Sesje, długie zapytania i blokady

Poziom: średni **Dotyczy: MySQL 5.7–8.0, MariaDB 10.x–11.x** **Czas: ~25 min**

Zapytanie, które wisi, i zapytanie, które czeka na blokadę trzymaną przez inne zapytanie, wyglądają z zewnątrz identycznie — oba stoją. Różnica widać dopiero, gdy zajrzymy głębiej niż `SHOW PROCESSLIST`*.*

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych

Fiszka

TYLKO ODCZYT

Wylistowanie pełnej listy procesów: SHOW FULL PROCESSLIST

```
SHOW FULL PROCESSLIST;
```

show = FortiOS: wypisuje konfigurację, w podstawowej postaci tylko ustawienia różniące się od domyślnych, a z wariantem full-configuration także wartości domyślne

ZMIENIA SYSTEM

Odpytanie listy procesów jako tabeli: information_schema.processlist

```
SELECT id, user, host, db, command, time, state, info
FROM information_schema.processlist
WHERE time > 60
```

```
ORDER BY time DESC;
```

□□ ZMIENIA SYSTEM

Przerwanie zapytania: KILL QUERY

```
KILL QUERY NumerId;
```

□□ **kill** = wyślij sygnał do procesu, domyślnie SIGTERM proszący o zamknięcie

□□ ZMIENIA SYSTEM

Przerwanie całego połączenia: KILL

```
KILL NumerId;
```

Wybór między nimi zależy od tego, czy aplikacja po drugiej stronie sama otworzy nowe połączenie po zerwaniu — pula połączeń (`connection pool`) zwykle to robi bezproblemowo, więc `KILL CONNECTION` jest tu prostsze niż celowanie w samo zapytanie.

□□ **kill** = wyślij sygnał do procesu, domyślnie SIGTERM proszący o zamknięcie

□□ TYLKO ODCZYT

Podgląd blokad wierszy InnoDB: performance_schema.data_locks

```
SELECT * FROM performance_schema.data_locks;
```

Od MySQL 8.0 i MariaDB 10.5 stan tych blokad odpytujemy bezpośrednio.

□□ TYLKO ODCZYT

Podgląd oczekujących blokad: performance_schema.data_lock_waits

```
SELECT * FROM performance_schema.data_lock_waits;
```

Widok `data_lock_waits` pokazuje wyłącznie sytuacje, w których jedna transakcja aktywnie czeka na blokadę trzymaną przez drugą — dokładnie to, czego szukamy przy diagnozowaniu zawieszonego zapytania.

TYLKO ODCZYT

Wyszukanie blokującej i czekającej transakcji: sys.innodb_lock_waits

```
SELECT * FROM sys.innodb_lock_waits\G
```

Przykład:

```
*************************** 1. row ***************************
                wait_started: 2026-03-14 10:02:18
                    wait_age: 00:00:12
               locked_table: `produkcja`.`zamowienia`
                locked_index: PRIMARY
                 locked_type: RECORD
              waiting_trx_id: 421983
         waiting_pid: 42
             waiting_query: UPDATE zamowienia SET wartosc = 100 WHERE id = 5
       blocking_trx_id: 421979
          blocking_pid: 38
        blocking_query: UPDATE zamowienia SET wartosc = 200 WHERE id = 5
```

Kolumna `blocking_pid` jest tym identyfikatorem, którego użylibyśmy w `KILL`, żeby odblokować sesję czekającą, jeśli blokująca transakcja rzeczywiście utknęła, a nie tylko chwilowo pracuje.

TYLKO ODCZYT

Wykrycie ostatniego zakleszczenia: SHOW ENGINE INNODB STATUS

```
SHOW ENGINE INNODB STATUS\G
```

Przykład (fragment sekcji):

```
------------------------
LATEST DETECTED DEADLOCK
------------------------
2026-03-14 10:15:44 0x7f2a1c200700
*** (1) TRANSACTION:
TRANSACTION 421990, ACTIVE 2 sec starting index read
mysql tables in use 1, locked 1
```

```
LOCK WAIT 3 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 44, OS thread handle, query id 88213 app01 nazwa_usera updating
UPDATE zamowienia SET wartosc = 300 WHERE id = 7
*** (2) TRANSACTION:
TRANSACTION 421989, ACTIVE 3 sec starting index read
*** WE ROLL BACK TRANSACTION (1)
```

Sekcja trzyma wyłącznie ostatni deadlock od ostatniego restartu serwera — kolejny nadpisuje poprzedni, więc do trwałej historii deadlocków ustawiamy `innodb_print_all_deadlocks = ON`, które kieruje każdy wykryty deadlock do logu błędów serwera.

