

1. Wprowadzenie

Bezpośrednie przekształcenie modelu bazy danych, przygotowanego zgodnie z ideą zorientowanego obiektowo podejścia do projektowania (tj. z wykorzystaniem języka definiowania obiektów – ODL, od ang. *object definition language*), do postaci modelu relacyjnego, może prowadzić do pewnych problemów podczas eksploatacji bazy danych [11]. Konkretnie, niewłaściwe zaprojektowanie schematów relacji może być przyczyną redundancji danych, ich niespójności i anomalii podczas ich aktualizowania. Rozważmy przykładową relację *PRACOWNICY*, której schemat przedstawiono na rys. 1 [1].

PRACOWNICY

<u>Id_prac</u>	Nazwisko	Adres	Numer_działu	Nazwa_działu
----------------	----------	-------	--------------	--------------

Rys. 1. Schemat relacji *PRACOWNICY*.

W relacji *PRACOWNICY* z rys. 1 we wszystkich krotkach opisujących pracowników zatrudnionych w danym dziale, wielokrotnie będzie powtarzana nazwa tego działu – jest to typowy przykład niepotrzebnego dublowania danych pamiętanych w bazie danych. Co więcej, podczas eksploatacji bazy danych zawierającej tak zaprojektowaną relację *PRACOWNICY*, mogą wystąpić następujące anomalie:

- *Anomalia wstawiania*: wprowadzenie pracownika zatrudnionego np. w dziale 5 wymaga zadbania o spójność nazwy działu z wartościami w innych krotkach; nie można wprowadzić informacji o dziale, który nie zatrudnia jeszcze pracowników.
- *Anomalia usuwania*: usuwając krotkę z ostatnim pracownikiem danego działu, tracimy informację, że taki dział w ogóle istnieje.
- *Anomalia modyfikacji*: zmiana nazwy działu wymaga aktualizacji krotek wszystkich pracowników zatrudnionych w tym dziale w celu zachowania spójności.

Wszystkie te zjawiska uniemożliwiają lub co najmniej bardzo utrudniają poprawną eksploatację bazy danych. Ich wystąpienie najczęściej pociąga za sobą konieczność przeprojektowania bazy danych. Jest to zawsze bardzo uciążliwe, gdyż wiąże się z koniecznością zmiany oprogramowania użytkowego. Dlatego każdy, kto przystępuje do projektowania bazy danych, powinien znać podstawy ich projektowania.

W ramach projektowania schematów bazy danych najistotniejszą czynnością jest proces *normalizacji relacji*, czyli doprowadzanie relacji do odpowiedniej *postaci normalnej* (ang. *Normal Form – NF*). W uproszczeniu, proces normalizacji relacji można traktować jako proces, podczas którego schematy relacji posiadające pewne niepożądane cechy są dekomponowane na mniejsze schematy o pożądanych własnościach.

Dekompozycja sprowadza się do podziału atrybutów relacji *R* między dwa schematy nowych relacji. Reguła dekompozycji obejmuje również podział krotek relacji wejściowej *R* między nowe relacje, dzięki operacji projekcji krotek *R*. Relację *R* o schemacie $\{A_1, A_2, \dots, A_n\}$ dekomponujemy między dwie relacje *S* i *T* o schematach odpowiednio $\{B_1, B_2, \dots, B_m\}$ oraz $\{C_1, C_2, \dots, C_k\}$ według następujących zasad [11]:

1. $\{A_1, A_2, \dots, A_n\} = \{B_1, B_2, \dots, B_m\} \cup \{C_1, C_2, \dots, C_k\}$.
2. Krotki relacji *S* i *T* powstają przez projekcję wszystkich krotek relacji *R* na zbiór atrybutów odpowiednio $\{B_1, B_2, \dots, B_m\}$ i $\{C_1, C_2, \dots, C_k\}$.

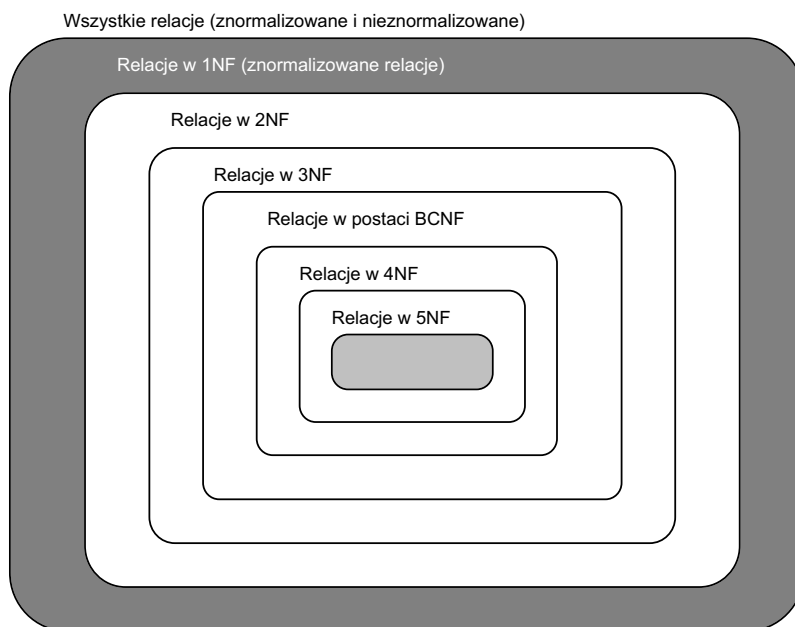
Celem normalizacji jest minimalizacja fizycznego rozmiaru bazy danych oraz uniknięcie anomalii związanych z wstawianiem, aktualizacją i usuwaniem krotek. Proces normalizacji musi posiadać trzy własności:

- żaden atrybut nie może zostać zagubiony w trakcie procesu normalizacji,
- dekompozycja relacji nie może prowadzić do utraty informacji,
- wszystkie zależności funkcyjne muszą być reprezentowane w pojedynczych schematach relacji.

W literaturze podano wiele definicji postaci normalnych (zob. rys. 2). Pierwsze trzy (1NF, 2NF, 3NF) zdefiniował Codd w pracy [3]. Jak wykazuje rys. 2 wszystkie relacje znormalizowane są w postaci 1NF, niektóre z nich są też w postaci 2NF, a część tych 2NF występuje w 3NF. Definicje Codd opierały się na tym, że postać 2NF jest bardziej pożądana niż 1NF, 3NF zaś z kolei bardziej niż 2NF. Projektant bazy danych powinien dążyć do skonstruowania projektu zawierającego relacje w 3NF, a nie tylko w 2NF czy 1NF.

W pracy [3] Codd wprowadził pojęcie procedury normalizacji, dzięki której relacja będąca w pewnej postaci normalnej, powiedzmy 2NF, może zostać przekształcona na zbiór relacji w bardziej pożądanej postaci, powiedzmy 3NF. Pierwotnie procedura ta była zdefiniowana tylko do poziomu 3NF, jednak później została rozszerzona do 5NF. Procedurę tę można określić jako kolejne dekompozycje danego zbioru relacji do innej, bardziej pożądanej postaci. Zauważmy, że jest ona odwracalna. Oznacza to, że zawsze można wziąć wynik wykonania tej procedury (powiedzmy zestaw relacji w 3NF) i przekształcić go z powrotem w relację wejściową (np. w 2NF). Odwracalność jest ważna, ponieważ gwarantuje, że nie ma utraty informacji w procesie normalizacji.

Okazało się, że pierwotna definicja Codd trzeciej postaci normalnej miała pewne niedostatki. Poprawioną i mocniejszą definicję podali Boyce i Codd w [4]. Była ona mocniejsza w tym sensie, że każda relacja, będąca zgodnie z nową definicją w 3NF, była na pewno w postaci 3NF według starej definicji, natomiast mogły pojawiać się relacje w starej postaci 3NF, nie spełniające warunków nowej definicji. Tę nową postać 3NF nazywa się zwykle postacią normalną Boyce'a/Codda (BCNF) po to, aby odróżnić ją od starej postaci.



Rys. 2. Poziomy normalizacji [6].

