Spis treści

Część I

Bazy danych

Rozdział 1

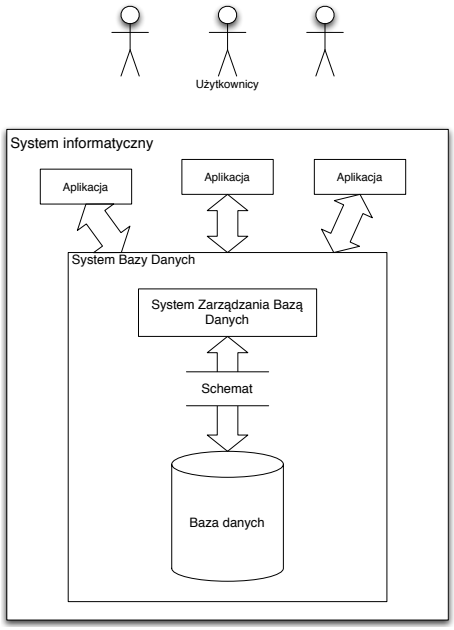
Wstęp

1.1 Motywacja

Przedmiotem badań informatyki są szeroko pojęte zagadnienia związane z kodowaniem, przechowywaniem i przetwarzaniem informacji. Owocem dociekań teoretycznych i rozwoju praktycznych technik są między innymi systemy przystosowane to przechowywania dużych ilości danych i ich obróbki. Takie systemy znajdują zastosowanie w niemal wszystkich dziedzinach działalności komercyjnej. Zazwyczaj system informatyczny (np. system rezerwacji biletów lotniczych) składa się z jednej lub więcej aplikacji, które są obsługiwane przez użytkowników. W przypadku systemu rezerwacji mogą to być: aplikacja obsługująca rezerwacje internetowe, aplikacja dla kasjerów i biur podróży, aplikacja obsługująca weryfikację wstępu pasażerów do samolotu. Te aplikacje są programami komputerowymi, które pracują równocześnie, w wielu kopiach i w różnych miejscach, ale mimo to muszą korzystać ze wspólnego zbioru danych. Ponadto często zdarza się, że aplikacje są znacząco modyfikowane albo wycofywane z użytku, a na ich miejsce wchodzą nowe zapewniające funkcjonalność dostosowaną do bieżących potrzeb. Natomiast dane, na których operują, pozostają niezmienione. W związku z tym korzystne jest oddzielenie aplikacji użytkowych od systemu przechowywania danych.
Istnieje wiele tzw. Systemów Zarządzania Bazą Danych (ang. *Database Management Systems (DBMS)*), które pozwalają na realizację poniższych postulatów:

1. Dane są niezależne od korzystających z nich aplikacji
2. Długi czas przechowywania (kilkadziesiąt lat)
3. Ilość danych przekracza rozmiar pamięci operacyjnej
4. Równoczesny dostęp wielu użytkowników
5. Wydajność
6. Bezpieczeństwo

Rozdzielenie warstwy aplikacji od systemu przechowywania danych jest szczególnie istotne np. w przypadku systemów związanych z bankowością, gdzie przepisy i wprowadzanie nowych produktów wymuszają aktualizację aplikacji i



Rysunek 1.1: Schemat typowego systemu informatycznego

równocześnie może być wymagany dostęp do danych sprzed wielu lat. Ponadto umożliwia tworzenie wysoce wydajnych, bezpiecznych i zachowujących standardy systemów DBMS. Łatwe jest dodawanie nowych aplikacji korzystających z tych samych danych. Teoretycznie możliwa jest również przy minimalnych zmianach w aplikacjach wymiana systemu DBMS na inny np. bardziej wydajny. Podstawowym wymaganiem stawianym bazie danych jest zachowywanie spójności, która jest rozumiana jako zestaw warunków:

1. Zgodność danych z rzeczywistością
2. Zgodność zależności między danymi z przyjętym modelem
3. Odporność na błędy i awarie
4. Brak anomalii związanych ze współbieżnym dostępem
5. Odporność na błędy użytkowników

Warunek 1 jest w pewnym sensie sformułowany życzeniowo, albowiem nie jest możliwa z poziomu systemu komputerowego weryfikacja, czy wprowadzone przez operatora dane są poprawne. Możliwe jest jednak zdefiniowanie pewnych reguł, które muszą być zawsze spełnione. Na przykład, nie można zweryfikować, czy student faktycznie otrzymał piątkę z egzaminu, ale można nie dopuścić do wprowadzenia oceny osobie, która nie jest zarejestrowana na zajęcia lub nie zaliczyła ćwiczeń. Innym aspektem spójności jest odporność na wszelkiego rodzaju błędy. Przykładowo, jeżeli próba zapisu zostanie przerwana w połowie na skutek awarii, system DBMS powinien przywrócić bazę danych do stanu poprzedniego.

Szczególnie ważne jest umożliwienie równoczesnego dostępu danych wielu użytkownikom/aplikacjom. Wiąże się to z koniecznością zagwarantowania, że równoczesne modyfikacje nie naruszą spójności bazy danych. Rozważmy przykładowy scenariusz:

1. Pan Kowalski podchodzi do bankomatu i rozpoczyna wypłatę środków ze wspólnego konta na którym jest 1000 złotych. Bankomat łączy się z bazą danych i sprawdza, że na koncie znajdują się wymagane środki.
2. Pani Kowalska podchodzi do innego bankomatu i rozpoczyna operację wypłaty 1000 zł. Bankomat potwierdza, że środki są dostępne.
3. Pan Kowalski zatwierdza operację. Bankomat zeruje stan konta i wypłaca gotówkę.
4. Pani Kowalska zatwierdza operację. Bankomat jeszcze raz wpisuje zero na konto i wypłaca pieniądze.

W opisanym przypadku następuje naruszenie spójności bazy danych, ponieważ bank wypłaca więcej gotówki, niż znika z konta. Problem ten można rozwiązać grupując operacje sprawdzenia i stanu konta w tzw. transakcję. System DBMS gwarantuje, że transakcje spełniają warunki ACID:

- Atomic – wykonują się w całości albo wcale
- Consistent – nie naruszają spójności bazy danych
- Isolated – są od siebie niezależne
- Durable – wynik zakończonej transakcji nie może zostać utracony

Gdyby w powyższym scenariuszu były zastosowane transakcje, zatwierdzenie jednej z transakcji nie powiodłoby się i jedno z małżonków nie otrzymałoby pieniędzy.
Można wyróżnić dwa podstawowe zastosowania baz danych:

- Przetwarzanie transakcyjne (On-Line Transaction Processing) - OLTP np. systemy ewidencyjne
- Przetwarzanie analityczne (On-Line Analytical Processing) - OLAP np. systemy wspomagające zarządzanie

Systemy OLTP charakteryzują się wielką liczbą prostych operacji (np. rejestracji zdarzeń bankowych, albo sprzedaży). Każda taka operacja dotyczy niewielkiego fragmentu danych. W przypadku systemów OLAP operacji jest relatywnie niewiele, ale zazwyczaj są one skomplikowane i dotyczą dużej ilości danych. Systemy OLAP są wykorzystywane między innymi do wspomagania zarządzania poprzez generowanie pogłębionych analiz zgromadzonych danych. Innym zastosowaniem OLAP jest system zliczający głosy w wyborach powszechnych, albo obsługujący spis ludności.
Występują również specjalizowane systemy baz danych do przechowywania informacji o szczególnej złożonej strukturze. Należą do nich między innymi:

- Computer Aided Desing - CAD
- Geographical Information Systems - GIS
- Protein Data Bank - PDB

1.2 Modele baz danych

Najprostszą i najstarszą strategią przechowywania danych jest składowanie ich w plikach (**model plikowy**), których struktura przypomina tabelę. Do tej pory przechowuje się w ten sposób np. listy użytkowników i hasła w systemach zgodnych z POSIX (np. Unix i Linux – pliki `/etc/passwd` i `/etc/shadow`). Podejście to nie umożliwia definiowania zależności pomiędzy danymi zawartymi w różnych plikach. Dodatkowo operacje na plikach polegające na wstawieniu lub usunięciu wiersza ze środka są niewydajne.
W **modelu hierarchicznym** dane przechowywane są w strukturze drzewiastej. Każdy zapis (z wyjątkiem pierwszego) posiada dokładnie jednego "rodzica". "Rodzic" może mieć wiele dzieci. Dane zorganizowane w ten sposób przypominają strukturę katalogów w systemie plików. W tym modelu można łatwo odwzorowywać wszelkiego rodzaju hierarchiczne związki pomiędzy danymi (np. strukturę organizacyjną przedsiębiorstwa, projekt budynku z podziałem na piętra, pomieszczenia i elementy wyposażenia). Model hierarchiczny nie pozwala natomiast na występowanie tego samego węzła w wielu miejscach hierarchii. Model hierarchiczny stosowany jest między innymi w tzw. rejestrze systemu Windows.
Model sieciowy jest rozwinięciem modelu hierarchicznego, w którym węzeł może mieć wielu rodziców.
Obecnie najczęściej stosuje się bazy danych w **modelu relacyjnym**, który można traktować jako sformalizowane rozwinięcie modelu plikowego. W modelu relacyjnym baza danych składa się z relacji (rozumianych w sensie matematycznym – jako relacje wieloargumentowe nad zbiorami dopuszczalnych wartości), które w sensie potocznym mogą być traktowane jak tabele z dobrze zdefiniowanymi ograniczeniami na rodzaj danych wpisywanych w poszczególne kolumny. Pomiędzy elementami relacji można definiować związki, które odzwierciedlają zależności zachodzące pomiędzy opisywanymi obiektami świata rzeczywistego. Model relacyjny jest dość silnie sformalizowany, co pozwala na łatwe konstruowanie skomplikowanych zapytań i innych operacji na danych.
Obiektowy model baz danych wywodzi się bezpośrednio z technik obiektowego projektowania i programowania. Obiekt posiada określony stan oraz zachowanie i zależności od innych obiektów. Obiekty są składowane w bazie danych. Zaletą takiego podejścia jest to, że w przypadku aplikacji tworzonych z użyciem obiektowego paradygmatu programowania nie ma konieczności każdorazowej konwersji danych zapisanych z bazie na obiekty używane w aplikacji.

