Architektura i konfiguracja

Silniki składowania, pliki, klient mysql oraz my.cnf.

- MySQL i MariaDB: jedna baza, dwie córki Monty'ego
- Architektura MySQL i MariaDB: silniki składowania, pliki, procesy
- Klient mysql i połączenie
- my.cnf: konfiguracja serwera

MySQL i MariaDB: jedna baza, dwie córki Monty'ego

Poziom: podstawy **Dotyczy: MySQL 5.7-8.0, MariaDB 10.x-11.x** **Czas: ~15 min**

Ta strona otwiera książkę „MySQL | MariaDB": dlaczego oba silniki noszą imiona córek jednego programisty, jak z jednego projektu zrobiły się dwa i co czeka na kolejnych stronach.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
GPL	General Public License: otwarta licencja oprogramowania, na której opierają się MySQL Community i MariaDB
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
MSSQL	Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Skąd się wzięło

W 1995 roku szwedzka firma MySQL AB, założona przez Davida Axmarka, Allana Larssona i fińskiego programistę Michaela Wideniusa, wydała pierwszą wersję silnika MySQL. Nazwa łączy imię starszej córki Wideniusa, My, ze skrótem SQL, a sam silnik od początku stawiał na prostotę i szybkość kosztem części funkcji, które oferowały wtedy dużo droższe silniki komercyjne. Przez kolejną dekadę MySQL stał się domyślnym wyborem dla stron internetowych na całym świecie, w dużej mierze dzięki darmowej licencji GPL i łatwości instalacji.

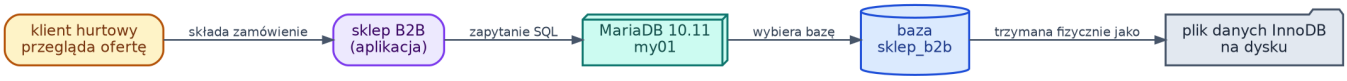
W 2008 roku MySQL AB zostało kupione przez Sun Microsystems, a niecały rok później samo Sun trafiło w ręce Oracle Corporation, firmy sprzedającej konkurencyjny silnik komercyjny. Michael Widenius i część oryginalnych twórców MySQL obawiali się, że Oracle ograniczy otwarty rozwój projektu, więc jeszcze w 2009 roku, zanim przejęcie Sun formalnie się zamknęło, założyli fork o nazwie MariaDB, nazwany od imienia jego młodszej córki. Pierwsza wersja MariaDB trafiła do użytkowników w październiku 2009 roku, zbudowana tak, żeby zostać w pełni zamiennikiem

MySQL, wiersz po wierszu kodu SQL.

Do wersji MySQL 5.5 oba projekty rozwijały się niemal identycznie: ten sam format plików, ten sam protokół sieciowy, te same komendy. Od tego momentu MariaDB zaczęła iść własną drogą, dopisując funkcje, których MySQL nie miał jeszcze przez lata, podczas gdy Oracle rozwijał MySQL w stronę własnych priorytetów, w tym funkcji dostępnych tylko w płatnej edycji Enterprise. Dziś oba silniki dzielą wspólny rdzeń SQL i ten sam domyślny silnik składowania InnoDB, ale formalnie są dwoma osobnymi produktami z osobnymi wydaniami.

Jak to działa w jednym obrazku

Zapytanie w MySQL i MariaDB przechodzi ten sam prosty łańcuch: aplikacja łączy się z procesem `mysqld`, ten wybiera właściwą bazę, a dane trafiają fizycznie na dysk jako plik silnika składowania InnoDB.



Aplikacja sklepu B2B wysyła zapytanie SQL do MariaDB na my01, która wybiera bazę sklep_b2b zapisaną fizycznie jako plik InnoDB na dysku (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

> **Z ŻYCIA WZIĘTE**
>
> Sklep B2B kupiony przez Zieloną od zewnętrznego dostawcy przyszedł z gotową bazą na MariaDB, jeszcze zanim ktokolwiek z zespołu IT zdążył zapytać, czy wolałby PostgreSQL. Ola przy pierwszym wdrożeniu zauważyła w dokumentacji dostawcy zapytania z `RETURNING` w `INSERT`, których na starym MySQL po prostu by nie uruchomiła. Dopiero wtedy zespół sprawdził w wersji na serwerze `my01`, że stoi tam MariaDB, nie MySQL, mimo że klient `mysql` łączy się z oboma tak samo.

Na `my01` MariaDB trzyma bazę `sklep_b2b`, obsługującą zamówienia odbiorców hurtowych, i podobnie jak przy Optimie na MSSQL, wybór silnika przyszedł razem z zakupioną aplikacją. Różnica polega na tym, że MariaDB i MySQL wyglądają na pierwszy rzut oka identycznie, więc łatwo przeoczyć, którego dokładnie silnika dotyczy dokumentacja, jaką komendę backupu trzeba użyć i której wersji funkcji można w ogóle oczekiwać. Reszta tej książki uczy rozpoznawać te różnice,

zanim zrobi to produkcja.

Mapa tej książki

Książka „MySQL | MariaDB" składa się z czterech rozdziałów, budowanych na słowniku z książki „Bazy danych: fundamenty".

Rozdział	Na jakie pytanie odpowiada
MySQL i MariaDB: jedna baza, dwie córki Monty'ego	Skąd wzięły się oba silniki i czym się dziś od siebie różnią
Architektura i konfiguracja	Czym jest proces `mysqld`, silniki składowania i plik `my.cnf`
Rozpoznanie i diagnostyka	Jak sprawdzić rozmiary, sesje, długie zapytania i blokady
Użytkownicy i uprawnienia	Konta, hosty, role i konta techniczne
Backup i replikacja	Logi binarne, `mysqldump`, backup fizyczny i replikacja

Powiązane: Baza danych, silnik, instancja i schemat · Architektura MySQL i MariaDB, silniki składowania, pliki, procesy · Rozpoznanie serwera i baz

Źródła: Wikipedia, „MySQL" · Wikipedia, „MariaDB" · Wikipedia, „Michael Widenius" · Tecmint, „The Story Behind Acquisition of MySQL by Sun Microsystem and the Rise of MariaDB"

Architektura MySQL i MariaDB: silniki składowania, pliki, procesy

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

Zanim zaczniemy cokolwiek administrować, ustalamy, czym w ogóle jest to, co uruchamiamy poleceniem `systemctl start mysqld` — jeden proces, jeden katalog danych i kilka silników składowania ukrytych pod jednym językiem SQL.