Następnie Fagin [8, 9] zdefiniował nową czwartą postać normalną – 4NF, a następnie jeszcze jedną, którą nazwał PJ/NF (ang. *projection/join normal form*), zwaną niekiedy piątą postacią normalną – 5NF.

W kolejnych rozdziałach przedstawimy i omówimy pięć kolejnych postaci normalnych relacji oraz dokonamy analizy procedury normalizacyjnej.

2. Pierwsza postać normalna relacji

Pierwsza postać normalna jest immanentną cechą każdej relacji, gdyż wymagania tej postaci są zawarte w definicji relacji. Proces normalizacji polega w tym przypadku na doprowadzeniu zbioru danych do postaci relacji (zbiór ten nazywa się potocznie *relacją nieznormalizowaną*, choć ściśle rzecz biorąc nie jest on relacją). Przykład relacji nieznormalizowanej przedstawiono na rys. 3.

Definicja pierwszej postaci normalnej [1, 5-7, 10-12]

Relacja jest w *pierwszej postaci normalnej*, jeśli każda wartość atrybutu w każdej krotce tej relacji jest wartością elementarną, czyli nierozkładalną (atomową).

Z definicji pierwszej postaci normalnej relacji wynika, że każdemu elementowi relacji znajdującemu się na przecięciu dowolnej krotki i dowolnego atrybutu odpowiada pojedyncza wartość, a nie zbiór wartości. Innymi słowy, pierwsza postać normalna zabrania definiowania złożonych atrybutów, które są wielowartościowe. Relacje, które dopuszczają definiowanie takich złożonych atrybutów nazywamy *relacjami zagnieżdżonymi* (ang. *nested relations*) – w ogólności, w relacjach zagnieżdżonych krotka może zawierać inną relację. Należy zauważyć, że koncepcja relacji zagnieżdżonych wykracza poza ramy klasycznego relacyjnego modelu danych.

<i>Nr_ zamówienia</i>	<i>Id_ dostawcy</i>	<i>Nazwa_ dostawcy</i>	<i>Adres_ dostawcy</i>	<i>Id_ części</i>	<i>Nazwa_ części</i>	<i>Ilość</i>	<i>Magazyn</i>	<i>Adres_ magazynu</i>
001	300	Nissan	Liege	53	Pompa	100	5	Warszawa
				57	Filtr	50	5	Warszawa
				59	Błotnik	500	6	Szczecin
002	400	HONDA	Glasgow	54	Pompa	500	5	Warszawa
				32	Koło	100	6	Szczecin
003	500	TOYOTA	Tokyo	88	Silnik	15	7	Poznań
004	600	HONDA	London	59	Błotnik	400	6	Szczecin
				21	Lampa	50	7	Poznań

Rys. 3. Relacja nieznormalizowana [1].

Rozważmy zestaw danych w postaci tabelarycznej przedstawiony na rys. 3. Zgodnie z podaną definicją zestaw ten nie jest relacją, gdyż istnieją w nim krotki, w których jednemu atrybutowi odpowiada zbiór wartości. Jak wspomnieliśmy, potocznie mówi się w tym przypadku o relacji nieznormalizowanej. Relację ZAMÓWIENIA w pierwszej postaci normalnej odpowiadającą relacji nieznormalizowanej z rys. 3 przedstawiono na rys. 4. W tej relacji każda wartość atrybutu w każdej krotce jest wartością elementarną.

Scharakteryzujemy krótko dziedzinę rzeczywistości, która została zamodelowana w postaci tej relacji, ponieważ posłuży nam ona za przykład przy przedstawianiu tej i kilku dalszych postaci normalnych. Relacja ta zawiera informacje o zamówieniach części u dostawców i skierowaniu ich do odpowiednich magazynów. Jedno zamówienie może dotyczyć wielu części dostarczanych przez tego samego dostawcę. Oczywiście, różne zamówienia mogą dotyczyć tego samego dostawcy i tych samych części. Ta sama część może być dostarczana przez różnych dostawców. Zakładamy,

że te same części są składowane zawsze w tym samym magazynie. W zamówieniach jest podawany identyfikator dostawcy i nazwa dostawcy, gdyż różni dostawcy mogą nosić tę samą nazwę (Honda). Podobna jest sytuacja w odniesieniu do części.

Nr_ zamówienia	Id_ dostawcy	Nazwa_ dostawcy	Adres_ dostawcy	Id_ części	Nazwa_ części	Ilość	Magazyn	Adres_ magazynu
001	300	NISSAN	Liege	53	Pompa	100	5	Warszawa
001	300	NISSAN	Liege	57	Filtr	50	5	Warszawa
001	300	NISSAN	Liege	59	Błotnik	500	6	Szczecin
002	400	HONDA	Glasgow	54	Pompa	500	5	Warszawa
002	400	HONDA	Glasgow	32	Koło	100	6	Szczecin
003	500	TOYOTA	Tokyo	88	Silnik	15	7	Poznań
004	600	HONDA	London	59	Błotnik	400	6	Szczecin
004	600	HONDA	London	21	Lampa	50	7	Poznań

Rys. 4. Relacja ZAMÓWIENIA w pierwszej postaci normalnej [1].

Na przykładzie relacji ZAMÓWIENIA możemy zilustrować wady relacji w pierwszej postaci normalnej i wynikające z nich niekorzystne zjawiska występujące przy eksploatacji bazy danych, o których wspominaliśmy uprzednio. Zauważmy, że w relacji ZAMÓWIENIA występuje dublowanie się danych – na przykład, wielokrotnie jest pamiętany adres dostawcy. Liczba powtórzeń adresu dostawcy jest równa iloczynowi liczby skierowanych do niego zamówień i liczby różnych części figurujących na tych zamówieniach. Konsekwencją dublowania się danych jest niepotrzebne zajmowanie przestrzeni na dysku i przedłużony czas wykonywania operacji relacyjnych, a także możliwość wystąpienia niespójności danych, na przykład na skutek uaktualnienia adresu dostawcy w pewnych krotkach i pozostawienia starego adresu w pozostałych. Dalsze problemy wynikające z pierwszej postaci normalnej relacji ZAMÓWIENIA powstają przy jej aktualizowaniu. W tej relacji nie możemy zapamiętać adresu dostawcy, do którego nie złożono żadnego zamówienia, choć informacja taka może być bardzo przydatna. Podobnie, usuwając informację o zamówieniu skierowanym do danego dostawcy, możemy utracić informacje o jego adresie. Analogiczna sytuacja występuje w odniesieniu do magazynu i jego adresu.

Tak więc, relacje w pierwszej postaci normalnej nie gwarantują jeszcze wyeliminowania anomalii jakie mogą wystąpić podczas eksploatacji bazy danych. Dlatego należy doprowadzić relacje do kolejnych postaci normalnych, aby te zjawiska wyeliminować.

3. Druga postać normalna relacji

Aby przedstawić drugą postać normalną, musimy przedtem wprowadzić kilka pojęć podstawowych.

Każdy podzbiór SK atrybutów relacji R , taki że dla każdych dwóch krotek relacji prawdą jest, że $t_1 [SK] \neq t_2 [SK]$, jest nazywany *nadkluczem* (*superklucz*) (ang. *superkey*) relacji R . Każda relacja ma co najmniej jeden superklucz – schemat relacji. *Kluczem* K schematu relacji R nazywamy „minimalny” nadklucz, tzn. taki, że nie istnieje $K' \subset K$ będący nadkluczem schematu relacji R .

Jeżeli relacja zawiera więcej niż jeden klucz, to są one nazywane *kluczami potencjalnymi* (*kandydującymi*) (ang. *candidate key*). Jeden z nich może pełnić rolę *klucza głównego* (ang. *primary key*) relacji. Pozostałe są wtedy kluczami drugorzędnymi (wtórnymi). Wyróżniamy klucze proste i złożone. Klucz jest *kluczem prostym*, jeśli superklucz, o którym mowa w definicji powyżej, jest zbiorem jednoelementowym; w przeciwnym razie mówimy o *kluczu złożonym*.

Atrybut X w schemacie relacji R będziemy nazywać atrybutem wtórnym, jeżeli nie należy on do żadnego z kluczy schematu relacji R .

