1. Bazy danych – wiadomości ogólne

Prowadzenie statku (każda działalność człowieka) poza czynnościami nawigacyjnymi (obsługi) wymaga prowadzenia najróżniejszych dokumentów i archiwizacji danych takich jak np. dziennik okrętowy. Dokonywane są w nich zapisy istotnych informacji, które wraz z upływem czasu mogą osiągnąć znaczne rozmiary. Dokumenty te w istocie umożliwiają gromadzić zbiory danych. Dane te można przeglądać, wynajdywać itd. Ze względu na formę dokumentów (zeszyt, księga itd.) a także formy zapisu (zapis ręczny długopisem) nie są one najwygodniejszym narzędziem zarządzania danymi. W chwili obecnej z pomocą przychodzi informatyka. Jej narzędzia i metody umożliwiają gromadzić dane wszelkiego typu i zarządzać nimi. Coraz częściej znajdują one zastosowanie w nawigacji morskiej.

1.1. Definicja bazy danych

Co to jest baza danych? Odpowiedź na to pytanie, bez wcześniejszego poznania programu MS Access, wyglądałaby następująco: swoista baza danych nie występuje w świecie rzeczywistym jest to zbiór danych posegregowanych według, interesującego konkretną osobę, organizację lub firmę, zjawiska. Przez bazę danych rozumieć można narzędzie do sprawnego i wydajnego przechowywania informacji oraz posługiwania się nią.

Aby wyjaśnić dokładniej termin "baza danych" należy cofnąć się do terminu poprzedzającego to pojęcie a mianowicie "dane". Dane są to wszystkie informacje, które przechowujemy aby mieć do nich w każdej chwili dostęp. W programie Microsoft Access danymi może być tekst, ale również liczby, daty, rysunki, pliki i wiele innych elementów.

Odpowiednim przykładem jest biblioteka. Zbiorem danych posiadanych przez tą instytucje jest lista książek (tytuł, autor oraz ilość książek) oraz lista osób wypożyczających książki (imię, nazwisko, adres do korespondencji, itp.). Interesującym zjawiskiem jest nie samo posiadanie książek, lecz to jaki jest w danej chwili stan posiadania, stan wypożyczenia, ilość osób które wypożyczyły książki oraz ilość osób które zalegają z oddaniem.

Naturalnie taką bazę danych w bibliotece można prowadzić w postaci kartotek, lecz jak dużo wysiłku potrzeba aby sprawdzić interesujące nas zjawisko. Przykładem może być

sprawdzenie, ile osób zalega z oddaniem książki. Aby upewnić się, trzeba sprawdzić kolejno każdą kartotekę książki, która została wypożyczona. Natomiast tworząc bazę danych w odpowiednim programie, wystarczy utworzyć zapytanie w postaci kwerendy, biorąc pod uwagę termin oddania.

Terminologia baz danych jest niemal równie niejasna, jak sformułowanie "programowanie obiektowe". Zwrot "baza danych" może być używany na określenie wszystkiego - począwszy od pojedynczego zbioru danych, takiego jak spis telefonów, aż po skomplikowany zestaw narzędzi, taki jak SQL Server, nie mówiąc już o całej masie obiektów pośrednich. Ten brak precyzji wynika z natury języka.

Chociaż nie można znaleźć analogii do relacyjnych baz danych w świecie rzeczywistym, większość z nich ma na celu modelowanie wybranych aspektów tego świata.**Błąd! Nie można odnaleźć źródła odwołania.** Ten fragment świata nazwać można ***przestrzenią zagadnienia***. Przestrzeń zagadnienia jest z natury dość skomplikowana i nieuporządkowana należy więc ją przybliżyć budując jej model. Podstawą jest ograniczenie tworzonego systemu bazy danych do konkretnego, dobrze zdefiniowanego zbioru obiektów i zależności między nimi.

Przez termin ***model danych*** rozumieć należy opis pojęciowy przestrzeni zagadnienia. Obejmuje on definicję pojęć ogólnych charakteryzujących przestrzeń zagadnienia, np. w firmie zajmującej się sprzedażą produktów mogą być nimi: produkty (encje), oraz opis konkretnego pojęcia. Opisem produktu jest: nazwa produktu, data produkcji, termin przydatności itp. (atrybuty). Model danych obejmuje również opisy powiązań pomiędzy encjami i ograniczeń nałożonych na te powiązania (na przykład każdy kierownik sprzedaży może mieć ograniczenie do maksymalnie pięciu osób bezpośrednio mu podległych). W modelu danych nie ma natomiast żadnych odniesień do fizycznej struktury systemu.

Definicja fizycznej struktury systemu - implementowanych tabel i widoków - stanowi ***schemat bazy danych*** lub po prostu ***schemat***. Jest to przełożenie modelu pojęciowego na jego fizyczną reprezentację, nadającą się do implementacji za pomocą systemu zarządzania bazą danych. Schemat należy do sfery pojęciowej, a nie fizycznej. Schemat to nic innego, jak model danych wyrażony w terminach wykorzystywany następnie do przekazania jego opisu ***aparatowi bazy danych*** (takich jak na przykład tabele czy zdarzenia). Jedna z korzyści płynących z posługiwania się aparatem bazy danych jest uniknięcie zajmowania się fizyczna implementacją.

Po otrzymaniu niezbędnych informacji o charakterze danych, aparat bazy danych może utworzyć za pomocą kodu bądź środowiska interakcyjnego, takiego jak Microsoft

Access, konkretne fizyczne obiekty (zazwyczaj, choć nie koniecznie, gdzieś na dysku twardym). Dopiero te obiekty będą umożliwiały przechowywanie danych. Kombinacja struktury i danych jest tym, co będzie nazywane dalej ***bazą danych***. Taka baza obejmuje zarówno fizyczne tabele, zdefiniowane widoki, kwerendy i procedury przechowywane, jak i reguły wymuszane przez aparat w celu ochrony danych.

Termin "baza danych" nie obejmuje ***aplikacji*** składającej się z formularzy, raportów i korespondencji seryjnej wykorzystywanych bezpośrednio przez użytkownika, ani też dodatków - takich jak oprogramowanie pośredniczące używane do realizacji różnych operacji na poziomie fizycznym. Termin "baza danych" nie obejmuje również aparatu bazy danych.

Do oznaczenia wszystkich tych składników - aplikacji, bazy danych, aparatu bazy danych i oprogramowania pośredniczącego - należy używać terminu ***system bazy danych***. Całe oprogramowanie stanowi, natomiast, część systemu bazy danych.

W przykładzie biblioteki wprowadzone powyżej pojęcia oznaczają kolejno:

- przestrzenią zagadnienia będzie zbiór książek,
- modelem danych będzie opis konkretnych encji i ich atrybutów, np. encja: książka, atrybuty: tytuł, imię i nazwisko autora, wydawnictwo, rok wydania, data zakupu itp.,
- schematem bazy danych będzie przełożenie pojęciowej struktury na fizyczną tzn. spostrzeganie określonych elementów i wiązanie ich w poszczególne problemy, np. czy tworzyć osobną tabelę "Pracownicy", czy wpisać pracowników do tabeli "Osoby_wypożyczające_książki",
- bazą danych będzie kombinacja struktury i danych, tzn. relacji i powiązań a konkretniej tabel powiązanych między sobą,
- aparatem bazy danych będzie narzędzie służące do przechowywania danych, np. Jet lub SQL Server,
- aplikacjami będą przygotowane formularze i raporty oraz korespondencja seryjna, czyli narzędzia bezpośrednio wykorzystywane przez użytkownika, np. formularz dotyczący wypożyczonej książki.

1.2. Typy i modele baz danych

Ze względu na sposób zarządzania bazy danych można podzielić na dwa rodzaje:

- operacyjne bazy danych,
- analityczne bazy danych.

Te pierwsze są to bazy danych, które znajdują zastosowanie w wielu instytucjach, np. w bankach, urzędach, bibliotekach, szkołach itp. Ten typ bazy przechowuje dane ***dynamiczne***, czyli takie, które ulegają ciągłym zmianom i aktualizacjom, a są wykorzystywane tam, gdzie zachodzi potrzeba gromadzenia, przechowywania i modyfikowania danych. Nawiążmy ponownie do przykładowej bazy danych: biblioteki. Dane zawarte w tej bazie muszą być aktualizowane z wielu powodów:

1. zakup nowej książki,
2. przyjęcie nowej osoby, która będzie wypożyczała książki,
3. proces wypożyczania (jedna książka może codziennie zmieniać miejsce wypożyczenia lub zarówno może pozostać nie wypożyczona).

Jest wiele przykładów operacyjnych baz danych i są one najczęściej stosowane, np. bazy inwentaryzacyjne, bazy obsługi klienta, bazy obsługi zamówień, bazy prenumeraty czasopism, bazy zmiany konfiguracji sprzętu, itp.

Natomiast analityczne bazy danych są typem, który przechowuje dane ***statyczne***, czyli takie, które nie ulegają zmianom i zawsze odzwierciedlają stan obiektów z pewnego ustalonego momentu. Można powiedzieć, że bazy te przechowują dane historyczne i informacje związane z pewnymi wydarzeniami. Bazy analityczne przydatne są zarówno do spraw naukowych (tablice chemiczne, próbki geologiczne, dane pomiarowe) jak również w problemach ekonomicznych (wyniki giełdowe z określonego okresu, wskaźniki danych statystycznych, itp.).

