RELACYJNY MODEL DANYCH

Baza danych stanowi zbiór tabel powiązanych ze sobą za pomocą tzw. atrybutów kluczowych. Powiązania występujące pomiędzy poszczególnymi wierszami nazywane są też relacjami, nie te relacje przesądziły o nazwie „Relacyjne bazy danych". Nazwa ta pochodzi od podstawowej struktury danych jakimi są tabele o strukturze relacji. Dokładniej pojęcia te omówiono i zilustrowano przykładami poniżej.

ILOCZYN KARTEZJAŃSKI

Podstawowym działem matematyki wykorzystanym w modelu relacyjnym baz danych jest teoria mnogości dotycząca relacji czyli podzbiorów iloczynu kartezjańskiego (szczególnym przypadkiem relacji jest funkcja, której definicję znamy z matematyki).

loczyn kartezjański zbiorów

Iloczynem kartezjańskim **A** x **B** zbiorów A i B nazywamy zbiór wszystkich uporządkowanych par (x,y) takich, że poprzednik **x** jest elementem zbioru A, a następnik y należy do zbioru B co symbolicznie zapisujemy jako:

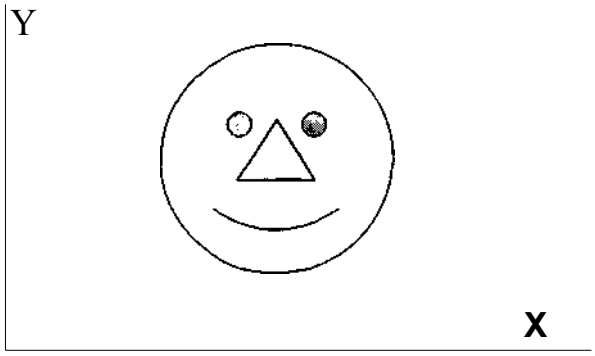
$$AxB = \{(x,y): \ x \in A \wedge y \in B \ \}$$

Podobnie dla dowolnej skończonej liczby zbiorów iloczynem kartezjańskim nazywamy zbiór wszystkich uporządkowanych krotek (xi x_2,...,x_n) takich, że kolejne Xj są elementami zbiorów A,, dla \-\ ,2,...n co zapisujemy jako:

$$\mathbf{A_1} \text{ x } \mathbf{A_2} \text{ x ..... x } \mathbf{A_n} = \{(\mathbf{x_1}, x_2, \ldots, x_n): \ xi \in Ai \ , \ i=l, \ldots, n\}$$

Relacje

Każdy podzbiór iloczynu kartezjańskiego nazywamy ***relacją.*** Jako przykład niech posłuży nam iloczyn kartezjański X **x** Y prostej X i prostej Y. Iloczynem kartezjańskim tych prostych będzie zbiór punktów płaszczyzny. Każdy podzbiór tego zbioru jest pewną relacją, w szczególności wykresy wszystkich funkcji lub zbiór punktów na rys. 1.



Rys. 1 Przykład relacji

Relacje oparte na iloczynie kartezjańskim zbiorów **o skończonej liczbie elementów** najlepiej zapisuje się w postaci tabeli składającej się z wierszy i kolumn. W wierszu podaje się elementy poszczególnych zbiorów, które są ze sobą w relacji. W poszczególnych kolumnach występują elementy tego samego zbioru. Typ elementów nazywa się dziedziną lub domeną.

Jako przykład rozważmy iloczyn kartezjański dwóch zbiorów - rzeczy i kolorów postaci:

Rzeczy x Kolory = {las, samochód, śnieg}x {zielony, niebieski, biały }

Każdy element tego iloczynu to jeden wiersz **relacji (tabeli)** o dwu kolumnach (para uporządkowana), a wszystkie podzbiory tego zbioru są **relacjami.** Przykład takiej relacji zapisanej w tabeli zamieszczono poniżej.

RZECZY	**KOLORY**
Śnieg	Biały
Samochód	Niebieski
Samochód	Zielony
Las	Zielony

Tabela 1 Przykład relacji

Rozważmy jeszcze jeden przykład: iloczyn kartezjański trzech zbiorów nazwisk, imion i numerów telefonów o postaci:

Nazwiska x Imiona x Nr telefonu

Relacją, czyli podzbiorem tego produktu kartezjańskiego będzie każda uporządkowana

trójka elementów, z których pierwszy należy do zbioru nazwisk, drugi należy do zbioru imion, a trzeci do zbioru numerów telefonów. Każdy podzbiór tego produktu będzie pewną relacją. Relacje te można zapisać w tabelach. Utwórzmy relację **Koledzy** i zapiszmy ją w tabeli nr 2

Nazwisko	**Imię**	**Nr telefonu**
Kowalski	Adam	816-14-15
Wilga	Ewa	855-65-66
Nowak	Piotr	578-35-12
Ligocki	Mateusz	967-88-99
Lisek	Urszula	789-98-76
Kowalski	Jan	814-45-55

Tabela 2 Relacja koledzy

Nazwa **relacyjne bazy danych** wywodzi się właśnie z pojęcia relacji w teorii mnogości przedstawionego powyżej, a nie z faktu relacji między poszczególnymi tabelami w bazie danych. Jak widać podstawową strukturą bazy danych jest **relacja.** Relację nazywa się też tabelą, podobnie jak wiersz nazywa się **rekordem.**

Podstawowe cechy relacji

Mimo, że relacja w relacyjnej bazie danych wyglądem przypomina zwykłą dwuwymiarową tabelę, posiada ona kilka specyficznych cech:

- Relacja musi składać się przynajmniej z jednej kolumny i może zawierać zero, jeden lub więcej wierszy danych.
- Każdy wiersz w tabeli stanowi abstrakcyjny opis pewnego obiektu określonego przez podane wartości atrybutów.
- Każdy wiersz relacji musi mieć wartości **unikalne.** Czyli musi się różnić przynajmniej wartością w jednej kolumnie od innych wierszy.
- Wartości atrybutów muszą być określonego typu zwanego dziedziną lub domeną. Niektóre (nie należące do klucza) domeny powinny uwzględniać wartość pustą NULL.
- Kolejność wierszy i kolumn w tabeli jest dowolna.
- Jeżeli baza danych jest przynajmniej w pierwszej postaci normalnej to w każdym wierszu tabeli pojedyncza kolumna zawiera tylko jedną wartość. Mówimy, że nie istnieją atrybuty wielowartościowe.
- Dane umieszczone w kolumnie należą do **domeny** tej kolumny, czyli do zbioru jej dopuszczalnych wartości. Każda relacja w relacyjnej bazie danych posiada klucz podstawowy, który składa się jednej lub większej liczby kolumn, o wartościach jednoznacznie identyfikujących każdy jej wiersz.

ATRYBUTY

Atrybuty charakteryzują cechy abstrakcyjnych obiektów umieszczonych w poszczególnych wierszach relacji (tabeli). Inaczej jest to opis poszczególnych kolumn relacji. Zakres wartości jakie mogą przyjmować atrybuty (kolumny) nazywa się ich typem, domeną lub dziedziną.

Typy atrybutów

We współczesnych bazach danych (32-bitowych) zaimplementowane są co najmniej typy danych przedstawione w poniższej tabeli.

