

Instalação ou Atualização do MySQLdump Personalizado

Preparação e Verificação

Primeiramente, é necessário verificar se a modificação do **MySQLdump** já está instalada, através do seguinte comando:

```
head -n 1 /usr/bin/mysqldump
```

Caso o retorno seja:

```
#!/usr/bin/php
```

A modificação está instalada e só poderá ser atualizada, seguir pro tópico ***Instalação do MySQLdump Modificado***.

Caso o retorno seja:

```
ELF>??@@8?r@8
```

Então a modificação não está instalada, seguir pro tópico abaixo.

Modificação do MySQLdump Original

Esse processo só pode ser executado se **não houver** sido instalado a modificação.

Será necessário renomear o **MySQLdump** original:

```
mv -f /usr/bin/mysqldump /usr/bin/mysqldump.ori
```

Instalação do MySQLdump Modificado

Não esqueça de verificar se a modificação já está instalada e se já foi preparada.

Criar o arquivo do **MySQLdump**:

```
echo "" > /usr/bin/mysqldump && chmod +x /usr/bin/mysqldump && nano /usr/bin/mysqldump
```

Inserir o seguinte comando:

```
#!/usr/bin/php
<?php

$command = array_shift($argv);
$credential = sys_get_temp_dir() . '/.mysql-credentials.cnf';
$extraFile = false;
$dataset = [];
$where = "";

$newArgs = [];
foreach ($argv as $arg) {
    if (strpos($arg, '--') !== false) {
        if (strpos($arg, '=') !== false) {
            $e = explode('=', $arg, 2);
            switch ($e[0]) {
                case '--defaults-extra-file':
                    $extraFile = $e[1];
                    $contentFile = file_get_contents($e[1]);
                    if (strpos($contentFile, 'db') !== false) {
                        if (!file_exists($credential)) {
                            file_put_contents($credential, str_replace(['db1', 'db2', 'localhost'], 'db3', $contentFile));
                        }
                        $e[1] = $credential;
                    }
                    break;
                case '--where':
                    $where = $e[1];
                    break;
            }
            $arg = "{$e[0]}='{$e[1]}'";
        }
    } else {
```

```

        $dataset[] = $arg;
        $arg = ""{$arg}"";
    }
    $newArgs[] = $arg;
}

if (in_array('--no-create-db', $newArgs) && $extraFile) {
    $database = array_shift($dataset);

    $contentFile = parse_ini_file($extraFile, true, INI_SCANNER_RAW);
    if (empty($contentFile['client']['default-character-set'])) {
        $contentFile['client']['default-character-set'] = 'utf8';
    }
    if (empty($contentFile['client']['port'])) {
        $contentFile['client']['port'] = '3306';
    }

    function isBinary($str)
    {
        return preg_match('~[^\x20-\x7E\t\r\n]~', $str) > 0;
    }

    $mysqlTypes = array(
        'numerical' => array(
            'bit',
            'tinyint',
            'smallint',
            'mediumint',
            'int',
            'integer',
            'bigint',
            'real',
            'double',
            'float',
            'decimal',
            'numeric'
        ),
        'blob' => array(
            'tinyblob',
            'blob',

```

```

        'mediumblob',
        'longblob',
        'binary',
        'varbinary',
        'bit',
        'geometry', /* http://bugs.mysql.com/bug.php?id=43544 */
        'point',
        'linestring',
        'polygon',
        'multipoint',
        'multilinestring',
        'multipolygon',
        'geometrycollection',
    )
);

$dbh = false;
try {

    $timezone = false;
    $timezoneFile = realpath(pathinfo($extraFile, PATHINFO_DIRNAME) . '/../timezone');
    if ($timezoneFile && file_exists($timezoneFile)) {
        $timezone = trim(str_replace(array("\r", "\n"), "", file_get_contents($timezoneFile)));
    }
    if (empty($timezone)) {
        $timezone = 'America/Sao_Paulo';
    }

    date_default_timezone_set($timezone);
    $off_set = date("P");

    $sslfile = '';
    if (file_exists('/var/www/BaltimoreCyberTrustRoot.crt.pem')) {
        $sslfile = '/var/www/BaltimoreCyberTrustRoot.crt.pem';
    }
    if (file_exists('/var/www/DigiCertGlobalRootCA.crt.pem')) {
        $sslfile = '/var/www/DigiCertGlobalRootCA.crt.pem';
    }
}