Przykładowo w relacji $ZAMÓWIENIA$ z rys. 4 jednym z potencjalnych kluczy jest para atrybutów $\{Nr_zamówienia, Id_części\}$. Zauważmy, że sam numer zamówienia nie jest kluczem, gdyż jedno zamówienie może dotyczyć wielu części. Podobnie identyfikator części nie jest kluczem, gdyż jedna część może być dostarczana przez wielu dostawców.

Innym ważnym pojęciem jest pojęcie zależności funkcyjnej, które możemy zdefiniować na dwa następujące sposoby.

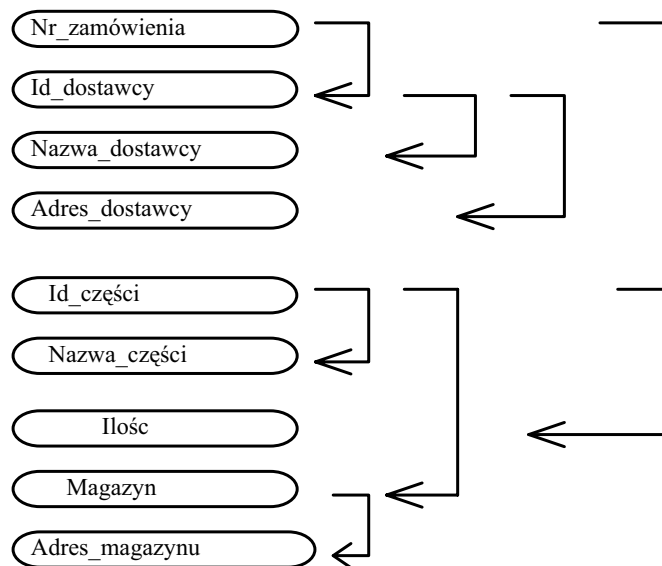
Definicja 1 zależności funkcyjnej – FD (od ang. *functional dependency – FD*)

Atrybut B relacji R jest funkcyjnie zależny od atrybutu A tej relacji, jeśli zawsze każdej wartości a atrybutu A odpowiada nie więcej niż jedna wartość b atrybutu B . Stwierdzenie, że B jest funkcyjnie zależne od A równoważne stwierdzeniu, że A identyfikuje (wyznacza) B .

Definicja 2 zależności funkcyjnej.

Dla danej relacji R ; w której X i Y są podzbiorami atrybutów schematu tej relacji, X wyznacza funkcyjnie Y , lub Y jest funkcyjnie zależne od X , co zapisujemy $X \rightarrow Y$, wtedy i tylko wtedy, jeżeli dla dwóch dowolnych krotek t_1, t_2 takich, że $t_1[X] = t_2[X]$ zachodzi zawsze $t_1[Y] = t_2[Y]$.

Należy podkreślić, że zależność funkcyjna między dwoma atrybutami A i B nie jest związana z przypadkowym układem ich wartości, lecz wynika z charakteru zależności między tymi atrybutami. Innymi słowy, zależności funkcyjne są z natury rzeczy semantyczne – dotyczą tego, co oznaczają dane. Zależności funkcyjne między atrybutami relacji $ZAMÓWIENIA$ przedstawiono na rys. 5.



Rys. 5. Diagram zależności funkcyjnych w relacji $ZAMÓWIENIA$ [1].

W tej relacji atrybut $Id_dostawcy$ jest funkcyjnie zależny od atrybutu $Nr_zamówienia$, gdyż jedno zamówienie jest kierowane tylko do jednego dostawcy, a zatem numer zamówienia jednoznacznie określa identyfikator dostawcy. Odwrotna zależność nie jest prawdziwa, gdyż do jednego dostawcy może być skierowanych wiele różnych zamówień. Podobnie atrybuty $Nazwa_dostawcy$ i $Adres_dostawcy$ są funkcyjnie zależne od atrybutu $Id_dostawcy$, atrybuty $Nazwa_części$ i $Magazyn$ są funkcyjnie zależne od atrybutu $Id_części$, gdyż część o danym identyfikatorze może mieć tylko jedną nazwę i może być przechowywana w tylko jednym magazynie. Ponadto atrybut $Adres_magazynu$ jest funkcyjnie zależny od atrybutu $Magazyn$. Natomiast atrybut $Ilość$ jest funkcyjnie zależny od zbioru atrybutów $\{Nr_zamówienia, Id_części\}$,

ponieważ zależy od obu tych atrybutów jednocześnie, ale nie zależy od żadnego z nich z osobna. Innymi słowy, *Ilość* zamawianej części jest funkcyjnie zależna od kombinacji wartości atrybutów *Nr_zamówienia* i *Id_części*, ale nie jest funkcyjnie zależna od atrybutu *Nr_zamówienia*, gdyż w tym samym zamówieniu różne części mogą występować w różnych ilościach, ani nie jest funkcyjnie zależna od atrybutu *Id_części*, gdyż ta sama część może w różnych ilościach figurować na różnych zamówieniach.

Mając zdefiniowane pojęcie zależności funkcyjnej możemy teraz zdefiniować pojęcie *pełnej zależności funkcyjnej*.

Definicja pełnej zależności funkcyjnej

Zbiór atrybutów Y jest w pełni funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i nie istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$. Zbiór atrybutów Y jest częściowo funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$.

Warto zauważyć, że wszystkie zależności funkcyjne w relacji *ZAMÓWIENIA* przedstawione na rys. 5 są pełnymi zależnościami funkcyjnymi.

Wprowadzone powyżej pojęcia i definicje pełnej zależności funkcyjnej umożliwiają zdefiniowanie drugiej postaci normalnej (2NF).

Definicja 1 drugiej postaci normalnej – 2NF

Relacja R o danym schemacie jest w drugiej postaci normalnej (2NF), jeżeli żaden atrybut wtórny tej relacji nie jest częściowo funkcyjnie zależny od żadnego z kluczy relacji R .

Definicja 2 drugiej postaci normalnej – 2NF

Relacja R o danym schemacie jest w drugiej postaci normalnej (2NF), jeżeli każdy atrybut wtórny tej relacji jest w pełni funkcyjnie zależny od klucza podstawowego relacji R .

Jak łatwo zauważyć, relacja *ZAMÓWIENIA* nie jest w drugiej postaci normalnej. Zgodnie z założeniami przyjętymi wcześniej, kluczem tej relacji jest para atrybutów $\{Nr_zamówienia, Id_części\}$. Atrybuty *Id_dostawcy*, *Nazwa_dostawcy*, *Adres_dostawcy*, *Nazwa_części*, *Magazyn* nie są w pełni funkcyjnie zależne od klucza. Atrybuty *Id_dostawcy*, *Nazwa_dostawcy*, *Adres_dostawcy* są funkcyjnie zależne od atrybutu *Nr_zamówienia*, będącego podzbiorem klucza, natomiast atrybuty *Nazwa_części* i *Magazyn* są funkcyjnie zależne od atrybutu *Id_części*, który również jest podzbiorem klucza.