Wyróżniamy kilka modeli baz danych:

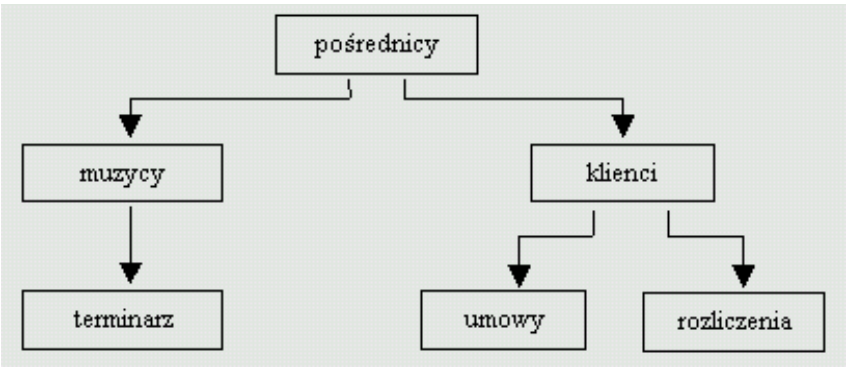
- model hierarchiczny,
- model sieciowy,
- model relacyjny.

1.2.1. Model hierarchiczny:

Model hierarchiczny jest najstarszym modelem bazy danych, z czasem określany został skrótem HMBD (**Hierarchiczny Model Bazy Danych**). W tym modelu dane przedstawione graficznie przybierają strukturę odwróconego drzewa, w którym jedna z tabel pełni role korzenia a cała reszta ma postać gałęzi. Jako przykład rozważmy bazę danych pośredników (agentów),**Błąd! Nie można odnaleźć źródła odwołania.** przedstawiona jest ona schematycznie na rysunku 2. Każdy pośrednik pracuje dla kilku muzyków i ma pewna liczbę klientów, którzy zamawiają u niego obsługę muzyczną różnych imprez. Klient zawiera umowę z danym muzykiem przez pośrednika i u tego pośrednika uiszcza należność za usługę.

Najważniejszymi elementami w każdej bazie danych są relacje, czyli powiązania pomiędzy kolejnymi tabelami. W hierarchicznym modelu bazy danych relacje występują w strukturze ojciec/syn. Co oznacza, że ***tabela-ojciec*** może być powiązana z wieloma ***tabelami-synami***, lecz kolejne tabele-synowie nie mogą przynależeć jednocześnie do kilku nadrzędnych tabel-ojców. Tabele te mogą być powiązane jawnie, przez wskaźniki, lub przez fizyczną organizację rekordów wewnątrz tabel. Aby uzyskać dane w modelu hierarchicznym bazy danych użytkownik musi przedrzeć się począwszy od samej tabeli-korzenia poprzez wszystkie gałęzie do interesującego miejsca, co oznacza, że użytkownik musi bardzo dobrze znać strukturę bazy danych.

Baza danych pośredników



Rysunek 1 Diagram modelu hierarchicznego.

Zalety modelu hierarchicznego:

Jedną z zalet jest to, że poszczególne tabele są ze sobą bezpośrednio powiązane, co daje możliwość przywołania potrzebnych danych. Kolejna zaleta jest taka, że tabele mają automatycznie wbudowaną integralność odwołań Oznacza to, że rekord z tabeli podrzędnej musi być powiązany z istniejącym rekordem w tabeli nadrzędnej, czyli jeżeli skasujemy

rekord z tabeli-ojca, skasowaniu ulegną również wszystkie rekordy w powiązanych z nią tabelach-synów.

Wady modelu hierarchicznego:

Problem zaczyna się pojawiać, gdy chcemy dopisać nowy rekord w tabeli podrzędnej. Jeżeli np. chcielibyśmy dodać nowego muzyka do tabeli MUZYCY a nie miałby on powiązanego rekordu w tabeli POŚREDNICY - nie jest to możliwe. Problem może zostać rozwiązany tylko w takim przypadku, gdy muzykowi przydzieli się konkretnego pośrednika lub w tabeli POŚREDNICY wpiszemy "sztuczne" dane.

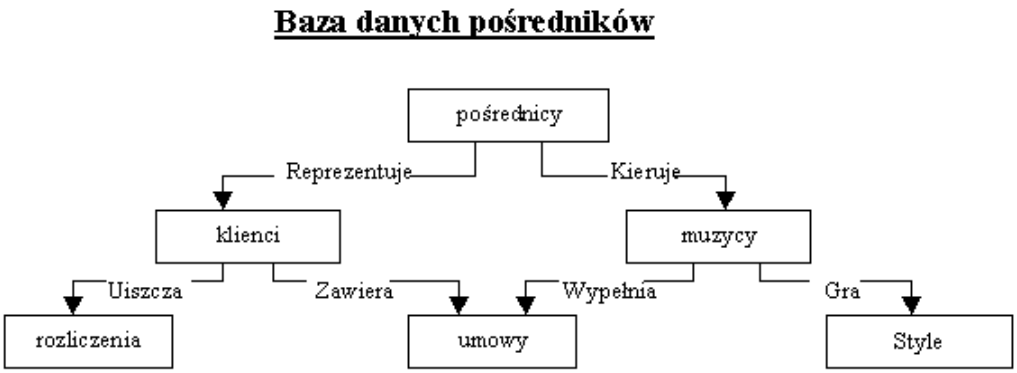
Kolejnym problemem jest nadmierna ilość danych wynikająca z tego, że HMBD nie obsługuje złożonych relacji. Z "Bazy danych pośredników" jasno wynika, że pomiędzy tabelami MUZYCY a KLIENCI występuje relacja wiele-do-wielu: jeden muzyk może pracować dla wielu klientów a klient może zatrudniać wielu muzyków. W takiej sytuacji zachodzi potrzeba wprowadzenia nadmiarowych danych do tabeli TERMINARZ i UMOWY. Pierwsza tabela będzie zawierała dane o klientach, informujące o miejscu występu danego muzyka, lecz dane te pojawiają się również w tabeli KLIENCI. Tabela TERMINARZ będzie również zawierała dane o muzykach, dając nam informację, który muzyk gra dla którego klienta, oczywiście te same dane znajdziemy w tabeli MUZYCY. Aby uniknąć sytuacji w których niektóre dane zostaną wprowadzone w sposób niekonsekwentny, co może zburzyć integralność bazy, możemy utworzyć osobną hierarchiczną bazę danych dla muzyków i drugą - dla pośredników.

Nowa baza muzyków będzie składać się jedynie z tabeli muzyków, natomiast baza pośredników będzie zawierać tabele pośredników, klientów, rozliczeń i umów. Tabela terminarzy nie jest już potrzebna, ponieważ można zdefiniować *logiczną relację ojciec-syn* między tabelą umów w bazie pośredników oraz tabelę muzyków w bazie muzyków. Po wprowadzeniu takiej relacji z bazy będzie można wyczytać informację takie jak: lista muzyków, z którymi zawarł umowy dany klient, czy terminarz występów danego muzyka

Metoda ta spełnia swoje zadanie, jeśli ją zastosujemy. Twórca bazy danych musi być jednak pewien konieczności uwzględnienia takiej relacji. W tym wypadku była ona dość oczywista, lecz wiele relacji jest znacznie bardziej skomplikowanych i potrzeba ich wprowadzenia może zostać zauważona dopiero w bardzo późnych stadiach procesu projektowania lub tez - co gorsza - po oddaniu bazy do użytku.**Błąd! Nie można odnaleźć źródła odwołania.**

1.2.2. Model sieciowy

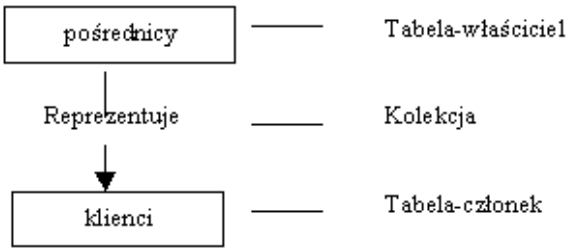
Model sieciowy bazy danych powstał po to aby rozwiązać problemy modelu hierarchicznego. Podobnie jak model hierarchiczny również model sieciowy został obdarzony skrótem (SMBD - **Sieciowy Model Bazy Danych**), a jego struktura wizualnie przedstawia odwrócone drzewo. Różnica pomiędzy HMBD a SMBD polega na tym, że w modelu sieciowym drzewa mogą dzielić ze sobą gałęzie a każde drzewo stanowi część ogólnej struktury bazy danych.



Rysunek 2 Diagram modelu sieciowego.

Baza danych pośredników w przypadku modelu sieciowego przedstawiona jest na przykładzie rysunku 4. Każdy pośrednik pracuje dla kilku muzyków i ma pewną liczbę klientów, którzy zamawiają u niego obsługę muzyczna różnych imprez i uiszczają opłaty. Każdy muzyk może zawrzeć wiele oddzielnych umów i specjalizować się w wielu różnych gatunkach muzyki.**Błąd! Nie można odnaleźć źródła odwołania.**

Sieciowy model logiczny definiuje relacje za pomocą ***kolekcji***. Kolekcja jest to struktura, dzięki której można połączyć dwie tabele nadając jednej z nich rolę ***właściciela*** a drugiej rolę - ***członka***. Kolekcje umożliwiają wprowadzenie relacji jeden-do-wielu, co znaczy, że rekord z tabeli właściciel może być powiązany z wieloma rekordami z tabeli-członka, lecz rekord z tabeli-członka może być powiązany tylko z jednym rekordem z tabeli-właściciel. Podobnie jak w przypadku HMBD, każdy rekord w tabeli podrzędnej (tabeli-członka) SMBD musi być powiązany z istniejącym rekordem w tabeli-właścicielu, np. każdy klient musi być podporządkowany określonemu pośrednikowi, lecz pośrednik może nie posiadać żadnych klientów. Strukturę kolekcji przedstawia rysunek 5.