TYP ATRYBUTU	**OPIS ATRYBUTU**
CHAR (n)	dowolny tekst, o stałej długość n-znaków
VARCHAR(n)	ciąg znaków zmiennej długości, max n znaków
BLOB	pole binarne o zmiennej długości, służy do pamiętania grafiki i danych multimedialnych
INTEGER	liczba całkowita, zakres wartości ± 2147483648
SMALL INT	liczba całkowita, zakres wartości ± 32768
DATA	data
DATATIME	data i czas
DECIMAL(p, s)	liczby dziesiętne, p oznacza liczbę cyfr (razem z kropką i znakiem), s < 6 oznacza liczbę cyfr po kropce dziesiętnej.
NUMERIC(p, s)	liczby dziesiętne, p oznacza liczbę cyfr (razem z kropką i znakiem), s oznacza liczbę cyfr po kropce dziesiętnej
FLOAT	liczba zmiennoprzecinkowa siedem cyfr znaczących
DOUBLE PRECISION	liczba zmiennoprzecinkowa podwójnej precyzji

Tabela 3 Typy atrybutów

Więzy nałożone na atrybuty

Na atrybuty w bazach danych można nałożyć dodatkowe ograniczenia zwane więzami. Podstawowymi więzami są:

- **NOT NULL** - wartość atrybutu nie może być pusta,
- **UNIQUE** - wartości atrybutów w danej kolumnie muszą być unikalne,
- **PRIMARY KEY** - dany atrybut jest kluczem podstawowym określonej relacji, co również oznacza, że musi być unikalny
- **FOREIGN KEY** - dany atrybut jest kluczem obcym.

Atrybuty kluczowe

To, że z osobnych relacji (tabel) może być utworzona spójna baza danych zawdzięczamy między innymi atrybutom kluczowym. Najczęściej tabele łączymy w relacje jeden do wielu za pomocą klucza głównego i obcego.

Aby jakiś atrybut mógł być kluczem musi w pierwszym rzędzie być unikalny i minimalny. Pojęcia te sprecyzowano i wyjaśniono poniżej.

Nadklucz lub **klucz kandydujący** schematu relacji $R\{A_1, A_2,...,A_n\}$ to zbiór atrybutów S e R, który jednoznacznie identyfikuje wszystkie krotki (rekordy) relacji **r** o schemacie R. Inaczej mówiąc, nie istnieją dwie takie krotki k_i i k_j, że $k_i[S] = k_j[S]$.

Kluczem K schematu relacji R nazywamy **minimalny nadklucz,** czyli **kluczem** nazywamy taki zbiór identyfikujący relacji, którego żaden podzbiór nie jest zbiorem identyfikującym. Znajdźmy przykład nadklucza i kluczy dla danych z tabeli poniżej.

A_1	A_2	A_3	A_4	A_5	A_6
0	0	1	A	B	C
1	1	0	D	E	F
0	1	0	G	H	I

Tabela 4 Przykład nadklucza i klucza

Zbiór **S = {A_1, A_2, A_3}** jest nadkluczem relacji r, ale nie jest kluczem, bo nie jest minimalny. Ponieważ nadklucz nie jest minimalny można utworzyć podzbiory, które już będą minimalne i będą kluczami np.

$S_1 = \{A_1,A_2\} = \{(0,0), (1,1), (0,1)\}$ $S_2 = \{A_1, A_3\} = \{(0,0),(1,0), (0,0)\}$

Natomiast podzbiór $S_3 = \{A_2, A_3\}$ nie jest nadkluczem, ani minimalnym nadkluczem

ponieważ $\mathbf{k_2(S_3) = k_3(S_3).}$

$\mathbf{S_3 = \{A_2,A_3\} = \{(0,1), (1,0), (1,0) \}}$

Rodzaje kluczy

Pojęcie atrybutu kluczowego jest bardzo ważne w teorii i praktyce relacyjnych baz danych, ponieważ w oparciu o atrybuty kluczowe są realizowane wszelkie powiązania relacyjne między tabelami bazy danych. Najogólniej atrybuty relacji dzielimy na 2 grupy:

- **Atrybuty podstawowe** - należące do któregokolwiek z kluczy,
- **Atrybuty** opisowe - nie należące do żadnego z kluczy.

W relacji można wyróżnić wiele potencjalnych kluczy. Wybrany spośród nich nazywamy głównym kluczem. Klucze dzielimy na proste i złożone:

- **Klucz prosty** - to klucz, którego zbiór identyfikujący jest jednoelementowy,
- **Klucz złożony** - to klucz, którego zbiór identyfikujący jest kilkuelementowy.

Jak już wyżej podano **kluczem** jest kolumna lub zestaw kolumn, który jednoznacznie identyfikuje resztę danych w dowolnie zadanym wierszu. Np. w tabeli Studenci, NrAlbumu jest **kluczem** ponieważ jednoznacznie określa dany wiersz. Oznacza to dwie rzeczy: nie może być dwóch wierszy w tej tabeli, które będą miały taki sam Nr_Albumu, a nawet jeśli dwóch studentów będzie miało te same nazwiska i imiona, kolumna NrAlbumu zapewnia, że tych dwóch studentów nikt nie pomyli ze sobą, ponieważ system bazy danych będzie raczej używać pola NrAlbumu do znalezienia studenta niż ich nazwisk i imion.

Obcy klucz jest atrybutem (kolumną) z obcej tabeli, gdzie ten atrybut jest podstawowym kluczem. Oznacza to, że dowolne dane w kolumnie obcego klucza muszą mieć odpowiedniki w odpowiadającej tabeli, w której ten klucz jest kluczem podstawowym. W języku relacyjnych baz danych, ta odpowiedniość określana jest jako integralność referencyjna.

RELACJE MIĘDZY TABELAMI

Relacyjna baza danych jest kolekcją dwuwymiarowych tabel (relacji), złożonych z kolumn (atrybutów) oraz wierszy (krotek). Struktura poszczególnych tabel jest przechowywane w słowniku danych (często określanego mianem tabel systemowych), który zawiera szczegółowe informacje o bazie danych. W relacyjnej bazie danych relacje między tabelami wyraża się przez umieszczenie identycznych kolumn w dwóch lub większej liczbie tabel. Jeśli któraś z tabel zawiera kolumnę lub kombinację kolumn odpowiadającą kluczowi podstawowemu innej tabeli, wówczas między tymi tabelami istnieje relacja logiczna. O tabeli zawierającej kopię klucza podstawowego innej tabeli mówimy, że zawiera klucz obcy. Każda różna od NULL wartość klucza obcego musi odpowiadać jednej z istniejących wartości klucza podstawowego, którego klucz ten jest kopią. Określa się to mianem integralności referencyjnej.

Powiązanie pomiędzy parą tabel nosi nazwę relacji. Relacja między tabelami istnieje wtedy, gdy dwie tabele są połączone przez klucz podstawowy i klucz obcy lub gdy istnieje dodatkowa, łącząca je tabela, zwana tabelą łączącą.

Relacje są ważne dla integracji bazy, ponieważ umożliwiają eliminowanie powtarzających się danych. Każda relacja jest opisywana przez typ powiązania istniejącego między dwoma tabelami i typ uczestnictwa jaki obie tabele mają w tej relacji. Każdej relacji między pewnymi dwoma tabelami możemy przypisać konkretny typ. Istnieją trzy typy relacji określające typ powiązania między wierszami tabel:

- **Jeden do jednego (1 : 1),**
- **Jeden do wielu (1 : n),**
- **Wiele do wielu (m :n).**

Typ uczestnictwa wierszy w relacji

Wiersze tabel mogą uczestniczyć obowiązkowo lub opcjonalnie w relacji między tabelami. Np. między tabelami Pracownicy i Narzędzia ustalmy relację narzędzie zostało wypożyczone. Relacja ta została określona przez wpisanie do kolumny IdPracownika tabeli Narzędzia odpowiedniego Id_Pracownika z tabeli Pracownicy. Ponieważ nie wszystkie narzędzia muszą być wypożyczone niektóre wiersze tabeli Narzędzia mają pustą kolumnę Id_pracownika (mówimy wtedy wartość **NULL).** Powiązanie to nazywamy opcjonalnym. Rozróżniamy dwa typy uczestnictwa wierszy tabel w relacji:

- **Uczestnictwo obowiązkowe,**
- **Uczestnictwo opcjonalne.**

Relacje jeden do jednego

Mówimy, że między dwoma tabelami istnieje relacja jeden do jednego, jeśli pojedynczemu rekordowi z pierwszej tabeli jest przyporządkowany dokładnie jeden rekord z drugiej tabeli i na odwrót, pojedynczemu rekordowi z drugiej tabeli jest przyporządkowany dokładnie jeden rekord z pierwszej tabeli.