```

```

/**
 *
 */
$dbMySQLi = mysqli_init();
mysqli_options($dbMySQLi, MYSQLI_SET_CHARSET_NAME, $contentFile['client']['default-character-set']);
if (ini_get('mysqlnd_azure.enableRedirect') && !empty($sslfile)) {
    mysqli_ssl_set($dbMySQLi, NULL, NULL, $sslfile, NULL, NULL);
    mysqli_real_connect($dbMySQLi, $contentFile['client']['host'], $contentFile['client']['user'],
$contentFile['client']['password'], $database, $contentFile['client']['port'], MYSQLI_CLIENT_SSL);
} else {
    mysqli_real_connect($dbMySQLi, $contentFile['client']['host'], $contentFile['client']['user'],
$contentFile['client']['password'], $database, $contentFile['client']['port']);
}

/**
 *
 */
mysqli_set_charset($dbMySQLi, $contentFile['client']['default-character-set']);
mysqli_query($dbMySQLi, 'USE ` ` . $database . ` `');
mysqli_query($dbMySQLi, 'SET NAMES ` ` . $contentFile['client']['default-character-set'] . ` `');
mysqli_query($dbMySQLi, 'SET time_zone = ` ` . $off_set . ` `');
mysqli_query($dbMySQLi, 'SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;');

/**
 *
 */
$options = [PDO::ATTR_PERSISTENT => false];
if (ini_get('mysqlnd_azure.enableRedirect') && !empty($sslfile)) {
    $options[PDO::MYSQL_ATTR_SSL_CA] = $sslfile;
}

$dbh = new PDO('mysql:host=' . $contentFile['client']['host'] . ($contentFile['client']['port'] ?
";port={$contentFile['client']['port']}": "") . ";dbname=" . $database, "{$contentFile['client']['user']}",
"{$contentFile['client']['password']}"; $options);

$dbh->exec('USE ` ` . $database . ` `');
$dbh->exec('SET NAMES ` ` . $contentFile['client']['default-character-set'] . ` `');
$dbh->exec('SET time_zone = ` ` . $off_set . ` `');
$dbh->exec('SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;');

```

```

$net_buffer_length = 1000000;
$sizeOf = 0;

function writeLn($str)
{
    global $sizeOf;
    $sizeOf += strlen($str);
    echo $str;
}

function generateInsert($table, $cols, $columnTypes)
{
    $vks = [];
    foreach ($columnTypes as $colName => $colProp) {
        if (!$colProp['is_pk'] && !$colProp['is_uk']) {
            $vks[] = "`{$colName}` = VALUES(`{$colName}`)";
        }
    }
    writeLn("INSERT" . (count($vks) > 0 ? " " : " IGNORE ") . "INTO `{$table}` (`" . join("`", $cols) . "`)
VALUES ");
}

function generateDuplicateKey($columnTypes)
{
    global $sizeOf;
    $vks = [];
    foreach ($columnTypes as $colName => $colProp) {
        if (!$colProp['is_pk'] && !$colProp['is_uk']) {
            $vks[] = "`{$colName}` = VALUES(`{$colName}`)";
        }
    }
    if (count($vks) > 0 && $sizeOf > 0) {
        writeLn(" ON DUPLICATE KEY UPDATE " . join(', ', $vks) . ";" . PHP_EOL);
    } else {
        writeLn("; " . PHP_EOL);
    }
    $sizeOf = 0;
}

echo '/*!50503 SET NAMES ' . $contentFile['client']['default-character-set'] . ' */;' . PHP_EOL;

```

```
echo '/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;' . PHP_EOL;
```

```
echo '/*!40103 SET TIME_ZONE=\'' . $off_set . '\'' */;' . PHP_EOL;
```

```
echo PHP_EOL;
```

```
foreach ($dataset as $table) {
```

```
    echo '--' . PHP_EOL;
```

```
    echo '-- Dumping data for table \'' . $database . '\'.\' ' . $table . '\'.\' . PHP_EOL;
```

```
    echo '--' . PHP_EOL;
```

```
    echo PHP_EOL;
```

```
    $pks = [];
```

```
    $uks = [];
```

```
    // echo "-- >>> " . PHP_EOL;
```

```
    // echo "-- >>> " . PHP_EOL;
```

```
    // echo "-- >>> " . PHP_EOL;
```

```
    // echo "-- >>> SQL (carrega os relacionamentos da tabela): " . "SELECT CONSTRAINT_NAME,
TABLE_NAME, COLUMN_NAME FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA =
'{$database}' AND TABLE_NAME = '{$table}';" . PHP_EOL . PHP_EOL;
```

```
    mysqli_real_query($dbMySQLi, "SHOW INDEX FROM `{$table}`");
```

```
    $res = mysqli_use_result($dbMySQLi);
```

```
    while ($row = mysqli_fetch_assoc($res)) {
```

```
        if ($row['Key_name'] === 'PRIMARY') {
```

```
            $pks[] = $row['Column_name'];
```

```
        } else {
```

```
            if ((int) $row['Non_unique'] === 0) {
```

```
                $uks[] = $row['Column_name'];
```

```
            }
```

```
        }
```

```
    }
```

```
    unset($res);
```

```
    $fks = [];
```

```
    // echo "-- >>> " . PHP_EOL;
```

```
    // echo "-- >>> " . PHP_EOL;
```

```
    // echo "-- >>> " . PHP_EOL;
```

```
    // echo "-- >>> SQL (carrega os relacionamentos da tabela): " . "SELECT CONSTRAINT_NAME,
TABLE_NAME, COLUMN_NAME FROM information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA =
'{$database}' AND TABLE_NAME = '{$table}';" . PHP_EOL . PHP_EOL;
```

```
    mysqli_real_query($dbMySQLi, "SELECT CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME FROM
```

```

information_schema.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA = '{$database}' AND TABLE_NAME =
'{$table}';");

$res = mysqli_use_result($dbMySQLi);
while ($row = mysqli_fetch_assoc($res)) {
    if ($row['CONSTRAINT_NAME'] !== 'PRIMARY') {
        $fks[] = $row['COLUMN_NAME'];
    }
}
unset($res);

$cols = [];
$columnTypes = [];
// echo "-- >>> " . PHP_EOL;
// echo "-- >>> " . PHP_EOL;
// echo "-- >>> " . PHP_EOL;
// echo "-- >>> SQL (carrega os nomes das colunas e os tipos): " . "SHOW COLUMNS FROM `{$table}`;" .
PHP_EOL . PHP_EOL;

mysqli_real_query($dbMySQLi, "SHOW COLUMNS FROM `{$table}`");
$res = mysqli_use_result($dbMySQLi);
while ($row = mysqli_fetch_assoc($res)) {
    $cols[] = $row['Field'];
    $colInfo = [];
    $colParts = explode(" ", $row['Type']);
    if ($fparen = strpos($colParts[0], "(")) {
        $colInfo['type'] = substr($colParts[0], 0, $fparen);
        $colInfo['length'] = str_replace(" ", "", substr($colParts[0], $fparen + 1));
        $colInfo['attributes'] = isset($colParts[1]) ? $colParts[1] : null;
    } else {
        $colInfo['type'] = $colParts[0];
    }
    $columnTypes[$row['Field']] = array(
        'is_numeric' => in_array($colInfo['type'], $mysqlTypes['numerical']),
        'is_blob' => in_array($colInfo['type'], $mysqlTypes['blob']),
        'is_pk' => in_array($row['Field'], $pks),
        'is_uk' => in_array($row['Field'], $uks),
        'is_fk' => in_array($row['Field'], $fks),
        'type' => $colInfo['type'],
        'type_sql' => $row['Type'],
        'is_virtual' => strpos($row['Extra'], "VIRTUAL GENERATED") !== false || strpos($row['Extra'],
"STORED GENERATED") !== false