□□ **show** = FortiOS: wypisuje konfigurację, w podstawowej postaci tylko ustawienia różniące się od domyślnych, a z wariantem full-configuration także wartości domyślne

□□ **ZMIENIA SYSTEM**

Włączenie logowania wolnych zapytań: slow_query_log i long_query_time

```
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 1;
```

Obniżenie progu do 1 sekundy albo niżej na czas diagnozowania problemu wydajnościowego jest bezpieczną zmianą tymczasową w locie — po zakończeniu diagnozy wracamy do wartości domyślnej, żeby log nie rósł nadmiernie na obciążonej instancji. Narzędzie `mysqldumpslow` (albo `pt-query-digest` z Percona Toolkit) agreguje wpisy logu w ranking najkosztowniejszych zapytań.

□□ **SET** = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP

□□ **TYLKO ODCZYT**

Uzyskanie planu wykonania zapytania: EXPLAIN

```
EXPLAIN SELECT * FROM zamowienia WHERE klient_id = 42;
```

Przykład:

```
+----+-------------+------------+------+---------------+--------------+---------
+-------+------+-------+
| id | select_type | table      | type | possible_keys | key          | key_len |
ref   | rows | Extra |
+----+-------------+------------+------+---------------+--------------+---------
+-------+------+-------+
|  1 | SIMPLE      | zamowienia | ref  | idx_klient_id | idx_klient_id | 4       | const
|  128 |       |
+----+-------------+------------+------+---------------+--------------+---------
+-------+------+-------+
1 row in set (0.00 sec)
```

□□ TYLKO ODCZYT

Uzyskanie planu z rzeczywistym czasem wykonania: EXPLAIN ANALYZE

```
EXPLAIN ANALYZE SELECT * FROM zamowienia WHERE klient_id = 42;
```

Przykład:

```
-> Index lookup on zamowienia using idx_klient_id (klient_id=42)
   (cost=25.80 rows=128) (actual time=0.045..0.312 rows=131 loops=1)
```

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

Dwóch handlowców z zespołu Kasi składa zamówienia na ten sam produkt niemal jednocześnie i dwie transakcje na my01 blokują się wzajemnie na tym samym wierszu. InnoDB sam wykrywa cykl i wycofuje jedną z transakcji, ale aplikacja sklepu B2B nie łapie tego błędu i pokazuje klientowi pustą stronę. Sprawdzamy `sys.innodb_lock_waits` i `SHOW ENGINE INNODB STATUS`, żeby pokazać zespołowi Oli dokładnie, które zapytania się zderzyły.

SHOW FULL PROCESSLIST kontra SHOW PROCESSLIST

Pojęcie: SHOW FULL PROCESSLIST. Rozszerzona wersja polecenia z poprzedniej strony Rozpoznanie serwera i baz — różnica jest jedna, ale istotna: kolumna `Info` z treścią wykonywanego zapytania nie jest obcinana do 100 znaków, tylko pokazuje pełen tekst. Bez `FULL` długie zapytanie z wieloma `JOIN`ami urywa się w połowie, co utrudnia rozpoznanie, które dokładnie zapytanie stoi.

Pojęcie: information_schema.processlist. Ten sam zestaw danych co `SHOW PROCESSLIST`, ale dostępny jako zwykła tabela do odpytania `SELECT`em z `WHERE` i `ORDER BY` — wygodne przy filtrowaniu po czasie trwania albo bazie, czego samo `SHOW PROCESSLIST` nie pozwala zrobić bez dodatkowego przetwarzania po stronie klienta.

KILL QUERY kontra KILL CONNECTION

Pojęcie: KILL. Polecenie przerywające sesję albo pojedyncze zapytanie po jego identyfikatorze `Id` z listy procesów. `KILL QUERY` przerywa wyłącznie aktualnie wykonywane zapytanie, zostawiając samo połączenie otwarte i gotowe na kolejne polecenia — `KILL` (odpowiednik `KILL CONNECTION`) zamyka całe połączenie, wraz z ewentualną otwartą transakcją, którą wtedy wycofuje.

performance_schema.data_locks i data_lock_waits

> **Pojęcie: blokada wiersza InnoDB.** Zabezpieczenie, które transakcja zakłada na wiersz przed jego modyfikacją, uniemożliwiające innej transakcji zmianę tego samego wiersza do czasu `COMMIT` albo `ROLLBACK` pierwszej — mechanizm szerzej opisany na stronie fundamentów Blokady: współdzielone, wyłączne, eskalacja.

