Bazy danych
przed 11 wykładem

Andrzej Giniewicz

24.05.2024

Błędy popełnione na etapie projektowania bazy danych mogą być trudne do usunięcia, gdy działa ona już „na produkcji". Aby uniknąć niektórych z nich, można użyć procesu normalizacji. Proces normalizacji jest formalną metodą szukania potencjalnych niespójności. Ponieważ jest to metoda formalna, musimy wprowadzić kilka terminów i oznaczeń, zatem zaczniemy blok od wstępu teoretycznego. Warto nadmienić, że będzie to wprowadzenie stosunkowo skrótowe, zbyt płytkie, gdyby ktoś był zainteresowany teoretycznymi podstawami projektowania baz danych. **Z materiału tego nie robimy kartkówki, ale będzie wymagany do projektu zaliczeniowego.**

1 Terminologia

Aby przejść do definicji, musimy wprowadzić nieco formalnej terminologii z projektowania baz danych. Baza danych składa się ze **zmiennych relacyjnych**. Wartość zmiennej relacyjnej nazywamy **relacją**. Relacja składa się z **krotek**. Jeśli nieformalnie mówimy o tabeli *Użytkownicy*, w której znajdują się dane

id	name	password
1	Tic	X
2	Tac	O
3	Toe	?

to formalnie *Użytkownicy* jest zmienną relacyjną, $(1, \text{Tic}, X)$ jest krotką, podobnie jak $(2, \text{Tac}, O)$ oraz $(3, \text{Toe}, ?)$. Zbiór krotek, czyli

$$\{(1, \text{Tic}, X), (2, \text{Tac}, O), (3, \text{Toe}, ?)\}$$

jest relacją. Ogólnie napiszemy

$$\mathit{Użytkownicy} = \{(1, \text{Tic}, X), (2, \text{Tac}, O), (3, \text{Toe}, ?)\}.$$

Osobnym terminem określa się **nagłówek relacji**, który składa się z **atrybutów**. Atrybuty to przyporządkowania nazw kolumn ich **dziedzinom**. W zmiennej relacyjnej z przykładu mamy trzy atrybuty: *id*, którego dziedziną są liczby całkowite oraz *name* i *password*, których dziedzinami są napisy. O dziedzinie atrybutu można myśleć jak o typie danych kolumny.

Atrybuty relacji możemy zapisać w nawiasie obok zmiennej relacyjnej. Na liście tej często podkreślamy atrybuty wchodzące w skład klucza głównego (nie mylić z atrybutami głównymi). Dziedziny można wypisać w indeksach dolnych nazw, choć często je pomijamy.

$$\mathit{Użytkownicy}(\mathit{id}_{\text{int}}, \mathit{name}_{\text{str}}, \mathit{password}_{\text{str}}) = \{(1, \text{Tic}, X), (2, \text{Tac}, O), (3, \text{Toe}, ?)\}.$$

Zwróćmy uwagę, że dziedziny nie muszą być zapisane za pomocą kodu SQL, wystarczy dowolny interpretowalny przez człowieka ciąg znaków. Gdy nie komunikujemy dziedzin, możemy zapisać też

$$\mathit{Użytkownicy}(\mathit{id}, \mathit{name}, \mathit{password}) = \{(1, \text{Tic}, X), (2, \text{Tac}, O), (3, \text{Toe}, ?)\}.$$

Jeśli interesuje nas tylko nagłówek relacji, a nie sama relacja, możemy napisać

$$\mathit{Użytkownicy}(\mathit{id}, \mathit{name}, \mathit{password}).$$

Dla relacji R o zbiorze atrybutów A rozważmy relację $\pi_B(R)$ powstałą przez pominięcie współrzędnych krotek nienależących do zbioru $B \subset A$. Operację tę nazywamy **projekcją**. Wracając do przykładowej relacji

$$\mathit{Użytkownicy}(\mathit{id}, \mathit{name}, \mathit{password}) = \{(1, \text{Tic}, X), (2, \text{Tac}, O), (3, \text{Toe}, ?)\},$$

jej projekcja $S = \pi_{\{\mathit{name},\mathit{password}\}}(\mathit{Użytkownicy})$, to relacja

$$S(\mathit{name}, \mathit{password}) = \{(\text{Tic}, X), (\text{Tac}, O), (\text{Toe}, ?)\}.$$

