Warunek w ON a warunek w WHERE: różnica, która zmienia wynik

> Przy INNER JOIN nie ma znaczenia, czy warunek na drugą tabelę stoi w ON czy w WHERE — przy LEFT JOIN ma znaczenie zasadnicze, bo ON filtruje wiersze przed dołączeniem, a WHERE filtruje już gotowy wynik, razem z wierszami, które LEFT JOIN dopiero co uratował przed usunięciem.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
ISO	International Organization for Standardization: organizacja, która wydaje normy techniczne używane na całym świecie

Fiszka

TYLKO ODCZYT

Wariant 1 — INNER JOIN: ON i WHERE dają ten sam wynik

```
SELECT k.nazwa, z.id AS zamowienie_id, z.data, z.status, z.wartosc
FROM klienci k
JOIN zamowienia z ON k.id = z.klient_id AND z.status = 'zrealizowane'
ORDER BY z.id;
```

Warunek na status stoi tu w `ON`, obok warunku łączącego klienta z zamówieniem. Przy INNER JOIN wynik byłby identyczny, gdyby ten sam warunek przenieść do `WHERE`, bo INNER JOIN i tak odrzuca każdy wiersz bez pełnego dopasowania — nie ma tu żadnych „uratowanych" wierszy z NULL-ami, które WHERE mogłoby po drodze wyciąć.

Przykład:

```
            nazwa             | zamowienie_id |    data    |    status    | wartosc
------------------------------+---------------+------------+--------------+---------
 Nowak i Wspólnicy Sp. z o.o. |           101 | 2024-01-15 | zrealizowane | 1250.00
 Baltic Trade Sp. z o.o.      |           107 | 2024-03-25 | zrealizowane | 3200.00
(2 rows)
```

join = połącz dwa pliki tekstowe po wspólnej kolumnie, jak złączenie tabel w bazie danych

TYLKO ODCZYT

Wariant 2 — LEFT JOIN, warunek w ON

```
SELECT k.nazwa, z.id AS zamowienie_id, z.data, z.status, z.wartosc
FROM klienci k
LEFT JOIN zamowienia z ON k.id = z.klient_id AND z.status = 'zrealizowane'
ORDER BY k.id;
```

Ten sam warunek na status, ale teraz złączenie jest zewnętrzne — `ON` decyduje tylko o tym, które zamówienia dopasować, natomiast każdy klient zostaje w wyniku niezależnie od tego, czy znalazł się jakikolwiek pasujący wiersz.

Przykład:

```
            nazwa             | zamowienie_id |    data    |    status    | wartosc
------------------------------+---------------+------------+--------------+---------
 Nowak i Wspólnicy Sp. z o.o. |           101 | 2024-01-15 | zrealizowane | 1250.00
 Nowak i Wspólnicy Sp. z o.o. |           102 | 2024-02-03 | zrealizowane |  430.50
 Kowalski Handel              |               |            |              |
 Zielona Dolina S.A.          |               |            |              |
 Baltic Trade Sp. z o.o.      |           107 | 2024-03-25 | zrealizowane | 3200.00
 Podlaski Handel              |               |            |              |
 Elektro-System Kowalczyk     |               |            |              |
(7 rows)
```

Siedem wierszy: Nowak i Wspólnicy ma oba swoje zamówienia zrealizowane, więc pojawia się dwukrotnie, z każdym z nich osobno. Każdy pozostały klient bez ani jednego zamówienia spełniającego oba warunki naraz — `k.id = z.klient_id` i `z.status = 'zrealizowane'` — dostaje jeden wiersz z pustymi kolumnami, ale sam wciąż jest na liście. Ani jeden klient nie zniknął.

◫ TYLKO ODCZYT

Wariant 3 — LEFT JOIN, ten sam warunek w WHERE

```
SELECT k.nazwa, z.id AS zamowienie_id, z.data, z.status, z.wartosc
FROM klienci k
LEFT JOIN zamowienia z ON k.id = z.klient_id
WHERE z.status = 'zrealizowane'
ORDER BY k.id;
```

Warunek na status przeniesiony z `ON` do `WHERE` — złączenie samo w sobie jest teraz zwykłym, niczym nieograniczonym LEFT JOIN po kliencie, a filtrowanie po statusie następuje dopiero na gotowym wyniku.

Przykład:

```
            nazwa             | zamowienie_id |    data    |    status    | wartosc
------------------------------+---------------+------------+--------------+---------
 Nowak i Wspólnicy Sp. z o.o. |           101 | 2024-01-15 | zrealizowane | 1250.00
 Baltic Trade Sp. z o.o.      |           107 | 2024-03-25 | zrealizowane | 3200.00
(2 rows)
```

Ten wynik jest identyczny z wariantem 1 — mimo że zapytanie deklaruje `LEFT JOIN`, wynik zachowuje się tak, jakby to był `INNER JOIN`. Kowalski Handel, Zielona Dolina, Podlaski Handel i Elektro-System Kowalczyk zniknęli, chociaż LEFT JOIN miał ich zachować. To jest najczęstsza pomyłka początkujących przy pisaniu raportów opartych na LEFT JOIN — i dokładny powód, dla którego ta strona istnieje osobno.