Rys.2 przedstawia przykładową relację jeden do jednego między tabelą pracownicy i tabelą Stawki. W tym przypadku każdemu rekordowi z tabeli Pracownicy przypisany jest jeden rekord z tabeli Stawki, a pojedynczemu rekordowi z tabeli Stawki odpowiada dokładnie jeden rekord z tabeli Pracownicy. Łącznikiem relacji między tymi tabelami jest atrybut kluczowy Id Pracownika.

Id Pracownika	**Imię**	Nazwisko
100	Jan	Kowalski
101	Jan	Nowak

1:1

Id Pracownika	**Stawka**
100	5.5
101	6.5

Rys. 2 Relacja jeden do jeden

Obydwie relacje z rys. 2 są obowiązkowe (bez kółek), ponieważ każdy aktywny pracownik ma określoną stawkę wynagrodzenia i każda stawka w tej tabeli musi należeć do jakiegoś pracownika.

Relacje jeden do wielu

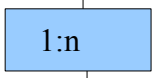
Mówimy, że między dwiema tabelami istnieje relacja jeden do wielu, jeżeli z każdym rekordem z pierwszej tabeli jest związany jeden lub kilka rekordów z drugiej tabeli, natomiast każdemu rekordowi z drugiej tabeli odpowiada dokładnie jeden rekord z pierwszej tabeli.

Przykładowa relacja jeden do wielu, łącząca tabele Pracownicy i Narzędzia, jest pokazana na rys. 3. W tym przypadku pojedynczemu rekordowi z tabeli Pracownicy odpowiada jeden lub więcej rekordów z tabeli Narzędzia, a pojedynczemu rekordowi z tabeli Narzędzia - dokładnie jeden rekord z tabeli Pracownicy. Łącznikiem relacji między tymi tabelami jest atrybut kluczowy IdPracownika. Na rysunku linie obrazującą powiązanie zakończono strzałką co oznacza relację jeden do wielu. Strzałka wskazuje, z której tabeli jest dołączanych wiele wierszy do pojedynczego wiersza pierwszej tabeli.

Uczestnictwo opcjonalne zaznaczono kółkiem. W tym przypadku oznacza to, że nie wszystkie narzędzia muszą być aktualnie wypożyczone.

Relacje jeden do wielu stanowią zdecydowanie najczęstszy typ relacji występujący w rzeczywistych bazach danych. Typ ten jest szczególnie ważny, ponieważ umożliwia ograniczenie nadmiarowości danych do absolutnego minimum.

IdPracownika	Imię	Nazwisko
100	Jan	Kowalski
101	Jan	Nowak



Id Pracownika	Nazwa narzędzia
100	Klucze nasadowe
100	Młotek
101	Piłka
101	Klucz Ml2

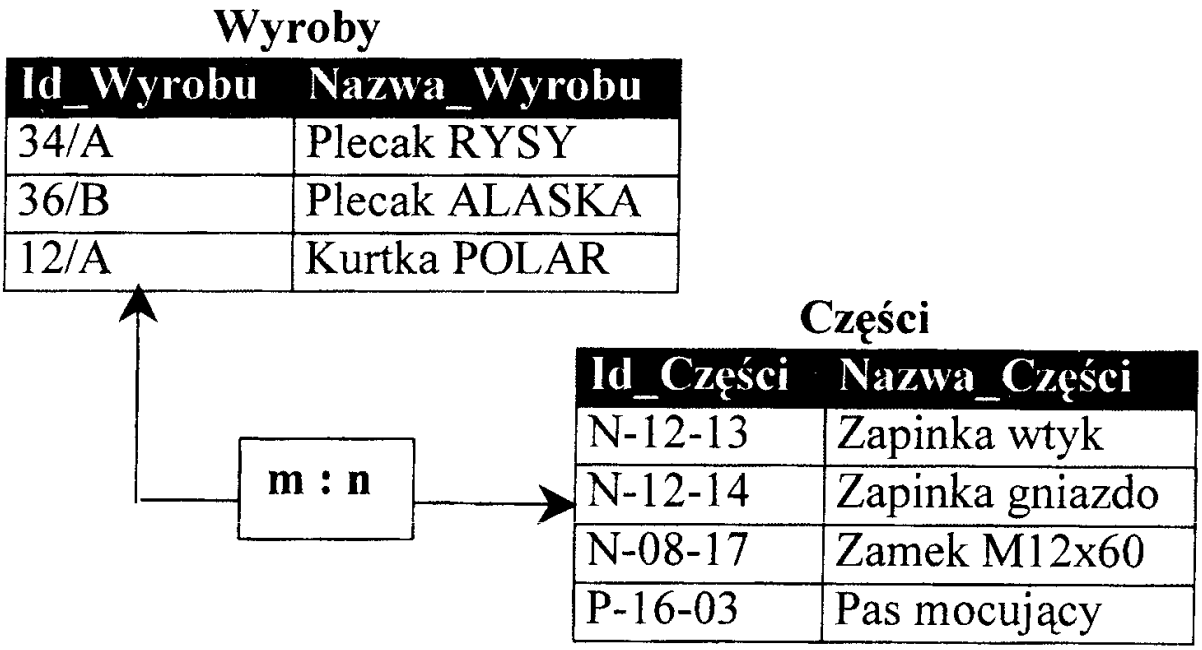
Relacja jeden do wielu

Relacje wiele do wielu

Mówimy, że między dwiema tabelami występuje relacja wiele do wielu, jeżeli pojedynczemu rekordowi w pierwszej tabela może odpowiadać jeden lub więcej rekordów

w drugiej tabeli i na odwrót - pojedynczy rekord z drugiej tabeli może być powiązany z jednym lub większą liczbą rekordów z pierwszej.

Poniżej podano przykład relacji wiele do wielu. W tym przypadku pojedynczemu rekordowi z tabeli Wyroby może odpowiadać wiele rekordów w tabeli Części i odwrotnie. Linia łącząca tabele jest z obu stron zakończona strzałkami, co oznacza właśnie relację wiele do wielu.