Rozdział 2

Modelowanie związków encji

2.1 Wstęp

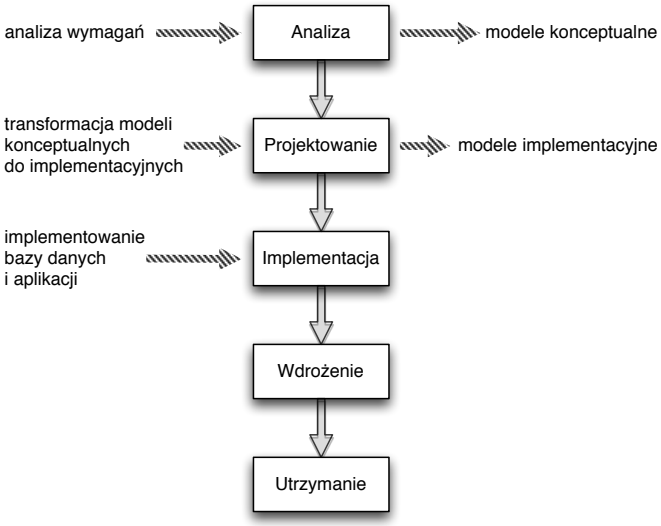
Modelowanie polega na odwzorowaniu obiektów świata rzeczywistego w systemie informatycznym (bazie danych). Jego celem jest stworzenie schematu bazy danych, który będzie zawierał reguły umożliwiające zachowanie spójności. Model musi między innymi zawierać informacje o rodzaju przechowywanych danych i zależnościach pomiędzy nimi.

Zazwyczaj modeluje się na dwóch poziomach abstrakcji. **Modele konceptualne** zawierają wyłącznie informacje wynikające z opisu rzeczywistości, **modele implementacyjne** dodatkowo zawierają założenia o przyjętej technologii realizacji bazy danych. W pewnym sensie model implementacyjny jest projektem wykonawczym bazy danych, zaś model konceptualny zbiorem wymagań, które musi ona spełniać. Modele konceptualne z definicji są niezależne od technik realilzacji bazy danych. Ich analiza może stanowić istotną wskazówkę pozwalającą wybrać właściwą technologię. Modelowanie związków encji (ang. *Entity Relationship Modeling*) jest formalizmem często stosowanym w modelowaniu konceptualnym.

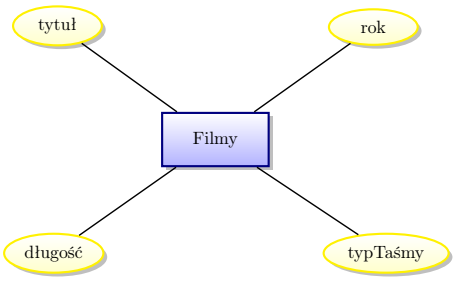
2.2 Encje

Modelowanie rozpoczyna się od zidentyfikowania obiektów świata rzeczywistego, które mają być odwzorowane w bazie danych. Należy pamiętać, że rozważane mogą być zarówno obiekty materialne takie jak towary, pojazdy, nieruchomości albo zasoby ludzkie (np. pracownicy, klienci) jak i niematerialne odpowiadające zdarzeniom (sprzedaż, rezerwacja) lub stanom rzeczywistości (np. stan konta). Obiekty świata rzeczywistego reprezentowane są w modelu związków encji przez encje (ang. *entity*). Kolekcja podobnych encji tworzy zbiór. Ze zbiorem encji związane są atrybuty opisujące zawarte w nim encje.

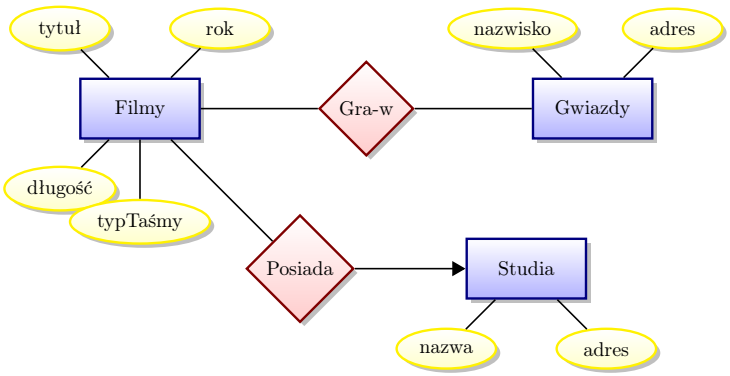
W następujących przykładach będziemy rozważać bazę danych wspomagającą zarządzanie studiem filmowym. Będą w niej przechowywane dane o filmach. Każdy film stanowi encję. Zbiór filmów jest zbiorem encji. Zbiór encji *Filmy* może mieć następujące atrybuty:



Rysunek 2.1: Typowy schemat tworzenia systemu informatycznego



Rysunek 2.2: Graficzna reprezentacja zbioru encji



Rysunek 2.3: Przykładowy diagram związków encji

- tytuł
- rok
- długość

Rysunek 2.2 przedstawia graficzną reprezentację zbioru encji *Filmy*. Zbiory encji oznacza się prostokątami. Atrybuty oznacza się owalami.

2.3 Związki

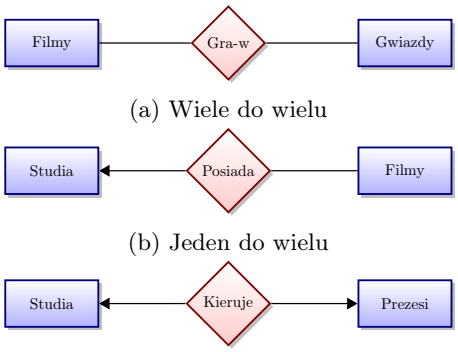
W formalizmie ER zależności pomiędzy zbiorami encji modeluje się przy pomocy związków (ang. *relationship*). Związki mogą obejmować dwa lub więcej zbiorów encji. Nazwa związku powinna jednoznacznie określać rzeczywiste powiązanie, które jest modelowane.

Przykładowo, związek *Gra-w* łączy zbiory *Filmy* i *Gwiazdy*. Film f i gwiazda g należą do związku *Gra-w* jeżeli g występuje w filmie f. Związki na diagramie oznacza się przy pomocy rombów (rys. 2.3). Konkretną instancję związku można przedstawić przy pomocy tabeli. Jeżeli w bazie znajdowałyby się filmy "Nagi instynkt" i "Całe wspomnienie" oraz aktorzy Sharon Stone i Arnold Schwarzenegger, związek *Gra-w* miałby postać:

Filmy	*Gwiazdy*
Nagi instynkt	Sharon Stone
Całe wspomnienie	Arnold Schwarzenegger
Całe wspomnienie	Sharon Stone

2.3.1 Krotność związków

Na rysunku 2.3 od związku *Posiada* do zbioru encji *Studia* poprowadzona jest strzałka. Oznacza ona, że film może być posiadany przez co najwyżej jedno studio. Tego typu właściwość związku nazywamy jego krotnością. Wyróżnia się związku typu:



(a) Wiele do wielu

(b) Jeden do wielu

(c) Jeden do jednego

Rysunek 2.4: Krotność związków

1. Wiele do wielu – np. *Gra-w* pomiędzy filmami i gwiazdami w nich występującymi – Gwiazda w swojej karierze może zagrać w wielu filmach. W filmie może być zaangażowanych wiele gwiazd.
2. Jeden do wielu – np. *Posiada* pomiędzy studiami filmowymi i filmami – Film jest własnością jednego studia. Studio może posiadać wiele filmów.
3. Jeden do jednego – np. *Kieruje* pomiędzy studiami i prezesami – Studio ma co najwyżej jednego prezesa. Prezes może kierować co najwyżej jednym studiem.

Graficzna reprezentacja wymienionych rodzajów związków jest przedstawiona na rysunku 2.4.

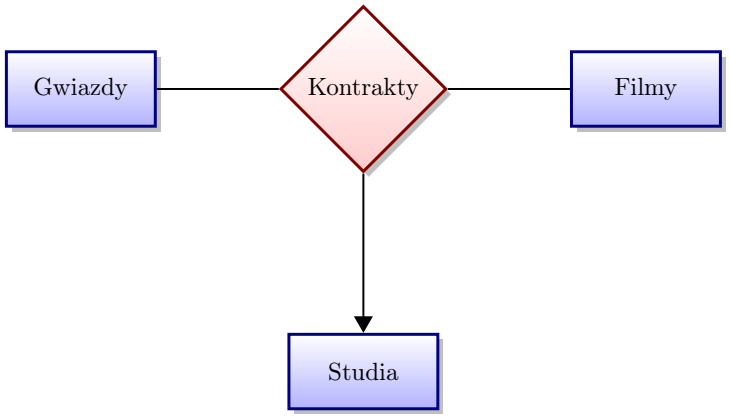
2.3.2 Związki wieloargumentowe

Związki mogą obejmować więcej niż dwa zbiory encji. W przypadku związków wieloargumentowych strzałka oznacza, że pozostająca w związku encja ze zbioru wskazanego strzałką jest jednoznacznie określona przez pozostałe encje w związku. Jest to uogólnienie związku jeden do wielu, gdzie encja po stronie "jeden" jest jednoznacznie określona przez encję po stronie "wiele".

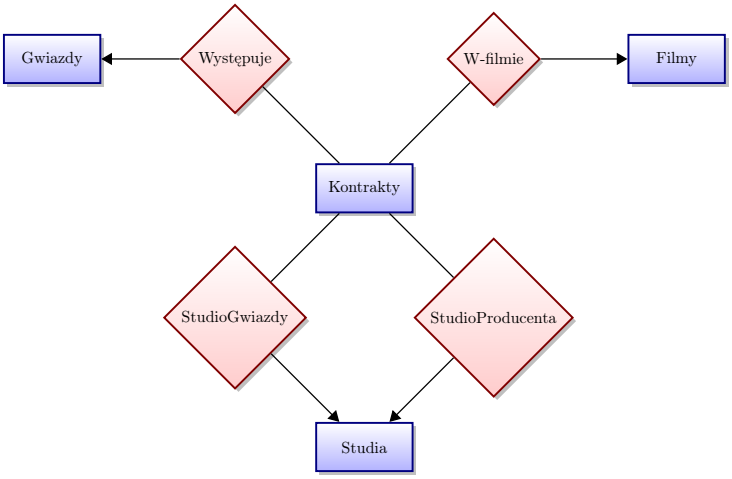
Przykład (rys. 2.5): Związek *Kontrakty* zawiera informację o fakcie występowania gwiazdy w filmie, do którego została zaangażowana przez studio filmowe:

- Gwiazda może zawrzeć kontrakt na występ w konkretnym filmie tylko z jednym studiem.
- Studio może zaangażować wiele gwiazd do jednego filmu.
- Gwiazda może występować w wielu filmach realizowanych przez to samo studio.