Po co to istnieje

Strona SELECT: kolejność zapisu a kolejność wykonania ustaliła logiczną kolejność wykonania zapytania: `FROM`, potem `WHERE`, dopiero potem `SELECT`. Złączenia mieszczą się w kroku `FROM` — a to znaczy, że `ON` działa w trakcie budowania wyniku złączenia, natomiast `WHERE` działa dopiero na gotowym wyniku tego złączenia, już po tym, jak LEFT JOIN zdążył dołożyć wiersze z NULL-ami za niedopasowanych klientów.

Ta różnica w momencie wykonania tłumaczy wszystko, co widać w trzech wariantach powyżej. `ON` filtruje kandydatów do dopasowania — decyduje, które konkretne zamówienie połączyć z którym klientem, zanim LEFT JOIN zdąży cokolwiek uratować. Klient bez pasującego zamówienia i tak zostaje w wyniku, tylko z pustymi kolumnami, bo LEFT JOIN działa niezależnie od tego, jak surowy

był warunek w `ON`. `WHERE` filtruje gotowe wiersze wynikowe — i nie wie ani nie chce wiedzieć, że wiersz z samymi NULL-ami po prawej stronie powstał właśnie po to, żeby klient bez zamówień nie zniknął. Skoro `NULL = 'zrealizowane'` nigdy nie daje prawdy, `WHERE` wyrzuca taki wiersz tak samo, jak wyrzuciłby dowolny inny niepasujący wiersz — i klient znika, mimo że LEFT JOIN dosłownie przed chwilą go zachował.

Przy INNER JOIN ta różnica nie istnieje, bo INNER JOIN i tak nigdy nie dokłada wierszy z NULL-ami za niedopasowanie — nie ma czego uratować, więc nie ma co stracić. Stąd wariant 1 i wariant 3 dają identyczny wynik: oba są, w praktyce, tym samym INNER JOIN, tylko jeden deklaruje to wprost, a drugi dochodzi do tego samego przez LEFT JOIN, którego efekt WHERE po drodze anuluje.

Jak to działa

Rozwiń: jak to działa pod spodem

Regułę da się streścić jednym zdaniem: warunek dotyczący tego, które wiersze prawej tabeli w ogóle wziąć pod uwagę przy dopasowywaniu, należy do `ON`; warunek dotyczący tego, które z gotowych wierszy wynikowych zostawić w raporcie, należy do `WHERE`. Przy INNER JOIN te dwa pytania sprowadzają się do tego samego, więc miejsce warunku nie ma znaczenia. Przy LEFT, RIGHT i FULL OUTER JOIN to dwa różne pytania, i przeniesienie warunku między nimi zmienia sens zapytania, nie tylko jego zapis.

Praktyczna wskazówka wynikająca z tej reguły: warunek na tabelę, która ma pozostać „zewnętrzna" (czyli tę, której niedopasowane wiersze mają przetrwać), niemal zawsze trzeba trzymać w `ON`. Warunek na tabelę „główną" — tę, po której stronie LEFT JOIN gwarantuje pełną listę — można bezpiecznie trzymać w `WHERE`, bo ona i tak nie ma wierszy z NULL-ami do stracenia.

```
SELECT k.nazwa, z.id, z.status
FROM klienci k
LEFT JOIN zamowienia z ON k.id = z.klient_id
WHERE k.miasto = 'Warszawa';
```

Ten warunek w `WHERE` jest bezpieczny, bo dotyczy `klienci`, tabeli głównej LEFT JOIN — nie filtruje wierszy dołączonych z `zamowienia`, więc nie zamienia niczego w INNER JOIN.

Ta sama zasada przy FULL OUTER JOIN. Zjawisko opisane na tej stronie dotyczy każdego złączenia zewnętrznego, nie tylko LEFT JOIN. Warunek na dowolną ze stron `FULL OUTER JOIN` postawiony w `WHERE` zamiast w `ON` odetnie niedopasowane wiersze z tej strony dokładnie tak samo jak przy LEFT JOIN — a skoro `FULL OUTER JOIN` z założenia ma pokazywać wiersze niedopasowane z obu stron naraz, błąd ten potrafi po cichu zamienić go w zwykły LEFT albo

nawet INNER JOIN, w zależności od tego, na którą stronę warunek trafił. Strona FULL OUTER JOIN wraca do tego mechanizmu przy uzgadnianiu dwóch list.

Częste błędy

Filtrowanie „ostatniego miesiąca" w WHERE zamiast w ON w raporcie LEFT JOIN. Raport, który ma pokazać wszystkich klientów i ich zamówienia z ostatniego miesiąca — łącznie z klientami, którzy w tym miesiącu nic nie zamówili — z warunkiem `WHERE z.data >= CURRENT_DATE - INTERVAL '30 days'` po cichu wyrzuca właśnie tych klientów bez zamówień, których raport miał pokazać. Warunek na datę zamówienia musi stać w `ON`, żeby LEFT JOIN zdążył zdecydować „brak zamówień w tym oknie" zamiast „brak zamówień w ogóle", zanim `WHERE` cokolwiek zobaczy.