Rysunek 3 Typowa struktura kolekcji.

Każda tabela może uczestniczyć w wielu różnych kolekcjach, a między każdymi dwoma tabelami można zdefiniować dowolną ilość powiązań. Przykładem może być tabela MUZYCY, która jest powiązana z tabelą STYLE MUZYCZNE przez kolekcję GRA oraz z tabelą UMOWY za pomocą kolekcji WYPEŁNIA. Tabela UMOWY, oprócz tabeli MUZYCY, jest powiązana również z tabelą KLIENCI kolekcją ZAWIERA.

Zalety modelu sieciowego:

W SMBD dostęp do interesujących nas danych można uzyskać poruszając się po kolekcjach i w przeciwieństwie do hierarchicznego modelu bazy danych, przeszukiwanie możemy zacząć od dowolnej tabeli a następnie poruszać się w górę lub w dół po tabelach z nią powiązanych.

Kolejną zaletą jest możliwość tworzenia bardziej skomplikowanych zapytań niż to było możliwe w HMDB, lecz również dużym plusem jest szybkość, z jaką można odczytywać z niego dane.

Wady modelu sieciowego:

Za wadę można uznać to, ze podobnie jak w przypadku hierarchicznego tak i w sieciowym modelu bazy danych należy mieć bardzo dobre wyobrażenie na temat struktury, aby móc szybko i sprawnie poruszać się po bazie danych i uzyskać interesujące nas informacje.

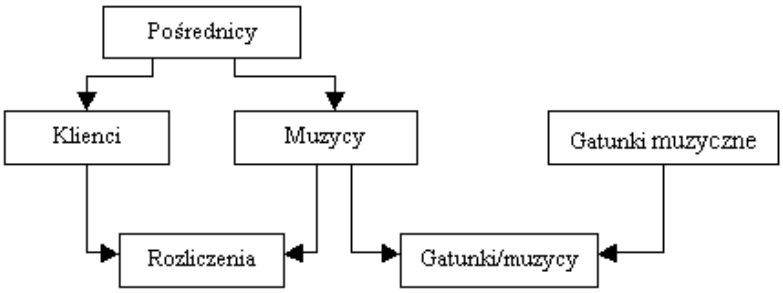
Kolejną wadą jest niemożność zmiany struktury bazy danych bez ponownego tworzenia obsługujących ja programów. Jak już wiemy, relacje w SMBD są zdefiniowane za pomocą kolekcji. Danej kolekcji nie można zmienić bez modyfikowania aplikacji bazodanowej, ponieważ aplikacja ta polega na kolekcjach do wyszukiwania odpowiednich danych. Jeśli któraś kolekcja ulegnie zmianie, wszystkie odwołania do tej kolekcji zawarte w aplikacji obsługującej bazę danych musza również zostać zmienione.**Błąd! Nie można odnaleźć źródła odwołania.**

1.2.3. Model relacyjny:

Obecnie najczęściej stosowanym modelem bazy danych jest relacyjny model logiczny, oznaczony RMBD (**Relacyjny Model Bazy Danych**).

Ważnym przełomem dla obsługi dużych ilości informacji była opublikowana w 1970r. praca E. F. Codd'a p.t. "Relacyjny model logiczny dla dużych banków danych".**Błąd! Nie można odnaleźć źródła odwołania.** Autor zauważa w niej, że zastosowanie struktur i procesów matematycznych w zarządzaniu danymi rozwiązuje wiele problemów trapiących poprzednie wersje. Codd zaprezentował w pracy założenia modelu relacyjnego baz danych. Model ten (RMBD) oparty jest na dwóch gałęziach matematyki - teorii mnogości i rachunku predykatów pierwszego rzędu.

W relacyjnym modelu dane przechowuje się w domenach, czyli tabelach, każda tabela składa się z krotek, czyli rekordów, oraz atrybutów, czyli pól. Fizyczna kolejność pól i rekordów w tabeli jest całkowicie bez znaczenia. Każdy rekord jest wyróżniony przez jedno pole zawierające unikatową wartość. Użytkownik nie musi znać fizycznego położenia rekordu, który go interesuje - to odróżnia RMBD od modelu hierarchicznego i sieciowego.



Rysunek 4 Diagram logicznego modelu relacyjnego.

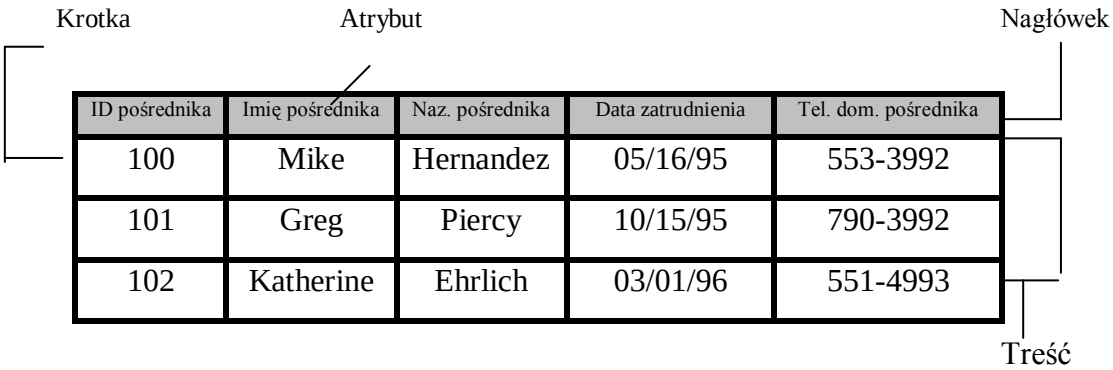
Diagram logicznego modelu relacyjnego**Błąd! Nie można odnaleźć źródła odwołania.** przedstawiony został za pomocą bazy danych pośredników na rysunku 6. Każdy pośrednik pracuje dla kilku muzyków i ma pewna liczbę klientów. Ponadto klienci i muzycy są ze sobą powiązani przez tabelę umów, ponieważ klient może zatrudnić dowolną liczbę muzyków, a każdy muzyk może grać dla wielu różnych klientów. Oprócz tego każdy muzyk reprezentuje jeden lub więcej gatunków muzyki, co odzwierciedla tabela "Gatunki/muzycy".

Mówiąc ogólnie systemy relacyjnych baz danych charakteryzują się następującymi właściwościami:

- wszystkie dane są reprezentowane koncepcyjnie jako zbiór wartości uporządkowanych w formie wierszy i kolumn zwanych relacją,
- wszystkie wartości są skalarne. Oznacza to, że dowolna pozycja relacji na przecięciu kolumny wiersza i kolumny zawiera zawsze jedną i tylko jedna wartość,
- wszystkie operacje są wykonywane na całej relacji, a ich wynikiem jest również relacja. Taka właśnie operacja nazywa się domknięciem.**Błąd! Nie można odnaleźć źródła odwołania.**

Reguła domknięcia - to znaczy, że zarówno tabele podstawowe, jak i wyniki wykonywanych na nich operacji są reprezentowane koncepcyjnie jako relacje - umożliwia wykorzystanie wyników jednej operacji jako danych wejściowych dla innej operacji.**Błąd! Nie można odnaleźć źródła odwołania.** To oznacza, że rezultaty jednej kwerendy mogą stanowić podstawę dla innej kwerendy.

Aby poznać dobrze relacje, trzeba umieć rozpoznać jej podstawowe składniki. Struktura relacji opiera się na wierszu danych. Każdy wiersz danych nazywa się ***krotką***. Liczba krotek w relacji określa jej ***moc***. Każda kolumna krotki nazywana jest ***atrybutem***. Liczba atrybutów relacji określa jej ***stopień.*** Relacja składa się z dwóch sekcji: ***nagłówka*** i ***treści***. Krotki stanowią treść relacji, a nagłówek tytuły kolejnych atrybutów. Strukturę relacji przedstawia rysunek 7.



Rysunek 5 Składniki relacji

Kolejnym ważnym faktem jest to, aby pamiętać, że relacje są pojęciem całkowicie abstrakcyjnym, a z chwilą zaistnienia bazy danych zamieniają się one w "zestawy rekordów" w aparacie Microsoft Jet. W aparacie Jet atrybut nazywa się ***polem***, natomiast krotka

rekordem. W zasadzie terminy te odpowiadają sobie. Należy jednak pamiętać, że relacje są pojęciami abstrakcyjnymi i używa się tej nazwy jedynie w chwili projektowania. Zestawy rekordów są obiektami fizycznymi i występują w bazach danych.

Podobnie dzieje się z nazwą powiązanie. W fazie projektowej powiązanie występuje pomiędzy dwoma "tematami relacji", natomiast w istniejącej bazie danych powiązanie (lub częściej używane sformułowanie: relacja) jest to wiązanie dwóch tabel za pośrednictwem jednego pola. Jeżeli więc mowa jest o relacji w fazie projektowania, należy przez to rozumieć zbiór krotek, atrybutów i zawartych w nich treści. Jeżeli zaś dyskusja toczy się wokół relacji czy też powiązań pomiędzy tabelami, należy przez to rozumieć powiązanie pomiędzy "zestawami rekordów" za pomocą jednego pola w istniejącej już bazie danych.

