Bazy danych

4. Normalizacja relacyjnych baz danych

P. F. Góra

http://th-www.if.uj.edu.pl/zfs/gora/

2020

Pierwsza postać normalna

Tabela jest w pierwszej postaci normalnej (1PN), jeżeli

1. Tabela posiada klucz.
2. Wszystkie składowe krotek są atomowe.

Można powiedzieć, że pierwsza postać normalna jest warunkiem tego, żeby w ogóle można było mówić o sytemie *relacyjnym*. Warunek posiadania klucza jest równoważny temu, że *tabela jest zbiorem krotek*.

Atomowość danych

Atomowość danych oznacza, że składowych krotek nie można podzielić. Warunek atomowości uniemożliwia to, żeby składowymi krotek były złożone struktury danych, takie jak tablice, listy itp.

W zasadzie wymóg atomowości nakazuje dzielić też atrybuty, które *można podzielić*, na części logicznie niepodzielne. Na przykład zamiast atrybutu "`Imię i Nazwisko`", powinniśmy mieć dwa atrybuty: "`Imię`", "`Nazwisko`". Można sobie jednak wyobrazić sytuacje, w których taki podział byłby niepotrzebny lub niewskazany*. O ile zatem pierwsza postać normalna z całą pewnością wyklucza złożone struktury danych, o tyle interpretacja pojęcia "można podzielić" może niekiedy zależeć od natury samych danych, które reprezentować ma konstruowana przez nas baza danych.

*Rozważmy na przykład nazwy osobowe z Chin czy Korei.

Anomalie baz danych

- Redundancja — ta sama informacja jest niepotrzebnie przechowywana w kilku krotkach.
- Anomalia modyfikacji — informacja zostanie zmodyfikowana w pewnych krotkach, a w innych nie. Która informacja jest wówczas prawdziwa?
- Anomalia usuwania — usuwanie części informacji powoduje utratę innej informacji, której nie chcielibyśmy stracić.
- Anomalia dołączania — wprowadzenie pewnej informacji jest możliwe tylko wtedy, gdy jednocześnie wprowadzamy jakąś inną informację, która może być obecnie niedostępna.

Celem normalizacji baz danych jest unikanie powyższych anomalii.

Druga postać normalna

Tabela jest w drugiej postaci normalnej (2PN), jeżeli

1. Tabela jest 1PN.
2. Wszystkie atrybuty niekluczowe zależą funkcyjnie od pełnego klucza.

Atrybuty niekluczowe mają zależeć od *pełnego* klucza, a nie od jego podzbioru właściwego (który nie musi być kluczem!).

Wszystkie tabele 1PN, które mają klucze jednokolumnowe, są automatycznie 2PN.

Przykład

Załóżmy, że zależności funkcyjne pomiędzy pewnymi atrybutami mają postać $A, B \to C$, $A \to D$. Poniższa tabela (podkreślenia oznaczają klucz)

$\underline{A}$	$\underline{B}$	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_1
a_1	b_3	c_3	d_1
a_2	b_1	c_4	d_2

nie jest 2PN, gdyż atrybut D zależy *tylko* od atrybutu A, a więc od części klucza, nie od całego klucza.

Schemat tabeli ze strony 6 nie chroni przed wymienionymi wyżej anomaliami:

- Nie można wprowadzić informacji o tym, że $a_3 \to d_3$, nie wprowadzając jednocześnie informacji, że $(a_3, b_\bullet) \to c_\bullet$ (anomalia dołączania).
- Usunięcie informacji o tym, że $a_2 \to d_2$ wymaga jednoczesnego usunięcia informacji, iż $(a_2, b_1) \to c_4$ (anomalia usuwania).
- Wartość d_1 przechowywana jest niepotrzebnie w trzech różnych krotkach (redundancja).

Po rozbiciu powyższej tabeli na dwie tabele będące w 2PN, anomalie dołączania i usuwania nie występują, a wartość d_1 przechowywana jest tylko w *jednej* krotce.

$\underline{A}$	$\underline{B}$	C
a_1	b_1	c_1
a_1	b_2	c_2
a_1	b_3	c_3
a_2	b_1	c_4

$\underline{A}$	D
a_1	d_1
a_2	d_2

Proszę pomyśleć, że w początkowym przykładzie d_1 mogłoby występować nie w 3, ale w 300 lub w 3000 krotek.