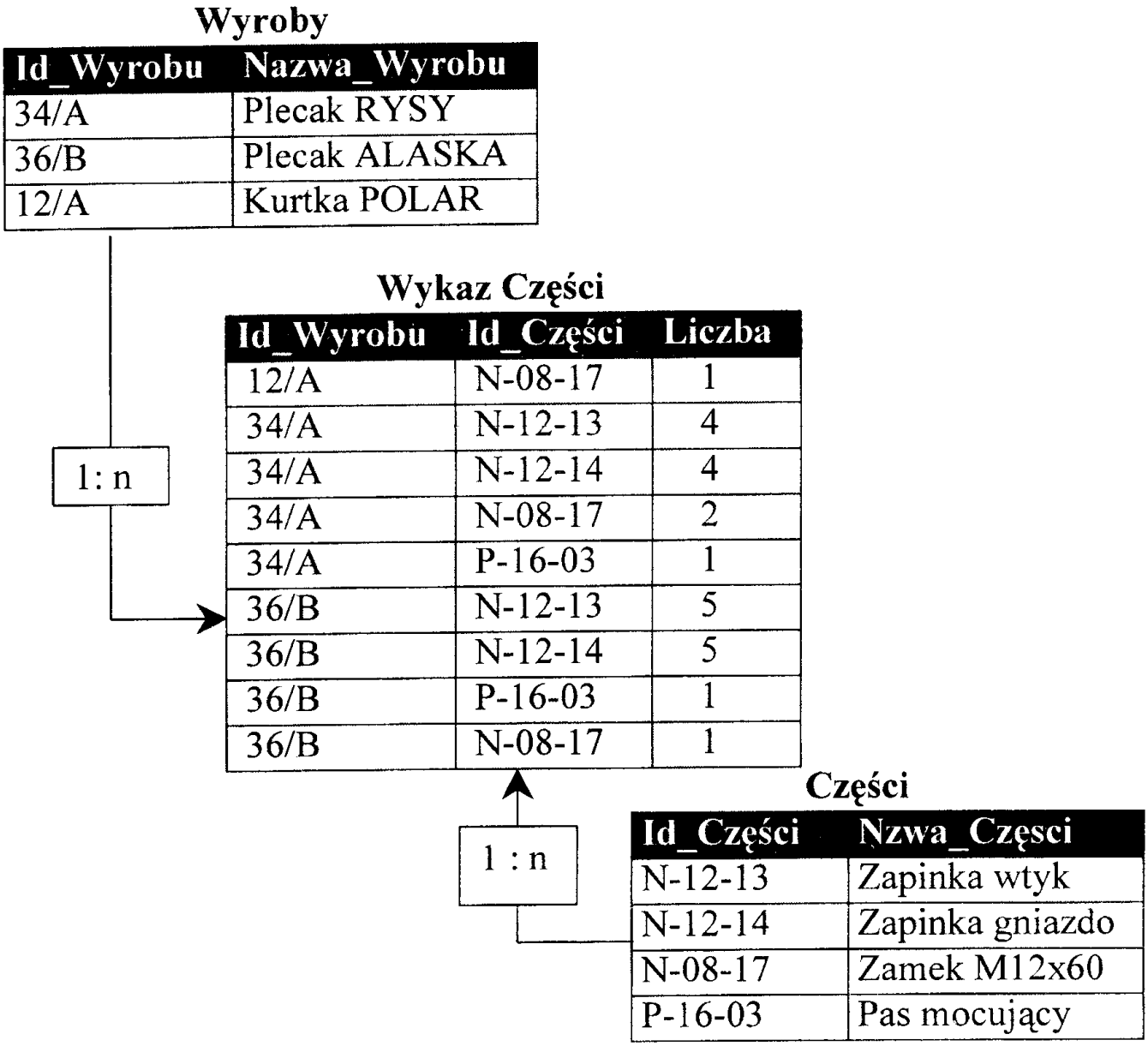


Rys. 4 Relacja wiele do wielu

Zaprogramowanie bezpośredniej relacji wiele do wielu między dwoma tabelami jest trudne, ponieważ wiąże się z wprowadzeniem dużej ilości nadmiarowych danych do jednej z tych tabel. Co więcej, dopisywanie, modyfikacja i usuwanie danych z tak powiązanych tabel sprawiłoby duże kłopoty. Dlatego powiązania te są eliminowane i zastępowane w procesie normalizacji dwoma połączeniami typu jeden do wielu z wykorzystaniem dodatkowej tabeli.

Jako przykład zmodyfikujmy zależność z rys. 4 dodając dodatkową tabelę łączącą *Wykaz Części.* Po dodaniu tej tabeli otrzymujemy jednoznaczny obraz zależności między wyrobami, a częściami składowymi wyrobu. Kolumna liczba określa ile sztuk danej części potrzeba na wytworzenie jednego wyrobu. Kluczem głównym (Primary Key) tej tabeli może być klucz złożony z klucza obcego *Id_Wyrobu* i *Id_Części*

Umiejętność określania typu relacji i stopnia uczestnictwa pomiędzy tabelami jest bardzo ważna, ponieważ decyduje to o sposobie powiązania tabeli, o zależności znajdujących się w nich rekordów oraz o maksymalnej liczbie powiązań między rekordami, jakie relacja ta powinna dopuszczać.



Rys. 5 Zastąpienie relacji wiele do wielu dwoma relacjami jeden do wielu

NORMALIZACJA

Normalizacja relacji jest to proces, podczas którego schematy relacji posiadające pewne niepożądane cechy są przekształcone na mniejsze schematy relacji o pożądanych własnościach. Celem normalizacji jest uzyskanie optymalnej struktury bazy danych.

Rozróżniamy pięć kolejnych postaci normalnych relacji. W praktyce stosuje się relacje w trzeciej postaci normalnej jako optymalne do zaprogramowania systemu baz danych. Proces

normalizacji musi spełniać następujące warunki:

- żaden atrybut nie zostanie zagubiony w trakcie normalizacji,
- dekompozycja relacji nie prowadzi do utraty informacji,
- wszystkie zależności funkcyjne są reprezentowane w pojedynczych schematach relacji,
- relacje między tabelami mogą być tylko typu jeden do jeden lub jeden do wielu.

Jako przykład rozważmy schemat relacji nieznormalizowanej dotyczącej przyjmowania zamówień na wyroby. Najprościej będzie bez zbędnych rozważań zapisać obok siebie informacje dotyczące konkretnego zamówienia (tabela 5). Tabela w tej postaci byłaby trudna do przetwarzania. Dlatego będzie się dążyć do jej normalizacji.

ID_Klienta	ID_Zamównienia	Wyroby
5	7	5 łożysk, 18 wiertarek, 3 pilniki
6	8	3 młotki, 2 pilniki
5	9	7 młotków, 2 pilniki

Tabela 5 Przykład tabeli nieznormalizowanej

Pierwsza postać normalna

Pierwsza postać normalna **(1PN)** wymaga, aby wszystkie wartości jej kolumn były elementarne (niepodzielne). Jak łatwo zauważyć kolumna o nazwie ***Wyroby*** nie spełnia tego warunku. Dlatego schemat relacji musimy tak przekształcić aby w kolumnach były tylko wartości elementarne oraz aby w trakcie przekształceń nie utracić żadnej informacji. Warunki te spełnia tabela 6.

D_Klienta	ID_Zamównienia	Nr_Pozycji	Ilość	Wyroby
5	7	1	5	Łożysko
5	7	2	18	Wiertło
5	7	3	3	Pilnik
6	8	1	3	Młotek
6	8	2	2	Pilnik
5	9	1	7	Młotek
5	9	2	2	Pilnik

Tabela 6 Relacja w pierwszej postaci normalnej

W tabeli tej pojawiła się nowa kolumna *Nr_Pozycji.* Określa ona pozycję towaru w zamówieniu. Zabieg ten spowodował, że tabela znajduje się w pierwszej postaci normalnej, czyli nie ma w niej powtarzających się kolumn, a każda kolumna zwiera tylko jedną wartość. Kluczem głównym tej tabeli jest klucz złożony z pary kolumn ***ID_Zamówienia*** i

Nr_Pozycji. Kluczem obcym jest *ID_Klienta.*

Druga postać normalna

Aby tabela przyjęła drugą postać normalną, musi być w pierwszej postaci normalnej i każda z jej kolumn musi być w pełni zależna od klucza głównego i od każdego atrybutu klucza głównego, jeśli klucz ten składa się w wielu kolumn. Oznacza to, że elementy każdej kolumny tabeli, nie będącej kluczem muszą być jednoznacznie identyfikowane przez klucz główny tabeli.

