:: DEVELOPER ZONE

MySQL Manual | 12.4.2 The BLOB and TEXT Types

Search the MySQL manual:

Go

Additional languages

- French
- German
- Japanese
- Portuguese
- Russian

Additional formats

- PDF (A4)
- PDF (US letter)
- HTML, one page per chapter, tarball
- HTML, one page per chapter, ZIP
- HTML, all on one page, tarball
- HTML, all on one page, ZIP

Previous / Next / Up / Table of Contents

12.4.2 The BLOB and TEXT Types

A BLOB is a binary large object that can hold a variable amount of data. The four BLOB types, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB, differ only in the maximum length of the values they can hold. See section 12.5 Column Type Storage Requirements.

The four `TEXT` types, `TINYTEXT`, `TEXT`, `MEDIUMTEXT`, and `LONGTEXT`, correspond to the four `BLOB` types and have the same maximum lengths and storage requirements.

`BLOB` columns are treated as binary strings, whereas `TEXT` columns are treated according to their character set. Sorting and comparison for `BLOB` values is not case sensitive. As of MySQL 4.1, values in `TEXT` columns are sorted and compared based on the collation of the character set assigned to the column. Before MySQL 4.1, `TEXT` sorting and comparison are based on the collation of the server character set.

No lettercase conversion takes place during storage or retrieval.

If you assign a value to a `BLOB` or `TEXT` column that exceeds the column type's maximum length, the value is truncated to fit.

In most respects, you can regard a `TEXT` column as a `VARCHAR` column that can be as big as you like. Similarly, you can regard a `BLOB` column as a `VARCHAR BINARY` column. The ways in which `BLOB` and `TEXT` differ from `CHAR` and `VARCHAR` are:

- You can have indexes on `BLOB` and `TEXT` columns only as of MySQL 3.23.2. Older versions of MySQL did not support indexing these column types.
- For indexes on `BLOB` and `TEXT` columns, you must specify an index prefix length. For `CHAR` and `VARCHAR`, a prefix length is optional:
- There is no trailing-space removal for `BLOB` and `TEXT` columns when values are stored or retrieved. This differs from `CHAR` columns (trailing spaces are removed when values are retrieved) and from `VARCHAR` columns (trailing spaces are removed when values are stored).
- `BLOB` and `TEXT` columns cannot have `DEFAULT` values.

From version 4.1.0, `LONG` and `LONG VARCHAR` map to the `MEDIUMTEXT` data type. This is a compatibility feature.

Connector/ODBC defines `BLOB` values as `LONGVARBINARY` and `TEXT` values as `LONGVARCHAR`.

Because `BLOB` and `TEXT` values may be extremely long, you may encounter some constraints in using them:

- If you want to use `GROUP BY` or `ORDER BY` on a `BLOB` or `TEXT` column, you must convert the column value into a fixed-length object. The standard way to do this is with the `SUBSTRING` function. For example:

  ```
  mysql> SELECT comment FROM tbl_name,SUBSTRING(comment,20) AS substr
      ->                 ORDER BY substr;
  ```

  If you don't do this, only the first `max_sort_length` bytes of the column are used when sorting. The default value of `max_sort_length` is 1024; this value can be changed using the `--max_sort_length` option when starting the `mysqld` server. See section 5.2.3 Server System Variables. You can group on an expression involving `BLOB` or `TEXT` values by using an alias or by specifying the column position:

  ```
  mysql> SELECT id,SUBSTRING(blob_col,1,100) AS b
      -> FROM tbl_name GROUP BY b;
  mysql> SELECT id,SUBSTRING(blob_col,1,100)
      -> FROM tbl_name GROUP BY 2;
  ```

- The maximum size of a `BLOB` or `TEXT` object is determined by its type, but the largest value you actually can transmit between the client and server is determined by the amount of available memory and the size of the communications buffers. You can change the message buffer size by changing the value of the `max_allowed_packet` variable, but you must do so for both the server and your client program. For example, both `mysql` and `mysqldump` allow you to change the client-side `max_allowed_packet` value. See section 7.5.2 Tuning Server Parameters, section 8.3 `mysql`, the Command-Line Tool, and section 8.8 The `mysqldump` Database Backup Program.