Skróty użyte na tej stronie (10)

Skrót	Co znaczy
ACID	Atomicity, Consistency, Isolation, Durability: cztery obietnice transakcji w bazie danych
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
CSV	w Windowsie: Cluster Shared Volume, dysk widoczny naraz dla wszystkich węzłów klastra; w danych: Comma-Separated Values, plik z wartościami oddzielonymi przecinkami
CTE	Common Table Expression: nazwany kawałek zapytania SQL pisany w WITH
GTID	Global Transaction Identifier: unikalny w całym klastrze numer nadawany każdej transakcji w MySQL i MariaDB, ułatwia przełączenie repliki na nowe źródło bez ręcznego szukania pozycji
MVCC	Multi-Version Concurrency Control: baza trzyma kilka wersji wiersza, żeby czytający nie blokował piszącego
RAM	Random Access Memory: pamięć operacyjna
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia

Fiszka

TYLKO ODCZYT

Sprawdzenie zainstalowanych silników składowania: SHOW ENGINES

```
SHOW ENGINES;
```

Sprawdzamy, które silniki składowania są wkompilowane i dostępne w danej instancji oraz który z nich jest domyślny.

Przykład:

```
+--------------------+---------
+-------------------------------------------------------------+--------------+------
+------------+
| Engine             | Support |
Comment                                                       | Transactions | XA   |
Savepoints |
+--------------------+---------
+-------------------------------------------------------------+--------------+------
+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign
keys     | YES          | YES  | YES        |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary
tables      | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage
engine                                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage
engine                                              | NO           | NO   | NO
|
| ARCHIVE            | YES     | Archive storage
engine                                          | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance
Schema                                              | NO           | NO   | NO
|
+--------------------+---------
+-------------------------------------------------------------+--------------+------
+------------+
6 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

Sprawdzenie silnika konkretnej tabeli: SHOW TABLE STATUS

```
SHOW TABLE STATUS FROM NazwaBazy LIKE 'NazwaTabeli';
```

Sprawdzamy, na jakim silniku leży istniejąca tabela, razem z jej przybliżonym rozmiarem i liczbą wierszy — przydatne przy przejmowaniu bazy po kimś, gdzie nie wiadomo, czy ktoś zostawił tabele na MyISAM.

Przykład:

```
*************************** 1. row ***************************
           Name: zamowienia
         Engine: InnoDB
Row_format: Dynamic
           Rows: 48213
 Avg_row_length: 128
    Data_length: 6176768
Auto_increment: 48214
    Create_time: 2026-03-01 09:12:04
```

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

□□ TYLKO ODCZYT

Sprawdzenie wersji serwera: SELECT VERSION()

```
SELECT VERSION();
```

Sprawdzamy pełny numer wersji wraz z sufiksem dystrybucji — sufiks `-MariaDB` w wyniku odróżnia MariaDB od MySQL, bo obie odpowiadają na to samo polecenie.

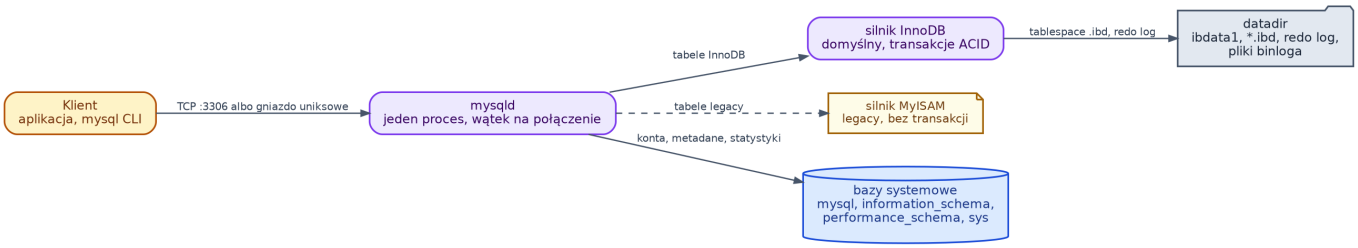
Przykład:

```
+-----------------+
| VERSION()       |
+-----------------+
| 10.11.6-MariaDB |
```

