LEFT JOIN i RIGHT JOIN

LEFT JOIN zachowuje w wyniku każdy wiersz z lewej tabeli, nawet gdy nie znalazł pary po prawej — wypełnia wtedy brakujące kolumny NULL-ami zamiast wyrzucić cały wiersz, jak zrobiłby to INNER JOIN.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
ANSI	American National Standards Institute: tu chodzi o kody sterujące kolorem i pozycją kursora w terminalu
ISO	International Organization for Standardization: organizacja, która wydaje normy techniczne używane na całym świecie
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

TYLKO ODCZYT

Zachowanie wszystkich wierszy lewej tabeli: LEFT JOIN

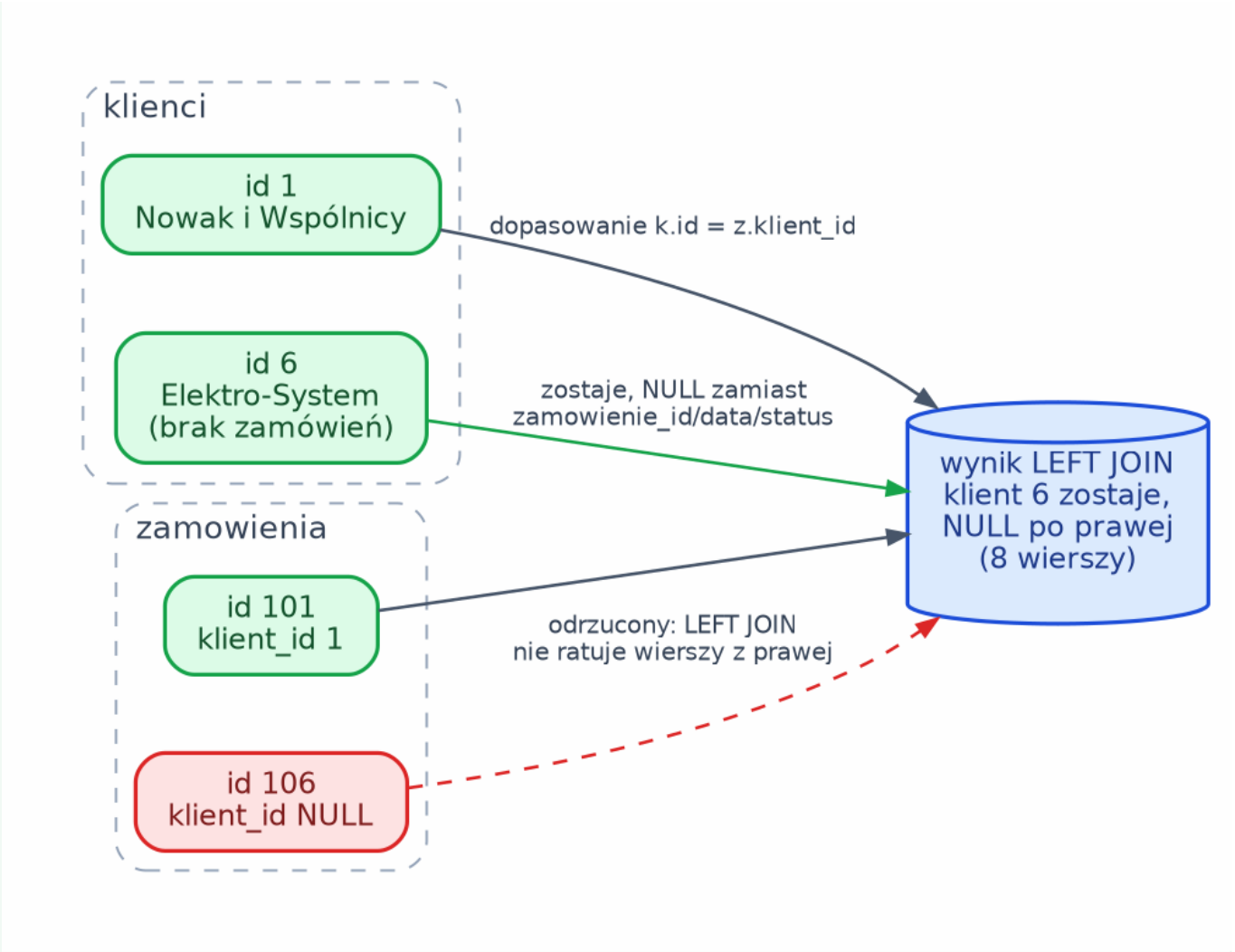
```
SELECT k.id, k.nazwa, z.id AS zamowienie_id, z.data, z.status, z.wartosc
FROM klienci k
LEFT JOIN zamowienia z ON k.id = z.klient_id
ORDER BY k.id;
```

Każdy klient pojawia się w wyniku co najmniej raz — ci z zamówieniami tyle razy, ile mają zamówień, a ci bez zamówień dokładnie raz, z NULL-ami w miejscu kolumn pochodzących z `zamowienia`.

