Relacje: 1:1, 1:N, N:M i tabela łącząca

Poziom: podstawy **Dotyczy: wszystkie silniki** **Czas: ~15 min**

1:1, 1:N i N:M opisują, ile wierszy jednej tabeli może odpowiadać ilu wierszom drugiej — i tylko jedna z tych trzech kardynalności wymaga trzeciej tabeli, żeby w ogóle dało się ją zapisać.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

ZMIENIA SYSTEM

Realizacja relacji 1:N kluczem obcym po stronie „wielu"

```
CREATE TABLE pozycje_zamowienia (
    id SERIAL PRIMARY KEY,
    zamowienie_id INTEGER NOT NULL REFERENCES zamowienia(id),
    produkt_id INTEGER NOT NULL,
    ilosc INTEGER NOT NULL CHECK (ilosc > 0)
);
```

Klucz obcy `zamowienie_id` siedzi po stronie tabeli, która ma być „wieloma" — jedno zamówienie może mieć wiele pozycji, ale każda pozycja wskazuje dokładnie jedno zamówienie.

Przykład:

```
CREATE TABLE
```

□□ **id** = pokaż użytkownika, jego numer UID oraz grupy, do których należy

□□ ZMIENIA SYSTEM

Realizacja relacji N:M tabelą łączącą

```
CREATE TABLE uzytkownicy_role (
    uzytkownik_id INTEGER NOT NULL REFERENCES uzytkownicy(id),
    rola_id INTEGER NOT NULL REFERENCES role(id),
    PRIMARY KEY (uzytkownik_id, rola_id)
);
```

Żadna z tabel `uzytkownicy` ani `role` nie dostaje klucza obcego do drugiej — relacja N:M mieszka wyłącznie w tabeli pośredniej.

Przykład:

```
CREATE TABLE
```

Po co to istnieje

Na stronie Model relacyjny: tabela, wiersz, kolumna, dziedzina słowo „relacja" oznaczało tabelę. Tutaj to samo słowo pojawia się w drugim, potocznym znaczeniu — jako powiązanie między dwiema tabelami: klient ma zamówienia, zamówienie ma pozycje, użytkownik ma role. Oba znaczenia funkcjonują w rozmowach o bazach danych równolegle, i kontekst zwykle wystarcza, żeby się nie pomylić, ale warto mieć to rozróżnienie nazwane wprost, zanim padnie reszta tej strony.

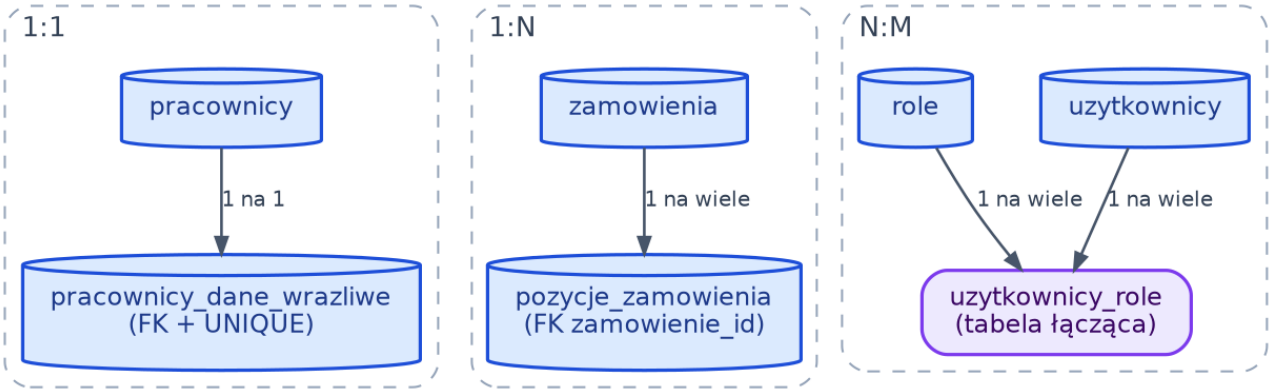
> **Pojęcie: relacja między tabelami.** Powiązanie logiczne, które mówi, jak wiersze jednej tabeli odnoszą się do wierszy drugiej. Kluczowa własność takiej relacji to jej kardynalność — czyli ile wierszy po jednej stronie może odpowiadać ilu wierszom po drugiej. Kardynalności są trzy: jeden do jednego, jeden do wielu i wiele do wielu, i to one — nie technika ich zapisu — decydują, jak relację zaprojektować.