W tabeli 6 kluczem głównym jest złożenie ***(ID_Zamówienia, Nr_Pozycji).*** Aby tabela była w drugiej postaci Normalnej **(2PN),** każda niekluczowa kolumna musi w pełni zależeć od klucza. Warunku tego nie spełnia ***ID_Klienta*** ponieważ jednoznacznie zależy od części klucza tj. od ***Nr_Zamówienia,*** a niecałego klucza. Z tego powodu zachodzi potrzeba rozdzielnia tabeli 6 na dwie mniejsze (tabele 7, 8).

Id_Klienta	Id_Zamówienia	Data
5	7	1999-04-15
6	8	1999-04-16
5	9	1999-04-20

Tabela 7 Relacja znormalizowana

Id_Zamówienia	Nr_Pozycji	Ilość	Id_Wyrobu	Wyroby
7	1	5	33	Łożysko
7	2	18	44	Wiertło
7	3	3	15	Pilnik
8	1	3	17	Młotek
8	2	2	15	Pilnik
9	1	7	17	Młotek
9	2	2	15	Pilnik

Tabela 8 Relacja w drugiej postaci normalnej

Trzecia postać normalna

Aby tabela była w trzeciej postaci normalnej, każda z jej kolumn musi być całkowicie zależna od klucza głównego i niezależna od pozostałych kolumn. A zatem tabela musi spełniać warunki drugiej postaci normalnej, a ponadto każda kolumna, nie będąca kluczem, musi być niezależna od pozostałych kolumn niekluczowych.

Poprzednio otrzymaliśmy dwie tabele 7 i 8. Tabela nr 7 jest już co najmniej w 3PN ponieważ nie ma żadnej zależności między kolumnami niekluczowymi. Natomiast tabela nr 8 nie jest w 3PN ponieważ między atrybutami niekluczowymi ***IDWyrobu*** i ***Nazwawyrobu*** zachodzi zależność. Aby otrzymać trzecią postać normalną należy dokonać jej dekompozycji na dwie tabele 9 i 10.

Id_Zamówienia	Nr_Pozycji	Ilość	Id_Wyrobu
7	1	5	33
7	2	18	44
7	3	3	15
8	1	3	17
8	2	2	15
9	1	7	17
9	2	2	15

Tabela 9 Relacja w postaci normalnej

Id_Wyrobu	Nazwa_Wyrobu
33	Łożysko
44	Wiertło
15	Pilnik
17	Młotek

Tabela 10 Relacja w postaci normalnej

Nasze tabele osiągnęły **3PN,** co w praktyce oznacza zadowalającą i efektowną strukturę bazy danych. Dekompozycja relacji, sięgająca poza trzecią postać normalną, prowadzi na ogół do utworzenia bardzo skomplikowanego modelu, którego implementacja może być nieopłacalna. Warto dodać, że współczesne systemy baz danych dobrze obsługują klucze złożone. Dlatego trzecia postać normalna jest zupełnie wystarczająca dla tabel bazy danych.

ALGEBRA RELACYJNA

W języku **SQL** bardzo łatwo można zapisać podstawowe **działania algebry relacyjnej** [10]. W matematyce algebrą nazywamy dowolny zbiór z działaniem, którego wynik należy do tego zbioru. W naszym przypadku zbiorem tym będzie zbiór tabel, a działaniem takie działanie na zbiorze tabel, które w wyniku daje znów tabelę. Działania mogą być w tym przypadku z jednym argumentem (udział w działaniu bierze jedna tabela) lub wieloargumentowe (udział w działaniu bierze kilka tabel, a wynikiem jest tabela).
W każdej bazie danych powinno dać się zrealizować przynajmniej następujące działania algebry relacyjnej i ich kombinacje.

Selekcja

Selekcja wybiera z relacji (tabeli) tylko te wiersze, które spełniają podany warunek. W języku SQL selekcję można zrealizować za pomocą polecenia SELECT z klauzulą WHERE, po której następuje predykat wyboru. Dla przykładu rozważmy relację STUDENCI i działanie SELECT:

Nr_Albumu	Nazwisko	Imię	Grupa
125/98	Kowalska	Anna	A
137/99	Kania	Jan	B
111/98	Rogowski	Adam	A
145/99	Nowak	Ewa	B

Tabela 11 Studenci

SELECT * FROM STUDENCI ***WHERE*** Grupa = B';

Wynik powyższego działania podano w tabeli 12.

Nr_Albumu	*Nazwisko*	*Imię*	*Grupa*
137/99	Kania	Jan	B
145/99	Nowak	Ewa	B

Tabela 12 Relacja otrzymana w wyniku selekcji

Projekcja

W czasie projekcji z relacji wybiera się tylko określone atrybuty (kolumny). Po wykonaniu na tabeli STUDENCI projekcji, w wyniku otrzymujemy znów relację (tabela 13).

SELECT Nazwisko, Imię ***FROM***
STUDENCI;

Nazwisko	Imię
Kowalska	Anna
Kania	Jan
Rogowski	Adam
Nowak	Ewa

Tabela 13 Wynik projekcji

Złączenie naturalne

Złączenie naturalne umożliwia pobranie danych z dwu lub większej liczby tabel połączonych odpowiednimi relacjami. Jako przykład weźmy pod uwagę dwie relacje AUTORZY (tabela 14) i KSIĄŻKI (tabela 15).

Nr_Autora	Autor
4	Henryk Sienkiewicz
2	Aleksander Fredro
3	Adam Mickiewicz

Tabela 14 AUTORZY

Nr_Książki	Tytuł	Nr_Autora
7	Ogniem i mieczem	4
9	Pan Tadeusz	3
8	W pustyni i w puszczy	4
6	Potop	4

Tabela 15 KSIĄŻKI

Wykonajmy następujące złączenie naturalne połączone z projekcją za pomocą polecenia select:

SELECT Tytuł, Autor
FROM **AUTORZY** ***JOIN*** KSIĄŻKI
ON AUTORZY.NrAutora = KSIĄŻKI.NrAutora

W wyniku otrzymujemy relację z odpowiednio dopasowanymi wierszami:

Autor	Tytuł
Henryk Sienkiewicz	Ogniem i mieczem
Adam Mickiewicz	Pan Tadeusz
Henryk Sienkiewicz	W pustyni i w puszczy
Henryk Sienkiewicz	Potop

Tabela 16 Wynik połączenia naturalnego i projekcji

Złączenia zewnętrzne

Złączenie zewnętrzne (FULL OUTER JOIN) umożliwia pobranie danych z dwu lub większej liczby tabeli połączonych odpowiednimi relacjami. Jeżeli w tych relacjach są wiersze, do których nie można dopasować wierszy z innej tabeli, to atrybuty tych wierszy oznaczane są jako NULL. Jako przykład napiszmy następujące zapytanie. W wyniku otrzymujemy tabelę 17.

