How to create a useful MySQL bug report

...and make sure it is properly processed

Valerii Kravchuk, Principal Support Engineer, MariaDB

vkravchuk@gmail.com

Who am I and why this is about MySQL bugs?

Valerii (aka **Valeriy**) **Kravchuk**:

- MySQL Support Engineer (bugs verification team) in MySQL AB, Sun and Oracle, 2005 - 2012
- Principal Support Engineer in Percona, 2012 - 2016
- Principal Support Engineer in MariaDB Corporation since March 2016
- **http://mysqlentomologist.blogspot.com** - my blog about MySQL (a lot about MySQL bugs, but some HowTos as well)
- **https://www.facebook.com/valerii.kravchuk** - my Facebook page, a lot about MySQL (mostly bugs, rants and links to blog posts…)
- **http://bugs.mysql.com** - it used to be my personal playground
- **@mysqlbugs** - **#bugoftheday** on Twitter
- I like FOSDEM, see slides from my previous talks:
  - http://www.slideshare.net/valeriikravchuk1/fosdem2015-gdb-tips-and-tricks-for-my-sql-db-as
  - http://www.slideshare.net/ValeriyKravchuk/more-on-gdb-for-my-sql-db-as-fosdem-2016
  - https://www.slideshare.net/ValeriyKravchuk/applying-profilers-to-my-sql-fosdem-2017

Topics to discuss

- Why should we report bugs at https://bugs.mysql.com (or other **public** tracker)?
- What is a “Useful MySQL Bug Report”?
- What to do before reporting a bug?
- How to search for known MySQL bugs efficiently?
- What tools may help to create a useful bug report?
- Life cycle of Oracle/MySQL public bug report
- Life cycle of MariaDB JIRA bug report
- Examples of useful bug reports
- Examples of useless bug reports
- How to escalate bug reports?

Always report bugs in public!

- Why? I am customer!
  - I tried to explain in a separate blog post
  - **Public bug reports are easy to search** for: **site:bugs.mysql.com drop add index InnoDB**
  - **Public bug reports becomes visible** to other customers, community members and MySQL support providers and developers outside of Oracle
- Public bug trackers:
  - https://bugs.mysql.com - MySQL software from Oracle
  - https://jira.mariadb.org/browse/MDEV - MariaDB Server
  - https://jira.percona.com/browse/PS - Percona Server
- What about security bugs?
  - Do NOT report bug as a "security" one unless it is really that serious

Features of a useful bug report

- It’s a real, new bug that was never reported before (or was once fixed but now appeared again, “**regression**”)
- **It’s clear what the bug is and how to reproduce it**
- Bug reporter cared to check on the latest officially released version in the branch (?) *What if it’s on a custom build from GitHub source code?*
- All important details about the environment and impact are provided
- It’s clear that bug reporter spent enough efforts on it (search, testing, minimal test case) (?)
- Anything wrong or missed in the above? (**use proper tags**)

Before reporting a bug...

- **Try to find similar already reported bugs**
- Try to find a solution/reason by reading the manual
- Try to reproduce the bug in a “clean environment”
- Make sure you’ve read vendor-specific (and product specific, if any) instructions:
  - https://mariadb.com/kb/en/mariadb/development/debugging-mariadb/reporting-bugs/
  - https://bugs.mysql.com/how-to-report.php
  - http://mysqlworkbench.org/faq/faq-bugreporting/
- Make sure you’ve read this entirely at least once: **http://www.catb.org/esr/faqs/smart-questions.html**

How to search for known bugs?

- Use Google like this:
  site:bugs.mysql.com drop table slow
  and check at least first 3 pages of results and links there
- Search for specific error messages, lines from stack trace, versions, filenames and line numbers, names
- Do **not** ignore bug tracker specific search:
  - Specific categories
  - Bugs vs. Feature requests
  - Specific version, bug status, q etc
- Study features and limitations of your search tools
- Keep your own notes and test cases “database”

Tools for bug reporters (and QA engineers)