Ten sam zapis stosujemy do projekcji krotek, gdzie $\pi_B : R \to \pi_B(R)$ traktujemy jako funkcję przekształcającą krotki relacji R w krotki relacji $\pi_B(R)$ (z pominiętymi kolumnami, których nie ma w zbiorze B). Na przykład

$$\pi_{\{\mathit{name},\mathit{password}\}}(1, \text{Tic}, X) = (\text{Tic}, X).$$

Zbiór atrybutów, który pozwala jednoznacznie identyfikować krotki relacji, nazywamy **nadkluczem**. Relacja może mieć wiele nadkluczy. Formalnie X jest nadkluczem relacji R, jeśli

$$\forall r_1, r_2 \in R \quad (\pi_X(r_1) = \pi_X(r_2)) \implies r_1 = r_2.$$

Jeśli dla nadklucza X nie istnieje $Y \subsetneq X$ będący również nadkluczem, X nazywamy **kluczem kandydującym**. Klucze kandydujące to takie nadklucze, które są minimalne w sensie zawierania atrybutów, czyli po usunięciu dowolnego atrybutu przestają być nadkluczami. Atrybuty należące do dowolnego klucza kandydującego nazywamy **atrybutami głównymi**. Pozostałe, czyli te, które nie należą do żadnego klucza kandydującego, nazywamy **atrybutami niegłównymi**. Spośród wszystkich kluczy kandydujących jeden wskazujemy jako **klucz główny**. Projektując bazę mamy dowolność w wyborze klucza głównego, warto jednak dobrać go tak, aby był użyteczny i wydajny w danej implementacji silnika bazy danych.

Przypomnijmy przykład z pierwszej strony

id	name	password
1	Tic	X
2	Tac	O
3	Toe	?

Załóżmy, że do logowania używamy nazwy użytkownika, musi być zatem unikatowa. Załóżmy, że pole id jest automatycznie numerowane, więc również będzie unikatowe. Oczywiście, kilku użytkowników może mieć takie samo hasło, na przykład „admin1". Przy takich założeniach, nadkluczami w tej relacji są $\{id\}$, $\{name\}$, $\{id, name\}$, $\{id, password\}$, $\{name, password\}$ oraz $\{id, name, password\}$. Kluczami kandydującymi są jedynie $\{id\}$ oraz $\{name\}$, w związku z czym atrybuty główne to *id* oraz *name*, natomiast atrybuty niegłówne to *password*.

Zwróćmy uwagę, że jest to moment, w którym istotne zaczyna być to, co wiemy o danych w tabeli. To, że dany zbiór atrybutów jednoznacznie identyfikuje krotki w relacji, wymaga zewnętrznej wiedzy, zatem ustalenie co jest kluczem kandydującym, spoczywa na człowieku — automat tego nie zrobi — chyba, że sformalizujemy wiedzę o tabeli w postaci tak zwanych zależności funkcyjnych.

Niech X i Y będą podzbiorami atrybutów relacji. Jeśli dla wszystkich krotek w relacji zachodzi

$$\forall r_1, r_2 \in R \quad (\pi_X(r_1) = \pi_X(r_2)) \implies (\pi_Y(r_1) = \pi_Y(r_2)),$$

mówimy, że zachodzi zależność funkcyjna $X \to Y$. W praktyce oznacza to, że wiersze równe sobie na kolumnach ze zbioru X, muszą być sobie równe na kolumnach ze zbioru Y. Rozważmy tabelę z danymi osobowymi. Jeśli w tabeli przechowujemy datę urodzenia i dzień tygodnia, w którym się dana osoba urodziła, mamy w tej tabeli zależność funkcyjną data $\to$ dzień tygodnia, ponieważ dwie osoby urodzone w ten sam dzień, na pewno urodziły się w ten sam dzień tygodnia.