Przykład:

```
 id |            nazwa             | zamowienie_id |    data    |    status    | wartosc
----+------------------------------+---------------+------------+--------------
+---------
  1 | Nowak i Wspólnicy Sp. z o.o. |           101 | 2024-01-15 | zrealizowane | 1250.00

  1 | Nowak i Wspólnicy Sp. z o.o. |           102 | 2024-02-03 | zrealizowane |  430.50

  2 | Kowalski Handel              |           103 | 2024-01-20 | anulowane    |    0.00

  2 | Kowalski Handel              |           105 | 2024-03-15 | nowe         |  150.00

  3 | Zielona Dolina S.A.          |           104 | 2024-03-10 | w realizacji | 2890.00

  4 | Baltic Trade Sp. z o.o.      |           107 | 2024-03-25 | zrealizowane | 3200.00

  5 | Podlaski Handel              |           108 | 2024-03-28 | anulowane    |    0.00

  6 | Elektro-System Kowalczyk     |               |            |
|
(8 rows)
```

Klient numer 6, Elektro-System Kowalczyk, nie ma żadnego zamówienia — przy INNER JOIN ze strony INNER JOIN znikał z wyniku całkowicie, tutaj zostaje, z pustymi kolumnami `zamowienie_id`, `data`, `status` i `wartosc`. To jest cała różnica między INNER a LEFT JOIN: ten sam warunek `ON`, ale inna decyzja co zrobić z wierszem, który go nie spełnił.



Klient bez zamówień zostaje w wyniku LEFT JOIN z NULL-ami po prawej stronie (kliknięcie otwiera pełny rozmiar)

TYLKO ODCZYT

Ta sama logika z odwróconą kolejnością tabel: RIGHT JOIN

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

`RIGHT JOIN` zachowuje wszystkie wiersze tabeli po prawej stronie słowa `JOIN` — tutaj przestawiliśmy tabele miejscami względem poprzedniego przykładu, więc `klienci` jest teraz po prawej, i wynik jest identyczny co w pierwszej Fiszce. `LEFT JOIN A JOIN B` i `RIGHT JOIN B JOIN A` to zawsze dokładnie ta sama operacja, tylko zapisana z drugiej strony.

Przykład: identyczny z powyższym, 8 wierszy, ta sama treść.

W praktyce piszemy niemal wyłącznie `LEFT JOIN`. `RIGHT JOIN` da się zawsze zastąpić `LEFT JOIN`-em z tabelami zamienionymi miejscami, a trzymanie się jednego kierunku ułatwia czytanie dłuższych zapytań z kilkoma złączeniami naraz — nie trzeba w głowie odwracać kierunku za każdym razem, gdy w kodzie pojawia się `RIGHT`.

TYLKO ODCZYT

Wzorzec antyzłączenia: wiersze bez pary

```
SELECT k.id, k.nazwa
FROM klienci k
LEFT JOIN zamowienia z ON k.id = z.klient_id
WHERE z.id IS NULL;
```

> **Pojęcie: antyzłączenie.** Wzorzec `LEFT JOIN` plus `WHERE kolumna_z_prawej IS NULL`, który znajduje wiersze lewej tabeli nieposiadające żadnego dopasowania po prawej stronie. Nazwa mówi wprost, o co chodzi — to złączenie, które szuka braku złączenia.

Przykład:

```
 id |          nazwa
----+---------------------------
  6 | Elektro-System Kowalczyk
(1 row)
```

Warunek sprawdza kolumnę, która przy braku dopasowania zawsze jest `NULL` — tu `z.id`, bo klucz główny tabeli `zamowienia` nigdy nie jest `NULL` sam z siebie, więc jego obecność w wyniku jednoznacznie znaczy „nie znaleziono dopasowania". Sprawdzanie zamiast tego `z.klient_id IS NULL` dałoby ten sam efekt tutaj, ale ryzykowałoby pomyłkę, gdyby kolumna po prawej stronie warunku `ON` sama mogła być `NULL`-em niezależnie od dopasowania.

Po co to istnieje

> **Pojęcie: złączenie zewnętrzne (OUTER JOIN).** Złączenie, które — w przeciwieństwie do INNER JOIN — zachowuje w wyniku wiersze bez dopasowania z co najmniej jednej strony,

> uzupełniając brakujące kolumny wartościami `NULL`. LEFT JOIN i RIGHT JOIN to dwa warianty złączenia zewnętrznego, różniące się tym, którą stronę traktują jako „tę do zachowania w całości" — trzeci wariant, zachowujący obie strony naraz, to FULL OUTER JOIN, opisany osobno na stronie FULL OUTER JOIN. Słowo `OUTER` w składni jest opcjonalne — `LEFT JOIN` i `LEFT OUTER JOIN` to dokładnie to samo polecenie.