DOSTAWCA_NA_ZAMÓWIENIU

<i>Nr_zamówienia</i>	<i>Id_dostawcy</i>	<i>Nazwa_dostawcy</i>	<i>Adres_dostawcy</i>
001	300	NISSAN	Liege
002	400	HONDA	Glasgow
003	500	TOYOTA	Tokyo
004	600	HONDA	London

CZĘŚCI_W_MAGAZYNIE

<i>Id_części</i>	<i>Nazwa_części</i>	<i>Magazyn</i>	<i>Adres_magazynu</i>
53	Pompa	5	Warszawa
57	Filtr	5	Warszawa
59	Błotnik	6	Szczecin

54	Pompa	5	Warszawa
32	Koło	6	Szczecin
88	Silnik	7	Poznań
59	Błotnik	6	Szczecin
21	Lampa	7	Poznań

DOSTAWY_CZĘŚCI

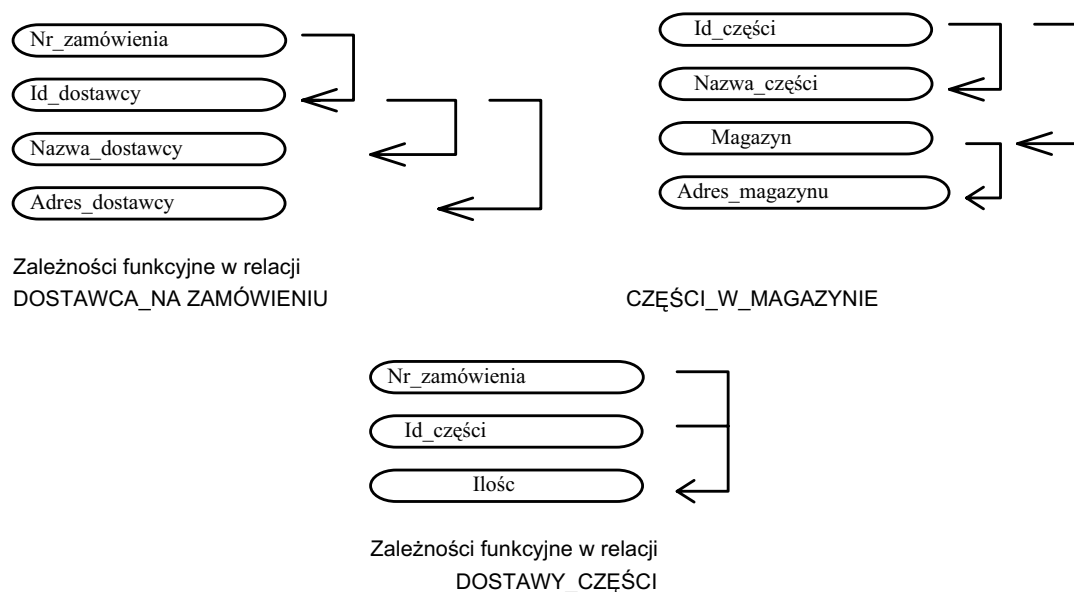
<i>Nr_zamówienia</i>	<i>Id_części</i>	<i>Ilość</i>
001	53	100
001	57	50
001	59	500
002	54	500
002	32	100
003	88	15

004	59	400	004	21	50
-----	----	-----	-----	----	----

Rys. 6. Relacje *DOSTAWCA_NA_ZAMÓWIENIU*, *CZEŚCI_W_MAGAZYNIE* i *DOSTAWY_CZEŚCI* w drugiej postaci normalnej [1].

W celu uzyskania drugiej postaci normalnej należy zdekomponować relację *ZAMÓWIENIA* na zbiór takich relacji, których wszystkie atrybuty będą w pełni funkcyjnie zależne od kluczy. Jak wynika z analizy zależności funkcyjnych występujących między atrybutami w relacji *ZAMÓWIENIA*, relację tę należy zdekomponować na następujące relacje: *DOSTAWCA_NA_ZAMÓWIENIU*, *CZEŚCI_W_MAGAZYNIE* i *DOSTAWY_CZEŚCI*, przedstawione na rys. 6. Relacja *DOSTAWCA_NA_ZAMÓWIENIU* zawiera te atrybuty, które są funkcyjnie zależne od atrybutu *Nr_zamówienia*, relacja *CZEŚCI_W_MAGAZYNIE* zawiera te atrybuty, które są funkcjonalnie zależne od atrybutu *Id_części*, natomiast relacja *DOSTAWY_CZEŚCI* została utworzona ze względu na atrybut *Ilość*, który jest funkcjonalnie zależny od pary atrybutów {*Nr_zamówienia*, *Id_części*}.

Łatwo zauważyć, że wszystkie te relacje są w 2NF, ponieważ ich klucze są kluczami prostymi, a zatem wszystkie zależności funkcyjne względem klucza są pełnymi zależnościami funkcyjnymi. W ogólności, relacje będące w pierwszej postaci normalnej, których wszystkie klucze potencjalne są kluczami prostymi, są również w drugiej postaci normalnej. Natomiast relacja *DOSTAWA_CZEŚCI* jest w drugiej postaci normalnej, gdyż atrybut *Ilość* jest w pełni funkcyjnie zależny od klucza {*Nr_zamówienia*, *Id_części*}. Diagramy zależności funkcyjnych tych relacji przedstawiono na rys. 7.



Rys. 7. Diagramy zależności funkcyjnych w relacjach *DOSTAWCA_NA_ZAMÓWIENIU*, *CZEŚCI_W_MAGAZYNIE* i *DOSTAWY_CZEŚCI* [1].

Analizując otrzymane w wyniku normalizacji relacje *DOSTAWCA_NA_ZAMÓWIENIU* i *CZEŚCI_W_MAGAZYNIE* będące w 2NF łatwo zauważyć, że nadal zawierają one dublujące się dane. W relacji *DOSTAWCA_NA_ZAMÓWIENIU* dotyczy to atrybutów *Nazwa_dostawcy* i *Adres_dostawcy*, a w relacji *CZEŚCI_W_MAGAZYNIE* – atrybutu *Adres_magazynu*. Oczywiście redundancja danych w tych relacjach pociąga za sobą wszystkie negatywne konsekwencje omówione poprzednio. Redundancja danych w relacjach w drugiej postaci normalnej wynika z istnienia tak zwanych *przechodnich zależności funkcyjnych* między atrybutami. Normalizacja do 2NF nie usuwa tych zależności, stąd konieczność normalizacji do trzeciej postaci normalnej.

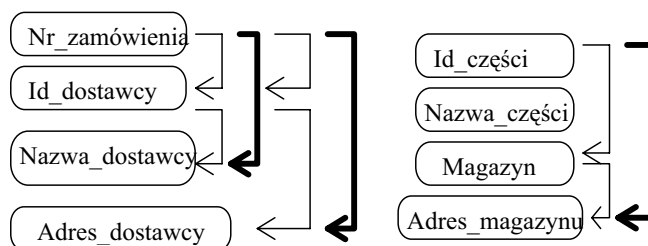
4. Trzecia postać normalna relacji

Definicja przechodniej zależności funkcyjnej

Zbiór atrybutów Y jest przechodnio funkcyjnie zależny od zbioru atrybutów X w schemacie relacji R , jeżeli $X \rightarrow Y$ i istnieje zbiór atrybutów Z , nie będący podzbiorem żadnego klucza schematu relacji R taki, że zachodzi $X \rightarrow Z$ i $Z \rightarrow Y$.

Przykładowo, w relacji *DOSTAWCA_NA_ZAMÓWIENIU* atrybuty *Nazwa_dostawcy* i *Adres_dostawcy* są przechodnio funkcyjnie zależne od atrybutu *Nr_zamówienia*, gdyż atrybuty *Nazwa_dostawcy* i *Adres_dostawcy* są funkcyjnie zależne od atrybutu *Id_dostawcy*, a ten atrybut jest funkcyjnie zależny od atrybutu *Nr_zamówienia*. Natomiast atrybut *Nr_zamówienia* nie jest funkcyjnie zależny od atrybutu *Id_dostawcy*, bo do jednego dostawcy może być skierowanych wiele zamówień, oraz atrybut *Id_dostawcy* nie jest funkcyjnie zależny od atrybutu *Nazwa_dostawcy*, bo kilku dostawców może tą samą nazwę. Atrybut *Id_dostawcy* nie jest funkcyjnie zależny od atrybutu *Adres_dostawcy*, bo pod jednym adresem może się mieścić kilku dostawców.

W relacji *CZĘŚCI_W_MAGAZYNIE* atrybut *Adres_magazynu* jest przechodnio funkcyjnie zależny od atrybutu *Id_części*, gdyż atrybut *Adres_magazynu* jest funkcyjnie zależny od atrybutu *Magazyn* (każdy magazyn mieści się pod jednym adresem), a atrybut *Magazyn* jest funkcyjnie zależny od atrybutu *Id_części* (każda część jest przechowywana w jednym magazynie). Atrybut *Magazyn* nie jest natomiast funkcyjnie zależny od atrybutu *Adres_magazynu*, gdyż pod jednym adresem może mieścić się wiele magazynów, oraz atrybut *Id_części* nie jest funkcyjnie zależny od atrybutu *Magazyn*, gdyż w jednym magazynie może być przechowywanych wiele różnych części. Przechodnie zależności funkcyjne w relacjach *DOSTAWCA_NA_ZAMÓWIENIU* i *CZĘŚCI_W_MAGAZYNIE* zilustrowano w postaci diagramu na rys. 8, oznaczając linią pogrubioną przechodnie zależności funkcyjne.