Bezstratne złączenie (ang. *lossless join*)

Normalizację baz danych (powyżej 1PN) przeprowadza się dzieląc tabele "wertykalnie" na tabele potomne. Tabele te jednak **muszą** pozwalać na pełne odtworzenie wyjściowej informacji po dokonaniu naturalnego złączenia. Niedopuszczalne jest także, aby naturalne złączenia kreowały informację fałszywą.

Dekompozycję tabeli R na tabele R_1, R_2, ..., R_n nazywamy dekompozycją bezstratnego złączenia (ze względu na pewien zbiór zależności funkcyjnych), jeśli naturalne złączenie R_1, R_2, ..., R_n jest równe tabeli R.

Dekompozycja tabeli R na dwie tabele R_1, R_2 jest dekompozycją bezstratnego złączenia, jeśli spełniony jest jeden z dwu warunków (symbol $\rightarrow$ oznacza zależność funkcyjną):

$$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$$

lub

$$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$$

Innymi słowy, wspólna część atrybutów R_1, R_2 musi zawierać klucz kandydujący R_1 lub R_2.

Twierdzenie Heatha

Tabelę R o atrybutach X, Y, Z, spełniającą zależność funkcyjną $X \to Y$ można bezstratnie zdekomponować na wyniki rzutowania $R_1 = \pi_{XY}(R)$ oraz $R_2 = \pi_{XZ}(R)$.

Przykład

Rozpatrzmy tabelę, spełniającą zależność funkcyjną ID $\rightarrow$ Imię:

Tabela S		
ID	Imię	Przedmiot
17	Alicja	Bazy danych
17	Alicja	Teoria języków formalnych
17	Alicja	SPK
112	Bogdan	Bazy danych
112	Bogdan	Systemy czasu rzeczywistego
119	Czesław	Teoria języków formalnych
119	Czesław	SPK

Tabela ta zawiera *zależności wielowartościowe*, do których jeszcze wrócimy.

Tabelę tę można bezstratnie podzielić na następujące dwie tabele:

Tabela S_1	
ID	Imię
17	Alicja
112	Bogdan
119	Czesław

Tabela S_1	
ID	Przedmiot
17	Bazy danych
17	Teoria języków formalnych
17	SPK
112	Bazy danych
112	Systemy czasu rzeczywistego
119	Teoria języków formalnych
119	SPK

Zachodzi $S = S_1 \bowtie S_2$. Zauważmy, że taka dekompozycja pozwala uniknąć anomalii redundancji, a przede wszystkim anomalii modyfikacji: Gdyby student o ID $= 17$ zmienił imię, zmianę tę trzeba by wprowadzić w trzech miejscach przed dekompozycją, ale tylko w jednym po dekompozycji.

Trzecia postać normalna

Tabela jest w trzeciej postaci normalnej (3PN), jeżeli

1. Tabela jest 2PN.
2. Dla wszystkich atrybutów tabeli zachodzi:
   Jeżeli $A_1, A_2, \ldots, A_n \to A_m$, to albo $\{A_1, A_2, \ldots, A_n\}$ jest nadkluczem, albo A_m jest elementem innego klucza.

Trzecia postać normalna jest postacią najczęściej występującą w zastosowaniach praktycznych. Druga część warunku definicyjnego (“albo A_m jest elementem innego klucza”) ma znaczenie tylko wówczas, gdy w tabeli występują zależności cykliczne (lub częściowe zależności cykliczne).

Zależności przechodnie

Jeżeli w tabeli nie występują zależności cykliczne, powiada się, że 3PN zakazuje wsytępowania zależności przechodnich. Istotnie, przyjmijmy, że spełnione są zależności funkcyjne $A \to B$, $B \to C$, $A \to C$; ostatnia z tych zależności wynika z dwu pierwszych na zasadzie przechodniości. Następująca tabela

$\underline{A}$	B	C

jest 2PN, ale **nie** jest 3PN, gdyż zachodzi zależność $B \to C$, zaś atrybut B nie jest nadkluczem. Sprowadzenie do 3PN oznacza rozbicie powyższej tabeli na dwie tabele:

$\underline{A}$	B

$\underline{B}$	C

Przykład

Niech zależności funkcyjne będą takie, jak na poprzednim ekranie. Rozważmy następującą instancję pierwszej tabeli:

$\underline{A}$	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_3	b_1	c_1
a_4	b_2	c_2

Występują anomalie dołączania (nie można wprowadzić $a_5 \to b_3$ bez jednoczesnego wprowadzenia $b_3 \to c_3$), usuwania (nie można usunąć $b_2 \to c_2$ bez jednoczesnego usunięcia $a_4 \to b_2$) oraz redundancja (wielkość c_1 przechowywana jest w trzech krotkach). Po sprowadzeniu do 3PN

$\underline{A}$	B
a_1	b_1
a_2	b_1
a_3	b_1
a_4	b_2

$\underline{B}$	C
b_1	c_1
b_2	c_2

anomalie te znikają.