Relacje w RMBD można podzielić na trzy grupy:

- jeden-do-jednego,
- jeden-do-wielu,
- wiele-do-wielu.

Gdy mówimy o istniejącej bazie danych, każde dwie tabele, między którymi istnieją relacje, są ze sobą jawnie powiązane, ponieważ wiążące je pola mają odpowiadające sobie wartości. Przykłady powiązań przedstawione zostały na rysunku 8. Pole: ID POŚREDNIKA z tabeli POŚREDNICY tworzy powiązanie z tabela KLIENCI. Każdy klient obsługiwany jest przez jednego pośrednika i dlatego jako pole wiążące wybrane zostało pole ID POŚREDNIKA, oczywiste staje się to, że każdy pośrednik może obsługiwać wielu klientów i to może zostać odznaczone w tabeli KLIENCI. Podobne pole ID MUZYKA w tabelach MUZYCY i UMOWY Tym razem zarówno klient jak i muzyk może podpisać wiele umów. Aby zrealizować takie powiązanie należy stworzyć dodatkową tabelę łączącą zarówno dane z tabeli KLIENCI oraz z tabeli MUZYCY. Jak widzimy tabela ta została nazwana UMOWY, łączy ona dane z dwóch tabel i umożliwia relację wiele-do-wielu.

Pośrednicy

ID pośrednika	Imię pośrednika	Naz. pośrednika	Data zatrudnienia	Tel. dom. pośrednika
100	Mike	Hernandez	05/16/95	553-3992
101	Greg	Piercy	10/15/95	790-3992
102	Katherine	Ehrlich	03/01/96	551-4993

Klienci

ID klienta	ID pośrednika	Imię klienta	Naz. klienta	Tel. dom. klienta	.....
9001	100	Stewart	Jameson	553-2236	...
9002	101	Shannon	McLain	125-3365	...
9003	102	Estela	Pundt	129-6695	...

Muzycy

ID muzyka	ID pośrednika	Imię pośrednika	Naz. pośrednika	.....
3000	100	John	Slade	...
3001	101	Mark	Jones	...
3002	102	Teresa	Weiss	...

Umowy

ID klienta	ID muzyka	Data usługi	Czas rozpoczęcia	Czas zakończenia
9003	3001	04/01/98	13:00	15:30
9009	3000	04/13/98	21:00	01:30
9001	3002	05/02/98	15:00	18:00

Rysunek 6 Przykłady powiązanych tabel w relacyjnym modelu baz danych.**Błąd! Nie można odnaleźć źródła odwołania.**

Zalety modelu relacyjnego:

Wielopoziomowa integralność relacyjna danych. Integralność na poziomie pól zapewnia dokładność wprowadzanych danych, integralność na poziomie tabel uniemożliwia powtarzanie się tego samego rekordu i pozostawia niewypełnionych pól wchodzących w skład klucza podstawowego, integralność na poziomie relacji gwarantuje ich odpowiednie

zdefiniowanie, a reguły integralności kontrolują poprawność danych z punktu widzenia tematu bazy.

Logiczna i fizyczna niezależność od aplikacji bazodanowych. Zarówno zmiany wprowadzane przez użytkownika do projektu logicznego, jak i modyfikacje sposobów fizycznej implementacji bazy przez producentów oprogramowania maja znikomy wpływ na działanie aplikacji obsługującej tę bazę.

Zagwarantowana dokładność i poprawność danych. Dane są poprawne i dokładne dzięki wprowadzaniu wielopoziomowej integralności.

Łatwy dostęp do danych. Dane można w prosty sposób odczytywać z pojedynczej tabeli lub z całej grupy powiązanych tabel, najczęściej za pomocą kwerendy.**Błąd! Nie można odnaleźć źródła odwołania.**

W RMBD dostęp do danych uzyskuje się, podając nazwę interesującego nas pola oraz tabeli, do których ono należy. Jak już wcześniej zaznaczono czyni się to za pomocą kwerend. Podstawa każdej kwerendy w Accessie jest język SQL. SQL jest standardowym językiem do wprowadzania, modyfikowania oraz odczytywania informacji z bazy danych. Można zaprojektować większość kwerend przy użyciu prostej siatki projektu, jednak Access i tak przechowa każdą z nich jako instrukcję SQL. Przy zaawansowanych typach kwerend, które korzystają z wyników innej kwerendy do tworzenia kryteriów, znajomość SQL-a jest konieczna do zdefiniowania drugiej kwerendy, zwanej także podkwerendą. Nie wszystkie rodzaje kwerend dostępne w Accessie można tworzyć korzystając z siatki projektu, dla niektórych należy użyć SQL.**Błąd! Nie można odnaleźć źródła odwołania.**

```
SELECT Naz. klienta, Imię klienta, Tel. dom. klienta
FROM Klienci
WHERE Miasto = "Szczecin"
ORDER BY Naz. klienta, Imię klienta
```

Powyżej przedstawione zostało przykładowe zapytanie w języku SQL o spis wszystkich klientów (imię, nazwisko, tel. dom. klienta) z miasta "Szczecin" umieszczonych wcześniej w tabeli KLIENCI.

1.3. Modele danych

Model danych jest najbardziej abstrakcyjnym poziomem projektu bazy danych, ponieważ ogarnia on opis pojęciowy przestrzeni zagadnienia. Modele danych są opisywane w kategoriach encji, atrybutów, domen i powiązań.

Encje:

Trudno jest podać dokładną definicję terminu encja, wprowadza się je w sposób opisowy. Znaczenie pojęcia "encja" jest intuicyjnie zrozumiałe: jest to coś, o czym system musi przechowywać informacje. W "hierarchii" modeli danych encja stoi najwyżej, a pozostałe elementy, tj. atrybuty, domeny, będą określeniami.

Encja jest bardzo ważnym elementem bazy danych, ponieważ to ona charakteryzuje dokładnie przestrzeń zagadnienia bazy danych i używana jest od samego początku tworzenia bazy. Jeżeli początkowo wybierzemy nieprawidłowe encje będzie to oznaczało dwie rzeczy:

- źle poznaliśmy przestrzeń zagadnienia,
- tworzenie bazy danych musimy zacząć od początku!!!

Gdy przystępujemy do tworzenia modelu danych, opracowanie wstępnej listy encji nie jest skomplikowane. W trakcie dyskusji o przestrzeni zagadnienia, większość używanych rzeczowników i czasowników to kandydaci na encje. "Klienci kupują produkty. Pracownicy sprzedają produkty. Dostawcy sprzedają nam produkty." Rzeczowniki "klienci', "pracownicy", "pracownicy" i "dostawcy" są naturalnymi encjami dla modelu danych. Również czasowniki "kupować" i "sprzedawać" są encjami jednak kryją w sobie parę pułapek. Możliwa jest sytuacja, w której "sprzedaż" i "kupno" mogą oznaczać to samo a zostaną źle zrozumiane i w konsekwencji podzielone na dwie encje lub odwrotnie. Czasowniki "sprzedaż" i "kupno" oznaczać będą dwie różne sytuacje a określone zostaną jako jedna encja np. zakup.

W celu przygotowania listy encji dobrze jest oprócz rozmów z klientami przejrzeć dostępne dokumenty związane z przestrzenią zagadnienia. Formularze do wprowadzenia danych, raporty oraz opisy postępowania stanowią dobre źródła kandydatów na encje.

Po opracowaniu wstępnej listy kandydatów na encje, należy sprawdzić czy jest ona kompletna i spójna. Szczególna uwagę powinno się zwrócić na powtórzenia oraz encje, które choć na pozór różne, mogą w praktyce oznaczać to samo. Do przeprowadzenia tego procesu

przydaje się zastosowanie pojęcia ***podtypu***. Przykładem może być sklep z garniturami, w którym raz klient zamawia garnitur wpłacając zaliczkę (ratę), a innym razem klient kupuje garnitur również płacąc tylko pierwsza ratę. W tym momencie spotykamy się z różnym nazewnictwem sytuacji (zamówienie, sprzedaż), która w gruncie rzeczy dla firmy oznacza to samo. Encje podaną w przykładzie można nazwać: zakup, a podtypy przynależące do encji to: zamówienie i sprzedaż.

Większość encji służy modelowaniu obiektów lub zdarzeń występujących w świecie rzeczywistym, takich jak klienci, produkty, pracownicy. Nazywamy je *encjami konkretnymi*. Encje mogą jednak modelować również pojęcia abstrakcyjne tj. fakt, ze pewien sprzedawca jest odpowiedzialny za zamówienia konkretnego klienta. W takim przypadku całe modelowanie sprowadza się do zaprezentowania samego istnienia powiązania.

1.3.1. Atrybuty:

W poprzednim punkcie został opisany najwyżej stojący w hierarchii modeli danych element, encja. Faktem jest, że przyszły system będzie musiał przechowywać również pewne dane dotyczące każdej encji. Dane te nazywane są atrybutami encji.