Każda tabela ma przynajmniej jedną zależność funkcyjną, ponieważ ma przynajmniej jeden atrybut. Definicja zależności funkcyjnej nie wyklucza przypadku $X \to X$, co jest oczywiste. Oczywiste są też wszystkie zależności funkcyjne postaci $X \to Y$, gdy $Y \subseteq X$. Zależności takie nazywamy **trywialnymi zależnościami funkcyjnymi**.

Jeśli dla różnych zbiorów atrybutów X, Y, Z zachodzi $X \to Y$ oraz $Y \to Z$, ale nie zachodzi $Y \to X$ mówimy, że $X \to Z$ jest **przechodnią zależnością funkcyjną**. Jeśli dla różnych zbiorów atrybutów X, Y, Z takich, że $Y \subsetneq X$ zachodzi $X \to Z$ oraz $Y \to Z$, to zależność $X \to Z$ nazywamy **częściową zależnością funkcyjną**. Nietrywialne zależności funkcyjne, które nie są częściowe, nazywamy **elementarnymi zależnościami funkcyjnymi**. Klucz kandydujący X nazywamy **kluczem elementarnym**, jeśli istnieje atrybut $\{a\}$ taki, że w relacji R prawdziwa jest elementarna zależność funkcyjna $X \to \{a\}$. Atrybut nazywamy **atrybutem elementarnym**, jeśli jest elementem pewnego klucza elementarnego.

Jeśli rozważamy zbiór zależności funkcyjnych Σ, **domknięciem** tego zbioru nazywamy zbiór Σ^+ zależności funkcyjnych, które możemy wywnioskować z Σ. Na przykład, jeśli

$$\Sigma = \{A \to B, B \to C, BD \to E, AD \to F\},$$

w relacji $R(A, B, C, D, E, F)$, to możemy wywnioskować, że w Σ^+ są:

1. $A \to A$, trywialna,
2. $A \to B$, ponieważ należy do Σ,
3. $A \to C$, ponieważ $A \to B$ i $B \to C$,
4. $A \to AB$, ponieważ $A \to A$ i $A \to B$,
5. $A \to AC$, ponieważ $A \to A$ i $A \to C$,
6. $A \to BC$, ponieważ $A \to B$ i $A \to C$,
7. $A \to ABC$, ponieważ $A \to A$ i $A \to BC$,
8. $B \to B$, trywialna,
9. $B \to C$, ponieważ należy do Σ,
10. $B \to BC$, ponieważ $B \to B$ i $B \to C$,
11. $BD \to BD$, trywialna,
12. $BD \to E$, ponieważ należy do Σ,
13. $BD \to BDE$, ponieważ $BD \to BD$ i $BD \to E$,

… (i tak dalej, w tym ponad 100 trywialnych zależności funkcyjnych, ciężko wypisać wszystkie, ale wypiszemy jeszcze jedną),

x. $AD \to ABCDEF$, ponieważ $A \to ABC$, więc $AD \to ABCD$, ponieważ $BD \to E$ to również $AD \to ABCDE$, ponieważ $AD \to F$, to również $AD \to ABCDEF$.

Ostatni przykład jest szczególny, ponieważ pokazuje sposób na szukanie nadkluczy. Jeśli wyliczymy domknięcie zbioru zależności funkcyjnych, następnie ze wszystkich wybierzemy te, które mają po prawej stronie wszystkie atrybuty relacji, po ich lewej stronie znajdować się będą nadklucze. W przykładzie powyżej, AD jest możliwym nadkluczem relacji, przy czym ponieważ nie można go zmniejszyć (nie zachodzi $A \to D$ ani $D \to A$), to jest on jednocześnie kluczem kandydującym i mógłby być kluczem głównym tej relacji.