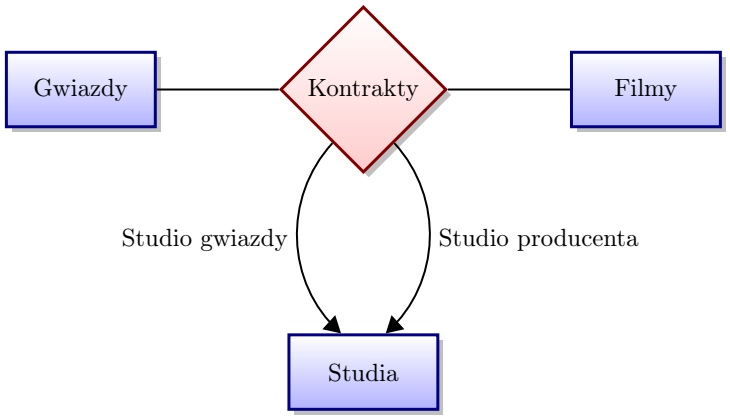
Uwaga (rys. 2.6): Związek wieloargumentowy może zostać przekształcony w zbiór encji i zestawi związków binarnych. Może to jednak prowadzić do utraty informacji o jednoznaczności.



Rysunek 2.5: Przykład związku wieloargumentowego



Rysunek 2.6: Zastępowanie związku wieloargumentowego przez binarne.



Rysunek 2.7: Role w związkach

2.3.3 Role w związkach

Jest dopuszczalne, aby zbiór encji występował w związku wielokrotnie. Jest to konieczne, aby móc odwzorować hierarchię służbową pomiędzy pracownikami. W związku *Jest-przełożonym* zbiór encji *Pracownicy* musi występować dwukrotnie. Encje z tego zbioru mogą pełnić w związku jedną z dwóch ról: *Przełożony*, *Podwładny*. Jeżeli związek nie jest typu "jeden do jednego" encja może pozostawać w związku z wieloma encjami i w każdym przypadku występować w innej roli (Kowalski może być przełożonym Iksińskiego i podwładnym Nowaka).

Przykład (rys. 2.7): Gwiazda może mieć stały kontrakt ze studiem, które "wypożycza" ją innemu studiu do konkretnego filmu. W kontrakcie dotyczącym występu gwiazdy w filmie biorą udział oba studia.

2.3.4 Atrybuty związków

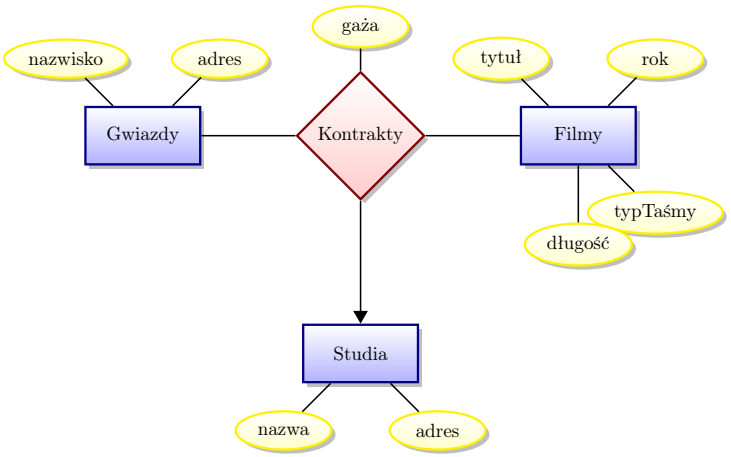
Niejednokrotnie wygodnie jest przyporządkowywać atrybuty bezpośrednio do związków. Formalizm ER to dopuszcza.

Przykład (rys. 2.8): Z każdym kontraktem (reprezentowanym jako instancja związku *Kontrakty*) jest związana informacja o wysokości gaży. Nie może ona być atrybutem żadnego ze zbiorów encji w związku *Kontrakty*. Najprostszym rozwiązaniem jest dodanie atrybutu *gaża* do związku *Kontrakty*. Alternatywnie można utworzyć zbiór encji *Gaże*.

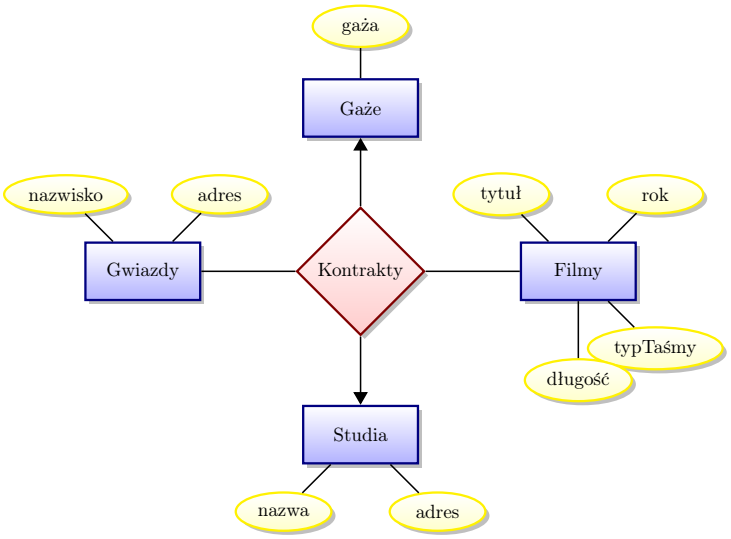
Atrybuty związków nie są niezbędne, ale niejednokrotnie upraszczają model.

2.3.5 Podklasy

W świecie rzeczywistym dość często zdarza się, że obiekt jest szczególnym przypadkiem pewnej szerszej kategorii. Posiada on wszystkie cechy tej kategorii oraz pewne właściwości właściwe tylko dla podkategorii, do której należy. W przypadku podejścia obiektowego taką sytuację modeluje się definiując klasę posiadającą właściwości szerszej kategorii oraz dziedziczącą z niej podklasę, która

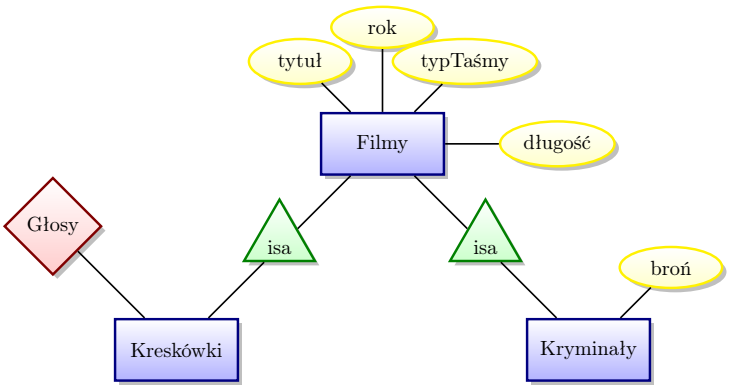


(a)



(b)

Rysunek 2.8: Atrybuty związków



Rysunek 2.9: Podklasy

dodatkowo posiada właściwości podkategorii. W modelowaniu ER stosuje się podobną technikę przy użyciu związku *isa*.

Przykład (rys. 2.9): Niektóre filmy są kreskówkami. Aktorzy w nich nie występują, ale podkładają głos. Jest to modelowane przy użyciu związku *Głosy*. Niektóre filmy są kryminałami. W nich używana jest broń (atrybut *broń*).

2.4 Więzy

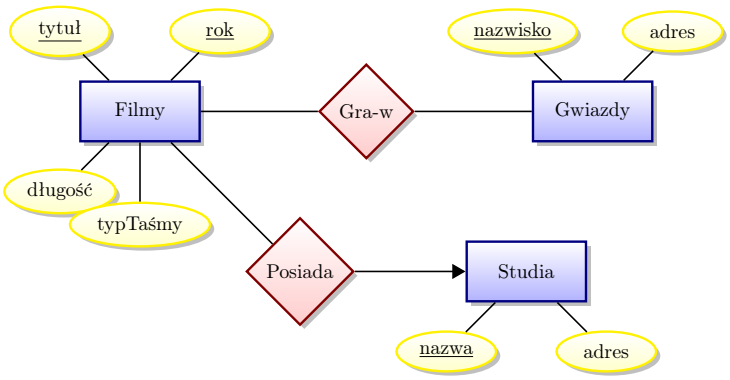
Oprócz zbiorów encji, związków pomiędzy nimi i ich atrybutów konstruując model zazwyczaj trzeba wskazać dodatkowe ograniczenia, które muszą być spełnione, jeżeli baza danych jest spójna. Należą do nich:

1. Klucze – zbiory atrybutów jednoznacznie identyfikujące encję w zbiorze encji
2. Więzy jednoznaczności – wymaganie, aby wartość w określonym kontekście była unikatowa: klucze, związki wiele do jeden
3. Więzy integralności referencyjnej – odwołania muszą wskazywać na obiekty, które faktycznie znajdują się w bazie
4. Więzy domenowe – wymaganie, aby wartość atrybutu należała do określonego zbioru lub zakresu
5. Więzy zasadnicze – inne arbitralnie narzucone warunki

2.4.1 Klucze

Klucz jest zbiorem atrybutów, które pozwalają jednoznacznie zidentyfikować encję w zbiorze, do którego należy. Przykładowo:

1. W zbiorze encji *Studenci* kluczem może być *nr albumu*.
2. W zbiorze encji *Samochody* kluczem może być *nr nadwozia*.



Rysunek 2.10: Przykładowy schemat. Atrybuty kluczowe są podkreślone.

Klucze mogą składać się z więcej niż jednego atrybutu. W zbiorze encji *Mecze* kluczem może być zbiór atrybutów: *drużyna gospodarzy*, *drużyna gości*, *data*.

Niejednokrotnie w zbiorze encji może występować więcej niż jeden potencjalny klucz. Na przykład studenci mogą być identyfikowani jednoznacznie zarówno przy pomocy numeru albumu, jak i PESELa. Zazwyczaj jeden z kluczy wyróżnia się jako klucz główny. Jeżeli zbiór encji należy do hierarchii "isa", wszystkie atrybuty klucza muszą należeć do korzenia hierarchii.