```



```

    );
}
unset($res);

/*
    Array
    (
        [0] => descricao
        [1] => valor
        [2] => observacao
        [3] => id_usuario
        [4] => data
    )
    Array
    (
        [descricao] => Array
            (
                [is_numeric] =>
                [is_blob] =>
                [type] => varchar
                [type_sql] => varchar(255)
                [is_virtual] =>
            )
        }
    */

$colStmt = [];
foreach ($columnTypes as $colName => $colType) {
    if ($colType['type'] == 'bit') {
        $colStmt[] = "LPAD(HEX(`{$colName}`),2,'0') AS `{$colName}`";
    } elseif ($colType['is_blob']) {
        $colStmt[] = "HEX(`{$colName}`) AS `{$colName}`";
    } else {
        $colStmt[] = "`{$colName}`";
    }
}

$sizeOf = 0;
$xRows = 0;
$withResult = false;

```

```

// echo "-- >>> " . PHP_EOL;
// echo "-- >>> " . PHP_EOL;
// echo "-- >>> " . PHP_EOL;
// echo "-- >>> SQL (extraí os dados das tabelas pra mandar os servidores locais): " . 'SELECT ' . join(",",
$colStmt) . ' FROM `'. $table . '`' . ($where ? 'WHERE ' . $where : '') . PHP_EOL . PHP_EOL;
mysqli_real_query($dbMySQLi, 'SELECT ' . join(",", $colStmt) . ' FROM `'. $table . '`' . ($where ? 'WHERE '
. $where : ''));

$res = mysqli_use_result($dbMySQLi);
if ($res) {
    while ($row = mysqli_fetch_assoc($res)) {
        $stop = false;

        if ($xRows === 0) {
            generateInsert($table, $cols, $columnTypes);
        } else {
            writeln(",");
        }

        writeln('(');
        $xCol = 0;
        foreach ($row as $col => $v) {

            if ($xCol > 0) {
                writeln(",");
            }

            $colType = $columnTypes[$col];
            if (is_null($v)) {
                writeln("NULL");
            } elseif ($colType['is_blob']) {
                if ($colType['type'] == 'bit' || !empty($v)) {
                    writeln("0x{$v}");
                } else {
                    writeln("''");
                }
            } elseif ($colType['is_numeric']) {
                writeln($v);
            } else {
                writeln($dbh->quote($v));
            }
        }
    }
}

```

```

        unset($colType);
        $xCol++;
    }
    unset($xCol);
    writeln('');

    $withResult = true;
    if ($sizeof > $net_buffer_length) {
        generateDuplicateKey($columnTypes);
        $xRows = 0;
    } else {
        $xRows++;
    }
}
unset($res, $xRows);

if ($withResult) {
    generateDuplicateKey($columnTypes);
}
}

echo PHP_EOL;
echo '/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;' . PHP_EOL;
} catch (PDOException $exc) {
    exit(1);
}
unset($dbh);
} else {
    $command .= '.ori ' . join(" ", $newArgs);
    passthru($command);
}

```

Revisão #: contagem de revisões

Criado: duração de tempo por usuário

Atualizado: duração de tempo por usuário