Jeśli $X \to Y$ jest zależnością funkcyjną należącą do zbioru zależności funkcyjnych Σ zachodzących dla danej relacji, atrybut $x \in X$ nazywamy **atrybutem nadmiarowym** w zależności $X \to Y$ nad zbiorem Σ, jeśli zastąpienie relacji $X \to Y$ w Σ przez $X \setminus \{x\} \to Y$ nie zmienia jego domknięcia, czyli formalnie

$$\Sigma^+ = (\Sigma \setminus \{X \to Y\} \cup \{X \setminus \{x\} \to Y\})^+ .$$

Zbiór zależności funkcyjnych Σ_c nazywamy **nieredundantnym pokryciem** zbioru zależności funkcyjnych Σ, jeśli $\Sigma_c \subseteq \Sigma$, $\Sigma_c^+ = \Sigma^+$ oraz dla każdej zależności funkcyjnej $X \to Y \in \Sigma_c$, zachodzi $\left(\Sigma_c \setminus \{X \to Y\}\right)^+ \neq \Sigma^+$. Oznacza to, że nierendundatne pokrycie Σ_c stanowi najmniejszy zbiór zależności funkcyjnych, z których możemy wywnioskować to samo, co ze zbioru Σ. Nieredundantne pokrycie nie jest wyznaczone jednoznacznie, a jego

praktyczne wyznaczenie można przeprowadzić metodą eliminacji, wybierając po jednej zależności funkcyjnej ze zbioru i wykreślając ją, jeśli da się ją wywnioskować z pozostałych lub trywialnych zależności. W szczególności oznacza to, że w nierendundatnym pokryciu nie ma ani jednej zależności trywialnej.

2 Postaci normalne

W notatkach poznamy pierwsze pięć postaci normalnych: pierwszą, drugą, trzecią, klucza elementarnego oraz Boyce'a-Codda. Istnieje więcej postaci normalnych, jednak na nasze potrzeby wystarczy omówienie powyższych.

- Relacja R o atrybutach $(A_1, \ldots, A_k)$ takich, że dziedziną A_i jest T_i dla $i = 1, \ldots, k$ jest w **pierwszej postaci normalnej (1NF)** wtedy i tylko wtedy, gdy każda krotka r z tej relacji należy do zbioru $T_1 \times T_2 \times \ldots \times T_k$.
- Relacja jest w **drugiej postaci normalnej (2NF)**, jeśli każda nietrywialna zależność funkcyjna albo nie zaczyna się od podzbioru właściwego nadklucza albo kończy się na atrybucie głównym. Oznacza to, że nie ma częściowych zależności funkcyjnych atrybutów niegłównych od kluczy kandydujących. Każda tabela w 2NF jest również w 1NF.
- Relacja jest w **trzeciej postaci normalnej (3NF)**, jeśli każda nietrywialna zależność funkcyjna albo zaczyna się od nadklucza albo kończy się na atrybucie głównym. Oznacza to, że nie ma tranzytywnych zależności funkcyjnych atrybutów niegłównych od kluczy kandydujących. Każda tabela w 3NF jest również w 2NF.
- Relacja jest w **postaci normalnej klucza elementarnego (EKNF)**, jeśli każda nietrywialna zależność funkcyjna albo zaczyna się od nadklucza albo kończy się na atrybucie elementarnym. Każda tabela w EKNF jest również w 3NF.
- Relacja jest w **postaci normalnej Boyce'a-Codda (BCNF)**, jeśli każda nietrywialna zależność funkcyjna zaczyna się od nadklucza. Każda tabela w BCNF jest również w EKNF.

Postaci normalne opisane powyżej się zawierają. Każda kolejna pomaga eliminować potencjalne anomalie w bazie danych. Do anomalii zaliczamy różnego rodzaju niespójności, które mogą wynikać z duplikacji danych w bazie lub niespełnionych zależności funkcyjnych. Przykład z datą urodzin i dniem tygodnia w jednej tabeli obrazuje potencjalny problem — jeśli okaże się, że ktoś modyfikuje datę, ponieważ podał niewłaściwą — co nie jest niemożliwe, nie wszyscy rejestrując się na stronach internetowych podają prawdziwą datę urodzin — musimy pamiętać, aby w innym miejscu tabeli zmienić też dzień tygodnia. Jeśli o tym zapomnimy, błąd taki spowoduje utratę cech zależności funkcyjnej i może doprowadzić do dziwnego zachowania się strony. Proces normalizacji pomaga usunąć duplikację i dzięki temu zapobiega wielu anomaliom.