Analogia

Wyobraźmy sobie szkołę. Uczeń i jego numer legitymacji szkolnej są w relacji 1:1 — jeden uczeń ma dokładnie jedną legitymację, a jedna legitymacja należy do dokładnie jednego ucznia; gdyby ktoś zgubił legitymację, dostaje nową o innym numerze, a nie cudzą. Uczeń i klasa, do której jest przypisany na stałe, to relacja 1:N — jedna klasa ma wielu uczniów, ale każdy uczeń w danym roku szkolnym należy do dokładnie jednej klasy. Uczeń i przedmioty, na które uczęszcza, to relacja N:M — jeden uczeń chodzi na wiele przedmiotów, a na jeden przedmiot (matematykę) chodzi wielu uczniów naraz.

Analogia pokazuje też, dlaczego N:M potrzebuje osobnego bytu. Sekretariat szkoły nie prowadzi tej relacji, dopisując w kartotece ucznia listę przedmiotów ani w kartotece przedmiotu listę uczniów — prowadzi osobny rejestr zapisów, gdzie każdy wiersz to jedno zdarzenie: ten uczeń zapisany na ten przedmiot w tym roku. Ten rejestr jest dokładnie tym, czym w bazie danych jest tabela łącząca.

Trzy kardynalności



Kardynalność 1:1, 1:N i N:M: dwie tabele połączone kluczem obcym, a przy N:M trzecia tabela łącząca pośrodku (kliknięcie otwiera pełny rozmiar)

1:1 — jeden do jednego. Jeden wiersz tabeli A odpowiada najwyżej jednemu wierszowi tabeli B, i odwrotnie. W praktyce spotyka się to rzadziej niż pozostałe dwie kardynalności, zwykle jako świadomy podział jednej logicznej encji na dwie tabele — na przykład `pracownicy` (dane podstawowe, czytane często) i `pracownicy_dane_wrazliwe` (numer konta, dane medyczne, czytane rzadko i z węższymi uprawnieniami). Realizuje się to kluczem obcym, na którym dodatkowo nałożone jest ograniczenie `UNIQUE` — bez `UNIQUE` kolumna klucza obcego pozwoliłaby wielu wierszom tabeli B wskazywać ten sam wiersz tabeli A, co przeistoczyłoby relację w 1:N.

1:N — jeden do wielu. Jeden wiersz tabeli A odpowiada wielu wierszom tabeli B, ale każdy wiersz tabeli B odpowiada dokładnie jednemu wierszowi tabeli A. To najczęstsza kardynalność w praktycznych schematach. Klasyczny przykład: jedno zamówienie (`zamowienia`) ma wiele pozycji (

`pozycje_zamowienia`), ale każda pozycja należy do dokładnie jednego zamówienia. Realizuje się ją kluczem obcym umieszczonym po stronie „wielu" — to `pozycje_zamowienia` trzyma kolumnę `zamowienie_id`, nie odwrotnie, bo gdyby `zamowienia` miały kolumnę wskazującą pozycję, jeden wiersz mógłby wskazać tylko jedną pozycję, co przeczyłoby idei zamówienia z wieloma pozycjami.

N:M — wiele do wielu. Wiele wierszy tabeli A odpowiada wielu wierszom tabeli B, i odwrotnie. Przykład: użytkownik może mieć wiele ról (administrator i redaktor naraz), a jedna rola (administrator) przysługuje wielu użytkownikom. Tej kardynalności nie da się zapisać kluczem obcym w żadnej z dwóch tabel wprost — kolumna w relacyjnej tabeli przechowuje jedną wartość, nie listę wartości, więc `uzytkownicy.rola_id` mogłoby wskazać tylko jedną rolę na użytkownika, co jest sprzeczne z założeniem. Rozwiązaniem jest trzecia tabela.

Dlaczego N:M zawsze potrzebuje tabeli łączącej

> **Pojęcie: tabela łącząca (pośrednia).** Tabela, której jedynym zadaniem jest przechowywanie par kluczy obcych wskazujących po jednym wierszu z każdej z dwóch tabel biorących udział w relacji N:M. Każdy wiersz tabeli łączącej reprezentuje jedno konkretne skojarzenie — ten użytkownik ma tę rolę.