Cykliczne zależności funkcyjne

Przykładem cyklicznych zależności funkcyjnych jest $A \to B, B \to C, C \to A$. W takiej sytuacji atrybuty A, B, C są sobie równoważne — określenie wartości jednego z nich, jednoznacznie ustala wartość dwu pozostałych. **Każda** z trzech tabel

$\underline{A}$	B	C

A	$\underline{B}$	C

A	B	$\underline{C}$

jest 3PN, gdyż co prawda występują zależności przechodnie, ale atrybuty niekluczowe są elementami *innych kluczy* (de facto *są* innymi kluczami).

Uwaga! Przy cyklicznych zależnościach funkcyjnych jak poprzednio, tabele rozbite w taki sposób:

$\underline{A}$	B

$\underline{B}$	C

$\underline{C}$	A

technicznie rzecz biorąc także są 3PN, a nie zawierają zależności przechodnich. Jednak taki projekt nie zapobiega redundancji, przeciwnie, wymusza ją, gdyż każda wartość każdego z atrybutów A, B, C jest przechowywana dwa razy. Projekty z poprzedniej strony są z tego względu zdecydowanie lepsze.

Procedura postępowania

1. Dany jest zbiór atrybutów, które chcemy reprezentować, i zbiór zależności funkcyjnych pomiędzy atrybutami.
2. Znajdujemy bazę minimalną zbioru zależności funkcyjnych.
3. Mając bazę minimalną, sumujemy zależności funkcyjne o takich samych lewych stronach i dla każdej wysumowanej zależności tworzymy tabelę z odpowiednią lewą stroną zależności jako kluczem.

Taki sposób postępowania prowadzi do projektu bazy, w której tabele są 3PN.

Procedura szukania bazy minimalnej

1. Każdy atrybut musi występować z lewej lub z prawej strony jednej zależności funkcyjnej w zbiorze.
2. Jeśli jakaś zależność funkcyjna jest włączona do zbioru, nie wszystkie zależności funkcyjne potrzebne do jej wyprowadzenia mogą występować w tym zbiorze.
3. Jeśli jakaś zależność funkcyjna zostaje wyłączona ze zbioru, zależności funkcyjne potrzebne do jej wyprowadzenia muszą zostać doń dołączone.

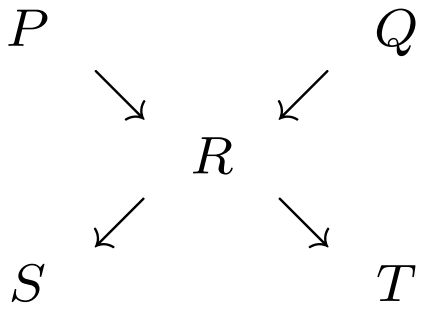
Jeżeli w zbiorze zależności funkcyjnych występują (częściowe) cykle, może istnieć więcej niż jedna baza minimalna.

Przykład 1

Zaprojektujmy tabele będące (co najmniej) w trzeciej postaci normalnej, spełniające zależności funkcyjne

$$P, Q \rightarrow R, S, T \tag{1a}$$
$$R \rightarrow S, T \tag{1b}$$

Czasami wygodnie narysować jest graf skierowany, obrazujący analizowane zależności funkcyjne. Dla zależności (1) graf taki ma postać



Zależności $P, Q \to S, T$ można usunąć, gdyż na mocy przechodniości wynikają one z zalezności $P, Q \to R$, $R \to S, T$. Ostatecznie tabele mają postać

$$T_1(\underline{P}, \underline{Q}, R) \qquad T_2(\underline{R}, S, T)\,,$$

gdzie podkreślone atrybuty stanowią klucz.