Najdogodniejszą forma wyjaśnienia tego pojęcia będzie podanie przykładu. Gdy mamy już listę encji, wybieramy kolejno po jednym elemencie i staramy się określić interesującą nas, wokół danej encji, przestrzeń zagadnienia. Przykładowo system ma zawierać encje: Klient. Będą potrzebne różne atrybuty w zależności od miejsca, w którym ma się znajdować dany system, ale na pewno potrzebne będzie imię klienta, nazwisko klienta, adres klienta.

Ustalanie atrybutów, które maja być uwzględnione w modelu, to proces natury semantycznej. Decyzje są podejmowane na podstawie znaczenia danych i tego, w jaki sposób będą one wykorzystywane. Konkretnym przykładem, przy którym mogą pojawić się wątpliwości w trakcie tworzenia modelu danych jest pojedyncza encja: Adres. Jeżeli system ma być stworzony dla osiedlowej biblioteki, która wypożycza książki jedynie osobom zamieszkującym dane osiedle, utworzenie atrybutów dla tej encji jest mało skomplikowane. Atrybutami będą: kod pocztowy, ulica, nr domu, nr mieszkania. Jeżeli jednak tworzymy system dla międzynarodowej firmy wysyłkowej, atrybutami encji Adres będzie dużo więcej elementów: kontynent, kraj, stan, województwo, miasto, dzielnica, osiedle, ulica i nr domu, (ponieważ w wielu krajach pisze się inaczej ulicę i nr lokalu, np. w Australii pierwsza liczba

to nr mieszkania, druga to nr domu a na końcu wpisuje się ulicę), itd. Ilość atrybutów nie można ocenić jedynie po samej nazwie encji bardzo ważna jest przestrzeń zagadnienia, w której opracowujemy bazę danych.

1.3.2. Domeny:

Definicja domeny określa rodzaj danych reprezentowanych przez atrybut. Aby dokładniej określić pojęcie domeny trzeba powiedzieć, że jest to zbiór wszystkich możliwych wartości, jakie może przyjmować atrybut.**Błąd! Nie można odnaleźć źródła odwołania.**

Bardzo często domeny mylone są z typami danych, a w rzeczywistości typ danych to jest np. liczba (pojęcie fizyczne), podczas gdy domena to będzie wiek (charakter logiczny). Kolejnym przykładem mogą być atrybuty: Nazwisko i NazwaUlicy, jeżeli określamy typ danych, obydwa reprezentowane są jako pola tekstowe, lecz oczywistym jest, że charakter logiczny mają inny.

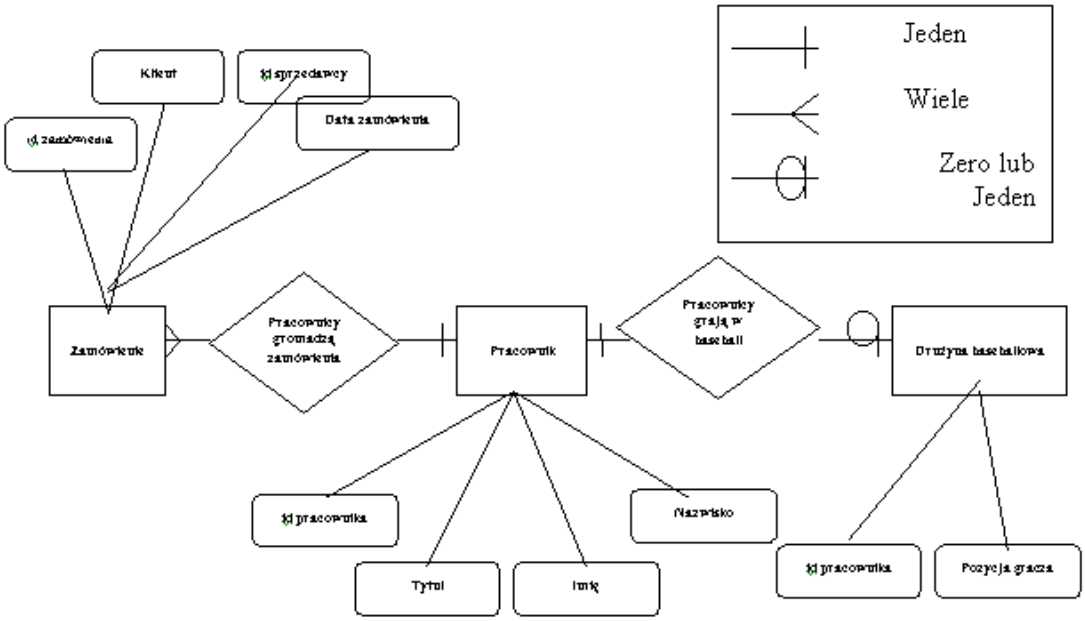
Domena jest dokładniejszym pojęciem niż typ danych, gdyż definicja domeny bardziej poprawnie opisuje dane. Przykładem może być domena Miesiące reprezentująca trzyliterowe skróty nazw wszystkich miesięcy. W schemacie bazy danych atrybut ten może być zdefiniowany jako Text,**Błąd! Nie można odnaleźć źródła odwołania.** ale w praktyce nie będą to oczywiście dowolne trzyliterowe ciągi znakowe, lecz dwunastoelementowy zbiór (STY, LUT, MAR, KWI, MAJ, CZE, LIP, SIE,WRZ, PAŹ, LIS, GRU).

Oczywiście nie wszystkie domeny mogą być zdefiniowane przez proste wyliczenie, przykładem może być domena WiekOsoby, która będzie liczyć około 100 wartości, ale gdy mamy do czynienia z domena WiekEksponatu, liczyć ona może dziesiątki tysięcy. W takich wypadkach wygodnie jest zdefiniować domenę za pomocą reguł umożliwiających sprawdzenie przynależności konkretnej wartości do zbioru stanowiącego tę domenę. Na przykład WiekOsoby można zdefiniować jako "liczba całkowita z przedziału od 0 do 120", a WiekEksponatu jako "liczbę całkowitą równą 0 lub większą".

1.3.3. Diagram powiązań:

Model powiązań między encjami, opisujący dane w terminach encji, atrybutów i relacji, wprowadził w 1976 roku Peter Pin Shan Chen. W tym samym czasie zaproponował on również metodę przedstawiania tych powiązań za pomocą diagramów (diagramów E/R0,

które spotkały się z powszechna akceptacja. W diagramach E/R na oznaczenie encji stosuje się prostokąty, atrybutów - elipsy, a powiązań - romby.**Błąd! Nie można odnaleźć źródła odwołania.**



Rysunek 7 Diagram E/R

Powiązania:

Poza podaniem atrybutów każdej encji model danych musi określać powiązania między nimi. Na poziomie pojęciowym *powiązania* są po porostu związkami łączącymi encje między sobą. W zdaniu "Dostawcy sprzedają produkty" występuje powiązanie pomiędzy encjami Dostawcy i Produkty. Encje występujące w powiązaniu nazywane są jego ***uczestnikami***. Liczba uczestników nazywana jest ***stopniem*** powiązania.

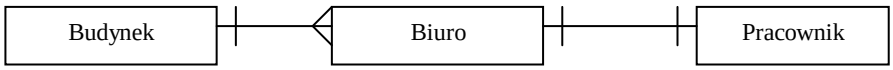
Większość powiązań, z którymi mamy do czynienia to ***powiązania binarne*** (gdzie uczestniczą dwie encje). Często występują również ***powiązania ternarne***, czyli trójczłonowe, są one wynikiem dwóch powiązań binarnych tj.: "Pracownicy sprzedają produkty" i "Klienci kupują produkty", czyli: "Pracownicy sprzedają produkty klientom". Same powiązania binarne nie pozwalają nam określić, którzy pracownicy sprzedali jakieś produkty klientom, na to zezwala nam dopiero powiązanie ternarne. Szczególnym przypadkiem powiązania

binarnego jest encja będąca w powiązaniu sama ze sobą. Tak jest w przypadku, gdy na przykład weźmiemy pod uwagę związek pomiędzy pracownikiem a menedżerem: dowolny pracownik może być menedżerem jak i podlegać innemu menedżerowi.

Powiązania dwóch dowolnych encji może być typu:

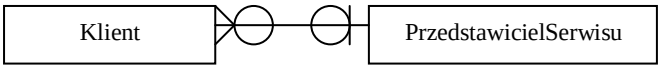
- jeden-do-jednego,
- jeden-do-wielu,
- wiele-do-wielu.

Powiązania jeden-do-jednego są rzadkie i zazwyczaj występują między encjami typu nadrzędnego i podrzędnego. Przykładem może być powiązanie pomiędzy encjami Sprzedawca i SzczegółoweDaneSprzedawców, bo każdy sprzedawca ma indywidualne dane chociażby PESEL itp. Kolejny przykład podany jest na rysunku 11. Tu relacja jeden-do-jednego występuje pomiędzy tabelami Biuro i Pracownik. Relacja taka ma sens, gdy występuje zapytanie: jak często zmienia się użytkownik biura.



Rysunek 8 Przykładowe powiązanie jeden-do-jednego.

Powiązania jeden-do-wielu są prawdopodobnie najbardziej typowe. Na przykład zamówienie obejmuje zwykle wiele produktów, a jeden sprzedawca wystawia wiele faktur. Oba te powiązania są typami jeden-do-wielu. Rysunek 12 przedstawia typ relacji jeden-do-wielu, gdzie encja PrzedstawicielSerwisu stoi po stronie jeden, natomiast Klient stoi po stronie wielu. Dzieje się tak, ponieważ biorąc pod uwagę przestrzeń zagadnienia każdy przedstawiciel serwisu może obsługiwać wielu klientów.