```
+-----------------+
1 row in set (0.00 sec)
```

Czym jest mysqld

> **Pojęcie: mysqld.** Pojedynczy proces serwera, który przyjmuje połączenia TCP i gniazda uniksowego, parsuje SQL, planuje wykonanie zapytania i deleguje właściwe czytanie oraz zapisywanie danych do wybranego silnika składowania. W przeciwieństwie do PostgreSQL, który na każde połączenie odpala osobny proces systemowy, `mysqld` jest wielowątkowy — jedno połączenie to jeden wątek wewnątrz tego samego procesu, nie osobny proces w `ps aux`. Widzimy więc w systemie jeden proces `mysqld`, niezależnie od tego, czy obsługuje pięć czy pięćset połączeń.



Jeden proces mysqld obsługuje połączenia wątkami i deleguje odczyt i zapis do silnika InnoDB albo MyISAM, które trzymają dane w katalogu datadir (kliknięcie otwiera pełny rozmiar)

Proces `mysqld` po starcie czyta plik konfiguracyjny, otwiera katalog danych (`datadir`), inicjalizuje bufor InnoDB w pamięci i zaczyna nasłuchiwać. Zatrzymanie usługi to zawsze zamknięcie tego jednego procesu — nie ma czegoś takiego jak proces `checkpointer` czy `wal writer` widoczny osobno w liście procesów, jak w PostgreSQL; te role pełnią wątki wewnętrzne InnoDB, niewidoczne z poziomu `ps`.

Z życia wzięte

> **Z ŻYCIA WZIĘTE**

> O 9:03 Tomek próbuje znaleźć proces checkpointer na my01 poleceniem `ps aux`, tak jak nauczył się na PostgreSQL. Widzi tylko jeden proces `mysqld` obsługujący sklep B2B i musi się przestawić na myślenie o wątkach wewnątrz jednego procesu zamiast osobnych procesów na sesję. Tłumaczymy mu, że InnoDB robi te same rzeczy co checkpointer, tylko w wątkach niewidocznych z poziomu `ps`.

MySQL i MariaDB — wspólny rodowód, rozjazd od 5.5

> **Pojęcie: MariaDB.** Fork MySQL, powstały w 2009 roku po przejęciu firmy MySQL AB przez Sun Microsystems, a następnie przez Oracle. Do wersji MySQL 5.5 oba projekty rozwijały się praktycznie równolegle i binarnie kompatybilnie — ta sama składnia, te same pliki `.frm`, ten sam protokół sieciowy. Od tego momentu drogi zaczęły się rozjeżdżać: MariaDB rozwija własny optymalizator zapytań, własne silniki (Aria, ColumnStore), własne rozszerzenia SQL (sekwencje jako osobne obiekty, `RETURNING` w `INSERT`/`UPDATE`/`DELETE` już w wersji 10.5, gdy MySQL dogonił to dopiero w 8.0.21 tylko dla `INSERT`), a Oracle rozwija MySQL w stronę funkcji analitycznych, JSON-a jako typu natywnego i klastra Group Replication.

Dziś oba silniki czyta się tym samym klientem `mysql`, obie rozumieją ten sam rdzeń SQL i obie polegają na InnoDB jako domyślnym silniku składowania — ale nie są już tożsame. Numeracja wersji też się rozjechała: MariaDB 10.x odpowiada w przybliżeniu MySQL 5.7, a od MariaDB 10.2 pojawiają się funkcje, których MySQL 5.7 nie ma wcale (okna analityczne, CTE), więc numer wersji między silnikami nie da się porównywać wprost. Migracja z jednego na drugi w prostych przypadkach (proste tabele InnoDB, standardowy SQL) przechodzi bez zmian, ale każda funkcja specyficzna dla jednego silnika — `JSON_TABLE` w MySQL 8, sekwencje w MariaDB — wymaga sprawdzenia z osobna.

> **Pojęcie: replikacja.** Mechanizm kopiowania zmian z jednego serwera (source) na drugi (replika) na podstawie logu binarnego, tak żeby replika trzymała u siebie bieżącą kopię tych samych danych, zwykle z niewielkim opóźnieniem. MySQL i MariaDB rozwijają dziś niekompatybilne ze sobą implementacje GTID, mimo wspólnego rodowodu. Szerzej: Baza danych, silnik, instancja i schemat.

Cecha	MySQL	MariaDB
Właściciel	Oracle Corporation	MariaDB Foundation / MariaDB plc
Domyślny silnik	InnoDB	InnoDB

Cecha	MySQL	MariaDB
Replikacja logiczna	GTID (od 5.6)	GTID (własna implementacja, niekompatybilna z MySQL)
Silniki kolumnowe	brak natywnego	ColumnStore
Szyfrowanie danych w spoczynku	Enterprise (płatne) / keyring w Community od 5.7	wbudowane we wszystkich edycjach
Licencja	GPL (Community) / komercyjna (Enterprise)	GPL

Silniki składowania — dlaczego dziś to zawsze InnoDB

> **Pojęcie: silnik składowania.** Warstwa odpowiedzialna za to, jak dane fizycznie leżą na dysku, jak są indeksowane i czy transakcje w ogóle istnieją — MySQL i MariaDB oddzielają tę warstwę od reszty serwera (parsera SQL, optymalizatora, warstwy sieciowej), dzięki czemu jedna instalacja potrafi trzymać różne tabele na różnych silnikach jednocześnie. Wybór silnika deklarujemy przy tworzeniu tabeli:

```
CREATE TABLE NazwaTabeli (id INT PRIMARY KEY) ENGINE=InnoDB;
```

MyISAM był domyślnym silnikiem do MySQL 5.5 włącznie. Nie zna transakcji, nie zna kluczy obcych, blokuje całą tabelę przy zapisie (nie wiersz), a po nieczystym zamknięciu potrafi zostawić tabelę uszkodzoną, wymagającą `REPAIR TABLE`. Jego jedyną realną przewagą był mniejszy narzut miejsca i szybsze `COUNT(*)` bez `WHERE`, bo licznik wierszy trzymał gotowy w metadanych — InnoDB musi ten count policzyć skanując.

InnoDB jest domyślnym silnikiem od MySQL 5.5 i jedynym, którego dziś świadomie wybieramy do nowych tabel. Ma pełne transakcje ACID, blokady na poziomie wiersza zamiast całej tabeli, klucze obce z realną walidacją, MVCC pozwalające czytającym nie blokować piszących i mechanizm `crash recovery` oparty na redo logu, opisany dalej na tej stronie. Różnica praktyczna jest brutalna: aplikacja webowa z wieloma równoległymi zapisami na MyISAM dławi się na blokadach tabelowych, na InnoDB nie.

Cecha	MyISAM	InnoDB
Transakcje	brak	pełne ACID
Blokady	poziom tabeli	poziom wiersza
Klucze obce	brak	tak

Cecha	MyISAM	InnoDB
Odporność na crash	wymaga naprawy ręcznej	automatyczny redo/undo recovery
Full-text search	tak (historycznie)	tak (od 5.6)
Zastosowanie dziś	tabele tymczasowe, systemy legacy	wszystko nowe

Pozostałe silniki (Memory, Archive, CSV, Aria w MariaDB, Federated) mają zastosowania niszowe — Memory do tabel tymczasowych trzymanych całkowicie w RAM bez trwałości, Archive do tanio kompresowanych logów tylko do zapisu. Sprawdzamy dostępne silniki poleceniem z sekcji Fiszka niżej.

Pliki i katalog danych

> **Pojęcie: datadir.** Katalog na dysku, w którym `mysqld` trzyma wszystkie dane, indeksy i metadane — domyślnie `/var/lib/mysql` na systemach z rodziny RHEL/Debian. Struktura wewnątrz zależy od wersji i ustawienia `innodb_file_per_table`, ale kilka plików wraca zawsze:

Plik / katalog	Zawartość
`ibdata1`	tablespace systemowy InnoDB — słownik danych (w starszych wersjach), bufor undo, w skrajnym przypadku też same tabele, jeśli `innodb_file_per_table=OFF`
`NazwaBazy/NazwaTabeli.ibd`	tablespace pojedynczej tabeli InnoDB, gdy `innodb_file_per_table=ON` (domyślne od 5.6.6) — dane i indeksy razem
`NazwaBazy/NazwaTabeli.frm`	definicja struktury tabeli w MySQL do wersji 5.7 i w MariaDB do dziś
`NazwaBazy/NazwaTabeli.sdi`	od MySQL 8.0 zastępuje `.frm` — słownik danych trzymany w formacie JSON osadzonym też wewnątrz `.ibd`, nie jako osobny plik na dysku
`ib_logfile0`, `ib_logfile1` (do 8.0) / `#innodb_redo/` (od 8.0.30)	redo log InnoDB
`NazwaHosta-bin.NNNNNN`	pliki binloga, opisane szerzej na stronie Logi binarne: archiwizacja i czyszczenie
`NazwaHosta.pid`	plik z PID-em działającego procesu
`NazwaBazy.opt` w każdym katalogu bazy (MariaDB, starszy MySQL)	domyślne kodowanie i collation bazy

> **Pojęcie: redo log InnoDB.** Cykliczny plik (lub zestaw plików w `#innodb_redo/`), w którym InnoDB zapisuje każdą zmianę strony danych, zanim potwierdzi transakcję klientowi — to samo zjawisko, które w PostgreSQL nazywa się WAL-em, opisane ogólnie na stronie Log transakcyjny: WAL, redo, transaction log. Po awarii serwera InnoDB odtwarza z tego pliku wszystkie zmiany, które zdążyły trafić do logu, zanim brudne strony danych trafiły na dysk. Rozmiar redo logu ustawiamy parametrem `innodb_log_file_size`, opisanym na stronie my.cnf: konfiguracja serwera.

Schemat = baza danych

> **Pojęcie: schemat w MySQL.** Tu leży najczęstsze zaskoczenie osoby przychodzącej z PostgreSQL albo Oracle: w MySQL i MariaDB słowa `SCHEMA` i `DATABASE` są synonimami tego samego obiektu — `CREATE SCHEMA` i `CREATE DATABASE` robią dokładnie to samo. Nie ma tu drugiego poziomu grupowania wewnątrz bazy, jak schemat w PostgreSQL grupujący tabele wewnątrz jednej bazy — każda tabela w MySQL należy wprost do bazy, a pełna kwalifikacja wygląda `NazwaBazy.NazwaTabeli`, nie `NazwaBazy.NazwaSchematu.NazwaTabeli`. Fizycznie baza to po prostu katalog wewnątrz `datadir`.

Bazy systemowe

> **Pojęcie: instancja.** Działający, uruchomiony proces `mysqld` razem z zaalokowaną pamięcią i otwartym katalogiem danych, obsługujący połączenia klientów. W MySQL i MariaDB jedna instancja to zawsze jeden proces systemowy, inaczej niż w Oracle, gdzie instancja jest osobnym bytem od plików bazy. Szerzej: Baza danych, silnik, instancja i schemat.

Każda świeżo zainstalowana instancja zawiera cztery bazy, które nie są danymi aplikacji:

mysql — baza serwera. Trzyma konta użytkowników i ich uprawnienia (tabele `user`, `db`, `tables_priv`), pomocnicze tabele czasowe, dane replikacji. To tutaj lądują zmiany po `CREATE USER` i `GRANT`.

information_schema — widoki tylko do odczytu, zgodne w części ze standardem SQL, opisujące metadane wszystkich pozostałych baz: listy tabel, kolumn, kluczy, uprawnień. Zapytania do tabel `TABLES` i `COLUMNS` tej bazy wracają na stronie Rozpoznanie serwera i baz przy liczeniu rozmiarów.

performance_schema — instrumentacja wydajnościowa niskiego poziomu, włączona domyślnie od 5.6.6. Zbiera statystyki czekań, blokad i czasu wykonania na poziomie wątków i instrumentów — surowsza i bardziej szczegółowa niż `sys`, kosztem czytelności.

sys — zestaw widoków zbudowanych na `performance_schema`, prezentujący te same dane w formie czytelnej dla człowieka bez znajomości wewnętrznych instrumentów — na przykład widok `sys.innodb_lock_waits` wykorzystywany na stronie Sesje, długie zapytania i blokady.

Częste błędy

Zakładanie, że schemat to coś innego niż baza. Osoba przychodząca z PostgreSQL szuka drugiego poziomu grupowania wewnątrz bazy i próbuje pisać `NazwaBazy.NazwaSchematu.NazwaTabeli` — w MySQL taka kwalifikacja nie istnieje, `SCHEMA` i `DATABASE` to jeden i ten sam obiekt.

Pozostawienie tabel na MyISAM po migracji ze starszej wersji. Instalacja aktualizowana od MySQL 5.5 potrafi wciąż mieć tabele systemowe albo tabele aplikacji założone na MyISAM z czasów, gdy był domyślny — brak transakcyjności na takiej tabeli ujawnia się dopiero przy pierwszym równoległym zapisie albo pierwszym nieczystym wyłączeniu serwera.

Powiązane: Klient mysql i połączenie · my.cnf: konfiguracja serwera · Log transakcyjny: WAL, redo, transaction log

Źródła: dokumentacja MySQL 8.0, rozdział „The InnoDB Storage Engine" · dokumentacja MariaDB, rozdział „Storage Engines" · notatki własne

Klient mysql i połączenie

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

`mysql` jest tym samym narzędziem od dwudziestu lat — jeden binarny klient wiersza poleceń, cztery parametry połączenia i garść komend zaczynających się od `SHOW`, które robią za brakujący w tym silniku odpowiednik metakomend `psql`.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia

Fiszka

TYLKO ODCZYT

Wylistowanie wszystkich baz danych: SHOW DATABASES

```
SHOW DATABASES;
```

Zwracamy listę baz danych widocznych dla zalogowanego użytkownika w całej instancji.

Przykład:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
```

```
| produkcja          |
| sys                |
+--------------------+
5 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

Przełączenie się na inną bazę: USE

```
USE NazwaBazy;
```

Ustawiamy bazę domyślną dla bieżącej sesji — kolejne zapytania bez kwalifikacji `NazwaBazy.NazwaTabeli` odnoszą się od teraz do tej bazy.

Przykład:

```
Database changed
```

TYLKO ODCZYT

Wylistowanie tabel bieżącej bazy: SHOW TABLES

```
SHOW TABLES;
```

Zwracamy tabele i widoki z bazy ustawionej wcześniej poleceniem `USE`.

Przykład:

```
+--------------------+
| Tables_in_produkcja |
+--------------------+
| klienci            |
| zamowienia         |
+--------------------+
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

Wyświetlenie struktury tabeli: DESCRIBE

```
DESCRIBE NazwaTabeli;
```

Zwracamy kolumny tabeli wraz z typem, informacją o `NULL`, kluczach i wartości domyślnej — odpowiednik `\d` w `psql`, tyle że bez indeksów i triggerów, które MySQL pokazuje osobnym poleceniem.