SELECT Tytuł, Autor
FROM AUTORZY *FULL OUTER JOIN KSIĄŻKI*
ON AUTORZY.Nr_Autora = KSIĄŻKI.NrAutora

Autor	***Tytuł***
Henryk Sienkiewicz	Ogniem i mieczem
Adam Mickiewicz	Pan Tadeusz
Henryk Sienkiewicz	W pustyni i w puszczy
Henryk Sienkiewicz	Potop
Aleksander Fredro	NULL

Tabela 17 Wynik złączenia zewnętrznego

Suma

Suma (UNION) jest wynikiem dodania do siebie wierszy dwóch tabel mających takie same kolumny określone na tych samych dziedzinach. W wyniku powstaje tabela zawierająca wiersze z obydwu tabel wyjściowych. Jako przykład wykorzystajmy tabelę STUDENCI i podwójne zapytanie select:

```
SELECT * FROM STUDENCI
 WHERE Grupa - B'
 UNION
SELECT * FROM STUDENCI
 WHERE Nazwisko = 'Rogowski'
```

W wyniku otrzymujemy:

Nr_Albumu	Nazwisko	Imię	Grupa
137/99	Kania	Jan	B
111/98	Rogowski	Adam	A
145/99	Nowak	Ewa	B

Tabela 18 Wynik obliczenia sumy dwóch zapytań

Iloczyn kartezjański

Z dwóch relacji będących argumentami iloczynu powstaje jedna relacja wynikowa złożona ze wszystkich możliwych kombinacji wierszy obydwu tabel. Jako przykład rozważmy iloczyn kartezjański dwóch następujących tabel i zapytania:

```
SELECT* FROM T1,T2;
```

A1	A2
A	B
C	D

Tabela 19 Tl

A3	A4
1	2
3	4

Tabela 20 T2

A1	*A2*	*A3*	*A4*
A	B	1	2
A	B	3	4
C	D	1	2
C	D	3	4

Tabela 21 Iloczyn kartezjański tabel Tl i T2

STRUKTURA I WŁASNOŚCI BAZY DANYCH

Zazwyczaj baza danych składa się z kilku lub kilkudziesięciu, a nawet kilkuset tabel połączonych ze sobą za pomocą odpowiednich relacji. Poniżej podano podstawowe cechy charakteryzujące prawidłowo zaprojektowaną bazę danych.

Struktura bazy danych

Struktura bazy danych została pokazana na rysunku poniżej. Elementarne obiekty składowe bazy danych jakimi są wiersze i kolumny widnieja$_u$ w centrum diagramu. Elementy te są zgrupowane w większe struktury - tabele i perspektywy. Od kilka do kilkuset tabel i perspektyw uzupełnionych indeksami tworzy z kolei bazę danych.

Rys. 6 Struktura bazy danych.

Integracja danych

Podstawowym zadaniem w czasie projektowania i użytkowania bazy danych jest zapewnienie integracji jej danych. Integracja danych oznacza, że w bazie danych nie ma powtarzających się i niepotrzebnych danych. Integracja danych ułatwia również ich przetwarzanie, ponieważ przy zmianie danych modyfikujemy je tylko w jednym miejscu.

Integralność danych

Integralność danych dotyczy powiązań między danymi w bazie. Zmiany po jednej stronie danego powiązania powinny być odzwierciedlone również po drugiej jego stronie. Jeśli np. wystawiamy dowód przyjęcia do magazynu wyrobu z produkcji to równocześnie musi się zmienić stan magazynu wyrobów gotowych.

Współdzielenie danych

Dane przechowywane w bazie danych są zazwyczaj wykorzystywane przez wielu użytkowników w tym samym czasie. Jedni użytkownicy modyfikują dane, inni w tym samym czasie korzystają z różnych raportów i zestawień, a wszystko to musi być realizowane tak, żeby nikt nikomu nie przeszkadzał w jego pracy. Dlatego przed systemem zarządzającym bazą danych stawia się wymaganie, bezkolizyjnego dostępu do tych danych

przez wielu użytkowników równocześnie. Współczesne systemy baz danych wykorzystując tzw. transakcje coraz lepiej rozwiązują ten trudny problem współdzielenia danych. Współdzielenie danych ma dwa aspekty. Po pierwsze, na tej samej bazie danych mogą współbieżnie pracować różne aplikacje. Po drugie, na tych samych danych może współbieżnie pracować ta sama aplikacja uruchomiona przez różnych użytkowników.

Bezpieczeństwo danych

Bazy danych służą do przechowywania wszelkich informacji o działalności firmy czy instytucji. Mogą to być banki, zakłady pracy, sądy, czy też organizacje wojskowe. Nie trzeba wielkiej wyobraźni aby zrozumieć potrzebę ochrony informacji przed nieuprawnionym dostępem. Dlatego bazy danych muszą mieć odpowiednie zabezpieczenia przed swobodnym dostępem do informacji. Dlatego w systemach baz danych informację zabezpiecza się przez jej szyfrowanie oraz przydzielanie zakresu uprawnień dla użytkowników do modyfikowania i odczytywania danych.

Abstrakcja danych

Baza danych może być traktowana jako pewien model rzeczywistości. Informacje zawarte bazie reprezentują tylko niektóre właściwości (atrybuty) obiektów świata rzeczywistego. Do bazy danych zapisuje się tylko te dane dotyczące obiektów świata rzeczywistego, które mają istotny wpływ na działalność danej instytucji.

Niezależność danych

Niezależność danych to oddzielenie danych od procesów operujących na tych danych. Czyli baza danych ma być formalnie niezależna od systemu baz danych. Poprawianie lub ulepszanie systemu baz danych musi odbywać się więc bez naruszania bazy danych.

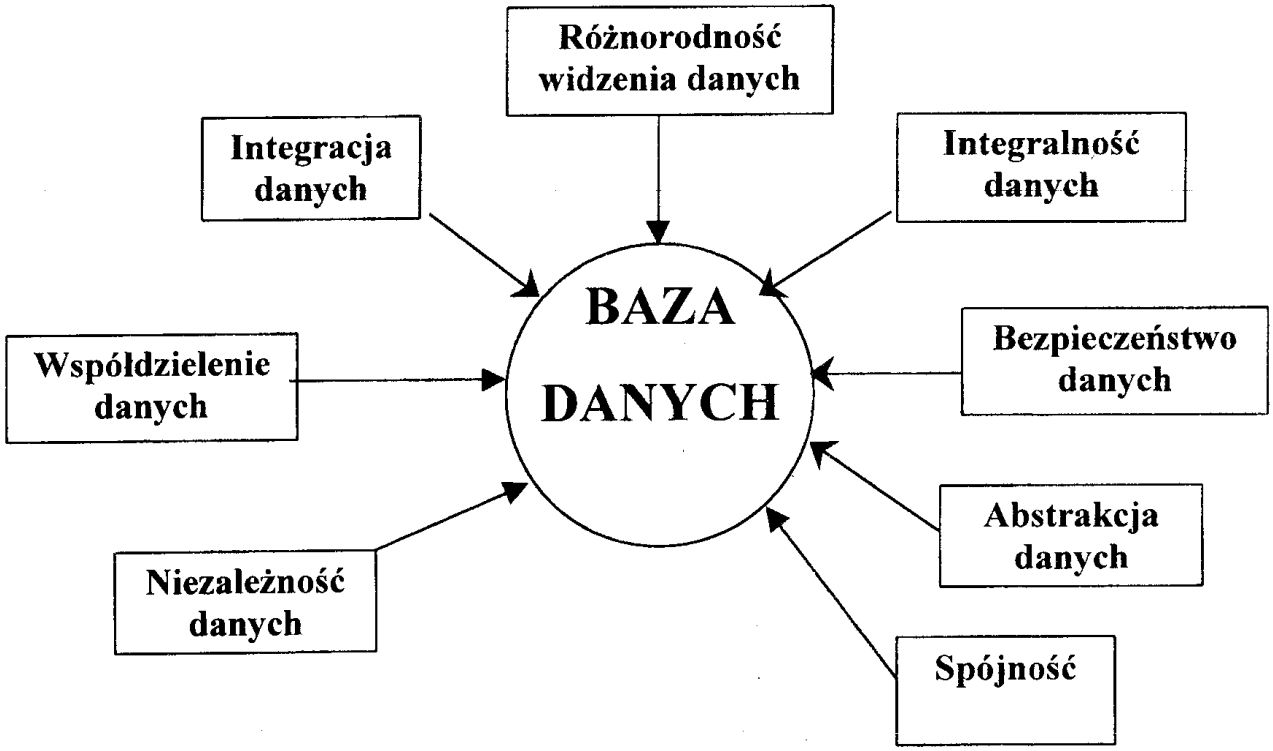
Różnorodność widzenia danych

Te same dane mogą być prezentowane użytkownikom na różne sposoby. Uzyskuje się to np. za pomocą perspektyw lub zapytań w języku SQL, które dotyczą tych samych lub wspólnych danych, lecz zestawianych na różne sposoby.