Rysunek 9 Przykładowe powiązanie jeden-do-wielu

Powiązania wiele-do-wielu, chociaż nie tak powszechne jak jeden-do-wielu, również spotykane są dość często. Klienci kupują wiele produktów a produkty są kupowane przez wielu klientów. Nauczyciele maja wielu uczniów, a uczniowie są nauczani przez wielu nauczycieli. Powiązania wiele-do-wielu nie dają się bezpośrednio implementować w relacyjnym modelu baz danych, lecz istnieje dość naturalny sposób implementowania ich metoda pośrednią. Metoda pośrednia polega na modelowaniu powiązania za pomocą trzeciej

tabeli, tzw. ***tabeli sprzęgającej***. Tabela sprzęgająca zawiera zazwyczaj tylko dwa atrybuty pobrane z encji uczestniczących w relacji, dzięki którym ma dojść do powiązania. Dzięki modelowaniu powiązań wiele-do wielu uzyskuje się dwa powiązania jeden-do-wielu, co przedstawione zostało na rysunku 13.

Rysunek 10 Modelowanie powiązań wiele-do-wielu.

Uczestnictwo jakiejś encji w powiązaniu może być częściowe lub całkowite. Jeśli obecność encji w powiązaniu jest warunkiem jej istnienia, to mówimy o uczestnictwie całkowitym. W przeciwnym wypadku jest to uczestnictwo częściowe. Na przykład nie ma sensu istnienie szczegółowych danych Sprzedawcy, o ile nie istnieje odpowiadający im Pracownik. Odwrotne stwierdzenie nie jest już jednak prawdziwe. Pracownik niekoniecznie musi być sprzedawcą. Tak, więc rekord Pracownik może istnieć nawet wtedy, gdy nie ma odpowiadającego mu rekordu Sprzedawca. Oznacza to, że uczestnictwo encji Pracownik w powiązaniu jest częściowe, a encji Sprzedawca całkowite.**Błąd! Nie można odnaleźć źródła odwołania.**

1.4. Struktura bazy danych

Aby omówić strukturę bazy danych należy przede wszystkim zająć się aspektami projektowania modelu relacyjnego. Podstawowym celem fazy projektowania jest:

a) zagwarantowanie modelowi zdolności do udzielenia odpowiedzi na każde rozsądne pytanie w ramach przestrzeni zagadnienia,

b) zminimalizowanie redundancji i związanych z nią problemów.

Redundancja jest to powtarzanie jednego atrybutu w kilku tabelach. Jest to bardzo często stosowane w bazach danych, w których odpowiedzi na konkretne pytania udzielane są już za pośrednictwem tabel. Redundancja wiąże się bezpośrednio z marnotrawstwem zasobów oraz stwarza skomplikowane problemy. Jeżeli ustalimy, że w tabeli Umowy będą znajdowały się informacje dotyczące pracowników (czyli nie utworzymy osobnej tabeli Pracownik), przy

każdym odnawianiu umowy informacje w polach: ImięPracownika, NazwiskoPracownika itp. pozostaną bez zmian. Natomiast pole DataZatrudnienia (czyli data podpisania umowy), będzie ulegało zmianie. W takim przypadku stracimy informacje o rzeczywistej dacie zatrudnienia pracownika i nie będziemy mieli możliwości obliczenia stażu pracy.

IDUmowy	DataZatrudnienia	ImięPracownika	NazwiskoPracownika	NrTelPracownika	.....
101	02/01/05	Hanna	Misiak	4690-456	...
102	02/01/05	Konrad	Grzegorczyk	4896-789	...
103	02/03/05	Rafał	Jackowski	4632-951	...
104	02/01/05	Marcin	Guzeł	4691-486	...
105	02/03/05	Halina	Grzyb	4961-852	...
106	02/04/05	Halina	Grzyb	4961-852	...

Rysunek 11 Duplikowanie danych w niepoprawnie zbudowanej tabeli

Jeżeli zdecydujemy się w podobnej sytuacji wpisywać wszystkie nowe dane do tabeli Umowy jako nowy rekord, może zdarzyć się, że doprowadzi to do wielu pomyłek. Gdy pracownik zmieni nr telefonu i my tę informacje zachowamy w tabeli Umowy w nowym rekordzie (uwzględniając oczywiście wszystkie dane, które się nie zmieniły), to przy potrzebie zatelefonowania do tego właśnie pracownika istnieje możliwość wybrania nieprawidłowego numeru telefonu. Przykłady tego typu przedstawia rysunek nr 14.

Pracownik

IDPracownika	ImięPracownika	NazwiskoPracownika	NrTelPracownika	.....
8	Henryk	Podwika	4690-336	...
9	Karolina	Mazur	4523-963	...
10	Paulina	Piłat	2513-963	...
11	Bartosz	Dobosz	4569-412	...
12	Dobromiła	Miłosz	4569-842	...

Umowy

IDUmowy	DataZatrudnienia	ImięPracownika	NazwiskoPracownika	NrTelPracownika	.....
102	02/02/05	Henryk	Podwika	4690-336	...
103	02/02/05	Karolina	Mazur	4523-963	...
104	02/03/05	Paulina	Piłat	2513-963	...
105	02/03/05	Bartosz	Dobosz	4125-963	...
106	02/03/05	Dobromiła	Miłosz	4569-842	...

Rysunek 12 Przykład anomalii aktualizacji w nieprawidłowej relacji.

Problemy tego typu, nazywane zazwyczaj ***anomaliami aktualizacji***, stają się jeszcze gorsze, jeśli redundantne dane są przechowywane w więcej niż jednej relacji. Z sytuacją,

którą widzimy na rysunku 15 spotkamy się często jeżeli te same dane są użyte w dwóch tabelach. W takiej strukturze wystarczy w tabeli umowy podać jedynie pole IDPracownika aby nie powtarzać wszystkich pól i nie doprowadzić do sytuacji, gdzie aby zmienić jedną daną należy tę wartość wprowadzać równocześnie do dwóch tabel, w innym wypadku jakakolwiek relacja jest błędna. W tej sytuacji pracownik zmienił numer telefonu i poinformował o tym fakcie, ale zmieniono wartość tylko w jednej tabeli.

Aby nie zdarzały się takie sytuacje jak opisane wcześniej należy bardzo poważnie rozpatrzyć wszystkie encje znajdujące się w przestrzeni zagadnienia. Nadać im numer indentyfikacyjny (IDUmowy, IDPracownika), i tworzyć relacje łączac tabele za pomocą identyfikatorów. Przykład z rysunku 15 został poprawiony i przedstawiono go na rysunku 16. W takiej sytuacji struktura relacji nie zostanie zachwiana. Dane pracownika zmieniamy w tabeli Pracownik, co jednocześnie podkreślone zostaje w tabeli Umowy.

Pracownik

ID Pracownika	Imię Pracownika	Nazwisko Pracownika	Nr Tel. Pracownika	.....
8	Henryk	Podwika	4690-336	...
9	Karolina	Mazur	4523-963	...
10	Paulina	Piłat	2513-963	...
11	Bartosz	Dobosz	4125-963	...
12	Dobromiła	Miłosz	4569-842	...

Umowy

ID Umowy	Data Zatrudnienia	ID Pracownika	.....
102	02/02/05	8	...
103	02/02/05	9	...
104	02/03/05	10	...
105	02/03/05	11	...
106	02/03/05	12	...

Rysunek 13 Prawidłowo utworzone powiązanie między tabelami.

Model relacyjny pozwala sprzęgać ze sobą relację na różne sposoby za pomocą łączenia atrybutów. Proces uzyskiwania w pełni znormalizowanego modelu danych wymaga usunięcia redundancji poprzez dzielenie relacji w taki sposób, aby uzyskane w efekcie relacje mogły być znowu połączone bez straty żadnej informacji. Jest to reguła ***bezstratnej dekompozycji*** i została przedstawiona za pomocą rysunków 15 i 16.

Bardzo ważnym elementem każdej relacji jest to aby kombinacja, składająca się z jednego lub więcej atrybutów jednoznacznie identyfikowała każdą jej krotkę. Taka kombinacja nazywana jest ***kluczem kandydującym***. Z definicji wszystkie relacje muszą mieć

przynajmniej jeden klucz kandydujący: zbiór wszystkich atrybutów składających się na krotkę. Klucze kandydujące mogą składać się z pojedynczego atrybutu (***klucz prosty***) lub z wielu atrybutów (***klucz złożony***). Klucz kandydujący musi spełniać pewien warunek - musi być nieredukowalny. Zatem zbiór wszystkich atrybutów niekoniecznie musi być kluczem kandydującym. Na poziomie logicznym posługujemy się terminem "klucz kandydujący", ale gdy przejdziemy na etap implementacji używamy pojęcia "klucz podstawowy".**Błąd! Nie można odnaleźć źródła odwołania.**

Czasami w relacji występuje wiele kluczy kandydujących. W takiej sytuacji jeden z kluczy kandydujących (dowolnie wybrany) określa się jako ***klucz główny***, a pozostałe jako ***klucze alternatywne***. Wybór klucza głównego nie ma wielkiego znaczenia na poziomie logicznym.

Zdarza się również, że dany atrybut nie spełnia zadania jako klucz kandydujący i podobnie dzieje się ze zbiorem. W takiej sytuacji najlepiej jest skorzystać z generowanego przez system identyfikatora liczbowego, takiego jak Autonumerowanie. Należy wtedy pamiętać, aby nie nadawać mu żadnego innego znaczenia.

