1. DROP INDEX Query returned successfully in 65 msec.
2. “!=”-Statement: Successfully run. Total query runtime: 251 msec. 199 rows affected.
3. NOT IN- Statement: Successfully run. Total query runtime: 76 msec. 199 rows affected.
4. WHERE NOT EXITS-Statement: Successfully run. Total query runtime: 76 msec. 199 rows affected.
5. LEFT JOIN & WHERE NULL-Statement: Successfully run. Total query runtime: 69 msec. 199 rows affected.

Für die Optimierung von Abfrageplänen bereitet PostgreSQL einen gehashten Subplan (Verwendung von HASH-Funktionen die eindeutigen HASH-Wert auf Daten von Subquery’s berechnet) vor, der den Zugriff auf Daten beschleunigt. Diese Optimierung ist nur in speziellen Fällen wirksam, wenn die Subquery nur eine geringe Anzahl an Zeilen zurückgibt..sprich es gibt wenige Verknüpfungen zwischen Zeilen der Haupttabelle und Unterabfragen.

Falls eine Subquery mehrere 1000 Zeilen returniert, könnte der Optimierungsansatz ineffizient werden. Bei größeren Ergebnismengen kommt es zu einer Materialisierung des gehashten Subplans (das Speichern der Ergebnisse in einer temporären Tabelle oder andere physische Struktur, Subquery wird nicht jedes Mal neu ausgeführt, sondern greift auf die gespeicherten Ergebnisse zu.)

Folgende Queries mit „WHERE NOT EXISTS“, und „LEFT JOIN“ liefern ähnliche bzw. leicht bessere Performances wie die ursprüngliche Lösung, die das NOT IN-Statement verwendet. Das auch nur weil der gehashte Subplan wenig Zeilen als Ergebnis ausgewählt hat.