Sens LEFT JOIN wynika wprost z pytań, na jakie INNER JOIN nie potrafi odpowiedzieć. „Pokaż wszystkich klientów i ich zamówienia" to zupełnie inne pytanie niż „pokaż klientów, którzy mają zamówienia" — pierwsze wymaga zachowania klientów bez zamówień w wyniku, drugie z definicji ich nie chce. INNER JOIN odpowiada tylko na to drugie pytanie; LEFT JOIN odpowiada na pierwsze.

NULL-e pojawiające się w kolumnach prawej tabeli po LEFT JOIN nie oznaczają błędu ani brakujących danych w sensie „ktoś zapomniał wypełnić" — oznaczają dokładnie to, że dla tego wiersza po lewej nie istnieje żaden pasujący wiersz po prawej. Strona NULL: trzecia wartość logiczna i pułapki, które z niej wynikają już zapowiadała ten mechanizm, mówiąc, że złączenie wypełnia brakującą stronę NULL-ami zamiast usuwać wiersz — to jest właśnie to miejsce.

Jak to działa

LEFT JOIN wykonuje się w dwóch krokach logicznych. Najpierw silnik liczy zwykłe INNER JOIN — wszystkie pary spełniające `ON`. Potem dokłada do wyniku każdy wiersz lewej tabeli, który w ogóle nie pojawił się w żadnej parze z pierwszego kroku, uzupełniając mu kolumny prawej strony `NULL`-ami. Stąd wynika ważna własność: liczba wierszy LEFT JOIN jest zawsze co najmniej taka, jak liczba wierszy zwróconych przez odpowiadające mu INNER JOIN, i rośnie dokładnie o liczbę niedopasowanych wierszy z lewej strony.

Ta sama mechanika tłumaczy, dlaczego `RIGHT JOIN A ON ...` i `LEFT JOIN` z odwróconą kolejnością tabel dają identyczny wynik — `RIGHT` i `LEFT` to tylko etykieta mówiąca, która tabela jest tą „do zachowania w całości", a nie osobny algorytm.

Stara składnia SQL Server ze znakami `*=` i `=*`. Wersje SQL Server sprzed 2005 roku pozwalały zapisać złączenie zewnętrzne bez słowa `OUTER JOIN` w ogóle, dopisując gwiazdkę do znaku równości w `WHERE`:

```
-- składnia usunięta z SQL Server od wersji 2005, tylko dla ilustracji
SELECT k.nazwa, z.wartosc
FROM klienci k, zamowienia z
WHERE k.id *= z.klient_id;
```

`*=` odpowiadało `LEFT JOIN`, `=*` odpowiadało `RIGHT JOIN`. Ta składnia bywała niejednoznaczna przy więcej niż dwóch tabelach naraz — silnik czasem nie potrafił jednoznacznie rozstrzygnąć kolejności

złączeń — i dlatego Microsoft usunął ją całkowicie z domyślnego trybu zgodności, zmuszając cały istniejący kod do migracji na jawne `LEFT JOIN` / `RIGHT JOIN`. To jeden z rzadkich przypadków, w których producent silnika wycofał się z własnego rozszerzenia na rzecz w pełni standardowej składni ANSI.

Częste błędy

Warunek na prawej tabeli w WHERE zamiast w ON. To jest najczęstsza i najkosztowniejsza pomyłka przy LEFT JOIN, na tyle poważna, że doczekała się własnej strony — Warunek w ON a warunek w WHERE: różnica, która zmienia wynik — ale warto zobaczyć jej skutek już tutaj, obok siebie.

Warunek w `ON`, LEFT JOIN zachowany:

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

```
            nazwa             | zamowienie_id |    status
------------------------------+---------------+--------------
 Nowak i Wspólnicy Sp. z o.o. |           101 | zrealizowane
 Nowak i Wspólnicy Sp. z o.o. |           102 | zrealizowane
 Kowalski Handel              |               |
 Zielona Dolina S.A.          |               |
 Baltic Trade Sp. z o.o.      |           107 | zrealizowane
 Podlaski Handel              |               |
 Elektro-System Kowalczyk     |               |
(7 rows)
```

Ten sam warunek przeniesiony do WHERE:

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

```
            nazwa             | zamowienie_id |    status
------------------------------+---------------+--------------
 Nowak i Wspólnicy Sp. z o.o. |           101 | zrealizowane
 Nowak i Wspólnicy Sp. z o.o. |           102 | zrealizowane
 Baltic Trade Sp. z o.o.      |           107 | zrealizowane
(3 rows)
```

Wszyscy klienci bez zamówienia o statusie `zrealizowane` — Kowalski Handel, Zielona Dolina, Podlaski Handel i Elektro-System Kowalczyk — zniknęli, chociaż zapytanie deklaruje LEFT JOIN. `WHERE` działa po złączeniu i odrzuca każdy wiersz, w którym `z.status` nie równa się `'zrealizowane'` — a wiersz z samymi NULL-ami po prawej stronie (bo klient nie miał żadnego zamówienia) też odpada, bo `NULL = 'zrealizowane'` nigdy nie daje prawdy. Efekt jest taki, jakby LEFT JOIN po cichu zamienił się w INNER JOIN.

