Rozpoznanie i diagnostyka

Rozmiary, sesje, długie zapytania i blokady.

- Rozpoznanie serwera i baz
- Sesje, długie zapytania i blokady

Rozpoznanie serwera i baz

Poziom: podstawy | Dotyczy: MySQL 5.7–8.0, MariaDB 10.x–11.x | Czas: ~20 min

Przed jakąkolwiek interwencją na przejętym serwerze sprawdzamy zawsze to samo: co to za wersja, jak długo działa, ile zajmuje, kto jest podłączony i co InnoDB robi w tej chwili wewnątrz siebie.

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

Sprawdzenie wersji serwera: SELECT VERSION()

```
SELECT VERSION();
```

Sprawdzamy dokładny numer wersji i to, czy odpowiada MySQL czy MariaDB, po sufiksie w wyniku.

Przykład:

```
+-----------------+
| VERSION()       |
+-----------------+
| 8.0.35          |
+-----------------+
1 row in set (0.00 sec)
```

🔒 TYLKO ODCZYT

Sprawdzenie czasu działania serwera: SHOW STATUS LIKE Uptime

```
SHOW STATUS LIKE 'Uptime';
```

Sprawdzamy, ile sekund minęło od ostatniego startu procesu — zerowanie tej wartości informuje, że gdzieś doszło do restartu, świadomego albo po awarii.

Przykład:

```
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| Uptime        | 362845 |
+---------------+--------+
1 row in set (0.00 sec)
```

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

🔒 TYLKO ODCZYT

Sprawdzenie rozmiaru wszystkich baz danych: information_schema.tables

```
SELECT table_schema AS baza,
       ROUND(SUM(data_length + index_length) / 1024 / 1024, 1) AS rozmiar_mb
FROM information_schema.tables
GROUP BY table_schema
ORDER BY rozmiar_mb DESC;
```

Liczymy sumę danych i indeksów każdej bazy w megabajtach — pierwsze pytanie przy planowaniu backupu albo migracji, kiedy trzeba wiedzieć, ile realnie danych jest do przeniesienia.

Przykład:

```
+--------------------+------------+
| baza               | rozmiar_mb |
+--------------------+------------+
```

```
| produkcja          |     4821.3 |
| mysql              |        8.2 |
| performance_schema |        0.0 |
| information_schema |        0.1 |
+--------------------+------------+
4 rows in set (0.14 sec)
```

🔒 TYLKO ODCZYT

Sprawdzenie rozmiaru tabel w jednej bazie: information_schema.tables z WHERE

```
SELECT table_name AS tabela,
       table_rows AS wierszy_przyblizonych,
       ROUND(data_length / 1024 / 1024, 1) AS dane_mb,
       ROUND(index_length / 1024 / 1024, 1) AS indeksy_mb
FROM information_schema.tables
WHERE table_schema = 'NazwaBazy'
ORDER BY data_length DESC
LIMIT 10;
```

Rozbijamy rozmiar bazy na poszczególne tabele, żeby znaleźć te największe — `table_rows` dla InnoDB jest wartością szacunkową ze statystyk optymalizatora, nie dokładnym `COUNT(*)`.

Przykład:

```
+-------------+----------------------+---------+------------+
| tabela      | wierszy_przyblizonych | dane_mb | indeksy_mb |
+-------------+----------------------+---------+------------+
| zamowienia  |                48213 |   589.2 |      112.4 |
| klienci     |                 9284 |    42.1 |       18.3 |
+-------------+----------------------+---------+------------+
2 rows in set (0.09 sec)
```

🔒 TYLKO ODCZYT

Sprawdzenie liczby aktywnych połączeń: SHOW STATUS LIKE Threads_connected

```
SHOW STATUS LIKE 'Threads_connected';
```

Sprawdzamy, ile połączeń jest aktualnie otwartych, w porównaniu z limitem `max_connections` opisanym na stronie my.cnf: konfiguracja serwera.

Przykład:

```
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Threads_connected | 24    |
+------------------+-------+
1 row in set (0.00 sec)
```

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

TYLKO ODCZYT

Wylistowanie wszystkich aktywnych procesów: SHOW PROCESSLIST

```
SHOW PROCESSLIST;
```

Zwracamy każde otwarte połączenie z jego bieżącym stanem i wykonywanym zapytaniem — bez uprawnienia `PROCESS` widzimy wyłącznie własne połączenia, z tym uprawnieniem widzimy wszystkie. Szersze omówienie i kasowanie konkretnych sesji rozwija strona Sesje, długie zapytania i blokady.

Przykład:

```
+----+----------+-----------+-----------+---------+------+----------
+-----------------------+
| Id | User     | Host      | db        | Command | Time | State    |
Info                  |
+----+----------+-----------+-----------+---------+------+----------
+-----------------------+
|  5 | nazwa_usera | app01:44210 | produkcja | Query   |    3 | Sending data | SELECT *
FROM zamowienia |
|  8 | nazwa_usera | localhost | NULL      | Sleep   |   42 |
|                       |
```

```
+----+----------+-----------+-----------+---------+------+----------
+-----------------------+
2 rows in set (0.00 sec)
```

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

TYLKO ODCZYT

Podgląd wewnętrznego stanu silnika InnoDB: SHOW ENGINE INNODB STATUS

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

> **Pojęcie: SHOW ENGINE INNODB STATUS.** Polecenie zwracające wieloekranowy zrzut wewnętrznego stanu InnoDB w danej chwili — sekcję `TRANSACTIONS` z listą aktywnych transakcji i ich blokad, sekcję `LATEST DETECTED DEADLOCK` z ostatnim wykrytym zakleszczeniem, sekcję `BUFFER POOL AND MEMORY` z wykorzystaniem bufora i sekcję `LOG` ze stanem redo logu. To najgęstsze źródło informacji o tym, co InnoDB naprawdę robi, ale format tekstowy nie nadaje się do parsowania w skryptach — do tego lepiej sięgnąć po widoki `performance_schema` opisane na stronie Sesje, długie zapytania i blokady.

