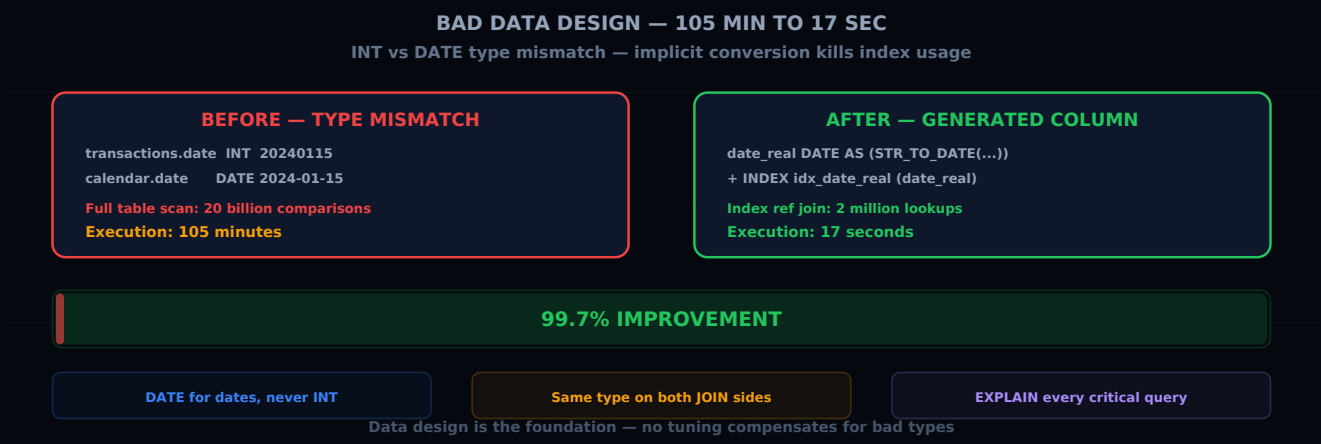
Zły projekt danych prowadzi do złej wydajności: ze 105 minut do 17 sekund

Sylvain ARBAUDIE · July 23, 2025

MARIADB PERFORMANCE OPTIMIZATION DATA-DESIGN



Objaw: 105 minut na jedno zapytanie

Klient dzwoni w trybie pilnym. Ich nocny batch zasilający raporty dzienne trwa coraz dłużej. To, co rok temu zajmowało 10 minut, teraz wymaga **105 minut**. Wolumen danych rzeczywiście wzrósł, ale nie na tyle, by uzasadnić dziesięciokrotne zwiększenie czasu wykonania.

Problematyczne zapytanie to klasyczny `JOIN` między tabelą transakcji a tabelą kalendarza:

```
SELECT
    t.transaction_id,
    t.amount,
    t.transaction_date,
    c.fiscal_year,
    c.fiscal_quarter
FROM transactions t
JOIN calendar c ON t.transaction_date = c.calendar_date
WHERE t.created_at >= '2024-01-01';
```

Nic nadzwyczajnego na pierwszy rzut oka. Dwie tabele, złączenie na dacie, filtr czasowy. A jednak 105 minut.

Diagnoza: niezgodność typów

Analiza planu wykonania (`EXPLAIN`) ujawnia pełne skanowanie tabeli `calendar` . Dziwne dla złączenia na czymś, co powinno być kluczem głównym.

Po zbadaniu struktur tabel problem staje się oczywisty:

```
-- Tabela transactions
CREATE TABLE transactions (
    transaction_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    amount DECIMAL(10,2),
    transaction_date INT NOT NULL,  -- ← przechowywane jako YYYYMMDD
    created_at DATETIME
);

-- Tabela calendar
CREATE TABLE calendar (
    calendar_date DATE NOT NULL PRIMARY KEY,
    fiscal_year SMALLINT,
    fiscal_quarter TINYINT
);
```

Kolumna `transaction_date` w tabeli `transactions` to `INT` przechowujący datę w formacie `YYYYMMDD` (na przykład `20240115` dla 15 stycznia 2024). Kolumna `calendar_date` w tabeli `calendar` to prawdziwy `DATE` .

Gdy MariaDB / MySQL wykonuje `JOIN` , musi porównać `INT` z `DATE` . Dla każdego wiersza `transactions` silnik niejawnie konwertuje `DATE` na `INT` (lub odwrotnie) dla każdego wiersza `calendar` . Ta niejawna konwersja sprawia, że indeks na `calendar_date` staje się bezużyteczny. Rezultat: pełne skanowanie tabeli `calendar` dla każdego wiersza `transactions` .

Przy 2 milionach transakcji i 10 000 wierszach w `calendar` daje to **20 miliardów porównań** z konwersją typów.

Dlaczego nie zmienić po prostu typu?