Wybierając klucz w zbiorze encji należy brać pod uwagę, że powinien on być wygodny w stosowaniu. Numer PESEL jest lepszy niż zbiór atrybutów: *nazwisko*, *imię*, *data urodzenia*, *miejsce urodzenia*. W pewnych przypadkach do klucza należą wszystkie atrybuty zbioru encji.

Istotna może również być jego niezmienność w czasie. *Nr rejestracyjny* samochodu nie spełnia tego wymagania. Może się zdarzyć, że wystąpi konieczność jego zmiany, nawet jeżeli samochód nie zmienia właściciela.

Zbiór zawierający dwie encje o jednakowych wartościach klucza nie jest spójny.

2.4.2 Integralność referencyjna

Dzięki określeniu kluczy możliwe jest identyfikowanie encji w możliwie prosty sposób. To z kolei pozwala wyobrazić sobie związek jako zbiór par (lub ogólniej krotek) wartości kluczowy dla powiązanych zbiorów encji. Oczywiście w takim zbiorze mogą pojawiać się tylko wartości kluczy encji faktycznie należących do odpowiednich zbiorów. Dodatkowo w pewnych przypadkach należy określić, że wszystkie encje muszą należeć do danego związku. Tego typu więzy nazywamy więzami integralności referencyjnej.

- Film jest posiadany przez dokładnie jedno studio
- Prezes kieruje dokładnie jednym studiem
- Studio ma co najwyżej jednego prezesa (może być wakat)

Rysunek 2.11: Więzy integralności referencyjnej. Film jest posiadany przez dokładnie jedno studio. Prezes kieruje dokładnie jednym studiem. Studio ma co najwyżej jednego prezesa (może być wakat).

2.4.3 Więzy domenowe

Większość wartości przechowywanych w bazie danych ma dobrze określony typ. Zazwyczaj są to wartości liczbowe całkowite, ułamkowe, walutowe, daty, napisy o określonej długości albo inne ściśle określone (np. PESEL, NIP, współrzędne geograficzne). Oczywiście niecelowe jest, aby dopuścić wprowadzanie wartości o typie nieodpowiadającym atrybutowi. Doprowadziłoby to do utraty spójności bazy danych, ponieważ z analizy problemu wynika, że w rzeczywistości informacje zawsze są danego typu. Dodatkowo dobrze (wąsko) określony typ danych atrybutu pozwala na efektywne przechowywanie danych i ich przeszukiwanie.

Niejednokrotnie, oprócz typu danych, określa się zakres dopuszczalnych wartości. Np. data urodzenia studenta powinna zawierać się w XX (lub XXI) wieku, a wysokość stypendium nieujemna.

2.4.4 Więzy zasadnicze

Poza wymienionymi powyżej rodzajami standardowych więzów (które zazwyczaj są implementowane przy pomocy gotowych mechanizmów systemu DBMS) czasami staje się konieczne zdefiniowanie dodatkowych. Przykładowo:

1. W filmie występuje nie więcej niż 10 gwiazd.
2. Data końca kontraktu nie może następować przed datą rozpoczęcia.
3. Gaża żadnego aktora nie może przekraczać połowy sumy gaż wszystkich gwiazd zaangażowanych do filmu.

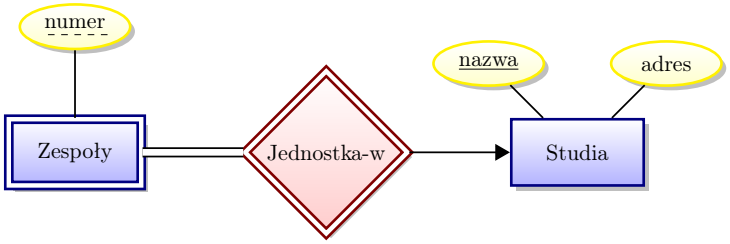
2.5 Słabe encje

Mogą istnieć encje, które nie posiadają wszystkich atrybutów swojego klucza. Dwie standardowe przyczyny przyczyny to:

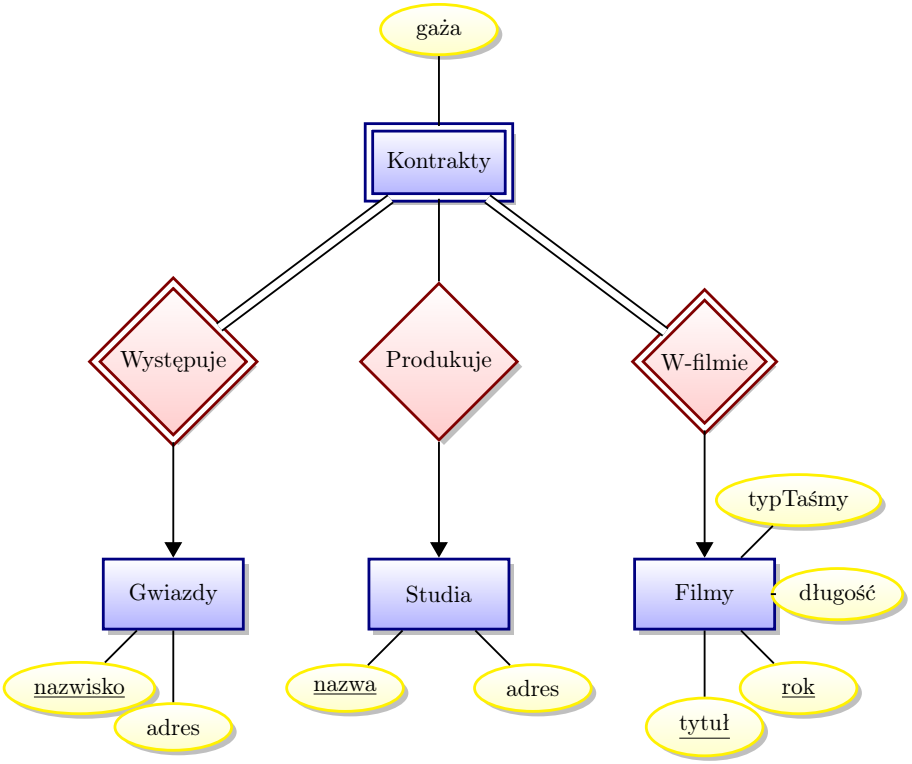
1. Encja jest podjednostką innej encji (ale nie w hierarchii “isa”) i do jej klucza należy klucz encji nadrzędnej (rys. 2.12).
2. Encja opisuje związek wieloargumentowy (rys. 2.13).

Związki typu jeden do wielu, które łączą zbiór słabych encji ze zbiorami encji zawierającymi ich atrybuty kluczowe nazywamy *związkami wspierającymi*.

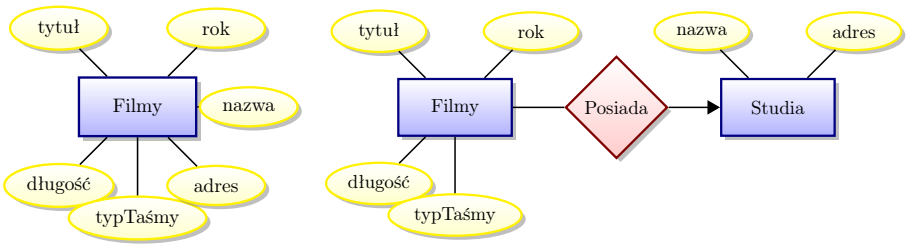
Przykład: Studia mogą mieć zespoły o tej samej nazwie. Para (nazwa, numer) jednoznacznie identyfikują zespół.



Rysunek 2.12: Numer zespołu identyfikuje go jednoznacznie jedynie w obrębie studia.



Rysunek 2.13: Kontrakt jest encją zastępującą związek 3-argumentowy. Ponieważ film jednoznacznie identyfikuje studio, do zidentyfikowania kontraktu wystarczają gwiazda i film.



(a) Źle (b) Dobrze

Rysunek 2.14: Przykład redundancji w modelu ER.

2.6 Zasady projektowania

Projektowanie baz danych i ogólnie systemów informatycznych dla wielu przypomina bardziej sztukę niż rzemiosło. Istnieje jednak kilka szczególnie istotnych aspektów, na które należy zwrócić uwagę, aby uniknąć problemów.

2.6.1 Dokładność

Model musi zawierać te encje, atrybuty i związki, które występują w rzeczywistości i żadne inne. Zawsze należy dokładnie zapoznać się z problemem i zidentyfikować istotne w nim obiekty. Jeżeli w modelu znajdą się nieistniejące encje albo inne obiekty, nie będzie możliwe wprowadzenie ich do stworzonej bazy danych. Dobrym kryterium pozwalającym na określenie, czy zbiór encji od danych atrybutach można umieścić w modelu, jest próba wskazania przykładowych encji wraz z wartościami wszystkich atrybutów.

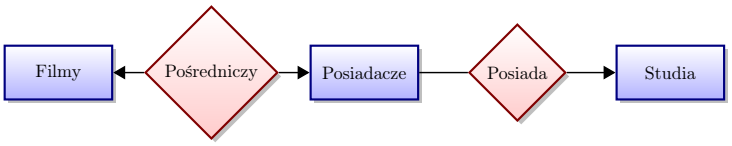
2.6.2 Unikanie redundancji

Modele, które zawierają tę samą informację w wielu kopiach są szczególnie niekorzystne. W szczególności łatwo jest doprowadzić do rozspójnienia bazy danych opartej o taki model poprzez wprowadzenie zmiany do części kopii powielonej informacji. Na rysunku 2.14a przedstawiono zbiór encji *Filmy* z atrybutem *adres*. Wiadomo, że studia filmowe mieszczą się w określonym miejscu i ich adres nie zmienia się w funkcji posiadanych przez nie filmów. Dlatego poprawne jest podzielenie atrybutów zbioru encji *Filmy* i stworzenie zbioru encji *Studia* (rys. 2.14b).

2.6.3 Prostota

Model nie powinien być skomplikowany bardziej niż to konieczne. Dość charakterystycznym przejawem potencjalnie zbędnej komplikacji są związki jeden do jednego. Każdorazowo należy sprawdzić, czy zbiorów encji powiązanych związkiem tego typu nie można połączyć. Być może możliwe jest uniknięcie tworzenia zbiorów encji przez przypisanie ich atrybutów związkom. Niejednokrotnie pomocny może być również związek “isa”.