Przykład (fragment):

```
=====================================
2026-03-14 09:41:02 0x7f2a1c3fe700 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 15 seconds
-----------------
BUFFER POOL AND MEMORY
-----------------
Total large memory allocated 8606711808
Buffer pool size   524288
Free buffers       12044
Database pages     511890
Modified db pages   842
Pending reads 0
```

```
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 128420, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Buffer pool hit rate 1000 / 1000
```

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

TYLKO ODCZYT

Sprawdzenie ogólnych liczników pracy serwera: SHOW STATUS

```
SHOW STATUS LIKE 'Com_%';
```

Sprawdzamy liczniki wykonanych poleceń według typu (`Com_select`, `Com_insert`, `Com_update`) od ostatniego startu serwera — pierwszy krok do oceny profilu obciążenia (czy przeważa odczyt, czy zapis) bez sięgania po zewnętrzne narzędzia monitoringu.

Przykład:

```
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| Com_select    | 984213 |
| Com_insert    |  48291 |
| Com_update    |  12043 |
| Com_delete    |    892 |
+---------------+--------+
4 rows in set (0.00 sec)
```

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

Z życia wzięte

Z ŻYCIA WZIĘTE

Przejmujemy serwer my01 po zewnętrznym dostawcy sklepu B2B i pierwsze pytanie brzmi, ile w ogóle waży baza `sklep_b2b`. Zapytanie do `information_schema.tables` pokazuje, że jedna

> tabela z historią koszyków zajmuje więcej miejsca niż wszystkie tabele zamówień razem wzięte. Okazuje się, że nikt nigdy nie ustawił czyszczenia starych, porzuconych koszyków.

Skąd bierzemy te informacje

Rozpoznanie serwera MySQL opiera się na trzech źródłach, które wracają na tej stronie wielokrotnie. information_schema odpowiada na pytania o metadane — ile jest baz, ile ważą, jakie mają tabele — bez wpływu na sam serwer, bo to zwykłe `SELECT` do widoków katalogowych. SHOW STATUS i pokrewne polecenia `SHOW` zwracają liczniki i stan bieżący samego procesu `mysqld`, zerowane przy każdym restarcie. SHOW ENGINE INNODB STATUS schodzi jeszcze niżej, pokazując wewnętrzny stan silnika InnoDB — kolejki, transakcje, ostatni deadlock — którego żaden widok `information_schema` nie ujawnia wprost.

Częste błędy

Traktowanie table_rows jako dokładnej liczby wierszy InnoDB. Dla tabel InnoDB `information_schema.tables.table_rows` jest estymacją opartą na statystykach optymalizatora, potrafiącą różnić się od rzeczywistości o kilkanaście procent na tabelach z dużym ruchem `UPDATE`/`DELETE` — dokładną liczbę daje wyłącznie `SELECT COUNT(*)`, kosztem pełnego skanu.

Czytanie SHOW ENGINE INNODB STATUS bez \G. Bez terminatora `\G` wynik wraca jako jedna szeroka kolumna tekstu, praktycznie nieczytelna w standardowym terminalu — terminator pionowy jest tu obowiązkowy, nie kosmetyczny.

Powiązane: Architektura MySQL i MariaDB: silniki składowania, pliki, procesy · Sesje, długie zapytania i blokady · my.cnf: konfiguracja serwera

Źródła: dokumentacja MySQL 8.0, rozdział „The INFORMATION_SCHEMA TABLES Table" i „SHOW ENGINE Statement"

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.

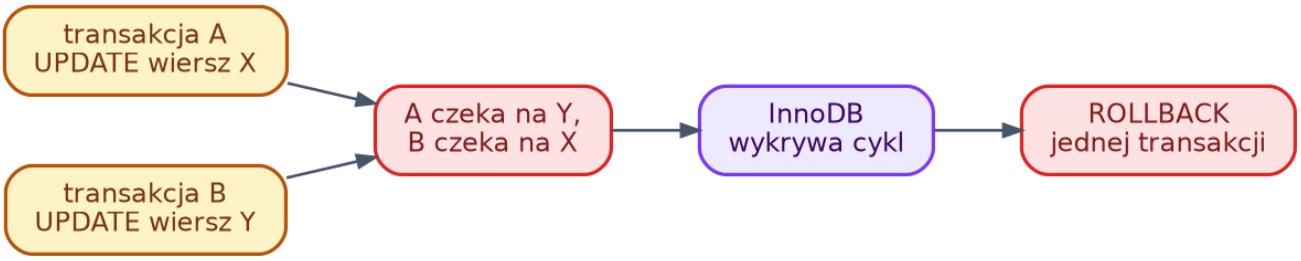
Na MySQL 5.7 ten sam obraz dawały tabele `information_schema.innodb_trx`, `innodb_locks` i `innodb_lock_waits`, usunięte w 8.0 na rzecz `performance_schema`.

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"