Я обыграл оптимизатор MariaDB: с 94 секунд до 55 миллисекунд

Sylvain ARBAUDIE · July 15, 2025

MARIADB OPTIMIZER PERFORMANCE SQL

Контекст: миграция MySQL 8 на MariaDB 11.4

Проект миграции с MySQL 8 на MariaDB 11.4 в целом идёт хорошо. Функциональные тесты зелёные, тесты производительности тоже — кроме одного запроса. Один единственный запрос, который выполнялся за 2 секунды на MySQL 8, теперь занимает **94 секунды** на MariaDB 11.4.

Это классическая регрессия при миграции: оптимизаторы MySQL и MariaDB существенно разошлись с момента форка. План выполнения, хорошо работавший на одном движке, может оказаться катастрофическим на другом.

Проблемный запрос

Исходный запрос использует классический паттерн LEFT JOIN для поиска записей, которые НЕ существуют в другой таблице:

```
SELECT p.product_id, p.product_name, p.category_id
FROM products p
LEFT JOIN discontinued_products dp
```

```
    ON p.product_id = dp.product_id
    AND p.category_id = dp.category_id
WHERE dp.product_id IS NULL
  AND p.status = 'active'
  AND p.created_at >= '2023-01-01';
```

Намерение понятно: найти все активные продукты, которые не фигурируют в таблице прекращённых продуктов. Это паттерн «анти-join», реализованный через LEFT JOIN + IS NULL.

Почему 94 секунды?

Анализ плана выполнения на MariaDB 11.4 выявляет проблему. Оптимизатор решает:

1. Просканировать таблицу `products` по индексу на `status` (200 000 строк)
2. Для каждой строки сканировать таблицу `discontinued_products` для проверки отсутствия соответствия

При 17 500 строках в таблице `discontinued_products` это даёт около **3,5 миллиарда сравнений**. Оптимизатор MySQL 8 выбирал другой план с hash join, значительно более эффективный для этого паттерна.

Фундаментальная проблема не в оптимизаторе MariaDB — а в самом запросе. LEFT JOIN + IS NULL для реализации анти-join — это исторический антипаттерн, появившийся в эпоху, когда в SQL не было лучших альтернатив.

Решение: CTE + EXCEPT

MariaDB поддерживает Common Table Expressions (CTE) с версии 10.2 и оператор `EXCEPT` с версии 10.3. Эти две функциональности позволяют переписать логику гораздо более явным образом:

```
WITH active_products AS (
    SELECT product_id, category_id
    FROM products
    WHERE status = 'active'
      AND created_at >= '2023-01-01'
),
```

```
still_active AS (
    SELECT product_id, category_id FROM active_products
    EXCEPT
    SELECT product_id, category_id FROM discontinued_products
)
SELECT p.product_id, p.product_name, p.category_id
FROM products p
JOIN still_active sa
    ON p.product_id = sa.product_id
    AND p.category_id = sa.category_id;
```

Почему это быстрее

1. **CTE** `active_products` материализует 200 000 активных продуктов в памяти. Один проход по таблице `products`.
2. **Оператор** `EXCEPT` выполняет теоретико-множественную операцию: берёт множество активных продуктов и вычитает из него те, что присутствуют в `discontinued_products`. Это разность множеств на основе хэша, а не построчное сравнение.
3. **Финальный JOIN** с CTE `still_active` — это простой поиск по уже отфильтрованному множеству.

Результат: 55 миллисекунд

Метрика	LEFT JOIN	CTE + EXCEPT	Улучшение
Время	94 секунды	55 мс	1 700x
Сравнения	~3,5 млрд	~217 500	99,99%
Подход	Построчный	Множественный	—

С 94 секунд до 55 миллисекунд. Коэффициент **1 700**. Не за счёт изменения параметра конфигурации. Не за счёт добавления индекса. За счёт **переосмысления логики запроса**.

Антипаттерн LEFT JOIN: почему он живучий

Паттерн LEFT JOIN + IS NULL для анти-join встречается повсюду. Он есть в учебниках, онлайн-курсах, ответах на Stack Overflow. Он живёт по нескольким причинам:

Историческая причина: до SQL:1999 не было ни `EXCEPT`, ни оптимизированного `NOT EXISTS`, ни CTE. LEFT JOIN + IS NULL был единственным переносимым вариантом.

Привычка: разработчики выучивают работающий паттерн и переиспользуют его. «Работает» — враг «оптимально».

Совместимость: LEFT JOIN работает на всех версиях всех СУБД. `EXCEPT` поддерживается только в новых версиях.

Альтернативы, которые нужно знать

Для анти-join существуют три альтернативы:

NOT EXISTS (часто лучший выбор)

```
SELECT p.product_id, p.product_name, p.category_id
FROM products p
WHERE p.status = 'active'
  AND p.created_at >= '2023-01-01'
  AND NOT EXISTS (
    SELECT 1 FROM discontinued_products dp
    WHERE dp.product_id = p.product_id
      AND dp.category_id = p.category_id
  );
```

`NOT EXISTS` обычно хорошо оптимизируется обоими движками MariaDB и MySQL. Оптимизатор может использовать обратный полусоединение (semi-join), значительно более эффективное, чем LEFT JOIN.

NOT IN (осторожно с NULL)

```
SELECT product_id, product_name, category_id
FROM products
WHERE status = 'active'
  AND created_at >= '2023-01-01'
  AND (product_id, category_id) NOT IN (
    SELECT product_id, category_id
```

```
    FROM discontinued_products
);
```

Внимание: `NOT IN` имеет коварное поведение с NULL-значениями. Если `discontinued_products.product_id` может содержать NULL, семантика NOT IN вернёт пустой результат. Всегда используйте NOT EXISTS, если возможны NULL.

EXCEPT (самый читабельный)

```
SELECT product_id, category_id FROM products
WHERE status = 'active' AND created_at >= '2023-01-01'
EXCEPT
SELECT product_id, category_id FROM discontinued_products;
```

`EXCEPT` — наиболее чистое выражение теоретико-множественной операции «А минус В». Но он возвращает только столбцы операции, а не дополнительные столбцы — отсюда использование CTE для повторного введения недостающих столбцов через JOIN.

Урок

Оптимизатор — это инструмент, а не чудо. Когда запрос медленный, первый вопрос должен быть не «какой хинт добавить?» или «какой параметр изменить?». Первый вопрос должен быть: **правильно ли мой запрос выражает моё намерение?**

LEFT JOIN + IS NULL выражает «соедини, затем отфильтруй несоответствия». `EXCEPT` напрямую выражает «вычти это множество из другого множества». Вторая формулировка ближе к бизнес-намерению, и она даёт оптимизатору гораздо лучший шанс найти эффективный план.

Побеждайте оптимизатор, думая как математик, а не как программист.

Эта статья была первоначально опубликована на Medium.