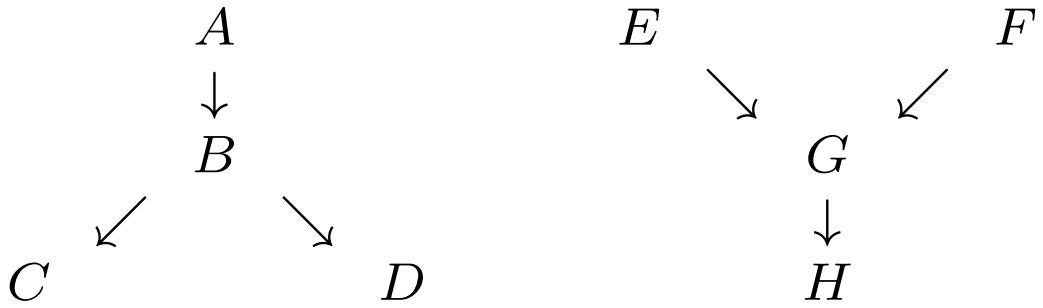
Przykład 2

Zaprojektujmy tabele będące (co najmniej) w trzeciej postaci normalnej, spełniające zależności funkcyjne

$$A \rightarrow B, C \tag{2a}$$
$$B \rightarrow C, D \tag{2b}$$
$$E, F \rightarrow G, H \tag{2c}$$
$$G \rightarrow H \tag{2d}$$
$$A, E \rightarrow D \tag{2e}$$

Zależności $A \rightarrow C$ oraz $E, F \rightarrow H$ eliminujemy na mocy przechodniości. Na pierwszy rzut oka pewną trudność może sprawiać ostatnia z zależności (2). Wiemy jednak, że na mocy przechodniości $A \rightarrow D$, a *poprzednik każdej zależności funkcyjnej można rozszerzyć*. Zatem zależność $A, E \rightarrow D$ wynika z pozostałych zależności funkcyjnych (2) i można ją pominąć.

Jako graf zależności funkcyjnych (2) otrzymujemy



zaś jako tabele

$$T_1(\underline{A}, B)\,, \quad T_2(\underline{B}, C, D)\,, \quad T_3(\underline{E}, \underline{F}, G)\,, \quad T_4(\underline{G}, H)\,.$$

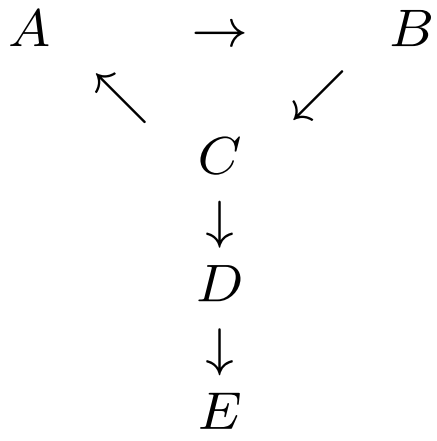
Przykład 3

Zaprojektujmy tabele będące (co najmniej) w trzeciej postaci normalnej, spełniające zależności funkcyjne

$$A \rightarrow B, D, E \tag{3a}$$
$$B \rightarrow C, D, E \tag{3b}$$
$$C \rightarrow A, D, E \tag{3c}$$
$$D \rightarrow E \tag{3d}$$

Widać, że atrybuty D, E zależą od każdego z atrybutów A, B, C, te zaś trzy tworzą cykl. Zależność $C \rightarrow E$ można wyeliminowac na mocy przechodniości.

Grafem zależności funkcyjnych jest



natomiast tabele mają postać

$$T_1(\underline{A}, B, C)\,, \quad T_2(\underline{C}, D)\,, \quad T_3(\underline{D}, E)$$

W tabeli T_1 kluczem mógłby być dowolny spośród atrybutów A, B, C. Podobnie w tabeli T_2 kluczem mógłby być któryś z atrybutów A, B, nie zaś koniecznie C.

Postać normalna Boyce'a-Codda

Tabela jest w postaci normalnej Boyce'a-Codda (BCNF, PNBC), jeżeli

1. Tabela jest 2PN.
2. Dla każdej zależności nietrywialnej, jeżeli $A_1, A_2, \ldots, A_n \to A_m$, to zbiór $\{A_1, A_2, \ldots, A_n\}$ jest nadkluczem.

PNBC jest "silniejszą" wersją 3PN. Każda tabela będąca PNBC jest także 3PN, ale wynikanie odwrotne nie musi zachodzić.

Do czego dążymy?

Przeprowadzając normalizację bazy danych, staramy się **jednocześnie** spełnić trzy warunki:

1. Bezstratne złączenie.
2. Zachowanie wszystkich zależności funkcyjnych.
3. PNBC.

Czy zawsze jest to możliwe?