Omawiane dalej ***reguły normalizacji*** są narzędziami do porządkowania struktury danych. Kolejne postacie normalne (sześć) określają w coraz to ściślejszy sposób strukturę relacji. Każda następna postać stanowi rozszerzenie poprzedniej i ma na celu zapobieżenie pewnym rodzajom anomalii towarzyszącym aktualizacji. Pierwsze trzy postacie normalne zostały wymienione w oryginalnej wersji teorii relacyjnej sformułowanej przez Codda.**Błąd! Nie można odnaleźć źródła odwołania.** Pozostałe postacie normalne - Boyce'a/Codda oraz czwarta i piąta - zostały opracowane z myślą o szczególnych przypadkach, występujących dość rzadko.

- ***Pierwsza postać normalna***:

Relacja jest w pierwszej postaci normalnej, jeśli domeny zdefiniowane dla jej atrybutów są skalarne (co oznacza, ze maja jedną i tylko jedną wartość). Jest to jednocześnie najprostsze jak i najtrudniejsze pojęcie w modelowaniu danych. Reguła jest dość oczywista: każdy atrybut krótki musi zawierać pojedynczą wartość. (Rys. 17). Problem pojawia się, gdy wartość używana do tej pory jako skalarna okazuje się domeną złożoną z kilku atrybutów. Takim przykładem jest data. Gdy za każdym razem posługujemy się datą jako pojedynczą wartością to uczestniczymy w pierwszej postaci normalnej relacji. Gdy natomiast odwołujemy się często do poszczególnych składników daty, lepiej jest przechowywać je jako oddzielne atrybuty.

Pracownik

ID Pracownika	ID Umowy	Imię Pracownika	Nazwisko Pracownika	Nr Tel. Pracownika
1	102	Katarzyna	Miłosz	4690-335
2	104	Sławomir	Raczek	4600-402
3	108	Karolina	Bąk	4632-856

Rysunek 14 Relacja w pierwszej postaci normalnej

- ***Druga postać normalna***

Relacja jest w drugiej postaci normalnej, jeśli jest w pierwszej postaci normalnej, a ponadto wszystkie jej atrybuty zależą od całego klucza kandydującego. Przykład widoczny jest na rysunku 18, gdzie klucz kandydującym, w tabeli Pracownik, jest pole IDPracownika, natomiast w tabeli Umowy jest to pole IDUmowy. Wszystkie domeny zdefiniowane dla jej atrybutów maja postać skalarną. Logicznie chodzi o to, aby nie reprezentować dwóch różnych encji (Pracownicy i Umowy) w ramach jednej relacji.

Pracownik

ID Pracownika	Imię Pracownika	Nazwisko Pracownika	Nr Tel. Pracownika
1	Katarzyna	Miłosz	4690-335
2	Sławomir	Raczek	4600-402
3	Karolina	Bąk	4632-856
4	Artur	Kocur	4563-962

Umowy

ID Umowy	Data Zawarcia	Miejsce Zawarcia
102	02/02/05	Szczecin
104	02/02/05	Szczecin
108	02/06/05	Szczecin
110	02/07/05	Szczecin

Rysunek 15 Relacje w drugiej postaci normalnej

- ***Trzecia postać normalna***

Relacja jest w trzeciej postaci normalnej, jeśli jest w drugiej postaci normalnej, a ponadto wszystkie atrybuty niekluczowe są wzajemnie niezależne. To znaczy, że każda domena zdefiniowana dla jej atrybutów przyjmuje wartości skalarne, wszystkie atrybuty zależą od klucza kandydującego oraz te same atrybuty są wzajemnie niezależne. Obie relacje widoczne na przykładzie rysunku 19 są w trzeciej postaci normalnej, ale w rzeczywistości jedyną płynącą z tego korzyścią jest możliwość automatycznego wyszukiwania kodu pocztowego w momencie wprowadzenia nowych rekordów.

Jak w przypadku każdej encji podejmowanej w trakcie procesu modelowania danych, to kiedy i jak należy implementować trzecią postać normalną, zależy wyłącznie od semantyki modelu. Tworzenie oddzielnych relacji powinno następować tylko wtedy, gdy dana encja jest istotna dla modelu, gdy dane często zmieniają się lub, gdy ma to zalety natury technicznej. Kody pocztowe zmieniają się dosyć rzadko i nie są najistotniejsze dla większość systemów.

Klient

Nazwa Firmy	Adres	Miasto	Region
Speede	Wojska Polskiego 134	Szczecin	ZP
Meteora	Kard. Wyszyńskiego 145/45	Wrocław	DS
KUJAWIAK Sp. z o.o.	Z. Nałkowskiej 23	Szczecin	ZP
Sokół	Pucka 55/18	Warszawa	M

Kod Pocztowy

Miasto	Region	Kod Pocztowy
Szczecin	ZP	70-856
Wrocław	DS	63-562
Szczecin	ZP	70-774
Warszawa	M	36-562

Rysunek 16 Relacje przedstawione w trzeciej postaci normalnej

- ***Postać normalna Boyce'a/Codda***

Postać normalna Boyce'a/Codda, traktowana jako odmiana trzeciej postaci normalnej, dotyczy szczególnego rodzaju relacji z wieloma kluczami kandydującymi. W praktyce, aby móc zastosować postać normalną Boyce'a/Codda, muszą być spełnione trzy warunki:

- relacja musi mieć co najmniej dwa klucze kandydujące,
- co najmniej dwa klucze kandydujące musza być kluczami złożonymi,
- klucze kandydujące musza mieć wspólne atrybuty.

Najprostszym sposobem wyjaśnienia postaci normalnej Boyce'a/Codda jest posłużenie się zależnościami funkcjonalnymi. Postać normalna Boyce'a/Codda określa przede wszystkim, ze nie mogą istnieć zależności funkcjonalne między kluczami kandydującymi. Weźmy na przykład relacje pokazaną na rysunku 20. Jest to relacja w trzeciej postaci normalnej, a mimo to nie może być źródłem poważnej redundancji.

Sprzedawca

IDSprzedawcy	NazwaSprzedawcy	IDProduktu	Ilość	CenaJednostkowa
5	Coperechec	11	2	11,00
9	Crasoola	12	5	15,50

25	Belmondo	52	3	14,90

Rysunek 17 Przykład relacji w trzeciej postaci normalnej

Dwoma kluczami kandydującymi w tym przypadku są {IDSprzedawcy, IDProduktu} oraz {NazwaSprzedawcy, IDProduktu}. Niezgodność z postacią normalną Boyce'a/Codda jest łatwa do uniknięcia, jeśli tylko zwraca się uwagę na logiczne znaczenie relacji. Jeśli relacja z rysunku 20 dotyczy produktów, nie powinna zawierać informacji o sprzedawcach. Dlatego relacje powinny być przedstawione jak na rysunku 21, gdzie są one w pełni znormalizowane.**Błąd! Nie można odnaleźć źródła odwołania.**

Sprzedawca

IDSprzedawcy	NazwaSprzedawcy
5	Coperechec
9	Crasoola
25	Belmondo

Produkt

IDSprzedawcy	IDProduktu	Ilość	CenaJednostkowa
5	11	2	11,00
9	12	5	15,50
25	52	3	14,90

Rysunek 18 W pełni znormalizowana wersja modelu z rysunku 20

- ***Czwarta postać normalna***

Czwarta postać normalna stanowi teoretyczną podstawę reguły: niezależne powtarzające się grupy nie powinny występować w jednej, tej samej, relacji. Przykładem może być produkt sprzedawany przez naszą firmę pod własną marką, ale pochodzących od różnych dostawców, mają oni różne wielkości opakowań i każdy z nich posiada w ofercie wszystkie wielkości opakowań. Przed normalizacją relacja wyglądałaby tak jak na rysunku 22.

Produkt

NazwaProduktu	NazwaDostawcy	WielkośćOpakowania
Batony Bamboo	Coperechec	1/4 kg, 1/2 kg, 1 kg
Wafle Calineczka	Crasoola	1/4 kg, 1/2 kg, 1 kg
Czekoladki Krasnalki	Belmondo	1/4 kg, 1/2 kg, 1 kg

Rysunek 19 Relacja nie znormalizowana.

Pierwszym krokiem doprowadzającym do czwartej postaci normalnej jest wyeliminowanie nieklarownej postaci atrybutu WielkośćOpakowania, co przedstawiono na rysunku 23. Relacja widoczna poniżej jest w postaci Boyle'a/Codda, gdyż cała jest "kluczem". Niestety widać w niej wyraźnie obecność redundancji, co może spowodować problemy z integralnością. Rozwiązanie tych problemów leży w koncepcji ***par zależności wielowartościowych*** i czwartej postaci normalnej.

Produkt

NazwaProduktu	NazwaDostawcy	WielkośćOpakowania
Batony Bamboo	Coperechec	1/4 kg
Batony Bamboo	Coperechec	1/2 kg
Batony Bamboo	Coperechec	1 kg
Wafle Calineczka	Crasoola	1/4 kg
Wafle Calineczka	Crasoola	1/2 kg
Wafle Calineczka	Crasoola	1 kg
Czekoladki Krasnalki	Belmondo	1/4 kg
Czekoladki Krasnalki	Belmondo	1/2 kg
Czekoladki Krasnalki	Belmondo	1 kg

Rysunek 20 Relacja w postaci normalnej Boyce'a/Codda.