Rys. 8. Diagram zależności funkcyjnych i przechodnich zależności funkcyjnych w relacji *DOSTAWCA_NA_ZAMÓWIENIU* i *CZĘŚCI_W_MAGAZYNIE* [1].

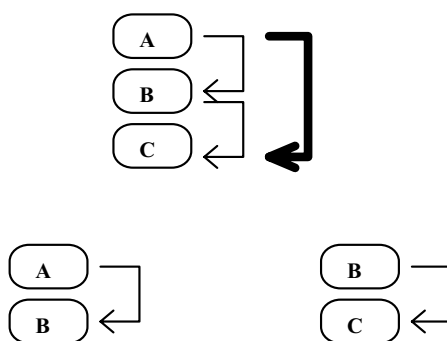
Korzystając z pojęcia przechodniej zależności funkcyjnej można zdefiniować trzecią postać normalną.

Definicja trzeciej postaci normalnej

Relacja R o danym schemacie jest w trzeciej postaci normalnej (3NF), jeżeli jest w drugiej postaci normalnej i żaden atrybut wtórny tej relacji nie jest przechodnio zależny od klucza podstawowego tej relacji.

Aby uzyskać trzecią postać normalną relacji, której atrybuty pozostają w przechodniej zależności funkcyjnej należy ją zdekomponować zgodnie ze schematem przedstawionym na rys. 9.

Przykładowo, w celu doprowadzenia relacji *DOSTAWCA_NA_ZAMÓWIENIU* do 3NF należy zdekomponować ją na dwie relacje: *ZAMÓWIENIE_DO_DOSTAWCY* i *DOSTAWCY* przedstawione na rys. 10. Natomiast w celu doprowadzenia do trzeciej postaci normalnej relacji *CZĘŚCI_W_MAGAZYNIE* należy podzielić ją na trzy relacje: *CZĘŚCI*, *CZĘŚĆ_W_MAGAZYNIE* oraz *MAGAZYNY* (rys. 9). Podział tej relacji na trzy relacje wynika z faktu występowania zależności funkcyjnej atrybutu *Nazwa_części* od atrybutu *Id_części* oprócz przechodniej zależności funkcyjnej atrybutu *Adres_magazynu* od atrybutu *Id_części* (por. rys. 8).



Zależności funkcyjne dwóch relacji w 3NF równoważnych relacji
o diagramie zależności funkcyjnych przedstawionych powyżej

Rys. 9. Schemat usuwania przechodnich zależności funkcyjnych [1].

ZAMÓWIENIE_DO_DOSTAWCY

Nr_zamówienia	Id_dostawcy
001	300
002	400
003	500
004	600

DOSTAWCY

Id_dostawcy	Nazwa_dostawcy	Adres_dostawcy
300	NISSAN	Liege
400	HONDA	Glasgow
500	TOYOTA	Tokyo
600	HONDA	London

MAGAZYNY

Magazyn	Adres_magazynu
5	Warszawa
6	Szczecin
7	Poznań

CZĘŚCI

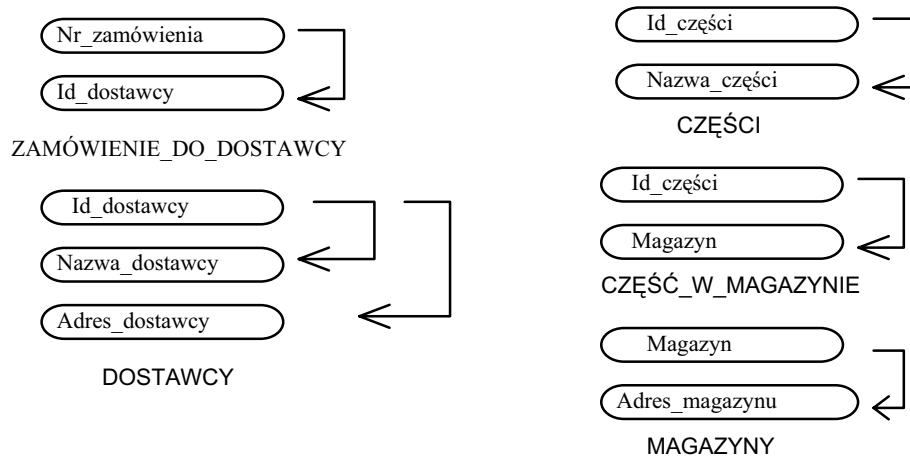
Id_części	Nazwa_części
53	Pompa
57	Filtr
54	Pompa
32	Koło
88	Silnik
59	Błotnik

CZĘŚĆ_W_MAGAZYNIE

Id_części	Magazyn
53	5
57	5
59	6
54	5
32	6
88	7
59	6

Rys. 10. Relacje ZAMÓWIENIA_DO_DOSTAWCY, DOSTAWCY, CZĘŚCI, CZĘŚCI_W_MAGAZYNIE, MAGAZYNY w trzeciej postaci normalnej.

Warto zauważyć, że relacja *DOSTAWY_CZĘŚCI*, powstała po normalizacji do drugiej postaci normalnej, jest w trzeciej postaci normalnej, ponieważ nie występują w niej zależności przechodnie. Diagramy zależności funkcyjnych relacji w trzeciej postaci normalnej przedstawiono na rys. 11.



Rys. 11. Diagramy zależności funkcyjnych w relacjach w trzeciej postaci normalnej [1].

5. Postać normalna Boyce'a/Codda

Jak wynika z naszych rozważań dotyczących 1NF, 2NF i 3NF zbiór różnych zależności funkcyjnych występujących w relacjach może być bardzo duży. Należałoby znaleźć jakiś sposób zmniejszenia tego zbioru do rozsądnych rozmiarów. Dlaczego powinno nam zależeć na tej redukcji? Jednym z powodów jest to, że zależności funkcyjne reprezentują więzy integralności, a zatem system zarządzania bazą danych (SZBD) musi je sprawdzać w chwili przeprowadzania aktualizacji. Jeżeli jest dany zestaw zależności o nazwie S , warto byłoby znaleźć inny zbiór T , który byłby znacznie mniejszy od S i miał tę własność, że każda zależność funkcyjna z S wynikałaby z zależności zawartych w T . Jeśli udałoby się znaleźć taki zbiór T , to wystarczałoby, aby SZBD wymuszał spełnienie zależności funkcyjnych w T , a tym samym zależności ze zbioru S byłyby spełnione automatycznie. Widzimy zatem, że problem znalezienia takiego zbioru T ma duże znaczenie praktyczne.

Jednym z oczywistych sposobów redukcji wielkości zbioru zależności funkcyjnych jest eliminacja tak zwanych *zależności trywialnych*. W ogólności, zależność jest trywialna, jeżeli nie może być niespełniona. Zależność jest trywialna wtedy i tylko wtedy, gdy prawa strona zależności jest podzbiorem lewej ($X, Y \rightarrow X$) lub odwrotnie lewa strona jest nadzbiorem prawej. Z powodów wspomnianych wyżej, analizując kolejne postaci normalne relacji będziemy rozważać tylko *zależności nietrywialne*.

Oryginalna definicja 3NF podana przez Codda (wprowadzona przez nas w rozdziale poprzednim) nie traktowała pewnych ogólnych przypadków zadowalająco. Poprawnie można było sobie radzić jedynie z relacjami, które:

1. miały dwa (lub więcej) klucze potencjalne takie, że
2. dwa klucze potencjalne były złożone oraz
3. miały przynajmniej jeden wspólny atrybut.

Dlatego właśnie pierwotna definicja 3NF została zastąpiona silniejszą definicją, podaną przez Boyce'a i Codda. Ponieważ ta nowa definicja faktycznie określała postać normalną silniejszą od starej 3NF, lepiej było nadać jej nową nazwę. Stąd pojawił się termin *postać normalna Boyce'a/Codda (BCNF)*. Kombinacja warunków 1, 2 i 3 rzadko występuje w praktyce. Dla relacji, w której warunki te nie są spełnione, określenia 3NF i BCNF są równoważne. Możemy teraz zdefiniować postać normalną BCNF.