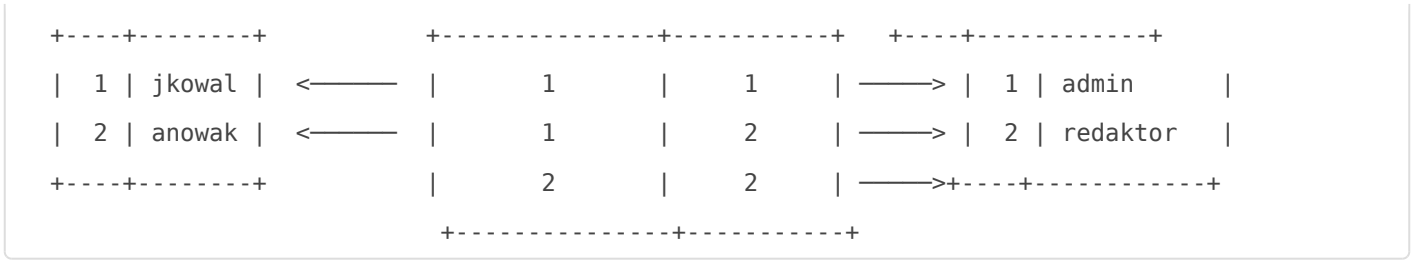
Powód, dla którego to jedyne rozwiązanie w modelu relacyjnym, wynika wprost z zasady, że kolumna tabeli przechowuje pojedynczą, atomową wartość — nie tablicę ani listę. Rozbicie relacji N:M na dwie relacje 1:N sprowadza problem do czegoś, co już umiemy zapisać: tabela łącząca jest w relacji 1:N zarówno z tabelą `uzytkownicy` (jeden użytkownik, wiele wierszy w tabeli łączącej), jak i z tabelą `role` (jedna rola, wiele wierszy w tabeli łączącej). Klucz główny tabeli łączącej jest zwykle złożony z obu kluczy obcych naraz (`PRIMARY KEY (uzytkownik_id, rola_id)`), co dodatkowo wymusza, że dana para użytkownik-rola nie powtórzy się dwa razy.

```
zamowienia (1) ------------------ (N) pozycje_zamowienia
    id  <──────────────────────────────  zamowienie_id



uzytkownicy (N) --- (N) role
                przez tabelę łączącą:

  uzytkownicy                uzytkownicy_role                role
 +----+--------+           +---------------+-----------+   +----+------------+
 | id | login  |           | uzytkownik_id | rola_id   |   | id | nazwa      |
```



Jak to działa

Silnik nie ma osobnego mechanizmu dla „relacji N:M" — dla niego istnieją wyłącznie tabele i klucze obce, a N:M to wzorzec zbudowany z dwóch relacji 1:N. Zapytanie, które chce zobaczyć role danego użytkownika, musi więc przejść przez tabelę pośrednią złączeniem — nie da się dostać z `uzytkownicy` prosto do `role` bez przejścia przez `uzytkownicy_role`:

```
SELECT r.nazwa
FROM role r
JOIN uzytkownicy_role ur ON ur.rola_id = r.id
WHERE ur.uzytkownik_id = 1;
```

Ta sama mechanika kluczy obcych, która wymusza istnienie tabeli nadrzędnej przy 1:N, działa też w tabeli łączącej — próba wstawienia pary z nieistniejącym `uzytkownik_id` albo `rola_id` zostanie odrzucona przez oba ograniczenia `FOREIGN KEY` naraz.

Relacja 1:1 bywa czasem realizowana jeszcze inaczej — zamiast osobnej kolumny z `UNIQUE`, obie tabele dzielą ten sam klucz główny (klucz główny tabeli B jest jednocześnie kluczem obcym do tabeli A). Wtedy `pracownicy_dane_wrazliwe.id` jest zarówno kluczem głównym tej tabeli, jak i kluczem obcym do `pracownicy.id` — co dodatkowo gwarantuje relację 1:1 bez osobnego ograniczenia `UNIQUE`, bo klucz główny z definicji już jest unikalny.

Usuwanie wierszy powiązanych relacją

Skasowanie wiersza, który jest stroną „jeden" w relacji 1:N albo jednym z uczestników relacji N:M, rodzi pytanie, co ma się stać z wierszami, które się do niego odwołują. Silnik pozwala to zdefiniować przy tworzeniu klucza obcego przez klauzulę `ON DELETE`.

Domyślne zachowanie w większości silników to odrzucenie usunięcia (`RESTRICT` albo `NO ACTION`) — nie da się skasować klienta, dopóki istnieją jego zamówienia. `ON DELETE CASCADE` usuwa razem z wierszem nadrzędnym wszystkie wiersze podrzędne — skasowanie zamówienia automatycznie

kasuje jego pozycje, co dla relacji 1:N między `zamowienia` i `pozycje_zamowienia` zwykle jest właściwym zachowaniem, bo pozycja bez zamówienia nie ma sensu. `ON DELETE SET NULL` zostawia wiersz podrzędny, ale zeruje klucz obcy — sensowne tylko wtedy, gdy kolumna klucza obcego dopuszcza `NULL` i gdy istnienie wiersza podrzędnego bez rodzica ma jakiekolwiek znaczenie biznesowe.

W tabeli łączącej N:M niemal zawsze stosuje się `ON DELETE CASCADE` po obu stronach — usunięcie użytkownika powinno usunąć jego przypisania do ról, nie zostawiać w `uzytkownicy_role` osierocone wiersze wskazujące nieistniejącego już użytkownika.