Przykład:

```
+------------+---------+------+-----+---------+----------------+
| Field      | Type    | Null | Key | Default | Extra          |
+------------+---------+------+-----+---------+----------------+
| id         | int(11) | NO   | PRI | NULL    | auto_increment |
| klient_id  | int(11) | NO   | MUL | NULL    |                |
| data       | date    | NO   |     | NULL    |                |
| wartosc    | decimal(10,2) | YES | | NULL  |                |
+------------+---------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

ZMIENIA SYSTEM

Wyświetlenie pełnej definicji obiektu: SHOW CREATE TABLE

```
SHOW CREATE TABLE NazwaTabeli;
```

Zwracamy komendę `CREATE TABLE`, którą trzeba by wykonać, żeby odtworzyć tabelę od zera, łącznie z silnikiem, kodowaniem i pełną definicją indeksów oraz kluczy obcych — dokładniejsze niż `DESCRIBE`.

Przykład:

```
*************************** 1. row ***************************
       Table: zamowienia
Create Table: CREATE TABLE `zamowienia` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `klient_id` int(11) NOT NULL,
  `data` date NOT NULL,
```

```
  `wartosc` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_klient_id` (`klient_id`),
  CONSTRAINT `zamowienia_ibfk_1` FOREIGN KEY (`klient_id`) REFERENCES `klienci` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

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

□□ TYLKO ODCZYT

Sprawdzenie stanu połączenia i serwera: STATUS

```
STATUS;
```

Zwracamy jednym poleceniem podsumowanie bieżącej sesji — wersję serwera i klienta, sposób połączenia (TCP czy gniazdo), kodowanie, uptime i podstawowe liczniki, bez konieczności odpytywania osobno kilku widoków.

Przykład:

```
--------------
mysql  Ver 15.1 Distrib 10.11.6-MariaDB

Connection id:          482
Current database:       produkcja
Current user:           nazwa_usera@localhost
SSL:                    Not in use
Using delimiter:        ;
Server version:         10.11.6-MariaDB
Protocol version:       10
Connection:              Localhost via UNIX socket
Server characterset:     utf8mb4
Uptime:                  4 days 2 hours 18 min 5 sec

Threads: 12  Questions: 384921  Slow queries: 3  Opens: 210  Flush tables: 1  Open
tables: 64  Queries per second avg: 1.114
--------------
```

□□ **TCP** = Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia

TYLKO ODCZYT

Wykonanie skryptu SQL z pliku: SOURCE

```
SOURCE SciezkaDoPliku;
```

Wczytujemy i wykonujemy zawartość wskazanego pliku wewnątrz bieżącej sesji, tak jakby jego treść wpisano ręcznie — standardowy sposób odtworzenia zrzutu z `mysqldump`, opisanego na stronie mysqldump i backup logiczny.

Przykład:

```
Query OK, 0 rows affected (0.02 sec)
Query OK, 48213 rows affected (1.84 sec)
Records: 48213  Duplicates: 0  Warnings: 0
```

SQL = Structured Query Language: język zapytań do relacyjnych baz danych · **source** = wykonaj plik w bieżącej powłoce zamiast w nowej, zmienne z niego zostają dostępne dalej

TYLKO ODCZYT

Terminator alternatywny dla wyniku pionowego: \G

```
SELECT * FROM NazwaTabeli WHERE id = 1\G
```

Kończymy polecenie znakiem `\G` zamiast średnika, żeby wynik wypisał się pionowo, jedna kolumna pod drugą zamiast szerokiej tabeli — odpowiednik `\x` w `psql`, tyle że tu decydujemy o tym przy każdym poleceniu z osobna, a nie przełącznikiem trybu na całą sesję.

Przykład:

```
*************************** 1. row ***************************
       id: 1
klient_id: 42
     data: 2026-01-15
  wartosc: 1250.00
1 row in set (0.00 sec)
```

TYLKO ODCZYT

Ustawienie pagera dla długich wyników: pager

```
pager less -SFX
```

Przekierowujemy długie wyniki przez zewnętrzny pager zamiast wypisywać je od razu na cały ekran — flaga `-S` w `less` wyłącza zawijanie wierszy, przydatne przy szerokich tabelach, które inaczej łamią się nieczytelnie. Wyłączamy pager poleceniem `nopager`.

Przykład:

```
PAGER set to 'less -SFX'
```

Z życia wzięte

Z ŻYCIA WZIĘTE

Sklep B2B na my01 przestaje odpowiadać dla jednego z klientów hurtowych i Tomek testuje port z serwera aplikacji poleceniem `mysql -h localhost`. Połączenie przechodzi, ale to nic nie mówi o dostępności portu 3306 z zewnątrz, bo `-h localhost` w MySQL nadal używa gniazda uniksowego. Dopiero `mysql -h 127.0.0.1` pokazuje, że port faktycznie jest zablokowany przez firewalld na my01.