W przypadku rzeczywistych rozwiązań informatycznych szuka się złotego środka pomiędzy prostotą modelu, a możliwością jego rozbudowy. Może się bowiem



(a) Źle

(b) Dobrze

Rysunek 2.15: Prostota.

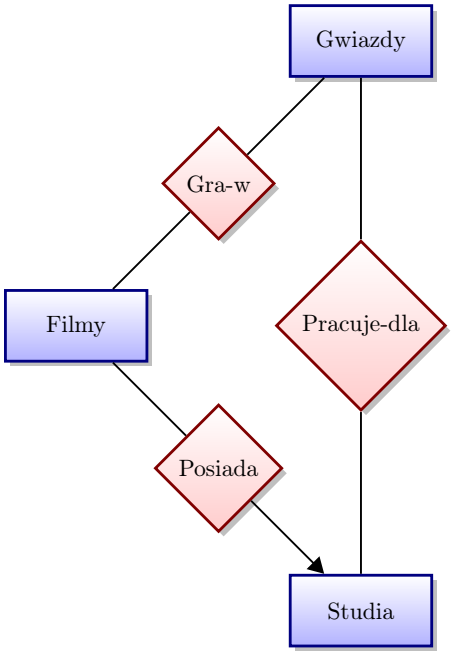
zdarzyć, że pewne zbiory encji lub związki są w danym momencie zbędne, ale ich pozostawienie ułatwi w przyszłości rozszerzenie funkcjonalności systemu. Na przykład, zbiór encji *Posiadacze* nie jest w przedstawionym przypadku konieczny (rys. 2.15), ale jeżeli przewidujemy, że w przyszłości mogłyby pojawić się inne podmioty posiadające filmy, należałoby rozważyć wprowadzenie takiego zbioru.

2.6.4 Wybór właściwych związków

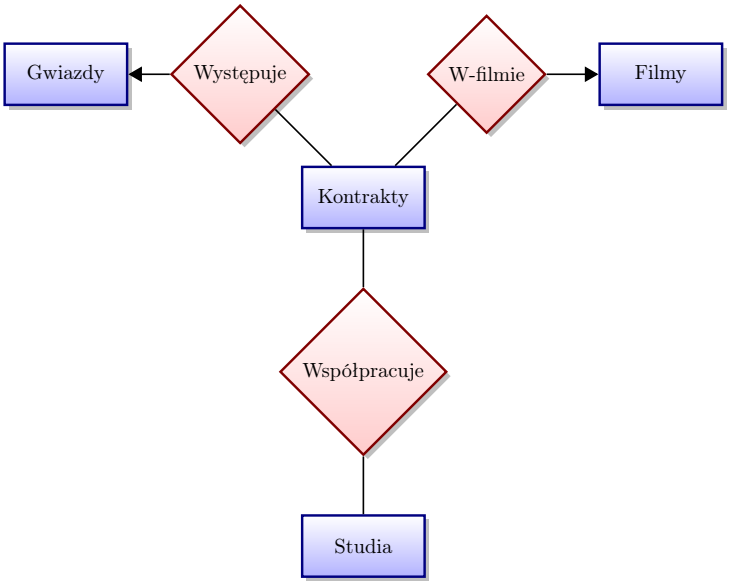
Związki, które występują w modelu muszą poprawnie odzwierciedlać rzeczywistość. Należy zwracać na to uwagę zwłaszcza w przypadku związków, w których krotność przynajmniej jednego zbioru encji jest ograniczona. Niech ilustracją będzie przykład z rysunku 2.16. Mogłoby się wydawać, że przedstawiony schemat jest równoważny z omawianym powyżej zastosowaniem wieloargumentowego związku *Kontrakty*. Tak jednak nie jest. Gwiazda może wystąpić w filmie (związek *Gra-w*) posiadanym przez pewne studio (związek *Posiada*) równocześnie dla tego studia nie pracując. Związki *Gra-w*, *Pracuje-dla* oraz *Posiada* pozwalają na wprowadzenie danych odzwierciedlających związek *Kontrakty*, ale dopuszczają również inne dane. W tym przypadku należy rozważyć, jakie sytuacje w rzeczywistości mogą mieć miejsce i wybrać możliwie restrykcyjny model.

2.6.5 Dobór właściwych elementów

W treści rozdziału zwracaliśmy uwagę, że związki i zbiory encji są do pewnego stopnia wymienne. Związki wieloargumentowe można implementować przy pomocy zbioru encji i związków binarnych. Związki mogą mieć atrybuty. Dlatego ważne jest, aby w modelu konceptualnym stosować związki i zbiory encji w sposób odzwierciedlający naturę opisywanych obiektów. Użycie zbioru encji *Kontrakty* zastępującego związek mogłoby być uzasadnione, jeżeli kontrakt byłby w rozumieniu systemu obiektem “materialnym” z szeregiem atrybutów, miejscem przechowywania itd.



Rysunek 2.16: Niewłaściwie dobrane związki.



Rysunek 2.17: Zbiór encji *Kontrakty* zamiast związku.

Rozdział 3

Model relacyjny

3.1 Podstawowe pojęcia

Iloczynem kartezjańskim zbiorów A i B nazywamy zbiór wszystkich par uporządkowanych (a, b) takich, że $a \in A$ i $b \in B$.

$$(a, b) \in A \times B \iff a \in A \wedge b \in B$$

Pojęcie to można łatwo uogólnić na większą liczbę zbiorów. Iloczynem kartezjańskim zbiorów $A_1, A_2, \ldots, A_n$ dla dowolnego n nazywamy zbiór wszystkich n-tek uporządkowanych $(a_1, a_2, \ldots, a_n)$ takich, że $a_i \in A_i$.

$$(a_1, a_2, \ldots, a_n) \in A_1 \times A_2 \times \ldots \times A_n \iff \forall_{1 \leq i \leq n} a_i \in A_i$$

Relacją nazywamy podzbiór iloczynu kartezjańskiego. Jeżeli $R \subset A_1 \times A_2 \times \ldots A_n$ jest relacją określoną na zbiorach $A_1, A_2, \ldots, A_n$ mówimy, że R jest relacją n-argumentową, a zbiory $A_1, A_2, \ldots, A_n$ nazywamy dziedziną relacji. W kontekście baz danych relacje często reprezentuje się w formie tabelarycznej.

Przykładowo 4-argumentowa relacja *Filmy* mogłaby mieć postać:

tytuł	rok	długość	typFilmu
Gwiezdne Wojny	1977	124	kolor
Potężne Kaczory	1991	104	kolor
Świat Wayne'a	1992	95	kolor

Jej dziedziną są zbiory: napisów, liczb reprezentujących lata, liczb oraz para wartości "kolor" i "czarno-biały". Dodatkowo powiemy, że jej atrybutami są: *tytuł*, *rok*, *długość*, *typFilmu*.

Często opisując relację będziemy posługiwać się notacją zawierającą nazwę relacji oraz jej atrybuty. Taki zapis nazywa się *schematem* relacji. Relacja *Filmy* ma schemat: *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*).

Zazwyczaj projekt relacyjnej bazy danych zawiera kilka relacji. Zbiór schematów relacji projektu nazywamy *schematem bazy danych*, zaś elementy relacji

(wiersze tabeli) nazywane są *krotkami*.

UWAGA: Model relacyjny bierze swoją nazwę od relacji (ang. *relations*) (potocznie tabel), z których składa się baza danych, a nie od związków (ang. *relationships*) pomiędzy zbiorami encji.

3.2 Budowanie schematu relacji na podstawie diagramu E/R

Rozważmy diagram E/R przedstawiony na rysunku 2.3. Znajdujące się w nim zbiory encji można reprezentować relacjami:

- *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*)
- *Gwiazdy*(*nazwisko*, *adres*)
- *Studia*(*nazwa*, *adres*)

Związki z kolei można modelować relacjami zawierającymi klucze powiązanych zbiorów encji:

- *Posiada*(*tytuł*, *rok*, *nazwaStudia*)
- *Gra-w*(*tytuł*, *rok*, *nazwiskoGwiazdy*)

W przypadku związków, w których zbiór encji występuje wielokrotnie w różnych rolach (np. rys. 2.7), każdej roli odpowiada wystąpienie klucza pod inną nazwą atrybutu:

- *Kontrakty*(*nazwiskoGwiazdy*, *tytuł*, *rok*, *studioGwiazdy*, *studioProducenta*)

3.2.1 Łączenie relacji

Jeżeli mamy do czynienia ze związkiem typu "jeden do jednego" albo "jeden do wielu", można rozważyć dołączenie relacji modelującej ten związek, do relacji modelującej odpowiedni zbiór encji. Relację *Posiada* (typu "jeden do wielu"), np.:

tytuł	rok	nazwaStudia
Gwiezdne Wojny	1977	Fox
Potężne Kaczory	1991	Disney
Świat Wayne'a	1992	Paramount

można dołączyć do relacji *Filmy* otrzymując relację o schemacie *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*, *nazwaStudia*) i przykładowej zawartości:

tytuł	rok	długość	typFilmu	nazwaStudia
Gwiezdne Wojny	1977	124	kolor	Fox
Potężne Kaczory	1991	104	kolor	Disney
Świat Wayne'a	1992	95	kolor	Paramount

Podobne działanie jest niewłaściwe w przypadku związków typu "wiele do wielu". Dołączając relację *Gra-w*, np.:

tytuł	rok	nazwiskoGwiazdy
Gwiezdne Wojny	1977	Carrie Fisher
Gwiezdne Wojny	1977	Mark Hamill
Gwiezdne Wojny	1977	Harrison Ford
Potężne Kaczory	1991	Emilio Estevez
Świat Wayne'a	1992	Dana Carvey
Świat Wayne'a	1992	Mike Meyers

otrzymalibyśmy relację *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*, *nazwiskoGwiazdy*):

tytuł	rok	długość	typFilmu	nazwiskoGwiazdy
Gwiezdne Wojny	1977	124	kolor	Carrie Fisher
Gwiezdne Wojny	1977	124	kolor	Mark Hamill
Gwiezdne Wojny	1977	124	kolor	Harrison Ford
Potężne Kaczory	1991	104	kolor	Emilio Estevez
Świat Wayne'a	1992	95	kolor	Dana Carvey
Świat Wayne'a	1992	95	kolor	Mike Meyers

Łatwo zauważyć, że dane takie jak długość i typ filmu są powtórzone tyle razy, ile występuje w nim gwiazd. Redundancja i jej negatywne konsekwencje są szerzej omówione poniżej.