Rozważmy tabelę

A	B	C

z następującymi zależnościami funkcyjnymi:

$$C \to A$$
$$A, B \to C$$

(Na przykład: A — nazwa banku, B — nazwisko klienta uprawnionego do *personal banking*, C — nazwisko bankiera. Pierwsza zależność mówi, że każdy bankier pracuje w określonym banku, druga, że każdego uprawnionego klienta w danym banku obsługuje określony bankier. Ale klient może mieć konta w więcej niż jednym banku. . .)

Powyższa tabela nie jest PNBC, gdyż C nie może być nadkluczem. Jednak żadne rozbicie na dwie tabele dwuatrybutowe albo nie zachowuje kompletu zależności funkcyjnych, albo nie jest dekompozycją bezstratnego złączenia, albo jedno i drugie. W tej sytuacji zadowalamy się niższą postacią normalną. Zachowanie kompletu zależności funkcyjnych oraz bezstratność złączeń muszą mieć priorytet! Nie każdą tabelę daje się znormalizować do PNBC.

Zależności wielowartościowe

Przypuśćmy, że kolumny (atrybuty) pewnej tabeli możemy podzielić na trzy wzajemnie rozłączne podzbiory: X, Y, Z. Wybierzmy teraz pewną wartość $x_c \in X$ *faktycznie występującą w tabeli*. Mówimy, że zbiory X, Y związane są zależnością wielowartościową, co zapisujemy $X \twoheadrightarrow Y$, jeżeli po utworzeniu zbioru wszystkich kombinacji $x_c y z$ *faktycznie występujących w tabeli* stwierdzamy, że x_c jest stowarzyszone z tymi samymi wartościami y bez względu na wartości z.

Oznacza to, że zbiory Y, Z są niezależne, czyli nie są ze sobą powiązane bezpośrednio, a co najwyżej poprzez zbiór X.

Mówimy, że zależność wielowartościowa $X \twoheadrightarrow Y$ jest trywialna, jeżeli $Y \subset X$ lub $X \cup Y$ stanowi zbiór wszystkich atrybutów tabeli.

Każda zależność funkcyjna jest zarazem zależnością wielowartościową (być może trywialną), jednak zależność wielowartościowa odnosi się do faktycznej *instancji* tabeli.

Przykład tabeli z zależnością wielowartościową już się pojawił na stronie 12. Redundancja i niebezpieczeństwo anomalii modyfikacji są oczywiste.

Czwarta postać normalna

Tabela jest w czwartej postaci normalnej (4PN), jeżeli

1. Tabela jest 3PN.
2. Dla każdej nietrywialnej zależności wielowartościowej $A \twoheadrightarrow B$, A jest nadkluczem.

Każda tabela PNBC, która nie zawiera nietrywialnych zależności wielowartościowych, jest automatycznie w 4PN.

W przykładzie ze strony 12 podaną tabelę należy rozbić na dwie mniejsze:

ID	Imię
...	...

ID	Przedmiot
...	...

Pewne badania wskazują, że około 20% praktycznie działających systemów bazodanowych zawiera tabele nie spełniające czwartej postaci normalnej, choć spełniające wymagania "niższych" postaci[†]. Zapewnienie spełniania 4PN jest więc ważnym, choć najwyraźniej niedocenianym problemem praktycznym. Jest on ważny w szczególności dla tabel pomostowych, opisujących związki wieloargumentowe.

[†]Można co prawda zastanawiać się, czy przynajmniej część z tych tabel jest w 1PN, w szczególności, czy posiadają *klucz*. Jednak wiele praktycznie działających RDMS dopuszcza duplikaty.

Normalizacja a wydajność

Normalizacja baz danych dostarcza **mechanizmu** pozwalającego unikać anomalii. Ma to jednak swoją cenę: Dostęp do danych w bazie znormalizowanej może być wolniejszy, gdyż RDBMS musi wykonywać złączenia. W praktyce czas wykonania zapytania ze złączeniem lub bez może zależeć od *fizycznego* stanu instancji bazy (np. fragmentacja plików dyskowych itp.).

Dlatego w wielkich bazach danych zoptymalizowanych na odczyt (na przykład w hurtowniach danych) często rezygnuje się z wyższych postaci normalnych, przechowując dane w tabelach 1PN. *To* także ma swoją cenę: Wprowadzając dane do takich tabel lub modyfikując istniejące dane należy dołożyć szczególnej staranności, aby nie dopuścić do anomalii usuwania lub dołączania, a szczególnie do anomalii modyfikacji (redundancja jest w tego typu bazy niejako wbudowana).