Normalizacja bazy danych odbywa się przez sprowadzanie do postaci normalnej poszczególnych relacji. Niekiedy tworzymy osobną tabelę, aby uniknąć duplikacji, niekiedy możemy usunąć wiersze lub przenieść je do innej tabeli. Niestety, nie zawsze możemy sprowadzić każdą relację do BCNF. Najwyższa postać normalna, do której da się sprowadzić każdą możliwą relację, to EKNF[1].

3 Sprowadzanie do postaci EKNF

Opiszemy teraz algorytm Bernsteina[2], który pozwala sprowadzić relację do postaci EKNF. Załóżmy, że mamy relację $R(\underline{A}, B, C, \underline{D}, E)$ ze zbiorem zależności funkcyjnych

$$\Sigma = \{A \to B, B \to C, B \to A, A \to C, ABD \to E\}.$$

Chcemy poddać ją normalizacji i sprowadzić do EKNF.

3.1 Krok 1: usuwamy nadmiarowe atrybuty w zależnościach funkcyjnych

W zależności funkcyjnej $ABD \to E$ atrybut B jest nadmiarowy, ponieważ $A \to B$ i $AD \to E$ daje razem $ABD \to E$. Oznacza to, że możemy wykreślić B z tej zależności funkcyjnej. W przykładzie nie ma innych atrybutów nadmiarowych. Analogicznie moglibyśmy wykreślić stamtąd A, ponieważ $B \to A$, ale nie oba jednocześnie. Wobec tego

$$\Sigma_1 = \{A \to B, B \to C, B \to A, A \to C, AD \to E\}.$$

3.2 Krok 2: wyliczamy nieredundantne pokrycie

W Σ_1 zależność funkcyjna $A \to C$ jest zbędna, ponieważ są tam już zależności $A \to B$ oraz $B \to C$, z których można wywnioskować $A \to C$. Nie ma innych zbędnych zależności, więc wyliczyliśmy pokrycie

$$\Sigma_2 = \{A \to B, B \to C, B \to A, AD \to E\}.$$

3.3 Krok 3: rozbijamy na partycje

Rozbijamy Σ_2 na zbiory $\Sigma_{3,i}$ takie, że każdy z nich zawiera relacje o identycznej lewej stronie.

$$\Sigma_{3,1} = \{A \to B\},$$
$$\Sigma_{3,2} = \{B \to C, B \to A\},$$
$$\Sigma_{3,3} = \{AD \to E\}.$$

[1] W związku z tym, sprowadzenie do EKNF **będzie wymagane w projekcie zaliczeniowym!**

[2] Oryginalny artykuł Bernsteina z 1976 roku dostępny jest pod adresem `https://www.comp.nus.edu.sg/~lingtw/papers/bernstein.pdf`. Uwaga — artykuł mówi o sprawdzaniu do 3NF, nie do EKNF, ponieważ w momencie jego publikacji EKNF nie była jeszcze znana — ale jak się okazuje sprowadza on nie tylko do 3NF, ale też do EKNF.

3.4 Krok 4: łączenie równoważnych grup

Ustalamy $J = \varnothing$. Dla każdej pary $\Sigma_{3,i}$ oraz $\Sigma_{3,j}$ o lewych stronach X oraz Y odpowiednio, sprawdzamy, czy w dopełnieniu $\left(\Sigma_{3,i} \cup \Sigma_{3,j}\right)^+$ znajdują się zależności $X \to Y$ oraz $Y \to X$, czyli czy X i Y są równoważne. Jeśli nie, pozostawiamy zbiory $\Sigma_{3,i}$ oraz $\Sigma_{3,j}$ bez modyfikacji. Jeśli tak, łączymy zbiory w nowy zbiór w następujący sposób:

1. $\Sigma^{'} = \Sigma_{3,i} \cup \Sigma_{3,j}$,
2. dodajemy $X \to Y$ oraz $Y \to X$ do zbioru J,
3. dla każdego podzbioru $X^{'} \subseteq X$, jeśli $Y \to X^{'}$ należy do $\Sigma^{'}$, to ją z niego usuwamy,
4. dla każdego podzbioru $Y^{'} \subseteq Y$, jeśli $X \to Y^{'}$ należy do $\Sigma^{'}$, to ją z niego usuwamy.