3.2.2 Słabe encje

Rozważmy schemat E/R przedstawiony na rys. 2.12. Schematy relacji zbudowane dla zbiorów encji mają postać:

- *Studia*(*nazwa*, *adres*, *rok*)
- *Zespoły*(*numer*, *nazwaStudia*)

W relacji *Zespoły* musi znaleźć się cały klucz włącznie z atrybutami nienależącymi do zbioru encji *Zespoły*.
Schemat relacji dla związku wspierającego wygląda następująco:

- *Jednostka-w*(*numer*, *nazwaStudia*, *nazwa*)

Łatwo zauważyć, że wartości atrybutuów *nazwaStudia*, *nazwa*, z których jeden należy do klucza zbioru encji *Zespoły*, a drugi do klucza zbioru encji *Studia* są sobie równe. Można zatem pominąć jeden z nich, co prowadzi do schematu relacji zawierającego się w schemacie zbudowanym dla zbioru słabych encji. Ta reguła jest spełniona dla wszystkich związków wspierających. Budując model relacyjny należy je więc pomijać.

3.2.3 Podklasy

Przedstawimy trzy sposoby tworzenia modelu relacyjnego dla hierarchii *isa*. We wszystkich przypadkach będziemy posługiwać się przykładem schematu E/R przedstawionym na rysunku 2.9.

Podejście E/R

Najbardziej naturalne jest traktowanie związku *isa* jako szczególnego przypadku związku wspierającego. Przy takim założeniu zbiory encji podklas traktujemy jak zbiory słabych encji i konstruując dla nich schematy relacji uzupełniamy je o atrybuty kluczowe nadklasy. W analizowanym przykładzie dostaniemy następujące schematy relacji:

- *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*)
- *Kryminały*(*tytuł*, *rok*, *broń*)
- *Kreskówki*(*tytuł*, *rok*)

Relacja *Filmy* zawierać będzie krotki odpowiadające wszystkim filmom. Dodatkowo relacje *Kryminały* i *Kreskówki* zawierać będą krotki odpowiadające filmom odpowiednich rodzajów. Kreskówka kryminalna będzie opisana przez trzy krotki. To rozwiązanie jest szczególnie korzystne jeżeli planuje się wykonywanie wielu operacji na wszystkich filmach niezależnie od ich rodzaju. Przykładowo wyszukanie filmów od zadanej długości i dowolnym rodzaju wymaga jedynie przeszukania relacji *Filmy*.

Podejście Obiektowe

Alternatywnym rozwiązaniem jest takie, w którym każdy film jest reprezentowany przez dokładnie jedną krotkę zawartą w odpowiedniej relacji. W omawianym przypadku konieczne jest stworzenie czterech schematów relacji odpowiadających filmom nie będącym kryminałami ani kreskówkami, kreskówkom niekryminalnym, kryminałom fabularnym oraz kryminalnym kreskówkom. Każdy z tych schematów zawiera wszystkie atrybuty wymagane do opisania encji danego rodzaju.

- *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*)
- *FilmyKres*(*tytuł*, *rok*, *długość*, *typFilmu*)
- *FilmyKrym*(*tytuł*, *rok*, *długość*, *typFilmu*, *broń*)
- *FilmyKrymKres*(*tytuł*, *rok*, *długość*, *typFilmu*, *broń*)

W tym podejściu łatwe jest uzyskanie dostępu do pełnej informacji o encjach konkretnego rodzaju. Mniej wygodne z kolei jest odnalezienie encji spełniającej wskazane warunki, jeżeli nie wiadomo, do których podklas ma należeć (np. znalezienie wszystkich filmów o określonej długości wymaga przejrzenia czterech relacji).

Podejście z wartością NULL

Ostatni sposób rozwiązania jest preferowany ze względu na łatwość implementacji, ale wymaga poczynienia dodatkowych założeń. Jeżeli w rozważanym modelu przynależność do podklasy można wywnioskować na podstawie posiadania przez encję pewnych atrybutów (np. z filmem kryminalnym jest zawsze związany rodzaj broni), można nie tworzyć osobnej relacji dla tej podklasy. W omawianym przykładzie zakładamy, że kryminałami są te filmy, których wartość atrybutu

broń jest niepusta (nie jest NULL). Podobnie kreskówki można identyfikować na podstawie występowania w związku *Głosy*:

- *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*, *broń*)

Takie rozwiązanie jest łatwe w implementacji i wydajne, ale wiąże się z dodatkowymi założeniami o modelu. Przykładowo w tym schemacie nie jest możliwe zareprezentowanie kryminału bez broni, albo niemej kreskówki. Problem ten można ominąć dodając dodatkowe atrybuty typu logicznego *czyKryminał*, *czyKreskówka*.

3.3 Zależności funkcyjne

3.3.1 Definicja

Mówimy, że atrybut B *zależy funkcyjnie* od atrybutów $A_1, A_2, \ldots, A_n$ w relacji R, jeżeli dla dowolnych dwóch krotek z R zgodność wartości atrybutów $A_1, A_2, \ldots, A_n$ implikuje zgodność wartości atrybutu B. Taką zależność zapisujemy następująco:

$$A_1 A_2 \ldots A_n \to B$$

Wiele zależności funkcyjnych:

$$\begin{gathered} A_1 A_2 \ldots A_n \to B_1 \\ \ldots \\ A_1 A_2 \ldots A_n \to B_m \end{gathered}$$

zapisujemy skrótowo jako:

$$A_1 A_2 \ldots A_n \to B_1 \ldots B_m$$

Przykładowo w relacji:

tytuł	rok	długość	typFilmu	nazwaStudia	nazwiskoGwiazdy
Gwiezdne Wojny	1977	124	kolor	Fox	Carrie Fisher
Gwiezdne Wojny	1977	124	kolor	Fox	Mark Hamill
Gwiezdne Wojny	1977	124	kolor	Fox	Harrison Ford
Potężne Kaczory	1991	104	kolor	Disney	Emilio Estevez
Świat Wayne'a	1992	95	kolor	Paramount	Dana Carvey
Świat Wayne'a	1992	95	kolor	Paramount	Mike Meyers

zachodzi zależność funkcyjna:

$$\textit{tytuł rok} \to \textit{długość typFilmu nazwaStudia}$$

Zależność:

$$\textit{tytuł rok} \to \textit{nazwiskoGwiazdy}$$

nie zachodzi. W relacji występują trzy krotki z jednakowymi wartościami atrybutów *tytuł* i *rok* różniące się wartością atrybutu *nazwiskoGwiazdy*.

3.3.2 Klucze

Powiemy, że atrybut lub zbiór atrybutów $\{A_1, A_2, \ldots, A_n\}$ tworzy *klucz relacji* R jeśli:

1. Wszystkie pozostałe atrybuty relacji są funkcyjnie zależne od tych atrybutów,
2. Nie istnieje podzbiór właściwy zbioru $\{A_1, A_2, \ldots, A_n\}$, od którego pozostałe atrybuty relacji R są zależne funkcyjnie.

Jeżeli klucz składa się z jednego atrybutu A, to mówimy, że A jest kluczem. Dowolny zbiór atrybutów, którego podzbiorem jest klucz nazywany jest *nadkluczem.*
Może się zdarzyć, że schemat relacji zawiera więcej niż jeden klucz. Wtedy klucze są od siebie zależne funkcyjnie. Przykładem może być kartoteka studentów, w której każda osoba jest jednoznacznie identyfikowalna przy pomocy numeru indeksu, lub numeru PESEL (w praktyce PESEL może okazać się nieskuteczny ze względu na obcokrajowców). Zazwyczaj jeden z kluczy jest wyróżniany jako klucz podstawowy. Do dobrej praktyki należy konsekwentne używanie klucza podstawowego we wszystkich relacjach modelujących związki odnoszące się do zbioru encji, którego klucz dotyczy. Niektóre klucze mogą być wręcz niepraktyczne w stosowaniu (np. numer dowodu osobistego identyfikuje posiadacza jednoznacznie, ale jest trudno dostępny i zmienia się w czasie).
Atrybuty należące do jakiegokolwiek klucza nazywamy atrybutami *kluczowymi* lub *podstawowymi.* Atrybuty, które nie należą do żadnego klucza nazywa się *niekluczowymi* lub *wtórnymi.*
W przykładowej relacji podanej w sekcji 3.3.1 kluczem jest zbiór atrybutów: *tytuł rok nazwiskoGwiazdy.*

3.3.3 Wnioskowanie z zależności funkcyjnych

Przechodniość

Jeżeli zachodzą zależności funkcyjne $A \to B$ i $B \to C$, to zachodzi zależność funkcyjna $A \to C$.

Dowód:
Weźmy dwie krotki: $t = (a, b_1, c_1)$, $u = (a, b_2, c_2)$

$$A \to B \implies b_1 = b_2$$
$$B \to C \implies c_1 = c_2$$

Zatem każde dwie krotki zgodne dla A są zgodne dla C.

$$A \to C$$

Równoważność i wynikanie

Powiemy, że zbiór zależności funkcyjnych S *wynika* ze zbioru T, jeśli każda instancja relacji spełniająca wszystkie zależności funkcyjne z T spełnia także zależności z S. Tak zdefiniowane wynikanie nie zależy w żadnym stopniu od

danych zawartych w instancji relacji. Jest ono związane wyłącznie z zależnościami funkcyjnymi. Przykładowo, jeżeli $S = \{A \to B, B \to C\}$, to wynika z niego $T = \{A \to C\}$.

- Dwa zbiory zależności funkcyjnych S i T są *równoważne*, jeśli zbiór instancji relacji spełniających S jest równy zbiorowi instancji spełniających T.
- Zbiór zależności funkcyjnych S *wynika* ze zbioru T, jeśli każda instancja spełniająca wszystkie zależności funkcyjne z T spełnia także zależności z S.

3.3.4 Reguły podziału i łączenia

Reguła podziału

Zależność funkcyjną $A_1A_2 \ldots A_n \to B_1B_2 \ldots B_m$ można zastąpić zbiorem zależności funkcyjnych $A_1A_2 \ldots A_n \to B_i$, $i = 1, 2, \ldots, m$.

Reguła łączenia

Zbiór zależności funkcyjnych $A_1A_2 \ldots A_n \to B_i$, $i = 1, 2, \ldots, m$ można zastąpić pojedynczą zależnością funkcyjną $A_1A_2 \ldots A_n \to B_1B_2 \ldots B_m$.