Domyślne traktowanie LEFT JOIN jak INNER JOIN po latach pisania jednego stylu. Zespół, który przez lata pisał głównie INNER JOIN, przenosi ten sam odruch — „warunek zawsze do WHERE" — na LEFT JOIN bez zastanowienia, bo składniowo obie wersje są równie poprawne i żadna nie zgłasza błędu. Skutek ujawnia się dopiero jako brakujące wiersze w raporcie, nie jako awaria zapytania.

Sprawdzanie wyniku tylko na danych, gdzie różnica akurat nie wychodzi. Jeśli każdy klient w danych testowych ma choć jedno pasujące zamówienie, warianty 2 i 3 dają identyczny wynik i różnica zostaje niewykryta aż do produkcji, gdzie pojawia się pierwszy klient bez dopasowania.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — przewidzenie wyniku

Bez uruchamiania zapytania przewidujemy, ilu klientów zwróci `SELECT k.nazwa FROM klienci k LEFT JOIN zamowienia z ON k.id = z.klient_id WHERE z.wartosc > 1000 OR z.wartosc IS NULL;`, wiedząc, że klienci 3, 4 mają zamówienia powyżej 1000, klient 6 nie ma żadnego zamówienia, a pozostali mają wyłącznie zamówienia poniżej tej kwoty.

Odpowiedź

Trzech: klienci 3 i 4 (mają zamówienie spełniające `z.wartosc > 1000`) oraz klient 6 (nie ma żadnego zamówienia, więc `z.wartosc IS NULL` jest dla niego prawdą po LEFT JOIN). Dopisanie `OR z.wartosc IS NULL` do warunku w `WHERE` to sposób na to, żeby świadomie zachować klientów bez dopasowania, mimo że warunek stoi w `WHERE`, a nie w `ON` — trzeba tylko jawnie przewidzieć przypadek `NULL`.

Ćwiczenie 2 — naprawa raportu

Raport miał pokazać wszystkich klientów razem z sumą ich zamówień z bieżącego roku, łącznie z klientami bez żadnego zamówienia w tym roku (suma równa zero albo NULL). Ktoś napisał `SELECT k.nazwa, SUM(z.wartosc) AS suma FROM klienci k LEFT JOIN zamowienia z ON k.id = z.klient_id WHERE z.data >= '2024-01-01' GROUP BY k.nazwa;` i klienci bez zamówień zniknęli z wyniku. Jak to naprawić?

Odpowiedź

Przenieść warunek na datę do `ON`: `LEFT JOIN zamowienia z ON k.id = z.klient_id AND z.data >= '2024-01-01'`, zostawiając `WHERE` puste albo z warunkami dotyczącymi wyłącznie `klienci`. Dokładnie ten sam mechanizm co w wariancie 3 tej strony — warunek na tabelę, która ma zachować niedopasowane wiersze, musi stać w `ON`, inaczej `WHERE` odetnie te wiersze razem z NULL-ami, które LEFT JOIN dopiero co dla nich stworzył.

Pytania kontrolne

1. Dlaczego warianty 1 i 3 z tej strony dają identyczny wynik, mimo że jeden deklaruje INNER JOIN, a drugi LEFT JOIN?

Odpowiedź

Bo warunek w `WHERE` po LEFT JOIN odrzuca każdy wiersz z NULL-ami wynikającymi z braku dopasowania — a to są dokładnie te same wiersze, których INNER JOIN nigdy by nie utworzył. Efekt końcowy wychodzi identyczny, tylko droga do niego jest inna: jeden odrzuca niedopasowane wiersze od razu w `FROM`, drugi dokłada je w `FROM` i zaraz potem odrzuca w `WHERE`.

2. Jaka jest ogólna reguła decydująca, czy warunek na drugą tabelę w LEFT JOIN powinien stać w ON czy w WHERE?

Odpowiedź

Warunek, który ma decydować, co zostanie dopasowane, zanim LEFT JOIN zachowa niedopasowane wiersze lewej tabeli, należy do `ON`. Warunek, który ma dodatkowo przefiltrować już gotowy wynik — łącznie z ryzykiem odrzucenia wierszy z NULL-ami — należy do `WHERE`. Jeśli intencją jest zachowanie wszystkich wierszy lewej tabeli, warunek na prawą tabelę prawie zawsze powinien trafić do `ON`.

Powiązane: LEFT JOIN i RIGHT JOIN · INNER JOIN · SELECT: kolejność zapisu a kolejność wykonania · WHERE: filtrowanie wierszy

Źródła: ISO/IEC 9075-2:2016 § 7.7, § 7.12 (query specification) · dokumentacja PostgreSQL, rozdział „Joins Between Tables"

Revision #47
Created 2026-09-24 22:42:02 UTC by Tomasz Konieczny
Updated 2026-10-26 14:11:05 UTC by Tomasz Konieczny