Parametry połączenia

Połączenie z instancją opisują cztery flagi, wszystkie opcjonalne — pominięte przyjmują wartości domyślne (host lokalny przez gniazdo uniksowe, port `3306`, użytkownik systemowy bez hasła):

```
mysql -h NazwaSerwera -P NumerPortu -u NazwaUsera -p
```

Flaga `-p` bez podanej wartości w tym samym słowie wymusza interaktywne pytanie o hasło — `-pHasloUzytkownika` sklejone bez spacji podaje hasło wprost, co jest wygodne w skryptach, ale zostawia hasło widoczne w liście procesów (`ps aux`) dla każdego użytkownika systemu, dopóki proces trwa. Przykład połączenia do zdalnej instancji z wykonaniem jednego polecenia bez wchodzenia w tryb interaktywny:

```
mysql -h my01 -P 3306 -u nazwa_usera -p -e "SHOW DATABASES;"
```

Flaga `-e` (execute) wykonuje podane polecenie i kończy sesję od razu — bez niej `mysql` zostaje w trybie interaktywnym, czekając na kolejne polecenia z klawiatury.

Plik .my.cnf i sekcja [client]

> **Pojęcie: .my.cnf.** Plik w katalogu domowym użytkownika systemowego, czytany automatycznie przez klienta `mysql` oraz przez `mysqldump` i pozostałe narzędzia CLI silnika przy każdym starcie — dzięki niemu nie powtarzamy flag `-h` / `-u` / `-p` w każdym wywołaniu. Sekcja `[client]` obowiązuje wszystkie narzędzia CLI naraz, sekcja `[mysql]` tylko sam klient interaktywny:

```
[client]
host=my01
port=3306
user=nazwa_usera
password=HasloUzytkownika

[mysql]
pager=less -SFX
```

Trzymanie hasła jawnym tekstem w `.my.cnf` wymaga uprawnień `0600` na pliku — z szerszymi uprawnieniami niektóre dystrybucje klienta ostrzegają albo odmawiają odczytu. Bezpieczniejszą alternatywą jest `mysql_config_editor` opisany niżej, który hasła jawnym tekstem w ogóle nie trzyma.

mysql_config_editor — hasło poza plikiem tekstowym

> **Pojęcie: login-path.** Zamiast trzymać hasło jawnym tekstem w `.my.cnf`, `mysql_config_editor` zapisuje zaszyfrowane (nie zahaszowane — odwracalnie zaszyfrowane, więc to ochrona przed przypadkowym podejrzeniem pliku, nie przed kimś z dostępem do konta systemowego) dane logowania do pliku `.mylogin.cnf` pod nazwaną ścieżką logowania:

```
mysql_config_editor set --login-path=NazwaSciezkiLogowania --host=NazwaSerwera --
user=NazwaUsera --password
```

Po ustawieniu łączymy się, podając samą nazwę ścieżki zamiast kompletu flag:

```
mysql --login-path=NazwaSciezkiLogowania
```

Gniazdo uniksowe kontra TCP

> **Pojęcie: gniazdo uniksowe.** Plik specjalny w systemie plików (domyślnie `/var/lib/mysql/mysql.sock` albo `/run/mysqld/mysqld.sock`, zależnie od dystrybucji), przez który proces lokalny łączy się z `mysqld` bez przechodzenia przez stos sieciowy TCP. Podanie `-h localhost` w MySQL i MariaDB, wbrew intuicji, nadal wymusza gniazdo uniksowe — dopiero `-h 127.0.0.1` albo `-h my01` wymusza połączenie przez TCP, nawet jeśli serwer siedzi na tym samym hoście. To zachowanie różni się od PostgreSQL, gdzie sama obecność `-h` (dowolna wartość) przełącza na TCP.

Różnica ma znaczenie praktyczne przy uprawnieniach: konto `'NazwaUsera'@'localhost'` odpowiada wyłącznie na połączenia przez gniazdo (albo `127.0.0.1` w niektórych konfiguracjach uwierzytelniania), a konto `'NazwaUsera'@'%'` odpowiada na połączenia z dowolnego hosta przez TCP — dwa różne konta opisane szerzej na stronie Użytkownicy i uprawnienia.

Częste błędy

Mylenie -h localhost z rzeczywistym TCP. Podanie `-h localhost` w MySQL i MariaDB nadal łączy przez gniazdo uniksowe, nie przez TCP — kto testuje w ten sposób dostępność portu 3306 z zewnątrz, testuje coś innego, niż mu się wydaje. Do wymuszenia TCP lokalnie potrzeba `-h 127.0.0.1`.

Hasło w -pHasloUzytkownika w skrypcie widocznym dla innych userów. Argument wywołania procesu jest widoczny w `ps aux` dla każdego zalogowanego użytkownika, dopóki polecenie trwa — `mysql_config_editor` albo `.my.cnf` z uprawnieniami `0600` są tu bezpieczniejszym rozwiązaniem.

Powiązane: Architektura MySQL i MariaDB: silniki składowania, pliki, procesy · my.cnf: konfiguracja serwera · Użytkownicy i uprawnienia

Źródła: dokumentacja MySQL 8.0, rozdział „mysql — The MySQL Command-Line Client" · `man 1 mysql`

my.cnf: konfiguracja serwera

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

Plik `my.cnf` jest jedynym miejscem, w którym zapisane ustawienia przeżywają restart serwera — wszystko, co zmienimy tylko poleceniem `SET GLOBAL`, po restarcie wraca do wartości z pliku.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
B2B	business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
RAM	Random Access Memory: pamięć operacyjna
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia

Fiszka

TYLKO ODCZYT

Odczytanie wartości pojedynczego parametru: SHOW VARIABLES LIKE

```
SHOW VARIABLES LIKE 'max_connections';
```

Sprawdzamy bieżącą, obowiązującą wartość parametru bez przeglądania całego pliku konfiguracyjnego.

