
| Table | Create Table
+-----+-----+
| prueba3 | CREATE TABLE `prueba3` (
  `a` int NOT NULL,
  `edad` int DEFAULT NULL,
  PRIMARY KEY (`a`),
  CONSTRAINT `mayordieciocho` CHECK ((`edad` > 18))
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci |
+-----+-----+
```

Eliminar clave primaria

```
mysql> alter table prueba3 drop primary key;
Query OK, 0 rows affected (0,06 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> describe prueba3;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| a     | int  | NO   |     | NULL    |       |
| edad  | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,00 sec)
```