Spójność bazy danych

Przez spójność bazy danych rozumie się jej zgodność z reprezentowaną rzeczywistością. Na przykład baza danych banku, aby była spójna musi odzwierciedlać aktualny stan kont klientów tego banku.

Cechy prawidłowo zaprojektowanej i eksploatowanej bazy danych podsumowano na rys. 7. Wiele tych cech wynika z teorii przedstawionej powyżej.



Rys. 7 Podstawowe cechy relacyjnej bazy danych

INDEKSY TABEL

Indeksy tabel są specjalną strukturą danych pozwalającą systemowi bazy danych na szybkie ich przeszukiwanie. W systemach baz danych kolejne rekordy są dokładane na koniec odpowiednich tabel. Ich kolejność jest na ogół przypadkowa, natomiast w trakcie przetwarzania poszczególne rekordy powinny być dostępne według wymaganego porządku zwanego kluczem.

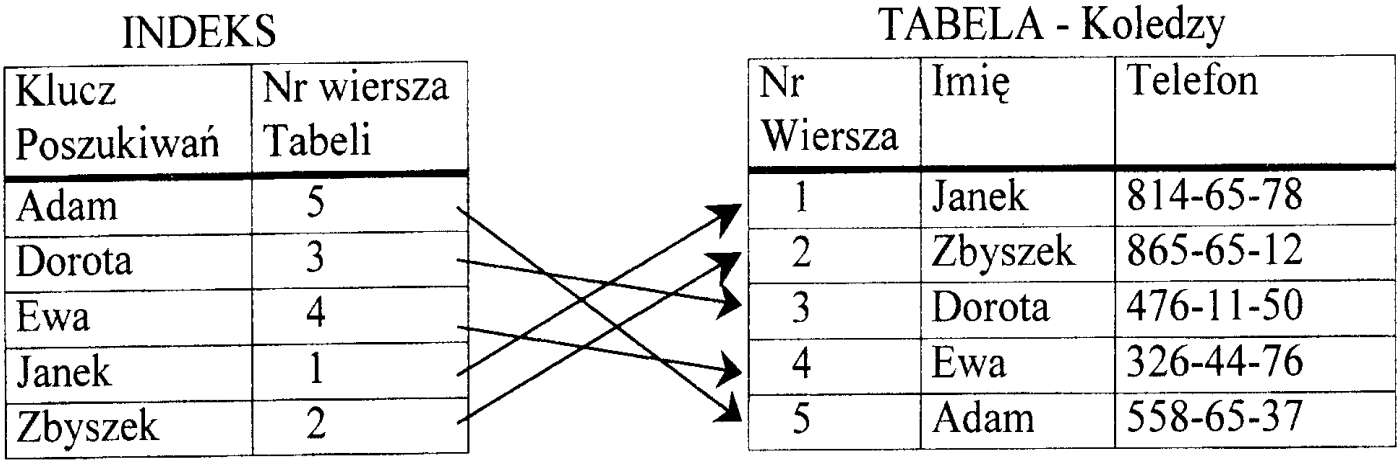
Jeżeli weźmiemy pod uwagę, że w relacji może być od kilku do kilkuset tysięcy rekordów to metoda systematycznego przeszukiwania jest w tym przypadku mało efektywna. Aby przyśpieszyć proces wyszukiwania rekordów wprowadzono indeksy.

Struktura indeksu

Indeks jest strukturą danych umożliwiającą szybki dostęp do wybranych wierszy, w oparciu o wartości jednej lub większej liczby kolumn (klucz indeksu). Ponieważ wartości kluczy indeksów są posortowane, to systemowi zarządzania łatwiej jest wyszukać potrzebne wiersze. Sam indeks zawiera uszeregowaną sekwencję kluczy, razem z danymi o położeniu w tabeli poszczególnych wierszy. Dane w tabeli bazowej są nieposortowane, ale alfabetyczny indeks umożliwia szybkie znalezienie interesujących nas danych -system zarządzania może wówczas na podstawie danych zawartych w indeksie wybrać te wiersze z tabeli, w których widnieją odpowiednie dane, unikając powolnego, sekwencyjnego przeszukiwania całej tabeli.

Dysponując gotowym indeksem, optymalizator zapytań będzie każdorazowo decydował, czy wykorzystanie go przyspieszy aktualną operację i w razie potrzeby odwoła się doń. Ty, jako użytkownik, nie będziesz już musiał przeprowadzać żadnych działań na indeksie, chyba, że postanowisz go skasować. Kiedy przypisujesz tabeli klucz podstawowy, system zarządzania automatycznie tworzy indeks tej tabeli, wykorzystując w charakterze klucza indeksu kolumny klucza podstawowego. W momencie wprowadzania nowego wiersza do tabeli, pierwszą czynnością wykonywaną przez system zarządzania bazą danych jest sprawdzenie czy klucz podstawowy tego wiersza występuje już w indeksie danej tabeli. Tak więc unikalność kluczy podstawowych jest w rzeczywistości kontrolowana na poziomie indeksu, a nie na poziomie tabeli bazowej. Zaletą tego mechanizmu jest znaczne przyspieszenie operacji dopisywania wierszy do tabeli, ponieważ przeszukiwanie indeksu trwa o wiele krócej niż tabeli.

Indeks jest to pomocniczy zbiór danej tabeli zawierający posortowaną według wymaganego klucza informację połączoną z numerami rekordów z tabeli macierzystej, co pozwala łatwo i bardzo szybko zlokalizować dany rekord (patrz rys.8).



Rys. 8 Zastosowanie indeksów do wyszukiwania danych

Szukanie określonego wiersza metodą połowienia

Szukanie danego wiersza metodą połowienia, przeanalizujmy na przykładzie szukania imienia Dorota w tabeli Koledzy (rys. 8). Metoda połowienia realizowana jest następująco:

- dzieli się indeks na pół i sprawdza pierwszą wartość w wierszu, w drugiej połowie tabeli,
- jeżeli jest ona większa od poszukiwanej, to porzuca się drugą połowę i dzieli pierwszą znów na pół. W przeciwnym przypadku porzucamy pierwszą połowę a dzielimy drugą i tak, aż do odnalezienia właściwej wartości,.
- następnie odczytuje się dla niej pozycję (Nr rekordu) i już bardzo szybko można ją zlokalizować, w tym przypadku imię Dorota w tabeli Koledzy.

Za pomocą pliku indeksowego czas dostępu do określonego rekordu jest rzędu **2+log2n,** natomiast przy systematycznym przeszukiwaniu czas ten wynosi n/2, gdzie n jest liczbą rekordów. Weźmy pod uwagę relację składającą się z **4096** wierszy. Przy pomocy pliku indeksowego czas dostępu jest rzędu

$$2+\log_2 4096 = 2+12 = 14$$

Natomiast przy systematycznym przeszukiwanie średni czas jest rzędu;

$$N/2 = 4096/2 = 2048.$$

Porównanie to wykazuje jak efektywne jest stosowanie indeksów do wyszukiwania informacji w tabelach.