W przykładzie należy połączyć $\Sigma_{3,1} oraz \Sigma_{3,2}$, ponieważ jest w nich $A \to B$ oraz $B \to A$. Wobec tego mamy

$$\begin{aligned} J &= \{A \to B, B \to A\}, \\ \Sigma_{4,1\cup2} &= \{B \to C\}, \\ \Sigma_{4,3} &= \{AD \to E\}. \end{aligned}$$

3.5 Krok 5: usuwanie zależności tranzytywnych

Dla każdego zbioru $\Sigma_{4,i}$ z poprzedniego kroku szukamy zbioru $\Sigma_{5,i}$ takiego, że $\Sigma_{5,i} \subseteq \Sigma_{4,i}$, zachodzi $(\Sigma_{4,i} \cup J)^+ = (\Sigma_{5,i} \cup J)^+$, który jest minimalny w sensie zawierania, czyli nie istnieje jego podzbiór właściwy F spełniający $(\Sigma_{4,i} \cup J)^+ = (F \cup J)^+$. Efektywnie, powoduje to wykreślenie ze zbiorów z poprzedniego kroku zależności, które spowodowałyby pojawienie się tranzytywnych zależności funkcyjnych. Po wykonaniu wykreślenia, do każdego zbioru $\Sigma_{5,i}$ dopisujemy odpowiednie elementy ze zbioru J.

W naszym przykładzie nie było zależności tranzytywnych, więc dodajemy elementy z J ponownie do każdego zbioru.

$$\begin{aligned} \Sigma_{5,1\cup2} &= \{A \to B, B \to A, B \to C\}, \\ \Sigma_{5,3} &= \{AD \to E\}. \end{aligned}$$

3.6 Krok 6: konstruujemy relacje

Dla każdego zbioru zbieramy wszystkie atrybuty występujące w zależnościach funkcyjnych. Atrybuty te określają relacje. Zbiory atrybutów po lewych stronach zależności funkcyjnych, są w nich kluczami kandydującymi.

Ze zbioru $\Sigma_{5,1\cup2} = \{A \to B, B \to A, B \to C\}$ tworzymy relację $R_1(\underline{A}, B, C)$ lub $R_1(A, \underline{B}, C)$. Ponieważ w oryginalnej relacji A było kluczem i A występuje w drugiej tabeli, to ma sens wybranie $R_1(\underline{A}, B, C)$. Ze zbioru $\Sigma_{5,3} = \{AD \to E\}$ tworzymy relację $R_2(\underline{A}, \underline{D}, E)$. W drugiej tabeli jest tylko jeden klucz kandydujący, więc musi być kluczem głównym.

3.7 Przydział zależności funkcyjnych

Po podzieleniu relacji rozdzielamy pomiędzy nie zależności funkcyjne. Do każdej relacji przyporządkowujemy wszystkie zależności funkcyjne, których suma atrybutów z lewej i prawej strony zawiera się w atrybutach relacji. Jeśli jakaś zależność funkcyjna trafia do dwóch tabel jednocześnie, warto rozważyć dodanie klucza obcego pomiędzy tabelami.

3.8 Podsumowanie normalizacji do EKNF

Zauważmy, że wychodziliśmy z założenia, że mamy relację $R(\underline{A}, B, C, \underline{D}, E)$ ze zbiorem zależności funkcyjnych $\Sigma = \{A \to B, B \to C, B \to A, A \to C, ABD \to E\}$.