Przykład

$$\left.\begin{array}{l} \textit{tytuł rok} \to \textit{długość} \\ \textit{tytuł rok} \to \textit{typFilmu} \\ \textit{tytuł rok} \to \textit{nazwaStudia} \end{array}\right. \iff \textit{tytuł rok} \to \textit{długość typFilmu nazwaStudia}$$

Uwaga, ta implikacja nie zachodzi:

$$\textit{tytuł rok} \to \textit{długość} \implies \begin{array}{l} \textit{tytuł} \to \textit{długość} \\ \textit{rok} \to \textit{długość} \end{array}$$

3.3.5 Trywialne zależności funkcyjne

Zależność funkcyjna $A_1A_2 \ldots A_n \to B_1B_2 \ldots B_m$ jest

- *trywialna*, jeśli zbiór atrybutów typu B jest podzbiorem zbioru atrybutów typu A;
- *nietrywialna*, jeśli co najmniej jeden z atrybutów typu B nie znajduje się sie pośród atrybutów typu A;
- *całkowicie nietrywialna*, jeśli żaden z atrybutów typu B nie znajduje sie pośród atrybutów typu A.

3.3.6 Reguła zależności trywialnych

Zależność funkcyjna $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ jest równoważna zależności $A_1A_2 \ldots A_n \rightarrow C_1C_2 \ldots C_k$, gdzie C są tymi elementami z B, które nie należą do A.
Przykład:

tytuł rok $\rightarrow$ *tytuł rok* – trywialna
tytuł rok $\rightarrow$ *długość rok* – nietrywialna
tytuł rok $\rightarrow$ *długość typFilmu* – całkowicie nietrywialna

3.3.7 Domknięcie zbioru atrybutów

Domknięciem zbioru atrybutów $\{A_1, A_2, \ldots, A_n\}$ nad zbiorem zależności funkcyjnych S nazywamy maksymalny zbiór atrybutów B dla którego zależność funkcyjna $A_1A_2 \ldots A_n \rightarrow B$ wynika z S. Domknięcie oznaczamy znakiem $^+$:

$$\{A_1, A_2, \ldots, A_n\}^+$$

Domknięcie zawiera wszystkie atrybuty zależne funkcyjnie od domykanego zbioru. Łatwo zauważyć, że dzięki regułom łączenia i podziału dla ustalonego zbioru atrybutów istnieje dokładnie jeden maksymalny (i zarazem najliczniejszy) zbiór będący jego domknięciem.

3.3.8 Algorytm wyznaczania domknięcia

Do wyznaczania domknięcia stosuje się prosty algorytm zachłanny.

1. $X := \{A_1, A_2, \ldots, A_n\}$
2. Znajdź zależność funkcyjną postaci $B_1B_2 \ldots B_m \rightarrow C$, dla których $B_1, B_2, \ldots, B_m$ należą do zbioru atrubutów X, a C nie należy. Wówczas dołącz C do X:
$$X := X \cup \{C\}$$
3. Powtarzaj krok 2 dopóki nie będzie można dołączyć do X żadnego nowego atrybutu.

Przykład:
Relacja: $R(A, B, C, D, E, F)$
Zależności funkcyjne: $S = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$
Obliczymy $\{A, B\}^+$:

1. $X := \{A, B\}$
2. $A, B \in X$, $AB \rightarrow C$, więc $X := X \cup \{C\} = \{A, B, C\}$
3. $B, C \in X$, $BC \rightarrow AD$, więc $X := X \cup \{D\} = \{A, B, C, D\}$
4. $D \in X$, $D \rightarrow E$, więc $X := X \cup \{E\} = \{A, B, C, D, E\}$

Lewa strona $CF \rightarrow B$ nie znajdzie się w X. Ostatecznie:

$$\{A, B\}^+ = \{A, B, C, D, E\}$$

3.3.9 Rzutowanie zależności funkcyjnych

Rzutowanie relacji polega utworzeniu relacji, której schemat zawiera wybrane atrybuty relacji rzutowanej. Poszczególne krotki rzutuje się poprzez usunięcie elementów odpowiadających eliminowanym atrybutom. Pozostaje określić jakie zależności funkcyjne zachodzą w relacji będącej wynikiem rzutowania.
Rozważmy relację $R(A, B, C, D)$ ze zbiorem zależności funkcyjnych $Z_R = \{A \to B, B \to C, C \to D\}$, z której chcemy wyeliminować atrybut B. W rezultacie otrzymamy relację o schemacie $S(A, C, D)$. Bez wątpienia zachodzi w niej zależność funkcyjna $C \to D$, nie mogą zaś zachodzić zależnosci $A \to B$ ani $B \to C$, ponieważ schemat nie zawiera atrybutu B. Czyżby zbiór zależności funkcyjnych w S zawierał tylko jedną zależność?
Zauważmy, że na mocy reguły przechodniości z zależności w Z_R wynika zależność $A \to C$. Obydwa atrybuty A i C należą do schematu S, zatem ta zależność jest spełniona w relacji S.
Z tego przykładu wynika, że przy wykonywaniu rzutowania zbioru zależności funkcyjnych pominięcie wszystkich zależności dotyczących usuwanych atrybutów nie jest uprawnione. Aby poprawnie rzutować należy obliczyć domknięcia wszystkich podzbiorów atrybutów nad zbiorem zależności funkcyjnych rzutowanej relacji i z prawych stron otrzymanych zależności wyrzucić pomijane atrybuty.

3.4 Postacie normalne i normalizacja

W poprzednich rozdziałach podaliśmy zasady budowania modeli konceptualnych i przekształcania ich w modele implementacyjne (w schemacie relacyjnym). Zachowanie tych zasad powinno prowadzić do intuicyjnie poprawnego schematu bazy danych. W tym rozdziale zdefiniujemy formalne kryteria, których spełnienie daje gwarancję, że schemat bazy danych nie zawiera anomalii, oraz metody poprawiania niewłaściwych schematów.

3.4.1 Anomalie

Rozważmy przedstawioną poniżej relację *Filmy*.

tytuł	rok	długość	typFilmu	nazwaStudia	nazwiskoGwiazdy
Gwiezdne Wojny	1977	124	kolor	Fox	Carrie Fisher
Gwiezdne Wojny	1977	124	kolor	Fox	Mark Hamill
Gwiezdne Wojny	1977	124	kolor	Fox	Harrison Ford
Potężne Kaczory	1991	104	kolor	Disney	Emilio Estevez
Świat Wayne'a	1992	95	kolor	Paramount	Dana Carvey
Świat Wayne'a	1992	95	kolor	Paramount	Mike Meyers

Łatwo zauważyć, że umieszczenie w jednej relacji atrybutów opisujących film oraz związku z relacją *Gwiazdy* jest dość niefortunne. Skutkuje ono szeregiem problemów:

- Redundancja – Informacje takie jak długość filmu występują w wielu kopiach (dokładnie tylu, ile gwiazd wystąpiło w filmie). Zwiększa to ilość danych, które muszą być przechowywane i zarazem koszt utrzymania systemu.

- Anomalie przy aktualizacji – Ze względu na redundancję aktualizacja pewnych informacji (np. czasu trwania "Gwiezdnych wojen" wymaga dokonania zmian w relacji. Jeżeli któraś z nich nie doszłaby do skutku (np. na skutek błędu w aplikacji lub awarii sprzętowej) baza danych utraci spójność.
- Anomalie przy usuwaniu – Nie jest możliwe usunięcie pewnych informacji bez usuwania innych (np. nie da się usunąć informacji, że Emilio Estevez zagrał w "Potężnych Kaczorach" bez usuwania pozostałych informacji o tym filmie.
- Anomalie przy wstawianiu – Nie można wpisać informacji o filmie bez podawania co najmniej jednej gwiazdy.

Pomijając potencjalną utratę wydajności wynikającą z redundancji anomalie niosą ryzyko utraty spójności. Sytuacja po nieudanej aktualizacji, gdy występują w relacji dwie konkurencyjne wartości, jest szczególnie niekorzystna ponieważ nie jest możliwe zidentyfikowanie poprawnej wartości, czyli następuje utrata informacji.
Normalizacja schematu bazy danych ma na celu usunięcie powyższych problemów. Polega ona na zidentyfikowaniu relacji, w których mogą wystąpić anomalie lub redundancja, na podstawie zachodzących w nich zależności funkcyjnych. Takie relacje są rozkładane na mniejsze. Proces trwa dopóki istnieją w schemacie problematyczne relacje. Normalizacja jest przeprowadzana w sposób gwarantujący zachowanie wszystkich atrybutów, informacji oraz zależności funkcyjnych. W szczególności możliwe jest odbudowanie relacji sprzed normalizacji na podstawie relacji wynikowych.
Do przeprowadzenia normalizacji konieczna jest znajomość zależności funkcyjnych. W praktycznych zastosowaniach poprawność procesu zależy od pełnej identyfikacji zależności funkcyjnych, którą przeprowadza się na podstawie analizy implementowanego zagadnienia.

3.4.2 Pierwsza postać normalna

Schemat relacji R znajduje się w *pierwszej postaci normalnej* (1NF), jeżeli wartości atrybutów są atomowe. (niepodzielne).
Przykład: Poniższa relacja posiada atrybut *gwiazdy*, który przyjmuje wartości będące zbiorami (relacjami jednoargumentowymi) nazwisk gwiazd. Schemat tej relacji możemy zapisać następująco: *Filmy*(*tytuł*, *rok*, *długość*, *nazwaStudia*, *Gwiazdy*(*nazwisko*)).

tytuł	rok	długość	typFilmu	nazwaStudia	gwiazdy
Gwiezdne Wojny	1977	124	kolor	Fox	Carrie Fisher, Mark Hamill, Harrison Ford
Potężne Kaczory	1991	104	kolor	Disney	Emilio Estevez
Świat Wayne'a	1992	95	kolor	Paramount	Dana Carvey, Mike Meyers

Aby doprowadzić relację do pierwszej postaci normalnej można zastosować poniższy algorytm:

1. Załóżmy, że relacja R zawiera relację zagnieżdżoną P.

2. Tworzymy relację R' dla relacji zewnętrznej R z atrybutów nie zawierających atrybutu złożonego zawierającego relację P.