Definicja postaci normalnej BCNF [6]

Relacja ma postać BCNF wtedy i tylko wtedy, gdy elementem determinującym każdej nietrywialnej, lewostronnie nieredukowalnej zależności funkcyjnej jest klucz potencjalny.

Lub mniej formalnie:

Relacja znajduje się w postaci BCNF wtedy i tylko wtedy, gdy jedynymi elementami determinującymi są klucze kandydujące, przy czym zakładamy, że wszystkie zależności funkcyjne są nietrywialne.

Innymi słowy, odnosząc treści zawarte w tych definicjach do diagramu zależności funkcyjnych, to jedynymi strzałkami wychodzącymi na diagramie zależności są strzałki wychodzące z kluczy potencjalnych (kandydujących). Definicja postaci BCNF stwierdza, że nie może być żadnych innych strzałek; nie ma więc strzałek symbolizujących zależności, które mogłyby być eliminowane w procedurze normalizacji.

Warto zauważyć, że definicja BCNF jest pojęciowo prostsza od starej definicji 3NF, ponieważ nie odwołuje się jawnie do 1NF i 2NF ani do pojęcia zależności przechodniej. Ponadto, chociaż BCNF jest ściśle mocniejsza niż 3NF, to nadal można bez utraty informacji rozłożyć dowolną relację na równoważny zbiór relacji w postaci BCNF.

6. Czwarta postać normalna relacji

Przy projektowaniu schematów relacyjnych czasami okazuje się, schemat ma postać BCNF, ale występuje w nim redundancja, która nie ma związku z zależnościami funkcyjnymi. Najpowszechniejszym źródłem redundancji w schematach BCNF jest niezależność dwóch lub więcej atrybutów wielowartościowych pewnej klasy, której projekt przekształcamy z postaci zapisanej w języku ODL do postaci relacyjnej.

Rozważmy następujący przykład. W bazie danych należy zapamiętać informacje o znajomości przez pracowników języków programowania i języków obcych: Kowalski zna C, C++ i Java oraz angielski i francuski; Malinowski zna Smalltalk i Java oraz angielski, włoski i hiszpański, a Nowak zna C i SQL PL/M oraz angielski i rosyjski. Powstaje problem w jaki sposób zapamiętać informacje przedstawione na rys. 12 w postaci relacji nieznormalizowanej.

<i>Nazwisko_pracownika</i>	<i>Język_programowania</i>	<i>Język_obcy</i>
Kowalski	C C++ Java	angielski francuski
Malinowski	Smalltalk Java	angielski włoski hiszpański
Nowak	C SQL PL/M	angielski rosyjski

Rys. 12. Relacja nieznormalizowana zawierająca informacje o pracownikach, językach programowania i językach obcych [1].

Zakładamy, że dany pracownik może znać dowolną liczbę języków programowania i dowolną liczbę języków obcych. Informacje o pracownikach, językach programowania i językach obcych są wzajemnie niezależne. Zamieńmy strukturę z rys. 12 na równoważną postać znormalizowaną (patrz rys. 13).

Zauważmy, że nie ma zupełnie w tych danych żadnych zależności funkcyjnych (poza trywialnymi). Oznacza to, że pojęcia i definicje wprowadzone wcześniej nie dostarczą formalnych podstaw do dekompozycji tej relacji na mniejsze. Relacja ta jest jednak w 3NF i BCNF, ponieważ jest typu „*all-key*”, tj. ma same klucze. Pomimo to występuje w niej dublowanie się danych; np. informacja, że Kowalski zna język C jest zapamiętana dwukrotnie. Oczywiście dublowanie się danych pociąga za sobą wszystkie negatywne konsekwencje omówione poprzednio.

<i>Nazwisko_pracownika</i>	<i>Język_programowania</i>	<i>Język_obcy</i>
Kowalski	C	angielski
Kowalski	C	francuski
Kowalski	C++	angielski
Kowalski	C++	francuski
Kowalski	Java	angielski
Kowalski	Java	francuski
Malinowski	Smalltalk	angielski
Malinowski	Smalltalk	włoski
Malinowski	Smalltalk	hiszpański
Malinowski	Java	angielski
Malinowski	Java	włoski
Malinowski	Java	hiszpański
Nowak	C	angielski
Nowak	C	rosyjski
Nowak	SQL PL/M	angielski
Nowak	SQL PL/M	rosyjski

Rys. 13. Znormalizowana relacja *PRACOWNICY* [1].

Przyczyną dublowania się danych są tym razem *wielowartościowe zależności funkcyjne* (ang. *multivalued functional dependency* – MVD). Jednej wartości atrybutu *Nazwisko_pracownika* (np. Kowalski) w ogólności odpowiada zbiór wartości atrybutu *Język_programowania* {C, C++, Java} oraz zbiór wartości atrybutu *Język_obcy* {angielski, francuski}. Innymi słowy, danej wartości atrybutu *Nazwisko_pracownika* równej Kowalski są przypisane wszystkie kombinacje wartości odpowiadających mu zbiorów wartości atrybutu *Język_programowania* i *Język_obcy*: {C, angielski}, {C, francuski}, {C++, angielski}, {C++, francuski}, {Java, angielski}, {Java, francuski}. Mamy tutaj do czynienia z dwoma wielowartościowymi zależnościami funkcyjnymi: $Nazwisko_pracownika \twoheadrightarrow Język_programowania$ i $Nazwisko_pracownika \twoheadrightarrow Język_obcy$ (zależności wielowartościowe będziemy dalej oznaczać podwójną strzałką).

Istotą czwartej postaci normalnej relacji jest dopuszczalność tylko jednej wielowartościowej zależności funkcyjnej w relacji.

Definicja wielowartościowej zależności funkcyjnej – MVD

Niech będzie dana relacja R o danym schemacie oraz niech X i Y oznaczają podzbiory atrybutów schematu relacji R . Mówimy, że podzbiór atrybutów Y jest *wielowartościowo funkcyjnie zależny* od podzbioru X , jeżeli dla dowolnej instancji relacji R o takim schemacie i dla dowolnej pary krotek t_1 i t_2 z relacji R takich, że $t_1[X] = t_2[X]$, istnieje para krotek t_3 i t_4 w relacji R , że:

$$t_3[X] = t_4[X] = t_1[X] = t_2[X] \text{ oraz}$$

$$t_3[Y] = t_1[Y] \text{ i } t_3[R-X-Y] = t_2[R-X-Y] \text{ oraz}$$

$$t_4[Y] = t_2[Y] \text{ i } t_4[R-X-Y] = t_1[R-X-Y].$$

Innymi słowy, jeżeli dla dowolnej pary krotek t_1 i t_2 z relacji R , takich, że wartości tych krotek dla atrybutów z podzbioru X są sobie równe ($t_1[X] = t_2[X]$), zamienimy w tych krotkach wartości atrybutów z podzbioru Y , to otrzymane w ten sposób krotki t_3 i t_4 również należą do relacji R .

Przykładowo, w relacji *PRACOWNICY* z rys. 13 podzbiór $X = \{Nazwisko_pracownika\}$, $Y = \{Język_programowania\}$, $R-X-Y = \{Język_obcy\}$. Parze krotek

$$t_1 = (\text{Kowalski}, \text{C}, \text{angielski})$$

$$t_2 = (\text{Kowalski}, \text{C++}, \text{francuski})$$

odpowiada para krotek

$$t_3 = (\text{Kowalski}, \text{C}, \text{francuski})$$

$$t_4 = (\text{Kowalski}, \text{C++}, \text{angielski})$$

utworzonych przez zamianę wartości atrybutu *Język_obcy*. Wystąpienie tych krotek w relacji *PRACOWNICY* wynika z faktu niezależnej znajomości kilku języków programowania i kilku języków obcych. Ze znajomości faktów, że Kowalski zna C i angielski oraz C++ i francuski wynika znajomość przez niego C i francuskiego oraz C++ i angielskiego.