Przykład:

```
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 151   |
+-----------------+-------+
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

ZMIENIA SYSTEM

Zmiana parametru dla całej instancji w locie: SET GLOBAL

```
SET GLOBAL max_connections = 500;
```

Podnosimy limit połączeń natychmiast, bez restartu serwera — zmiana obowiązuje tylko do najbliższego restartu, jeśli nie zapiszemy jej też w `my.cnf`.

Przykład:

```
Query OK, 0 rows affected (0.00 sec)
```

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

ZMIENIA SYSTEM

Trwała zmiana parametru bez edycji pliku (MySQL 8.0): SET PERSIST

```
SET PERSIST innodb_flush_log_at_trx_commit = 1;
```

Zmieniamy parametr w locie i jednocześnie zapisujemy go tak, żeby przetrwał restart serwera, bez ręcznej edycji `my.cnf`.

Przykład:

```
Query OK, 0 rows affected (0.01 sec)
```

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

Sprawdzenie, które parametry różnią się od wartości domyślnej: performance_schema.variables_info

```
SELECT VARIABLE_NAME, VARIABLE_SOURCE, VARIABLE_PATH
FROM performance_schema.variables_info
WHERE VARIABLE_SOURCE != 'COMPILED';
```

Sprawdzamy, które parametry zostały jawnie ustawione — w pliku konfiguracyjnym, wierszu poleceń albo poleceniem `SET PERSIST` — zamiast pozostać na wartości wkompilowanej w binarkę, przydatne przy audycie konfiguracji przejętego serwera.

Przykład:

```
+--------------------------+---------------+-----------------------+
| VARIABLE_NAME            | VARIABLE_SOURCE | VARIABLE_PATH         |
+--------------------------+---------------+-----------------------+
| innodb_buffer_pool_size  | CONFIG          | /etc/mysql/my.cnf     |
| max_connections          | GLOBAL          |                       |
+--------------------------+---------------+-----------------------+
2 rows in set (0.00 sec)
```

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

Z życia wzięte

Z ŻYCIA WZIĘTE

Sklep B2B na my01 zwalnia zauważalnie w listopadzie, kiedy ruch zamówień hurtowych rośnie przed świętami. Sprawdzamy `innodb_buffer_pool_size` i widzimy wartość ustawioną jeszcze przy pierwszej instalacji, dużo mniejszą niż dzisiejszy rozmiar bazy `sklep_b2b`. Podnosimy bufor do siedemdziesięciu procent dostępnego RAM-u serwera i zapisujemy zmianę przez `SET PERSIST`, żeby przetrwała najbliższy restart.

Gdzie leży plik i w jakiej kolejności jest czytany

> **Pojęcie: my.cnf.** Plik tekstowy w formacie INI, czytany przez `mysqld` przy starcie — na Linuksie zwykle `my.ini` na Windows. Serwer nie czyta jednego pliku, tylko sekwencję lokalizacji po kolei, a każda kolejna nadpisuje wartości ustawione we wcześniejszej. Kolejność (typowa dla dystrybucji opartych na RHEL/Debian, sprawdzana poleceniem `mysqld --verbose --help` na końcowej liście `Default options`):

```
/etc/my.cnf
/etc/mysql/my.cnf
/etc/mysql/mysql.conf.d/*.cnf   (albo mariadb.conf.d na MariaDB)
~/.my.cnf
```

Sprawdzamy dokładną, obowiązującą na danym hoście kolejność poleceniem:

```
mysqld --verbose --help | grep -A 5 "Default options"
```

Rozbicie na `/etc/mysql/mysql.conf.d/*.cnf` zamiast jednego pliku jest konwencją pakietów dystrybucyjnych — pozwala trzymać ustawienia sieciowe w jednym pliku, a limity zasobów w drugim, bez edytowania wspólnego monolitu.

Sekcje pliku

Plik dzieli się na sekcje w nawiasach kwadratowych, z których każda dotyczy innego programu czytającego ten sam plik:

```
[mysqld]
bind-address = 0.0.0.0
port = 3306

[client]
port = 3306

[mysqldump]
quick
```

```
max_allowed_packet = 256M
```

Sekcja `[mysqld]` konfiguruje sam serwer i jest tą, którą edytujemy najczęściej. Sekcja `[client]` obowiązuje klienta `mysql` i pozostałe narzędzia CLI, opisana też na stronie Klient mysql i połączenie. Osobne sekcje (`[mysqldump]`, `[mysqld_safe]`) pozwalają nadpisać domyślne parametry konkretnego narzędzia bez wpływu na resztę.

Najważniejsze parametry

bind-address — adres IP, na którym `mysqld` nasłuchuje połączeń TCP. Wartość `127.0.0.1` (domyślna w wielu dystrybucjach ze względów bezpieczeństwa) ogranicza dostęp do połączeń lokalnych, `0.0.0.0` otwiera nasłuch na wszystkich interfejsach sieciowych hosta.

port — numer portu TCP, domyślnie `3306`.

datadir — katalog danych opisany szerzej na stronie Architektura MySQL i MariaDB: silniki składowania, pliki, procesy, domyślnie `/var/lib/mysql`.

max_connections — maksymalna liczba równoczesnych połączeń klienckich. Przekroczenie limitu kończy się błędem `Too many connections` dla nowych połączeń, podczas gdy istniejące działają dalej — jedno dodatkowe połączenie zawsze zostaje zarezerwowane dla konta z uprawnieniem `SUPER`, żeby dało się zalogować i zdiagnozować problem nawet przy wyczerpanym limicie.

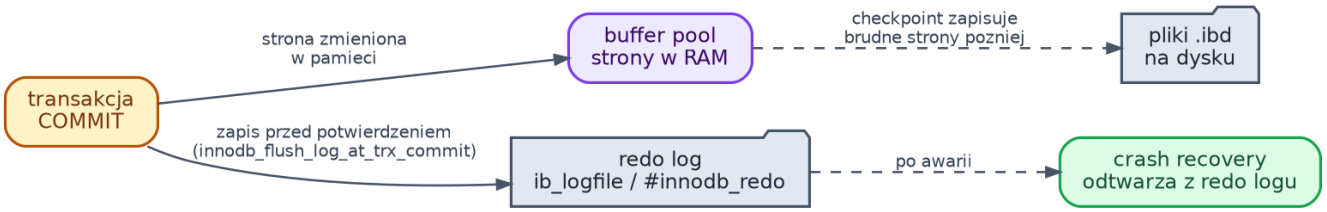
innodb_buffer_pool_size — rozmiar bufora w pamięci RAM, w którym InnoDB trzyma strony danych i indeksów odczytane z dysku. Najważniejszy pojedynczy parametr wydajnościowy całej instalacji — na serwerze dedykowanym pod bazę danych zaleca się 60–75% dostępnego RAM-u. Zbyt mały bufor oznacza, że InnoDB ciągle czyta te same strony z dysku zamiast z pamięci.

innodb_log_file_size (do 8.0) / innodb_redo_log_capacity (od 8.0.30) — rozmiar redo logu opisanego na stronie Architektura MySQL i MariaDB: silniki składowania, pliki, procesy. Większy redo log pozwala InnoDB opóźniać zapis brudnych stron na dysk i zbierać więcej zmian między checkpointami, kosztem dłuższego czasu odtwarzania po awarii.

innodb_flush_log_at_trx_commit — jak agresywnie InnoDB zrzuca redo log na dysk przy każdym `COMMIT`. Trzy wartości, kompromis między trwałością a wydajnością:

Wartość	Zachowanie	Ryzyko przy awarii
1 (domyślna)	zapis i `fsync` na dysk przy każdym commicie	brak utraty potwierdzonych transakcji
2	zapis do cache systemu operacyjnego przy każdym commicie, `fsync` raz na sekundę	utrata do 1 sekundy transakcji przy awarii systemu operacyjnego, nie samego MySQL

Wartość	Zachowanie	Ryzyko przy awarii
0	zapis i `fsync` raz na sekundę	utrata do 1 sekundy transakcji przy awarii samego procesu MySQL



COMMIT zapisuje strone w buffer poolu i w redo logu, checkpoint zrzuca brudne strony do plikow .ibd pozniej, redo log sluzy do crash recovery (kliknięcie otwiera pełny rozmiar)

log_bin — włącza binary log opisany szerzej na stronie Logi binarne: archiwizacja i czyszczenie, niezbędny do replikacji i do odtwarzania do punktu w czasie.

expire_logs_days (do 8.0) / binlog_expire_logs_seconds (od 8.0) — po ilu dniach (sekundach) serwer sam usuwa stare pliki binloga. Parametr `expire_logs_days` jest przestarzały od MySQL 8.0 i usunięty w nowszych wersjach na rzecz jednostki w sekundach.

slow_query_log — włącza zapisywanie zapytań przekraczających próg `long_query_time` do osobnego logu, opisane na stronie Sesje, długie zapytania i blokady.

sql_mode — zestaw flag zmieniających ścisłość walidacji SQL po stronie serwera. Dwie flagi mają praktyczne znaczenie przy migracji ze starszych wersji:

> **Pojęcie: ONLY_FULL_GROUP_BY.** Wymusza, żeby każda kolumna w `SELECT` przy użyciu `GROUP BY` była albo w klauzuli `GROUP BY`, albo wewnątrz funkcji agregującej — bez tej flagi MySQL do wersji 5.7 (i nadal, jeśli flagę świadomie wyłączymy) pozwalał zwrócić dowolną, niezdefiniowaną wartość kolumny spoza `GROUP BY`, co dawało niedeterministyczne wyniki. Domyślnie włączona od MySQL 5.7.5.

> **Pojęcie: STRICT_TRANS_TABLES.** Zamienia ciche obcinanie i konwersję nieprawidłowych wartości (na przykład wstawienie tekstu do kolumny liczbowej) na twardy błąd zamiast ostrzeżenia. Bez tej flagi `INSERT` z wartością zbyt długą dla kolumny `VARCHAR` po cichu ją obcina; z flagą kończy się błędem `Data too long for column`. Domyślnie włączona od MySQL 5.7.

Odczyt i zmiana parametru w locie

> **Pojęcie: SHOW VARIABLES.** Polecenie zwracające aktualną wartość parametrów działającego serwera — niekoniecznie tę zapisaną w pliku `my.cnf`, tylko tę, która faktycznie obowiązuje w tej chwili, po uwzględnieniu ewentualnych zmian wykonanych `SET GLOBAL` od czasu ostatniego startu.

```
SHOW VARIABLES LIKE 'NazwaParametru';
```

> **Pojęcie: SET GLOBAL.** Zmienia wartość parametru dla całej instancji, dla wszystkich przyszłych połączeń, ale wyłącznie w pamięci działającego procesu — restart serwera przywraca wartość z pliku `my.cnf`, jeśli tam ją zapisano, albo wartość domyślną wkompilowaną w binarkę, jeśli nie.

```
SET GLOBAL NazwaParametru = WartoscParametru;
```

> **Pojęcie: SET PERSIST.** Nowość MySQL 8.0 (brak w MariaDB) — działa jak `SET GLOBAL`, ale dodatkowo zapisuje wartość do pliku `mysqld-auto.cnf` wewnątrz `datadir`, skąd serwer odczytuje ją automatycznie przy każdym kolejnym starcie, bez ręcznej edycji `my.cnf`.

```
SET PERSIST NazwaParametru = WartoscParametru;
```

Nie każdy parametr da się zmienić w locie — kolumna `Dynamic` w tabeli `information_schema.SYSTEM_VARIABLES` (albo `performance_schema.variables_info` na nowszych wersjach) mówi, czy dany parametr jest `dynamic` czy tylko do odczytu w trakcie działania. Parametry wymagające restartu to typowo te wpływające na strukturę pamięci alokowaną raz przy starcie: `innodb_buffer_pool_size` w starszych wersjach (od 5.7.5 można go zmieniać w locie, ale ze skutkami ubocznymi na wydajność w trakcie zmiany), `innodb_log_file_size`, `port`, `datadir` — te ostatnie dwa nie zmienią się nigdy bez restartu, bo dotyczą samego procesu nasłuchującego i otwartego katalogu danych.

Częste błędy

Zmiana SET GLOBAL bez zapisania jej też w pliku. Parametr zmieniony wyłącznie w locie wraca do poprzedniej wartości po najbliższym restarcie serwera — który przy aktualizacji pakietu albo awarii zasilania przychodzi bez zapowiedzi. Trwałe zmiany na MySQL 8.0 wygodniej robić od razu przez `SET PERSIST`.

Ustawienie innodb_buffer_pool_size większego niż dostępny RAM. Serwer wystartuje, ale system zacznie swapować, gdy bufor rzeczywiście zapełni się danymi — objaw to gwałtowny spadek wydajności bez widocznego błędu w logu MySQL, bo problem leży w systemie operacyjnym, nie w samym silniku.

Powiązane: Architektura MySQL i MariaDB: silniki składowania, pliki, procesy · Klient mysql i połączenie · Logi binarne: archiwizacja i czyszczenie

Źródła: dokumentacja MySQL 8.0, rozdział „Server System Variables" i „The Server Configuration File" · dokumentacja MariaDB, rozdział „Server System Variables"