- MySQL Sandbox (dbdeployer), Docker, VMs
- MTR (MySQL regression test suite)
- OS tools: **gdb**, strace, lsof, perf...
- Percona Toolkit (pt-summary, pt-mysql-summary, **pt-pmp**)
- sysbench in case of performance problems
- Debug binaries and trace
- Valgrind and Massif:
  - http://www.percona.com/blog/2013/01/09/profiling-mysql-memory-usage-with-valgrind-massif/
- Randgen (RQG)
  - http://www.percona.com/blog/2014/04/17/how-to-find-bugs-in-mysql/
- Percona QA tools:
  - http://www.percona.com/blog/2014/09/03/reducer-sh-a-powerful-mysql-test-case-simplificationreducer-tool/
- *Anything else?*

Life cycle of a MySQL bug

http://mysqlentomologist.blogspot.com/2013/01/life-cycle-of-mysql-bug.html

- Open
- Analyzing (usually assigned)
- Need Feedback → No Feedback (expires in 30 days)
- **Can't repeat**
- **Verified** → copied to internal database → Closed one day
- **Not a Bug**
- **Duplicate**
- **Unsupported**
- **Won't fix**
- In progress… (real progress happens elsewhere)
- **Closed** ← when fixed in internal database, before release
- *What about feature requests?*

Life cycle of a MariaDB (Server) bug (JIRA issue)

http://mysqlentomologist.blogspot.com/2018/12/mariadb-jira-for-mysql-dbas.html

- **OPEN** - may be assigned, may need feedback
- **CONFIRMED** - usually also assigned
- **STALLED** - some work on it was performed
- **IN PROGRESS** - assignee is currently working on the fix for the bug.
- **IN REVIEW** - assignee is currently reviewing the fix for the bug.
- **CLOSED** - the bug is resolved somehow. "Resolution":
    - Fixed
    - Duplicate
    - Won’t Fix
    - Cannot reproduce
    - Incomplete
    - Not a Bug
- See also a table that matches MySQL “Status” with MariaDB “Status” + “Resolution” + “Labels” here

Examples of useful bug reports

- Bug #93963 - steps to create big enough table are missed, but otherwise it's clear and important detail (regression vs 5.7) is highlighted
- Bug #93957 - refers to older bug (with proper public test case) that was closed/fixed, but seems to re-appear in 5.7.24.
- Bug #91941 - gdb backtrace from production, collective efforts

- Bug #69979 - test case, how repeatable read works (NaB, but useful reading!)
- Bug #73837 - how to report optimizer bugs
- Bug #68705 - Valgrind in use. Still just "Verified".
- Bug #73825 - simple wrong results bug. Still just "Verified".
- Bug #73881 - RQG in action. Still just "Verified".
- Bug #68554 - optimizer trace, level of details, how ICP works
- Bug #69253 - with patch suggested (mind OCA)
- Bug #73018 - even perfect reports with a patch and OCA signed may stay Open or Verified for years

Examples of useless bug reports

- Bug #55796 - “it really sucks”, gotcha… Read the manual, ask for a feature...
- Bug #73960 - “this query is very slow”, no facts, no feedback
- Bug #73921 - free support request, no feedback eventually
- Bug #73844 - near zero information, no feedback

How to escalate the bug?

Verified (or Open but real) bug is not fixed. What to do?

- **Cooperate**, try to provide feedback when requested
- Comment, ask, ~~curse~~, complain in report
- Ask others to comment, use “Affects Me” button or similar
- ~~Report it again?~~ Only if closed bug is still repeatable
- Report it in other bug tracker (to other company)
- Ask some developer in private
- Blog about it
- Open issue (support request) about it
- Complain in social media
- Tell me (or somebody who cares) about it

They process your bug reports!

- **Umesh Shastry** (**Oracle**, 50+% of all bugs Verified)
- Shane Bester (MySQL, Sun, Oracle)
- Miguel Solorzano (MySQL, Sun, Oracle)
- **Elena Stepanova** (Sun, Oracle, **MariaDB**)
- Sveta Smirnova (MySQL, Sun, Oracle, **Percona**)
- Laurynas Biveinis (Percona)
- Roel Van de Paar (MySQL, Sun, Oracle, Percona)

Any specific bug reports you want to discuss?

- Find me somewhere around today
- Ask me at Facebook or Twitter
- That’s what I’d like to discuss today:

  “Private” and “Security” bugs in public bugs databases. Should they exist, who decides on these statuses, when they should become public (including test cases)?

Thank you!

Questions and Answers?

Please, report bugs at:

https://bugs.mysql.com

https://jira.mariadb.org

https://jira.percona.com