Zauważmy, że z definicji wielowartościowej zależności funkcyjnej wynika, podzbiór pusty atrybutów schematu *R* jest zawsze wielowartościowo funkcyjnie zależny od dowolnego podzbioru atrybutów *X*. Zauważmy również, że jeśli podzielimy schemat *R* na dwa dowolne rozłączne podzbiory atrybutów *X* i *Y* takie, że $X \cup Y = R$, to wówczas zawsze podzbiór atrybutów *Y* jest wielowartościowo funkcyjnie zależny od podzbioru atrybutów *X* i odwrotnie. W obu powyższych przypadkach mówimy o trywialnych wielowartościowych zależnościach funkcyjnych.

Definicja czwartej postaci normalnej – 4NF

Niech będzie dana relacja *R* o danym schemacie oraz trzy parami rozłączne, niepuste podzbiory atrybutów *X*, *Y*, *Z* tego schematu, takie że $X \cup Y \cup Z = R$ i podzbiór *Y* jest nietrywialnie wielowartościowo zależny od podzbioru *X*.

Dana relacja jest w czwartej postaci normalnej (4NF) wtedy i tylko wtedy, gdy jest w trzeciej postaci normalnej i wielowartościowa zależność podzbioru *Y* od *X* pociąga za sobą funkcjonalną zależność wszystkich atrybutów tej relacji od *X*.

Innymi słowy, jeżeli w danej relacji występuje wielowartościowa zależność funkcyjna zbioru atrybutów *Y* od dowolnego podzbioru atrybutów *X*, to podzbiór *X* musi zawierać klucz tej relacji. Łatwo zauważyć, że jeśli w danej relacji występuje tylko trywialna wielowartościowa zależność funkcyjna, to relacja ta jest w czwartej postaci normalnej.

Przykładowo, relacja *PRACOWNICY* z rys. 13 nie jest w 4NF, ponieważ występuje w niej nietrywialna wielowartościowa zależność funkcyjna między $X = \{\text{Nazwisko_pracownika}\}$ a $Y = \{\text{Język_programowania}\}$ i podzbiór $X = \{\text{Nazwisko_pracownika}\}$ nie zawiera klucza tej relacji, którym jest zbiór wszystkich atrybutów $\{\text{Nazwisko_pracownika}, \text{Język_programowania}, \text{Język_obcy}\}$. W celu doprowadzenia relacji *PRACOWNICY* do czwartej postaci normalnej należy podzielić ją na dwie relacje *JĘZYKI_PROGRAMOWANIA* i *JĘZYKI_OBCE* (rys. 14), w których występuje po jednej wielowartościowej zależności funkcyjnej. W relacji *JĘZYKI_PROGRAMOWANIA* występuje trywialna wielowartościowa zależność funkcyjna między $X = \{\text{Nazwisko_pracownika}\}$ a $Y = \{\text{Język_programowania}\}$. Analogiczna sytuacja występuje w przypadku relacji *JĘZYKI_OBCE*. Obie te relacje są zatem w czwartej postaci normalnej.

JĘZYKI_PROGRAMOWANIA

<i>Nazwisko_pracownika</i>	<i>Język_programowania</i>
Kowalski	C
Kowalski	C++
Kowalski	Java
Malinowski	Smalltalk
Malinowski	Java
Nowak	C
Nowak	SQL PL/M

JĘZYKI_OBCE

<i>Nazwisko_pracownika</i>	<i>Język_obcy</i>
Kowalski	angielski
Kowalski	francuski
Malinowski	angielski
Malinowski	włoski
Malinowski	hiszpański
Nowak	angielski
Nowak	rosyjski

Rys. 14. Relacje *JĘZYKI_OBCE* i *JĘZYKI_PROGRAMOWANIA* w 4NF.

Komentując przykład z relacją *PRACOWNICY* rozważany powyżej, należy podkreślić, że zapewne żaden doświadczony projektant baz danych, wykorzystujący jako narzędzie do przygotowania projektu pojęciowego bazy danych model związków encji, nie popełniłby takiego błędu, jaki celowo popełniliśmy w tym przykładzie, określając schemat tej relacji w taki a nie inny sposób. Dla tej dziedziny rzeczywistości, już na pierwszy rzut oka widać, że należało wydzielić

encje *JĘZYK_OBCY* i *JĘZYK_PROGRAMOWANIA*, pozostające w jednostronnie obligatoryjnym związku M:N z encją *PRACOWNIK*. Problemy sygnalizowane powyżej, wcale by się wówczas nie pojawiły. Pamiętajmy jednak, że pojęcie encji jest pojęciem intuicyjnym – wiele zależy zatem od doświadczenia projektanta. Te obecnie intuicyjnie oczywiste dla każdego doświadczonego projektanta prawdy, uzyskały solidne podstawy teoretyczne w postaci procesu normalizacji, a w szczególności, dzięki wprowadzeniu przez Fagina [8, 9] pojęcia zależności wielowartościowej i 4NF.

7. Piąta postać normalna relacji [6]

W dotychczasowych rozważaniach zakładaliśmy milcząco, że jedyną operacją występującą w procesie dekompozycji jest zastępowanie relacji (w sposób bezstratny) przez jej projekcje. Założenie to pozwoliło nam pomyślnie osiągnąć nawet czwartą postać normalną. Być może zaskakującą zatem wyda się informacja, że są takie relacje, których nie można bezstratnie zamienić na dwie relacje, ale za to można je rozłożyć na trzy lub więcej relacji. Taka sytuacja ma miejsce w przypadku relacji, w których występują tak zwane zależności połączeniowe (złączeniowe [6]).

Z doświadczenia wynika, że relacje takie stanowią przypadki patologiczne [6, 11] i w praktyce występują niezwykle rzadko. Co więcej, o ile stosunkowo łatwo można wskazać zależności funkcyjne i zależności wielowartościowe (ponieważ mają one bezpośrednie odpowiedniki w rzeczywistym świecie), o tyle sytuacja z zależnościami połączeniowymi – tymi, które nie są ani FD, ani MVD – jest trudniejsza, ponieważ intuicyjne znaczenie tych zależności nie musi być oczywiste. Z powyższych względów, ograniczymy się tutaj tylko do przedstawienia formalnej definicji zależności połączeniowych oraz piątej postaci normalnej, odsyłając zainteresowanych głębszą analizą tego problemu czytelników, do literatury podstawowej z dziedziny baz danych [6, 7, 11].

Definicja zależności połączeniowej – JD (od ang. *join dependency*) [6]

Niech R będzie relacją, zaś A, B, \dots, Z będą dowolnymi podzbiorami zbioru atrybutów R . Mówimy, że R spełnia JD

$$* (A, B, \dots, Z)$$

wtedy i tylko wtedy, gdy R jest równe połączeniu swoich projekcji na A, B, \dots, Z .

Definicja piątej postaci normalnej – 5NF

Dana relacja R jest w piątej postaci normalnej (5NF) wtedy i tylko wtedy, gdy każda zależność połączeniowa jest implikowana kluczami kandydującymi R .