Each `BLOB` or `TEXT` value is represented internally by a separately allocated object. This is in contrast to all other column types, for which storage is allocated once per column when the table is opened.

User Comments

Posted by [name withheld] on October 20 2003 5:59am [Delete] [Edit]

if we really go for up to 4 GB LONGBLOBS, some hints how to deal with your RAM that is most likely a lot smaller (http://bugs.mysql.com/bug.php?id=1605) and what to do when the communication gets interrupted (http://bugs.mysql.com/1606) - this not being that unlikely in presence of such large amounts of data - would be usefull

Posted by Daniel Applebaum on January 15 2004 11:05am [Delete] [Edit]

I retrieve large BLOBs by using repeatedly retrieving only sections of the BLOB
using substring, ie:
select substring(document, 1, 10240) from documents where docid='3';
and then
select substring(document, 10241, 10240) from documents where docid='3';
etc.
My servlet turns around and immediately writes the partial response to the
client so even the servlet does not need to allocate very much memory. In
practice, I use 10MB partial queries.

Posted by [name withheld] on February 10 2004 1:04am [Delete] [Edit]

How about a blob example? How does one get a binary into it? Is it like an attachment? I think I am probably not the only one who is puzzled.

Posted by [name withheld] on March 7 2004 4:29pm [Delete] [Edit]

Storing large files in a longblob I find is a bad idea. For a better implementation of mysql binary storage (in PHP, but concept could be implemented in any language) checkout this article:

http://php.dreamwerx.net/forums/viewtopic.php?t=6

Posted by andrei none on March 9 2004 5:38pm [Delete] [Edit]

Yes i would love to see some examples too, i have been trying to figure out a way to insert data into a text or blob but unsuccessfully.

Posted by [name withheld] on March 20 2004 12:46am [Delete] [Edit]

I don't understand the purpose of storing large amounts of data in a database. The host I'm using allows me 500MB of disk space and 128MB of database space, so I suppose I'd be more interested in conserving my database space than disk space. :>

I'd sooner store the glut of info, binary or otherwise, in a disk file and reference its filename in whichever table(s) I need.

It might get a tad messy if large numbers of files are stored, but it might be worth any potential savings in time, processing and headaches.

Posted by Remon Lapd on March 24 2004 8:33am [Delete] [Edit]

If you are connecting to a database server that is across the network, using local disk storage for Blobs/Clobs is not a good general solution. The storage used by the client writing the file may not be available to other clients. Only single-user applications that are always run from the same machine are guaranteed to work.

Posted by Laszlo Ujvari on April 18 2004 9:33am [Delete] [Edit]

Those of you who are asking about Inserting data into BLOBs. It has to be inserted in hex ie: 'A' = 0x41 and 'AB' = 0x4142 and so on. The prefix is a zero not a capital o. If you want some functions that can convert data to hex email me mailto:laszlo.ujvari@creativeintersection.com .

Posted by Scott Ellsworth on May 10 2004 3:54pm [Delete] [Edit]

Storing large amounts of information in a database can be a terrible idea, or it can be a very good one. It really depends on the data.

For example, if you have a library of documents in pdf form, you might want to put them on a web server, with the urls in a database, or you might want the pdfs streamed in as blobs, allowing web applications to generate download links as needed.

As a rule, consider the use of the data. If some ever would want to search by a field, it is probably worth creating a distinct database column to hold it. On the other hand, if someone is likely to want the content as a whole, then a blob would be a good idea. For example, even if I streamed the pdf in as a blob, I suspect I would want the title and abstract in a searchable db column.

Scott

Posted by George Johnson on May 31 2004 4:41am [Delete] [Edit]

Large amounts of binary data stored in a database does give one good advantage, when your backing up and restoring it, you only need backup the database files. As an Oracle DBA the general rule of thumb I was given was,I unless you do have some pressing need to store binary data in databases, do try to avoid it, as it does require little more effort to look after it and ensure the database continues to provide good performance.

Add your own comment.
Top / Previous / Next / Up / Table of Contents