

# J'ai battu l'optimiseur MariaDB : de 94 secondes à 55 millisecondes

Sylvain ARBAUDIE · 15 juillet 2025

MARIADB OPTIMIZER PERFORMANCE SQL

BEATING THE MARIADB OPTIMIZER — 94s TO 55ms  
LEFT JOIN anti-pattern → CTE + EXCEPT — 1,700x improvement

## LEFT JOIN + IS NULL

```
SELECT ... FROM products p
LEFT JOIN discontinued dp ON ...
WHERE dp.product_id IS NULL
```

3.5 billion row-by-row comparisons

## CTE + EXCEPT

```
WITH active AS (SELECT ...)
SELECT ... FROM active
EXCEPT SELECT ... FROM discontinued
```

Hash-based set difference — 217K ops

94 SECONDS

55 MILLISECONDS

1,700x FASTER — think sets, not loops

Beat the optimizer by thinking like a mathematician, not a programmer

## Le contexte : migration MySQL 8 vers MariaDB 11.4

Un projet de migration de MySQL 8 vers MariaDB 11.4 se passe bien dans l'ensemble. Les tests fonctionnels sont verts, les tests de performance aussi — sauf une requête. Une seule requête qui passait en 2 secondes sur MySQL 8 et qui prend désormais **94 secondes** sur MariaDB 11.4.

C'est une régression classique de migration : les optimiseurs de MySQL et MariaDB ont divergé significativement depuis le fork. Un plan d'exécution qui fonctionnait bien sur l'un peut être catastrophique sur l'autre.

## La requête problématique

La requête originale utilise le pattern classique du LEFT JOIN pour trouver les enregistrements qui n'existent PAS dans une autre table :

```
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';
```

L'intention est claire : trouver tous les produits actifs qui ne figurent pas dans la table des produits discontinués. C'est un pattern "anti-join" implémenté via LEFT JOIN + IS NULL.

## Pourquoi 94 secondes ?

---

L'analyse du plan d'exécution sur MariaDB 11.4 révèle le problème. L'optimiseur choisit de :

1. Scanner la table `products` en utilisant l'index sur `status` (200 000 lignes)
2. Pour chaque ligne, faire un scan de la table `discontinued_products` pour vérifier l'absence de correspondance

Avec une table `discontinued_products` de 17 500 lignes, cela représente environ **3,5 milliards de comparaisons**. L'optimiseur MySQL 8 choisissait un plan différent avec un hash join, beaucoup plus efficace pour ce pattern.

Le problème fondamental n'est pas l'optimiseur MariaDB — c'est la requête elle-même. Le LEFT JOIN + IS NULL pour implémenter un anti-join est un anti-pattern historique qui date de l'époque où SQL ne disposait pas de meilleures alternatives.

## La solution : CTE + EXCEPT

---

MariaDB supporte les Common Table Expressions (CTE) depuis la version 10.2 et l'opérateur `EXCEPT` depuis la version 10.3. Ces deux fonctionnalités permettent de réécrire la logique de manière bien plus explicite :

```
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;

```

## Pourquoi c'est plus rapide

1. **Le CTE `active_products`** matérialise les 200 000 produits actifs en mémoire. Un seul scan de la table `products`.
2. **L'opérateur `EXCEPT`** effectue une opération ensembliste : il prend l'ensemble des produits actifs et en soustrait ceux qui apparaissent dans `discontinued_products`. C'est un hash-based set difference, pas une comparaison ligne par ligne.
3. **Le JOIN final** avec le CTE `still_active` est un simple lookup sur un ensemble déjà filtré.

## Le résultat : 55 millisecondes

Métrique	LEFT JOIN	CTE + EXCEPT	Amélioration
Temps	94 secondes	55 ms	1 700x
Comparaisons	~3,5 milliards	~217 500	99,99%
Approche	Ligne par ligne	Ensembliste	—

De 94 secondes à 55 millisecondes. Un facteur **1 700**. Pas en changeant un paramètre de configuration. Pas en ajoutant un index. En **repensant la logique de la requête**.

## Le LEFT JOIN anti-pattern : pourquoi il persiste

Le pattern LEFT JOIN + IS NULL pour l'anti-join est partout. On le trouve dans les tutoriels, les cours en ligne, les réponses Stack Overflow. Il persiste pour plusieurs raisons :

**Historique** : Avant SQL:1999, il n'existait pas de `EXCEPT`, pas de `NOT EXISTS` optimisé, pas de CTE. Le LEFT JOIN + IS NULL était la seule option portable.

**Habitude** : Les développeurs apprennent un pattern qui fonctionne et le réutilisent. "Ca marche" est l'ennemi de "c'est optimal".

**Compatibilité** : Le LEFT JOIN fonctionne sur toutes les versions de tous les SGBD. `EXCEPT` n'est supporté que par les versions récentes.

## Les alternatives à connaître

---

Pour les anti-joins, trois alternatives existent :

### NOT EXISTS (souvent le meilleur choix)

```
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` est généralement bien optimisé par les deux moteurs MariaDB et MySQL. L'optimiseur peut utiliser un semi-join inversé, beaucoup plus efficace qu'un LEFT JOIN.

### NOT IN (attention aux 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
);
```

Attention : `NOT IN` a un comportement piègeux avec les valeurs NULL. Si `discontinued_products.product_id` peut être NULL, la sémantique de NOT IN retourne un résultat vide. Toujours utiliser NOT EXISTS si des NULL sont possibles.

## EXCEPT (le plus lisible)

```
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` est l'expression la plus pure de l'opération ensembliste "A moins B". Mais il ne retourne que les colonnes de l'opération, pas les colonnes supplémentaires — d'où l'utilisation d'un CTE pour réintroduire les colonnes manquantes via un JOIN.

## La leçon

---

L'optimiseur est un outil, pas un miracle. Quand une requête est lente, la première question ne devrait pas être "quel hint puis-je ajouter ?" ou "quel paramètre dois-je changer ?". La première question devrait être : **est-ce que ma requête exprime correctement mon intention ?**

Un LEFT JOIN + IS NULL exprime "joins puis filtre les non-correspondances". Un `EXCEPT` exprime directement "soustrais cet ensemble de cet autre ensemble". La seconde formulation est plus proche de l'intention métier, et elle donne à l'optimiseur une bien meilleure chance de trouver un plan efficace.

Battez l'optimiseur en pensant comme un mathématicien, pas comme un programmeur.

---

Cet article a été initialement publié sur [Medium](#).