8. Podsumowanie [6]

W rozdziałach wcześniejszych zajmowaliśmy się procesem normalizacji relacji, tj. bezstratnej dekompozycji jako narzędziem projektowania systemów baz danych. Główna idea jest następująca. Dana jest pewna relacja R w pierwszej postaci normalnej oraz lista zależności (więzów) (FD, MVD, JD) odnoszących się do tej relacji. Systematycznie redukujemy R do zestawu mniejszych relacji, równoważnych w określonym sensie relacji R , jednak w pewien sposób bardziej pożądanym niż R . Kolejny krok procesu redukcji (dekompozycji) polega na wykonaniu operacji projekcji relacji wynikających z poprzedniego kroku. W każdym kroku korzystamy z określonych zależności (więzów) przy wyborze rodzaju projekcji. Można nieformalnie przedstawić cały ten proces w postaci zbioru następujących zasad (zebranych również na rys. 15):

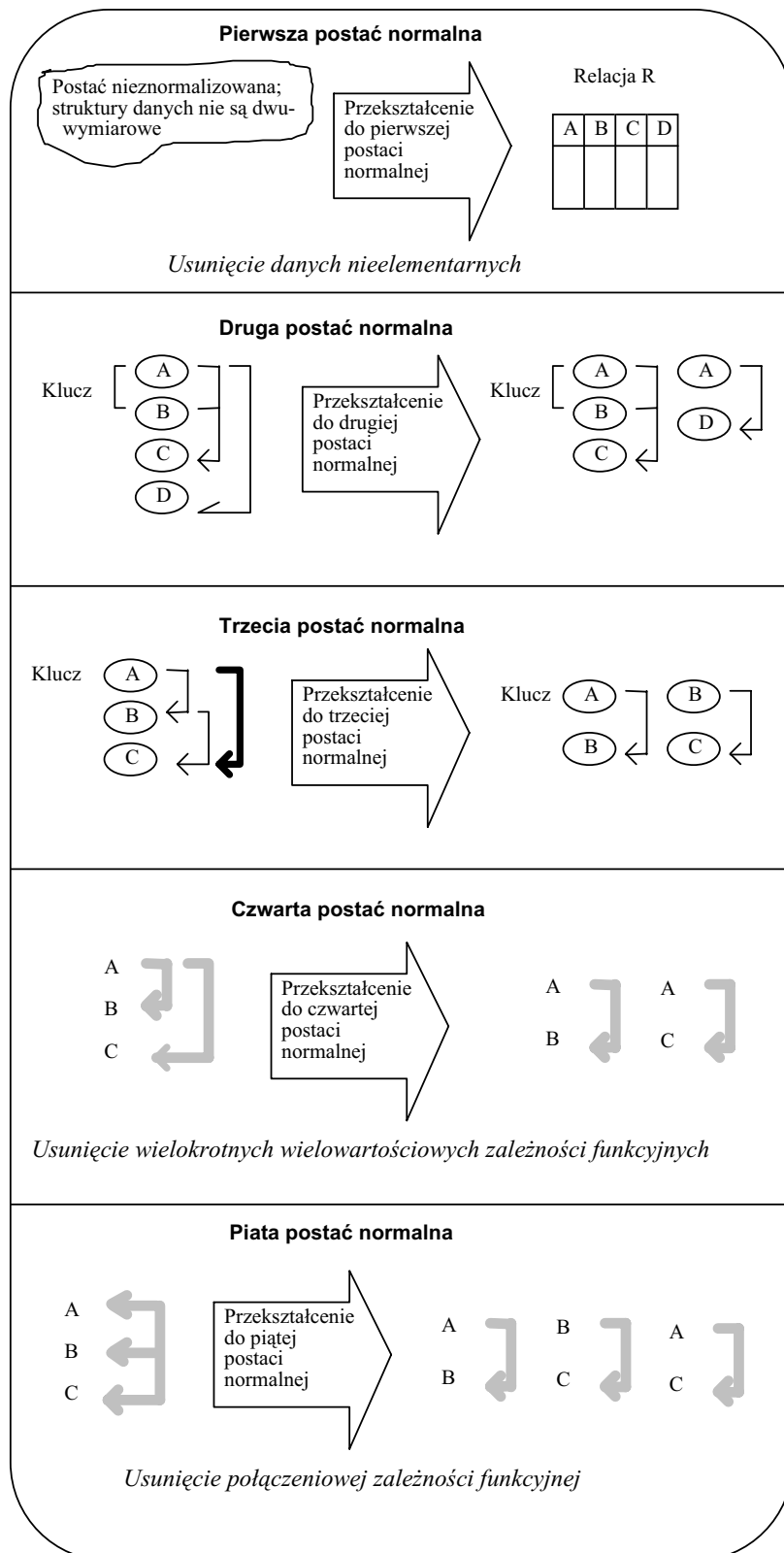
1. Wykonaj takie projekcje wyjściowej relacji w 1NF, aby wyeliminować wszystkie niepełne zależności funkcyjne. Ten krok da zestaw relacji będących w drugiej postaci normalnej.

2. Wykonaj takie projekcje tych relacji w 2NF, które usuną wszelkie przechodnie zależności funkcyjne. Krok ten da zestaw relacji w 3NF.
3. Wykonaj takie projekcje tych relacji w 3NF, które usuną wszelkie pozostałe zależności funkcyjne, w których element determinujący nie jest kluczem potencjalnym. Krok ten da zestaw relacji postaci BCNF. Kroki 1-3 można sprowadzić do jednego zalecenia: „Wykonaj takie projekcje relacji wyjściowej, które wyeliminują wszystkie zależności funkcyjne, które nie wychodzą z klucza potencjalnego”.
4. Wykonaj takie projekcje tych relacji w BCNF, aby wyeliminować wszystkie zależności wielowartościowe, nie będące jednocześnie zależnościami funkcyjnymi; lub inaczej, które usuną wszelkie wielokrotne wielowartościowe zależności funkcyjne. Ten krok prowadzi do zestawu relacji w 4NF.
5. Wykonaj takie projekcje tych relacji w 4NF, które usuną wszystkie zależności połączeniowe nie implikowane kluczami potencjalnymi. Krok ten da zestaw relacji w 5NF.

Zauważmy, że między definicjami BCNF, 4NF i 5NF występuje ciekawe podobieństwo.

- Relacja R jest w postaci BCNF wtedy i tylko wtedy, gdy każda FD w tej relacji jest implikowana kluczami potencjalnymi z relacji R .
- Relacja R jest w postaci 4NF wtedy i tylko wtedy, gdy każda MVD w tej relacji jest implikowana kluczami potencjalnymi z relacji R .
- Relacja R jest w postaci 5NF wtedy i tylko wtedy, gdy każda JD w tej relacji jest implikowana kluczami potencjalnymi z relacji R .

Anomalie związane z aktualizacją, omawiane wcześniej, są właśnie tymi trudnościami, jakie sprawiają zależności funkcyjne, zależności wielowartościowe czy zależności połączeniowe nie implikowane kluczami potencjalnymi.



Rys. 15. Etapy normalizacji – podsumowanie [1].

Podsumowując, ogólne cele procesu normalizacji są następujące:

- eliminacja pewnych rodzajów redundancji,
- uniknięcie niektórych anomalii związanych z aktualizacją,
- skonstruowanie projektu będącego „dobrą” reprezentacją świata rzeczywistego,
- uproszczenie wymuszania pewnych więzów integralności.

Dodajmy na koniec, że dobre metodyki projektowania z wykorzystaniem modelu związków encji przeważnie prowadzą i tak do w pełni znormalizowanych projektów.

Piśmiennictwo

1. Cellary W., Królikowski Z., *Wprowadzenie do projektowania baz danych*, WNT, W-wa, 1988.
2. Codd E., *The Relational Model for Database Management*, Addison-Wesley Pub. Comp., 1990.
3. Codd E.F., *Further Normalization of the Data Base Relational Model*, w: Data Base Systems, Courant Computer Science Symposia Series 6, Prentice-Hall, 1972.
4. Codd E.F., *Recent Investigations into Relational Data Base Systems*, w: Proc. IFIP Congress, Sztokholm, Szwecja, 1974.
5. Date C. J., *An Introduction to Database System*, vol. II, Addison-Wesley Pub. Comp., 1991, również WNT – W-wa.
6. Date C.J., *Wprowadzenie do systemów baz danych*, WNT, 2000.
7. Elmasri R., Navathe S., *Fundamentals of Database Systems*, Addison-Wesley Pub. Comp., 1995
8. Fagin R., *Multivalued Dependencies and a New Normal Normal Form for Relational Databases*, ACM TODS 2, nr. 3, 1977.
9. Fagin R., *Normal Forms and Relational Database Operators*, w: Proc. ACM SIGMOD Intern. Conf. on Management of Data, Boston, USA, 1979.
10. Ullman J.D., *Principles of database and knowledge base systems*, Vol. I and II, Computer Science Press, Rockville, Maryland, 1989.
11. Ullman J.D., Widom J., *Podstawowy wykład z systemów baz danych*, WNT, 2000.
12. Vossen G., *Data models, database languages and database management systems*, Addison-Wesley Pub. Company, 1991.