Częste błędy

Modelowanie N:M listą wartości w jednej kolumnie. Kolumna `uzytkownicy.role` z wartością `'admin,redaktor'` technicznie działa — silnik przyjmie taki tekst bez sprzeciwu — ale łamie założenie, że kolumna trzyma pojedynczą, atomową wartość. Wyszukanie wszystkich administratorów wymaga wtedy dopasowania podciągu tekstu zamiast prostego złączenia, dodanie roli wymaga edycji tekstu zamiast wstawienia wiersza, a nic nie broni literówki w nazwie roli w środku takiego ciągu.

Brak złożonego klucza głównego na tabeli łączącej. Tabela `uzytkownicy_role` z osobną kolumną `id` jako jedynym kluczem głównym, bez ograniczenia `UNIQUE (uzytkownik_id, rola_id)`, dopuszcza wstawienie tej samej pary użytkownik-rola wielokrotnie — z pozoru nieszkodliwe, dopóki raport licząc przypisania ról nie zacznie liczyć duplikatów.

Modelowanie 1:1 dwoma niezależnymi kluczami obcymi bez `UNIQUE`. Dwie tabele połączone zwykłym kluczem obcym bez dodatkowego ograniczenia unikalności są w relacji 1:N, nie 1:1 — nic nie stoi na przeszkodzie, żeby dwa wiersze tabeli B wskazywały ten sam wiersz tabeli A. Jeśli zamiarem było 1:1, taki brak ograniczenia jest cichym błędem projektowym, który ujawni się dopiero jako duplikat w produkcji.

□□ Ćwiczenia

> Poniższe zadania wykonujemy na dowolnym silniku SQL z uprawnieniami do tworzenia tabel we własnej, testowej bazie.

Ćwiczenie 1 — zaprojektowanie relacji

Mamy zaprojektować schemat dla biblioteki: książki, czytelnicy, wypożyczenia. Jeden czytelnik może wypożyczyć wiele książek w różnym czasie, a jedna książka (jeden konkretny egzemplarz) w

danym momencie może być wypożyczona przez jednego czytelnika, ale w historii — przez wielu, jeden po drugim. Jaka to kardynalność między `czytelnicy` i `ksiazki`, i gdzie ląduje klucz obcy?

Odpowiedź

To 1:N po obu stronach naraz, rozwiązywane trzecią tabelą `wypozyczenia` — nie dlatego, że jeden czytelnik może mieć wiele wypożyczonych książek w danym momencie (to samo w sobie dałoby się zapisać kluczem obcym w `wypozyczenia.czytelnik_id`), ale dlatego, że jedna książka jest wypożyczana wielokrotnie w czasie przez różnych czytelników — historia wypożyczeń wymaga osobnego wiersza na każde zdarzenie, nie da się jej zapisać jako pojedynczej kolumny w `ksiazki`. Tabela `wypozyczenia` ma kolumny `czytelnik_id`, `ksiazka_id`, `data_wypozyczenia`, `data_zwrotu` — i to jest wzorcowa tabela łącząca, tyle że z dodatkowymi kolumnami opisującymi samo zdarzenie, nie tylko parę kluczy.

Pytania kontrolne

1. Dlaczego relacji N:M nie da się zapisać, dodając do tabeli `uzytkownicy` kolumnę `rola_id` wskazującą tabelę `role`?

Odpowiedź

Bo kolumna `rola_id` w jednym wierszu trzyma dokładnie jedną wartość, a N:M zakłada, że jeden użytkownik może mieć wiele ról naraz. Taka kolumna zrealizowałaby co najwyżej relację N:1 (wielu użytkowników, jedna rola każdy), nie N:M.

2. Tabela łącząca `uzytkownicy_role` dostała dodatkową kolumnę `nadano_od DATE`. Czy to coś zmienia w jej roli jako tabeli łączącej?

Odpowiedź

Nie zmienia jej roli — tabela łącząca może przechowywać dodatkowe atrybuty samej relacji (kiedy rola została nadana, kto ją nadał), o ile nie łamie to klucza głównego złożonego z obu kluczy obcych. To bardzo częsty i poprawny wzorzec: relacja N:M z własnymi cechami, a nie tylko gołą parą identyfikatorów.

Powiązane: Model relacyjny: tabela, wiersz, kolumna, dziedzina · Klucze: główny, obcy, unikalny, naturalny, sztuczny · Czym jest złączenie: iloczyn kartezjański i warunek, który go ogranicza · Normalizacja do 3NF i świadoma denormalizacja

Źródła: C. J. Date, „An Introduction to Database Systems", rozdz. o związkach między relacjami · dokumentacja PostgreSQL, rozdz. „Foreign Keys"

Revision #47
Created 2026-09-24 22:41:37 UTC by Tomasz Konieczny
Updated 2026-10-26 14:10:36 UTC by Tomasz Konieczny