Przy korzystaniu z indeksów należy pamiętać co następuje:

- system dopisując nowy rekord automatycznie uzupełnia indeks,
- usuwając rekord z bazy, niektóre systemy baz danych nie usuwającego indeksu,
- odwołania do bazy poprzez indeksy są szybsze,
- nie należy zakładać nadmiernej ilość indeksów gdyż może to spowodować spowolnienie pracy systemu z powodu wydłużenia się czasu dodawania nowych rekordów. Warto pamiętać, że indeksy zajmują dodatkowe miejsce w pamięci RAM i na dyskach.

Jedną z zalet korzystania z indeksu jest możliwość wstępnej weryfikacji unikalności pola rekordu. Jeżeli założymy że zapisy w indeksie są unikalne to system bazy danych przed dopisaniem sprawdza czy dany zapis istnieje. Jeżeli tak, to nie kontynuuje zapisu tylko informuje użytkownika o próbie tworzenia duplikatów. Sprawdzenie to odbywa się dużo szybciej za pomocą indeksu niż bez niego.

TRANSAKCJE

Transakcje to specjalny sposób pracy z bazą danych, zapewniający wysoki stopień integralności i spójności danych (poprawności zapisów i ich celowości). Za przykład weźmy prosty na pozór przypadek zmiany stanów towarów w magazynie po przyjęciu nowej dostawy. Aby prawidłowo odnotować tę fizyczną transakcję należy zmodyfikować następujące tabele:

- w tabeli ***Dostawy*** należy wpisać nowy wiersz dotyczący tej transakcji,
- w tabeli ***Dostawy_pozycje*** należy wpisać tyle nowych wierszy ile przyjmuje się różnych towarów do magazynu,
- w tabeli ***Stany*** należy zmodyfikować te wiersze, które opisują ilościowo i wartościowo stany magazynowe dostarczonych towarów.

Biorąc pod uwagę to, że na tych tabelach równocześnie może pracować jeszcze wielu innych użytkowników mogą wystąpić następujące błędy zapisu:

- nie wprowadzono wiersza do tabeli Dostawy (utrata spójności),
- do tabeli *DostawyPozycje* wprowadzono o jeden wiersz za dużo (brak spójności),
- nie zmieniono kolumny wartość w tabeli ***Stany*** (brak spójności), itp.

Jak widać z tej krótkiej wyliczanki, możliwość poprawnego zakończenia zadania jest jedna, a błędnych możliwości jest parę. Wszelkie błędy zapisu są niedopuszczalne w systemach baz danych. Dlatego tak wiele uwagi projektanci i programiści systemów baz danych poświęcają zaprogramowaniu prawidłowych transakcji.

Każda poprawna transakcja powinna charakteryzować się następującymi właściwościami: niepodzielnością, spójnością, współbieżnością, izolacją, trwałością. Te właściwości rozważono szczegółowo w następnych podpunktach.

Niepodzielność

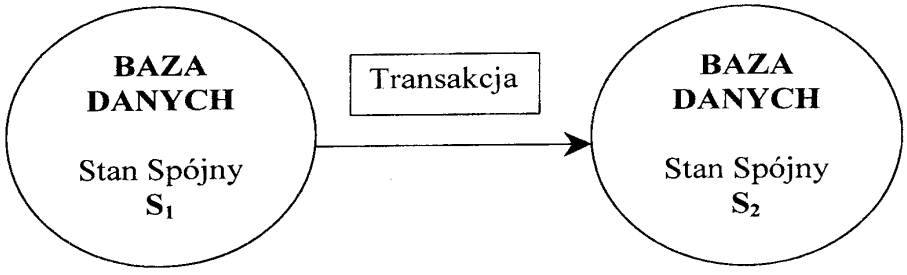
Zwykle transakcja składa się z kilku do kilkudziesięciu elementarnych akcji. System transakcji powinien zapewnić, że albo cała transakcja zostanie wykonana, albo w ogóle nic. Jest to najważniejsza zasada „dobrych" transakcji. Transakcja może być zdefiniowana jako taka najmniejsza logiczna jednostka pracy, która musi być wykonana w całości.

Współbieżność

Obecnie prawie wszystkie bazy danych umożliwiają równoległą pracę wielu użytkowników. Jeśli kilka transakcji jest wykonywanych równolegle na współdzielonej bazie danych, to ich wykonanie musi być zsynchronizowane i wzajemnie izolowane. Istnieje wiele metod kontroli współbieżności. Najbardziej znaną jest metoda blokad dostępu dla innych użytkowników w chwili zapisu.

Spójność

Jak wiadomo baza danych musi być zgodna z rzeczywistością, którą opisuje, co określamy jako spójność bazy danych. Wszystkie transakcje muszą zachowywać spójność i integralność bazy danych. Operacje wykonywane na przykład, przez transakcję modyfikującą nie powinny pozostawiać bazy danych w stanie niespójnym lub niepoprawnym. Dlatego, na przykład, w systemie linii lotniczej nie powinno być możliwości dokonania większej liczby rezerwacji na lot, niż jest miejsc w samolocie. Transakcję zapisu można traktować jako funkcję przejścia bazy danych ze stanu spójnego Si, w inny stan również spójny S_2, a w przypadku transakcji w czasie której następuje jedynie odczyt danych $Si = S_2$. Więzy integralności zdefiniowane dla bazy danych określają warunki początkowe wykonywania aplikacji i są jej niezmiennikiem, czyli w wyniku transakcji baza danych przechodzi z jednego stanu spójnego w drugi stan spójny.



Rys. 9 Transakcja jako niezmiennik spójności bazy danych

Izolacja

Jeżeli transakcja modyfikuje dane dzielone z innymi transakcjami, to te dane mogą być tymczasowo niespójne. Takie dane muszą być niedostępne dla innych transakcji dopóty, dopóki transakcja nie zakończy ich używać. System transakcji musi więc dostarczać iluzji, że dana transakcja działa w izolacji od innych transakcji.

Blokady

Na moment zapisu danych do bazy, dane muszą być „zablokowane" tzn. niedostępne w tym czasie dla innych użytkowników. Istnieją dwie metody blokowania:

- blokowanie totalne - polega na zablokowaniu na początku, transakcji wszystkich wierszy, które będą zmieniane,
- blokowanie optymistyczne - polegające na blokowaniu, kolejnych wierszy w miarę ich modyfikowania.

Blokowanie totalne wydłuża czas dostępu do bazy danych, natomiast blokowanie optymistyczne grozi zakleszczeniem które polega na wzajemnym zablokowaniu sobie dostępu do tabel przez dwóch użytkowników. Obecnie stosuje się na ogół blokowanie optymistyczne, ponieważ zostały opracowane skuteczne metody wychodzenia z zakleszczeń.

Trwałość

Gdy transakcja kończy się, wówczas zmiany dokonane przez nią powinny zostać w pełni utrwalone. To znaczy, nawet w wypadku awarii sprzętu lub oprogramowania powinny one zostać zachowane.

Transakcje opierają się na zasadzie, że jeżeli choć jeden z elementów zadania nie został zakończony pomyślnie to wszystkie zmiany związane z tym zadaniem powinny zostać anulowane. Przebieg transakcji można opisać w pięciu punktach:

- rozpoczęcie transakcji,
- zapisywanie kolejnych zmian tymczasowo w bazie,
- informowanie o rezultacie zakończenia (pomyślny lub nie),
- utrwalanie zmian jeżeli zakończenie transakcji jest pomyślne,
- wycofanie zmian jeżeli zakończenie transakcji niepomyślne.