Sprawdzanie kolumny mogącej być NULL zamiast klucza po prawej stronie. `WHERE z.status IS NULL` w antyzłączeniu zamiast `WHERE z.id IS NULL` zwróci nie tylko klientów bez zamówień, ale też — jeśli taki przypadek istnieje w danych — klientów z zamówieniem, którego status akurat jest `NULL`. Bezpieczniejszym testem na „brak dopasowania" jest zawsze kolumna klucza głównego drugiej tabeli, bo ta z definicji nigdy nie jest `NULL` sama w sobie.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — policzenie wierszy LEFT JOIN

Tabela `klienci` ma sześć wierszy. Klient 1 ma dwa zamówienia, klient 2 ma dwa zamówienia, klienci 3, 4 i 5 mają po jednym, klient 6 nie ma żadnego. Ile wierszy zwróci `SELECT * FROM klienci k LEFT JOIN zamowienia z ON k.id = z.klient_id;`, pomijając zamówienie 106 bez przypisanego klienta?

Odpowiedź

Osiem: 2 + 2 + 1 + 1 + 1 + 1 = 8. Klienci z dopasowaniami dają tyle wierszy, ile mają zamówień, a klient bez dopasowania (numer 6) dokłada dokładnie jeden wiersz z NULL-ami — LEFT JOIN nigdy nie usuwa wiersza lewej tabeli, nawet gdy nie ma dla niego żadnej pary.

Ćwiczenie 2 — antyzłączenie na trzech tabelach

Chcemy znaleźć produkty, których nigdy nie zamówiono — czyli takie, które nie pojawiają się w żadnej pozycji zamówienia. Jak zapisać to jako antyzłączenie na tabelach `produkty` i `pozycje`?

Odpowiedź

```
SELECT p.id, p.nazwa
FROM produkty p
LEFT JOIN pozycje poz ON poz.produkt_id = p.id
WHERE poz.id IS NULL;
```

Ten sam wzorzec co w Fiszce tej strony, tylko na innej parze tabel — `LEFT JOIN` zachowuje każdy produkt, dopasowany do pozycji albo nie, a `WHERE poz.id IS NULL` zostawia wyłącznie te, które nie znalazły żadnej pozycji. W naszych danych `pozycje` odwołuje się do produktów o id 1, 2, 3 i 5 — Fotel biurowy (4) i Lampka biurkowa (6) nigdy się tam nie pojawiają, więc to one wyjdą w wyniku tego zapytania.

Pytania kontrolne

1. Dlaczego `RIGHT JOIN` prawie nie występuje w prawdziwym kodzie, skoro jest częścią standardu SQL na równi z `LEFT JOIN`?

Odpowiedź

Bo każde `RIGHT JOIN` da się przepisać jako `LEFT JOIN` z zamienioną kolejnością tabel, a trzymanie się jednego kierunku w całej bazie kodu ułatwia czytanie zapytań z wieloma złączeniami — nie trzeba za każdym razem sprawdzać, w którą stronę akurat działa dany `JOIN`.

2. Zapytanie z LEFT JOIN i warunkiem na prawą tabelę w WHERE zwraca mniej wierszy niż programista się spodziewał. Jaka jest pierwsza rzecz do sprawdzenia?

Odpowiedź

Czy warunek dotyczący prawej tabeli powinien stać w `ON`, a nie w `WHERE`. Warunek w `WHERE` odfiltrowuje też wiersze, w których prawa strona jest cała `NULL`-em z braku dopasowania, co w

praktyce zamienia LEFT JOIN w INNER JOIN — dokładny mechanizm opisuje strona Warunek w ON a warunek w WHERE: różnica, która zmienia wynik.

Powiązane: INNER JOIN · FULL OUTER JOIN · Warunek w ON a warunek w WHERE: różnica, która zmienia wynik · NULL: trzecia wartość logiczna i pułapki, które z niej wynikają

Źródła: ISO/IEC 9075-2:2016 § 7.7 · dokumentacja PostgreSQL, rozdział „Joins Between Tables" · dokumentacja Oracle Database SQL Language Reference, sekcja „Outer Joins"

Revision #46
Created 2026-09-24 22:41:58 UTC by Tomasz Konieczny
Updated 2026-10-26 14:11:00 UTC by Tomasz Konieczny