3. Tworzymy relację P' dla relacji wewnętrznej, do której dodajemy klucz relacji zewnętrznej.

4. Kluczem nowej relacji P' jest suma klucza relacji zewnętrznej R i klucza relacji wewnętrznej P'.

W powyższym przykładzie relacje R, P, R' i P' będą miały schematy:

R**:** *Filmy*(*tytuł*, *rok*, *długość*, *nazwaStudia*, *Gwiazdy*(*nazwisko*))

P**:** *Gwiazdy*(*nazwisko*)

R'**:** *Filmy*(*tytuł*, *rok*, *długość*, *nazwaStudia*)

P'**:** *Gwiazdy*(*tytuł*, *rok*, *nazwisko*)

Jak widać jest to sytuacja analogiczna do zbiorów słabych encji omówionych w 2.5 i 3.2.2.

3.4.3 Druga postać normalna

Powiemy, że:

- zbiór atrybutów B jest *w pełni funkcyjnie zależny* od zbioru atrybutów A, jeżeli $A \to B$ i nie istnieje podzbiór właściwy $A' \subset A$ taki, że $A' \to B$,

- zbiór atrybutów B jest *częściowo funkcyjnie zależny* od zbioru atrybutów A, jeżeli $A \to B$ i istnieje podzbiór właściwy $A' \subset A$ taki, że $A' \to B$.

Schemat relacji R znajduje się w *drugiej postaci normalnej* (2NF), jeżeli żaden atrybut niekluczowy tej relacji nie jest częściowo zależny funkcyjnie od żadnego z kluczy relacji R.
Aby zrozumieć sens tego pojęcia zauważmy, że relacja podana w 3.4.1, nie spełnia 2NF. Kluczem tej relacji jest zbiór atybutów $\{tytuł, rok, nazwiskoGwiazdy\}$, oraz zachodzi zależność $tytuł, rok \to długość, typFilmu, nazwaStudia$. Dlatego informacje o długości, rodzaju i studiu dla danego filmu będą powtórzone tyle razy, ile gwiazd w nim wystąpiło. Widać wyraźnie, że zawsze w przypadku wystąpienia częściowej zależności od klucza atrybuty objęte przez tę zależność można (i należy) wydzielić do nowej relacji (dokonując dekompozycji).

3.4.4 Trzecia postać normalna

Powiemy, że:

- zbiór atrybutów B jest *przechodnio funkcyjnie zależny* od zbioru atrybutów A, jeżeli $A \to B$ i istnieje zbiór atrybutów C, nie będący podzbiorem żadnego klucza, taki, że $A \to C$ i $C \to B$,

- zależność funkcyjna $A \to B$ jest *zależnością przechodnią* jeżeli istnieje podzbiór atrybutów C taki, że zachodzi $A \to C$, $C \to B$ i nie zachodzi $C \to A$ lub $B \to C$.

Założenia o zbiorze C w powyższych definicjach służą zagwarantowaniu, że "przechodniość" jest rzeczywista. Przykładowo, gdyby A i C były różnymi kluczami, dla dowolnego zbioru B zachodziłyby zależności $A \to B$, $A \to C$ i $C \to B$.
Schemat relacji znajduje się w *trzeciej postaci normalnej* (3NF), jeżeli znajduje się w 2NF oraz nie zawiera zależności przechodnich atrybutów niekluczowych od klucza. Z tą definicją równoważny jest warunek, że każda całkowicie nietrywialna zależność funkcyjna $X \to A$ spełnia jeden dwóch warunków:

- X jest nadkluczem,
- A zawiera wyłącznie atrybuty kluczowe.

Rozważmy relację *StudiaFilmowe*(*tytuł*, *rok*, *długość*, *typFilmu*, *nazwaStudia*, *adresStudia*):

tytuł	rok	długość	typFilmu	nazwaStudia	adresStudia
Gwiezdne Wojny	1977	124	kolor	Fox	Hollywood
Potężne Kaczory	1991	104	kolor	Disney	Buena Vista
Świat Wayne'a	1992	95	kolor	Paramount	Hollywood

Relacja ta jest w 2NF. Jej kluczem jest para $\{tytuł, rok\}$, częściowe zależności od klucza nie występują. Problem sprawia atrybut *adresStudia*. Jeżeli w relacji znajduje się więcej niż jeden film wyprodukowany przez dane studio, informacja o jego adresie będzie występować wielokrotnie. Ponadto wystąpią anomalie związane ze wstawianiem i usuwaniem (usunięcie jedynego filmu wyprodukowanego przez studio spowoduje utratę informacji o jego adresie). Rozwiązaniem jest dekompozycja relacji *StudiaFilmowe*(*tytuł*, *rok*, *długość*, *typFilmu*, *nazwaStudia*, *adresStudia*) na relacje *Filmy*(*tytuł*, *rok*, *długość*, *typFilmu*, *nazwaStudia*) i *Studia*(*nazwaStudia*, *adresStudia*).

3.4.5 Postać normalna Boyce'a-Codda

Postać normalna Boyce'a-Codda jest warunkiem nieco silniejszym od 3NF. Zauważmy, że zarówno druga, jak i trzecia postać normalna dotyczą zależności funkcyjnych atrybutów niekluczowych od klucza zakazując odpowiednio występowania zależności częściowych (2NF) i przechodnich (3NF). Istnieją rzadkie przypadki, kiedy zależności funkcyjne pomiędzy atrybutami kluczowymi mogą powodować występowanie anomalii.
Powiemy, że schemat relacji R znajduje się w postaci normalnej Boyce'a-Codda (BCNF), jeżeli dla każdej całkowicie nietrywialnej zależności funkcyjnej $\{A_1, \ldots, A_n\} \to B$ zbiór $\{A_1, \ldots, A_n\}$ jest nadkluczem.
Łatwo zauważyć, że jeżeli relacja jest w BCNF, to jest również w 2NF i 3NF.
Rozważmy przykład relacji będącej w 3NF, w której mogą wystąpić anomalie. Relacja *Projekcje*(*kino*, *miasto*, *tytuł*) zawiera informacje o projekcjach filmów:

kino	miasto	tytuł
Polonia	Warszawa	Gwiezdne Wojny
Polonia	Warszawa	Świat Wayne'a
Moskwa	Warszawa	Potężne Kaczory
Dom Kultury Kolejarza "Stokrotka"	Kogutkowo Górne	Gwiezdne Wojny

Zachodzą w niej następujące zależności funkcyjne:

- *kino* $\to$ *miasto* – nie ma dwóch kin o tej samej nazwie, każde kino gdzieś się znajduje

- *tytuł miasto* $\rightarrow$ *kino* – film może być wyświetlany co najwyżej w jednym kinie w mieście

Zauważmy, że kluczami w tej relacji są pary atrybutów (*tytuł*, *miasto*), (*kino*, *tytuł*). Wszystkie atrybuty są zatem kluczowe i relacja jest w 3NF. Mimo to mogą wystąpić w niej anomalie związane z zależnością *kino* $\rightarrow$ *miasto*. Ta zależność powoduje również, że relacja nie jest w BCNF. Rozważmy jej dekompozycję do relacji *Kina*(*kino*, *miasto*) i *Projekcje*(*kino*, *tytuł*). Relacje te są w BCNF, ale zależność *tytuł miasto* $\rightarrow$ *kino* nie jest zachowana.
Powyższa relacja jest rzadkim przykładem zależności pomiędzy atrybutami, który nie pozwala na całkowite wyeliminowanie redundancji. Poprawnym modelem relacyjnym dla tego zagadnienia są dwie relacje *Projekcje*(*kino*, *miasto*, *tytuł*) i *Kina*(*kino*, *miasto*) połączone więzem integralności referencyjnej (każda para wartości atrybutów (*kino*, *miasto*) w relacji *Projekcje* musi występować w relacji *Kina*).

3.4.6 Normalizacja do relacji do BCNF

Normalizacja schematu bazy danych polega na zidentyfikowaniu relacji, które nie mają wymaganej postaci (będziemy zajmować się normalizacją do BCNF) i dokonywaniu ich dekompozycji, do momentu gdy wszystkie uzyskane relacje osiągną wymaganą postać. Dekompozycja relacji polega na zrzutowaniu jej na dwie relacje, których schematy zawierają wszystkie atrybuty relacji dekomponowanej. Atrybuty wspólne dla obydwu relacji służą do odbudowania zależności funkcyjnych. Załóżmy dla ustalenia uwagi, że dekomponujemy relację $R(A_1, A_2, \ldots, A_n)$ na dwie relacje $S(B_1, B_2, \ldots, B_m)$ i $T(C_1, C_2, \ldots, C_k)$ gdzie:

$$\{A_1, A_2, \ldots, A_n\} = \{B_1, B_2, \ldots, B_m\} \cup \{C_1, C_2, \ldots, C_k\}$$

Schematy relacji S i T są określone przez wybór zbiorów B i C. Krotki tych relacji otrzymujemy przez rzutowanie relacji R. Sposób rzutowania zależności funkcyjnych jest określony w 3.3.9. Oczywiście nie wszystkie wybory zbiorów B i C skutkują dekompozycją, która pozwala na odbudowanie zawartości relacji R na podstawie krotek relacji S i T.
Załóżmy, że w relacji R zachodzi zależność $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$. Jeżeli dokonamy dekompozycji na relacje $S(B_1, B_2, \ldots, B_m)$ i $T(C_1, C_2, \ldots, C_k)$, gdzie zbiór $\{C_1, C_2, \ldots, C_k\}$ zawiera wszystkie atrybuty R, które nie należą do $\{A_1, A_2, \ldots, A_n\}$ ani do $\{B_1, B_2, \ldots, B_m\}$, odbudowanie krotek z R będzie możliwe. Do każdej krotki z T będziemy dołączać krotkę z S, która jest z nią zgodna na atrybutach $\{A_1, A_2, \ldots, A_n\}$. Taka krotka dla ustalonych $\{A_1, A_2, \ldots, A_n\}$ istnieje tylko jedna dzięki zależności funkcyjnej $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$.
Teraz łatwo zauważyć, że normalizację można przeprowadzić znajdując zależności funkcyjne zaburzające BCNF i wykonując dekompozycję według opisanego powyżej schematu. Aby zagwarantować wydajność tego procesu (czyli wykonać możliwie niewiele podziałów), powinno się wybierać możliwie pełne zależności funkcyjne. W szczególności należy prawą stronę zależności funkcyjnej mającej posłużyć do dekompozycji uzupełnić o domknięcie zbioru atrybutów występującego po jej lewej stronie.