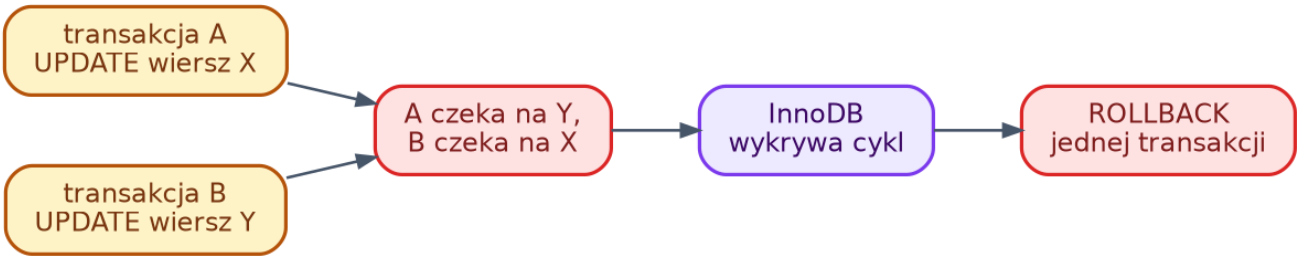
Na MySQL 5.7 ten sam obraz dawały tabele `information_schema.innodb_trx`, `innodb_locks` i `innodb_lock_waits`, usunięte w 8.0 na rzecz `performance_schema`.

sys.innodb_lock_waits — czytelny widok gotowy do użycia

> **Pojęcie: sys.innodb_lock_waits.** Widok z bazy `sys`, opisanej na stronie Architektura MySQL i MariaDB: silniki składowania, pliki, procesy, łączący dane z `performance_schema.data_locks` i `data_lock_waits` w jedną czytelną tabelę — pokazuje wprost, które zapytanie blokuje, które czeka i jak długo, bez ręcznego pisania złączeń po trzech tabelach `performance_schema`.

Zakleszczenie (deadlock)

> **Pojęcie: zakleszczenie (deadlock).** Sytuacja, w której dwie transakcje czekają wzajemnie na blokady trzymane przez siebie nawzajem, bez możliwości postępu żadnej z nich — InnoDB wykrywa taki cykl automatycznie i sam wycofuje (`ROLLBACK`) jedną z transakcji, żeby druga mogła kontynuować, zamiast czekać w nieskończoność. Ostatni wykryty deadlock trafia do sekcji `LATEST DETECTED DEADLOCK` polecenia opisanego na stronie Rozpoznanie serwera i baz.



Dwie transakcje czekają wzajemnie na swoje blokady, InnoDB wykrywa cykl i wycofuje jedną z nich (kliknięcie otwiera pełny rozmiar)

long_query_time i slow query log

> **Pojęcie: slow query log.** Osobny plik logu, do którego serwer zapisuje treść i czas wykonania każdego zapytania przekraczającego próg ustawiony parametrem `long_query_time` (domyślnie 10 sekund) — włączany parametrem `slow_query_log` opisanym na stronie my.cnf: konfiguracja serwera.

EXPLAIN i EXPLAIN ANALYZE

> **Pojęcie: EXPLAIN.** Polecenie zwracające plan wykonania zapytania bez jego faktycznego uruchomienia — pokazuje, którą tabelę optymalizator czyta pierwszą, czy używa indeksu i jakiego, oraz szacowaną liczbę przeglądanych wierszy.

> **Pojęcie: EXPLAIN ANALYZE.** Od MySQL 8.0.18 i MariaDB 10.1 — w odróżnieniu od zwykłego `EXPLAIN`, faktycznie wykonuje zapytanie i dokłada do planu rzeczywisty czas wykonania każdego kroku oraz rzeczywistą (nie szacowaną) liczbę wierszy, co pozwala zobaczyć, w którym miejscu szacunek optymalizatora rozjeżdża się z rzeczywistością.

Częste błędy

Diagnozowanie zawieszonego zapytania samym SHOW PROCESSLIST. Stan `Sending data` albo `Waiting for table metadata lock` w kolumnie `State` mówi, że coś się dzieje, ale nie mówi, na czym zapytanie czeka ani co je blokuje — dopiero `sys.innodb_lock_waits` albo `performance_schema.data_lock_waits` wskazują konkretną transakcję blokującą.

Traktowanie EXPLAIN jako gwarancji, a nie szacunku. Bez faktycznego wykonania optymalizator opiera się na statystykach tabeli, które po dużym imporcie danych bez odświeżenia (`ANALYZE TABLE`) potrafią być mocno nieaktualne — plan z `EXPLAIN` bywa wtedy inny niż rzeczywiste zachowanie, które ujawnia dopiero `EXPLAIN ANALYZE`.

Powiązane: Rozpoznanie serwera i baz · Architektura MySQL i MariaDB: silniki składowania, pliki, procesy · Blokady: współdzielone, wyłączne, eskalacja

Źródła: dokumentacja MySQL 8.0, rozdział „InnoDB Lock and Deadlock Information" i „EXPLAIN Statement" · dokumentacja MariaDB, rozdział „Slow Query Log"

Revision #41
Created 2026-09-25 00:00:34 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:21 UTC by Tomasz Konieczny