Common - Bug #20

Statement overflow

16/10/2009 15:23 - Sven Vanpoucke

Status:	Bug resolved	**Start date:**	16/10/2009
Priority:	Normal	**Due date:**	
Assignee:	Sven Vanpoucke	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0	**Spent time:**	0.50 hour
Complexity:	Normal		

Description

When executing batch scripts around chamilo 2.0 sometimes an issue occures where you reach the max allowed prepared statements of your mysql installation. We should find a solution as to why he comes to that number of prepared statements because normally a prepared statement is made to be reusable.

History

#1 - 18/10/2009 22:32 - Yannick Warnier

Following: http://www.mysqlperformanceblog.com/2006/08/02/mysql-prepared-statements/:

"
Do not forget to close prepared statements – Many memory leaks reported in MySQL Server turned out to be prepare statements or cursors which were forgotten to be closed. Watch Com_stmt_prepare and Com_stmt_close to see if you're closing all prepared statements. In newer versions you can also use prepared_stmt_count variable to track number of open statements diretly. You can also adjust max_prepared_stmt_count variable which limits how many statements can be open at the same time to avoid overload.
"

#2 - 19/10/2009 08:32 - Sven Vanpoucke

Yes we've already free'd every prepared statement in the system and hereby solved the problem for now but aren't prepared statements meant to be reused instead of recreated every time a query is executed?

#3 - 20/10/2009 02:29 - Yannick Warnier

Sven Vanpoucke wrote:

> ... but aren't prepared statements meant to be reused instead of recreated every time a query is executed?

I would have thought so too, but maybe we are looking at it in a wrong way. Following the same MySQL performance page, prepared statements find their advantage in passing the query to MySQL in binary representation, instead of passing strings that have to be re-interpreted at the database server level. This means that you gain mostly on the interpretation time for this query. You gain even more if there is repetition, but in this case repetition seems to mean "immediate" repetition.
There is also a big filtering advantage in prepared statements, but this is not the point here.

#4 - 08/11/2009 20:05 - Yannick Warnier

I just reached a small section on prepared statements in the "High Performance MySQL" book (which I highly recommend), in the "Advanced MySQL features", page 225. Between other things, it says:

Quote starts here:
"When you create a prepared statement, the client library sends the server a prototype of the actual query you want to use. The server parses and processes this "skeleton" query, stores a structure representing the partially optimized query, and returns a **statement handle** to the client. The client library can execute the query repeatedly by specifying the statement handle.

[...]

Using prepared statements can be more efficient than executing a query repeatedly, for several reasons:

- The server has to parse the query only once, which saves some parsing and other work
- The server has to perform some query optimization steps only once, as it caches a partial query execution plan
- Sending parameters via the binary protocol is more efficient than sending them as ASCII text [...]
- Only the parameters - not the entire query text - need to be sent for each execution, which reduces network traffic
- MySQL stores the parameters directly into buffers on the server, which eliminates the need for the server to copy values around in memory

[..]

Many client libraries let you "prepare" statements with question-mark placeholders and then specify the values for each execution, but these libraries are often only emulating the prepare-execute cycle in client-side code and are actually sending each query to the server with mysql_query().

[...]

Limitations of Prepared Statements

Prepared statements have a few limitations and caveats:

- Prepared statements are local to a connection, so another connection cannot use the same handle. For the same reason, a client that disconnects and reconnects loses the statements. (Connection pooling or persistent connections can alleviate this problem)
- Prepared statements cannot use the MySQL query cache in MySQL versions prior to 5.1.
- It's not always more efficient to use prepared statements. **If you use a prepared statement only once, you may spend more time preparing it than you would just executing it as normal SQL.** Preparing a statement also requires an extra round-trip to the server.
- You cannot currently use a prepared statement inside a stored function (but you can use prepared statements inside stored procedures)
- **You can accidentally "leak" a prepared statement by forgetting to deallocate it.** This can consume a lot of resources on the server. Also, because there is a single global limit on the number of prepared statements, a mistake such as this can interfere with other connections' use of prepared statements.

"
Quote ends here.

All in all, it seems like prepared statements should be reserved to UPDATE and INSERT queries that affect a group of rows (let's say above 3) in the same script execution. I think the global limit for prepared statement could actually be a very big problem if we have more than 500 users connected simultaneously.

Could we make a coding convention out of this? Something of the like of **don't use prepared statements for less than 4 repetitions of any query** ?

#5 - 09/11/2009 13:51 - Sven Vanpoucke

- Assignee set to Sven Vanpoucke

I agree we use the prepared statements to much. Hans and me will replace most of the prepared statements in the system.

#6 - 16/11/2009 12:54 - Sven Vanpoucke

- Status changed from New to Bug resolved

- % Done changed from 0 to 100

Hans and me removed all the prepared statements and changed them to query. I hope this resolves a lot of problems in the performance.

#7 - 01/04/2011 15:39 - Stefaan Vanbillemont

- Project changed from Chamilo LCMS Connect to Common

- Category deleted (21)

#8 - 14/04/2011 13:17 - Stefaan Vanbillemont

- Target version changed from 2 to 1.0.0