# 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.

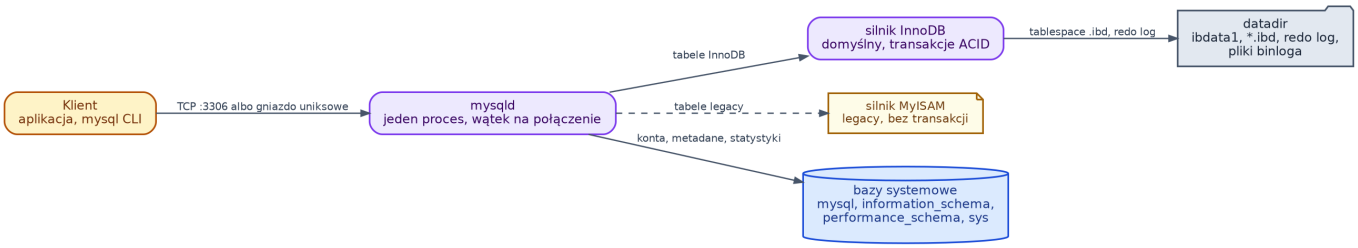
Przykład:

```
+-----------------+
| 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

---

Revision #42
Created 2026-09-25 00:00:28 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:15 UTC by Tomasz Konieczny