Oczywistą odpowiedzią byłoby przekonwertowanie kolumny `transaction_date` z `INT` na `DATE` . Ale w realiach systemów produkcyjnych:

- Tabela waży 15 GB. `ALTER TABLE` trwałby godzinami i zablokowałby tabelę.

- 47 procedur składowanych i 12 widoków odwołuje się do `transaction_date` jako `INT`.
- Aplikacja PHP używa porównań arytmetycznych na tej kolumnie (`WHERE transaction_date > 20240101`).
- Batch ładujący ETL wysyła daty w formacie `INT` z systemu legacy.

Zmiana typu to właściwe rozwiązanie długoterminowe, ale nie natychmiastowe rozwiązanie, którego klient potrzebuje tego wieczoru.

Rozwiązanie: wirtualna kolumna generowana

MariaDB / MySQL obsługuje kolumny wirtualne (lub generated columns). To kolumny obliczane dynamicznie na podstawie innych kolumn, bez fizycznego przechowywania (VIRTUAL) lub z przechowywaniem (STORED).

```
ALTER TABLE transactions
ADD COLUMN transaction_date_real DATE AS (
    STR_TO_DATE(CAST(transaction_date AS CHAR(8)), '%Y%m%d')
) VIRTUAL;
```

Ta kolumna konwertuje `INT` na `DATE` w locie. Ale sama kolumna wirtualna nie rozwiązuje problemu wydajności. Potrzebny jest indeks:

```
ALTER TABLE transactions
ADD COLUMN transaction_date_real DATE AS (
    STR_TO_DATE(CAST(transaction_date AS CHAR(8)), '%Y%m%d')
) STORED,
ADD INDEX idx_transaction_date_real (transaction_date_real);
```

Używamy `STORED` zamiast `VIRTUAL`, aby móc utworzyć indeks. Kolumna jest fizycznie przechowywana, a indeks jest automatycznie aktualizowany przy wstawieniach i aktualizacjach.

Poprawione zapytanie

```
SELECT
    t.transaction_id,
    t.amount,
    t.transaction_date,
    c.fiscal_year,
```

```
    c.fiscal_quarter
FROM transactions t
JOIN calendar c ON t.transaction_date_real = c.calendar_date
WHERE t.created_at >= '2024-01-01';
```

`JOIN` porównuje teraz `DATE` z `DATE`. Indeks jest użyteczny. Plan wykonania pokazuje `ref` zamiast pełnego skanu.

Wynik: 17 sekund

Metryka	Przed	Po	Poprawa
Czas wykonania	105 min	17 s	99,7%
Przeanalizowane wiersze	~20 mld	~2 mln	99,99%
Typ skanu	Pełne skanowanie	Index ref	—

Ze 105 minut do 17 sekund. Poprawa o **99,7%** bez zmiany istniejącego schematu, bez modyfikacji aplikacji, bez dotykania procedur składowanych.

Dlaczego niejawne konwersje są pułapką

Ten przypadek ilustruje fundamentalny problem: niejawne konwersje typów w złączeniach i klauzulach WHERE to ciche zabójcy wydajności.

MariaDB / MySQL wykonuje niejawne konwersje w wielu przypadkach:

- `INT` porównywany z `VARCHAR`: INT jest konwertowany na VARCHAR
- `INT` porównywany z `DATE`: DATE jest konwertowany na liczbę
- `VARCHAR(utf8)` porównywany z `VARCHAR(latin1)`: konwersja zestawu znaków
- `DECIMAL` porównywany z `FLOAT`: konwersja na zmiennoprzecinkowy

W każdym przypadku konwersja sprawia, że indeks staje się bezużyteczny, ponieważ silnik nie może przeszukiwać bezpośrednio indeksu B-tree, jeśli wartość musi być najpierw przekształcona.

Lekcja: projektowanie danych to fundament

Wydajność bazy danych rozstrzyga się w momencie projektowania, nie w momencie tuningu. Żaden indeks, żadna konfiguracja buffer pool, żaden sprzęt nie skompensuje złego wyboru typu danych.

Fundamentalne zasady:

1. **Data powinna być przechowywana jako `DATE` lub `DATETIME`**, nigdy jako `INT` czy `VARCHAR`.
2. **Kolumny złączeń muszą mieć ten sam typ i ten sam charset/collation.**
3. **Używaj `EXPLAIN` systematycznie**, by weryfikować, że złączenia korzystają z indeksów.
4. **Monitoruj niejawne konwersje** narzędziem `EXPLAIN ANALYZE` (MariaDB 10.1+).

Projektowanie danych nie jest efektowne. Nie jest tak ekscytujące jak tuning zmiennych systemowych czy budowanie klastra Galera. Ale to fundament. A gdy fundament jest wadliwy, wszystko inne się wali — 105 minut za razem.

Ten artykuł został pierwotnie opublikowany na Medium.