Mozout

Generated on 15/11/2026 00:38

KNOWLEDGE BASE · PUBLIC DOCUMENT

MySQL server has gone away: the causes, in order of likelihood

Connection & SQL Errors

CODE	VERSION	COUNTRY · LANGUAGE
RLX-INT-KB-2144	v2026.10.18	International · English

MozOut International
RLX Holdings (Pty) Ltd. · Group RLX Holdings
Tax no. (SARS) 943 907 6234 · Registration no. (CIPC) 2025/667953/07

https://www.my.mozout.com/knowledgebase/2144

MySQL server has gone away: the causes, in order of likelihood

Connection & SQL Errors Created on **September 27, 2026** Updated on **October 18, 2026** 5 min read



It is one of the most misleading messages there is, because it seems to say the database server fell over. It almost never did. Taken literally, what it says is that the connection your program was holding **stopped existing partway through**, and the program only noticed when it tried to use it again.

There are four causes worth your time. They are listed here in order of likelihood, and the first three each have a number behind them, given as we go.

1. You sent a statement bigger than allowed

This is cause number one during an import, and the least suspected. Every statement reaching the engine has a maximum size, and here that maximum is **256 MB** (the stored value is `268435456` bytes). If one single statement goes over it, the engine does not return a tidy error: it closes the connection. At your end that reads as «gone away».

Note carefully what is being measured: the size of **one statement**, not the size of the file. An export several gigabytes long goes in without trouble if it is broken into ordinary lines. A far smaller file blows up if it holds one enormous statement inside it, or an image stored in a column.

> **If the import dies at the same point every time, you can prove it in two minutes:**
> Open the file and look at the size of the line it stops on. If it is huge, you have found it. The cure is to export the source database again in smaller statements, an option nearly every export tool has. For big files the reliable route is **importing over SSH**.

2. The server waited 30 seconds for you and gave up

Once the connection is accepted, the engine allows **30 seconds** to receive what comes next. That is `net_read_timeout`. Thirty seconds is an age for an ordinary query, and next to nothing for a file crawling up a weak connection.

It happens in two situations, and both are recognisable. The first is an import run from your own computer over a slow link: the file is still on its way, the engine gets tired of waiting and hangs up. The second is a program that opens the connection at the top, goes off to fetch something from elsewhere on the internet, and only then sends its query.

Open the connection when you need it, not at the head of the program.
It is one line of code moved, and it makes this entire cause disappear. And if it is the import that is slow, put the file on the server first and import it from there: then there is no network in the middle.

3. The connection sat idle for eight hours

This is the cause everyone blames and the rarest of the four. A connection left open and unused is closed after **28,800 seconds**, which is exactly **8 hours**. Those are the values of `wait_timeout` and `interactive_timeout`, the same on both.

Eight hours is a very long time. A web page opens and closes its connection in fractions of a second, so **if the error turns up on an ordinary website, this is not it**. Where it really bites is in a program that runs all day: a queue worker, a service of your own that connects once at startup, a sync job that wakes up now and then. Those have to know the connection may have died, and reconnect.

4. The query was cut short over usage

On shared servers there is a brake, CloudLinux's **MySQL Governor**, which stops one account taking over everybody's database. When an account goes past its share, what it is doing gets held back and can be cut off midway. Seen from the application, a query cut off midway is precisely this: the connection that was there is no longer there. The brake belongs to the server and has no screen in your panel; what the panel gives you is the account's resource usage page.

You recognise it by its behaviour: it happens at busy hours, on the heaviest pages, and it clears on its own when the traffic drops. To find what is doing the spending, see **what eats CPU on a website** and **watching your usage in real time over SSH**.

The table for deciding fast

The situation	The likely cause
During an import, at the same point in the file every time	The size of one statement, against the 256 MB. **Cause 1.**
During an import, at a different point each time	The 30-second wait, if the file is going up over the network. **Cause 2.**
In a program that runs all day, after some hours	The connection idle for eight hours. **Cause 3.**
On an ordinary site, only at busy hours, clearing on its own	Usage. **Cause 4.**
On an ordinary site, always on the same page	One query on that page doing far too much, and getting cut. Still cause 4, but with a named culprit.
On everything at once, and phpMyAdmin will not open either	Then it genuinely is something at our end. Change nothing and tell us.

Do not confuse it with the locking error.

If the message mentions *lock wait timeout*, that is another story: two operations wanting the same row at the same moment, and one of them giving up after **50 seconds** (`innodb_lock_wait_timeout`). The connection did not drop; the operation was refused. The cure is in the code that holds the row that long, not in the engine's configuration.

Whatever the cause, the line with the time and the file is in your account's PHP error log, and that is where to start: **where the PHP error log is**. If the error is a different one and the site will not connect at all, the list is here: **error establishing a database connection**.

NEED A HAND?

Tell us at which step it appears and what you were doing. With the time, we can see what happened on the server side.

Open a support ticket →

SEE ALSO

- Hosting plans and what each one includes →
- Support Policy: how far our help goes →

Mozout

MozOut International

RLX Holdings (Pty) Ltd. · Group RLX Holdings

TAX NO. (SARS)	943 907 6234
REGISTRATION NO. (CIPC)	2025/667953/07
REGISTERED OFFICE	1111 Olive Street Saint Louis 63101, Missouri, United States
CONTACT	info@mozout.com · +27 78 823 3325 · www.mozout.com
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

Online version of this document

RLX-INT-KB-2144 · v2026.10.18 · Generated on 15/11/2026 00:38