Algorytm pozwolił nam na rozbicie jej na dwie relacje $R_1(\underline{A}, B, C)$ oraz $R_2(\underline{A}, \underline{D}, E)$ z odpowiadającymi im zależnościami funkcyjnymi $\Sigma_1 = \{A \to B, B \to A, B \to C\}$ oraz $\Sigma_2 = \{AD \to E\}$. Z konstrukcji wiemy, że jest to postać EKNF. Lewa strona każdej nietrywialnej zależności funkcyjnej z tych zbiorów jest nadkluczem, więc tabele te są również w postaci BCNF. Takie rozbicie bazy zapewni, że unikniemy anomalii.

4 Przykład relacji w EKNF, która nie jest BCNF

Rozważmy relację $R(\underline{A}, \underline{B}, C)$ ze zbiorem zależności $\Sigma = \{AB \to C, C \to B\}$. Zastosowanie algorytmu Bernsteina prowadzi do relacji $R_1(\underline{A}, \underline{B}, C)$ identycznej z relacją R oraz $R_2(B, \underline{C})$. Ponieważ zależność funkcyjna $C \to B$ powinna być spełniona w obu relacjach, (C, B) w relacji R_1 powinno być kluczem obcym wskazującym na klucz unikatowy (C, B) w R_2. Tak stworzone tabele R_1 i R_2 są w EKNF, ale R_1 nie jest (i nie może być) w BCNF, ponieważ zależność funkcyjna $C \to B$ w R_1 nie ma nadklucza po swojej lewej stronie. Jeśli nie podjęlibyśmy próby normalizacji do EKNF, mogłoby to prowadzić do niespójności w danych. Stworzenie pomocniczej tabeli R_2 i dodanie klucza obcego gwarantuje, że niespójność się nie pojawi.

Aby być może nieco skonkretyzować przykład, rozważmy relację

$$\text{Ulubiony}(\underline{\text{Uczeń}}, \underline{\text{Przedmiot}}, \text{Nauczyciel}).$$

Załóżmy, że każdy nauczyciel uczy jednego przedmiotu, czyli Nauczyciel $\to$ Przedmiot oraz każdy uczeń z danego przedmiotu na swojego ulubionego nauczyciela, czyli

$$\{\text{Uczeń}, \text{Przedmiot}\} \to \text{Nauczyciel}.$$

Jest to dokładnie taka sytuacja jak w przykładzie powyżej. Weźmy przykładowe dane.

Uczeń	Przedmiot	Nauczyciel
Adaś	Polski	Julia Słowacka
Albert	Fizyka	Nikola Kopernik
Albert	Niemiecki	Iza Newton
Adaś	Fizyka	Nikola Kopernik

Jeśli kluczem jest uczeń i przedmiot, baza zadba jedynie o to, aby nie było duplikatów wierszy, czyli aby dany uczeń nie miał dwóch ulubionych nauczycieli z tego samego przedmiotu, co przeczy definicji słowa „ulubiony". Niestety, wykonując aktualizację bazy, mogłoby dojść do problemu. Baza dopuszcza zmianę krotki (Adaś, Fizyka, Nikona Kopernik) na (Adaś, Niemiecki, Nikola Kopernik), ponieważ nie ma innej krotki zaczynającej się od (Adaś, Niemiecki). Niestety z powodu tej zmiany Nikola Kopernik będzie według bazy uczyć jednocześnie fizyki i niemieckiego, co przeczy zależności Nauczyciel → Przedmiot. Oznacza to, że taki projekt bazy pozwala na wystąpienie anomalii.

Dodanie drugiej tabeli Nauczyciele(Nauczyciel, Przedmiot) i powiązanie tych tabel kluczem obcym zagwarantuje, że taka anomalia nie wystąpi, ponieważ tabela Nauczyciele ma atrybut Nauczyciel jako klucz główny, nie pojawi się w niej duplikat. Klucz obcy z kolei nie pozwoli zmienić wiersza w tabli Ulubiony tak, aby był niezgodny z tabelą Nauczyciele — co oznacza, że anomalia nie jest już możliwa.