Para zależności wielowartościowych to dwa niezależne zbiory atrybutów. Widoczne jest to na rysunku 22 gdzie NazwaProduktu określa Nazwę Dostawcy oraz NazwaProduktu określa WielkośćOpakowania. Pomimo, że atrybuty te znajdują się w jednej relacji określają dwa odmienne problemy. Czwarta postać normalna określa stan takich atrybutów i fakt ich rozbicia na oddzielne relacje, jak to jest pokazane na rysunku 24.

Produkt

NazwaProduktu	WielkośćOpakowania
Batony Bamboo	1/4 kg
Batony Bamboo	1/2 kg
Batony Bamboo	1 kg
Wafle Calineczka	1/4 kg
Wafle Calineczka	1/2 kg
Wafle Calineczka	1 kg
Czekoladki Krasnalki	1/4 kg
Czekoladki Krasnalki	1/2 kg
Czekoladki Krasnalki	1 kg

Dostawca

NazwaProduktu	NazwaDostawcy
Batony Bamboo	Coperechec
Batony Bamboo	Coperechec
Batony Bamboo	Coperechec
Wafle Calineczka	Crasoola
Wafle Calineczka	Crasoola
Wafle Calineczka	Crasoola
Czekoladki Krasnalki	Belmondo
Czekoladki Krasnalki	Belmondo
Czekoladki Krasnalki	Belmondo

Rysunek 21 Znormalizowana relacja do czwartej postaci normalnej.

- ***Piąta postać normalna***

Piąta postać normalna ma zastosowanie tylko w wyjątkowo rzadkich przypadkach ***zależności sprzężeń***. Zależność sprzężeń określa więzy o charakterze cyklicznym: "Jeśli encja 1 jest sprzężona z encją 2, encja 2 jest sprzężona z encją 3, a encja 3 jest sprzężona z encja 1, to wszystkie trzy encje muszą obowiązkowo występować w tej samej krotce".**Błąd! Nie można odnaleźć źródła odwołania.**

Mówiąc potocznie, jeśli {Dostawca} dostarcza {Produkt}, {Klient} zamawia ten {Produkt} i {Dostawca} dostarcz coś {Klientowi}, to{Dostawca} dostarcza ten {Produkt}{Klientowi}. W świecie rzeczywistym nie jest to jednak poprawnie wnioskowane. {Dostawca} może dostarczyć {Klientowi}, coś innego, nie musi to być koniecznie {Produkt}. Zależność sprzężeń istnieje tylko wtedy, gdy obowiązuje dodatkowy warunek mówiący o tym, że podane wnioskowanie jest poprawne.

1.5. Integralność danych.

Proces modelowania danych to nie tylko tworzenie modelu encji występujących w przestrzeni zagadnienia i powiązań między nimi. Trzeba również ustalić reguły, których system bazy danych będzie używał w celu zapewnienia, że przechowywanie w nim fizyczne dane będą prawidłowe albo przynajmniej akceptowane,. Innymi słowy, należy dokonać modelowania ***integralności danych***.

Więzy integralności uważane są przez niektórych ludzi za reguły przedsiębiorstwa. Te ostatnie są jednak znacznie szerszym pojęciem, gdyż obejmują wszystkie więzy nałożone na system, a nie tylko te, które dotyczą integralności danych.

Integralność danych jest implementowana na kilku poziomach. Więzy domen, przejść i encji określają reguły służące utrzymaniu integralności poszczególnych relacji. Więzy integralności odniesienia zapewniają utrzymanie niezbędnych powiązań pomiędzy relacjami. Więzy integralności bazy danych rządzą bazą danych jako całością, a więzy integralności transakcji kontrolują sposób manipulowania danymi zarówno w ramach jednej, jak i wielu baz danych.**Błąd! Nie można odnaleźć źródła odwołania.**

1. Integralność domeny

Domena to zbiór wszystkich możliwych wartości danego atrybutu. Więzy integralności domeny - nazywane ***więzami domeny*** - to reguła definiująca te poprawne

wartości. Oczywiście do opisania całej domeny konieczne może być skorzystanie z więcej niż jednej takiej reguły.

Integralność domeny może być określana za pomocą typu danych. Mogą one być wygodnym zapisem skrótowym i z tego powodu wybieranie logicznego typu danych jest często pierwszym krokiem podczas ustalania więzów domeny w systemie. Przez logiczny typ danych rozumiemy "data", "łańcuch" czy "obraz" - a nie coś bardziej konkretnego. Po wybraniu logicznego typu danych można ustalić wielkość i dokładność typu numerycznego albo maksymalną długość łańcuchów znakowych.

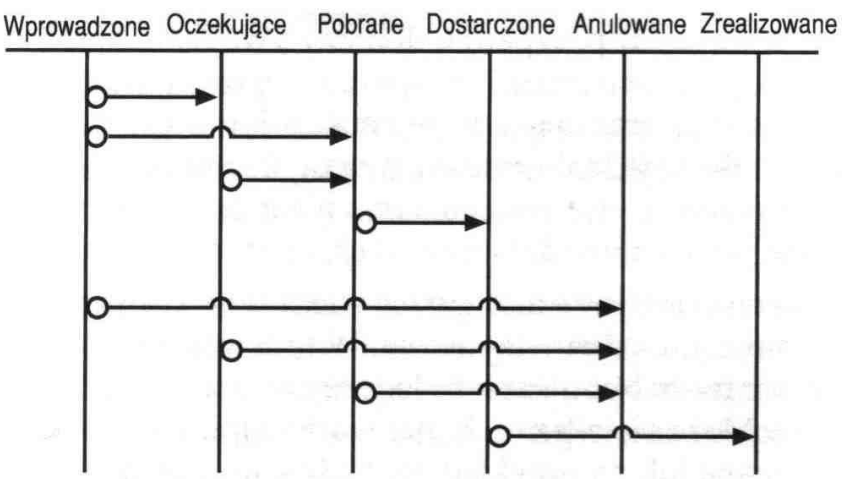
Kolejnym do rozważenia aspektem integralności domeny jest to, czy domena może zawierać nieznane lub nieistniejące wartości. Są dwa problemy, o których należy pamiętać przy ustalaniu zawartości domeny:

- wartości "nieznana" i "nieistniejąca" są różne,
- w przypadku domeny uwzględniającej wartość nieznaną lub nieistniejącą, obie te wartości mają być akceptowane przez system.

Ostatnią sprawą, którą należy uwzględnić zastanawiając się nad integralnością domeny jest chęć maksymalnie precyzyjnego zdefiniowania zbioru wartości. Na przykład domena DataTransakcji nie może być zbiorem wszystkich dat, lecz jedynie zbiorem dat od momentu rozpoczęcia handlu przez firmę do dnia dzisiejszego.

1.6. Integralność przejść

Więzy integralności przejść definiują stany, przez które w sposób poprawny może przejść krotka. Na przykład diagram przejść stanów z rysunku 25 pokazuje stany, przez które może przejść zamówienie.



Rysunek 22 Diagram pokazujący stany przez które może przejść zamówienie.

Więzy integralności przejść pozwalają na przykład zagwarantować, że status danego zamówienia nie zmieni się z "Wprowadzone" na "Zrealizowane", bez przejścia stanów pośrednich, a zamówienie anulowane nigdy już nie zmieni swojego statusu.**Błąd! Nie można odnaleźć źródła odwołania.**

3. Integralność encji

Więzy integralności encji mają na celu zapewnienie integralności encji modelowanych w systemie. Więzy encji można podzielić na dwie grupy:

- Więzy integralności samej encji,
- Więzy integralności na poziomie encji.

Te pierwsze wyrażone są za pomocą reguły: "każda encja musi się dawać jednoznacznie zdefiniować", czyli określa konieczność występowania klucza głównego w encji. Natomiast więzy zdefiniowane na poziomie encji mogą dotyczyć pojedynczego atrybutu, wielu atrybutów lub relacji jako całości.

Integralność pojedynczego atrybutu jest modelowana przede wszystkim poprzez zdefiniowanie konkretnej domeny dla tego atrybutu. Atrybut występujący w relacji dziedziczy więzy integralności zdefiniowane dla jego domeny. Na poziomie encji, więzy encji mogą wyznaczać podzbiór więzów domeny, ale nie mogą go rozszerzać.

Oprócz zawężania zakresu wartości jednego atrybutu więzy encji mogą również wpływać na wiele atrybutów jednocześnie. Przykładem takich więzów jest wymaganie, aby DataDostawy wypadała nie wcześniej niż DataZamówienia.

4. Integralność odniesienia.

Więzy integralności odniesienia utrzymują i chronią związki między relacjami, które dotyczą dekomponowania relacji w celu zminimalizowania redundancji oraz kluczy obcych służących do implementowania związków między relacjami.

Warunkiem więzów integralności odniesienia jest to, że klucz obcy nie może zostać "osierocony". To znaczy, że żaden rekord w tabeli obcej nie może zawierać klucza obcego, dla którego nie istnieje odpowiadający mu rekord w tabeli głównej. Krotki zawierające klucze obce niemające odpowiadających im kluczy kandydujących w relacji głównej nazywane są encjami osieroconymi. Istnieją trzy przyczyny powstawania encji osieroconych:

- dodanie do tabeli obcej krotki z kluczem niemającym odpowiadającego mu klucza kandydującego w tabeli głównej,
- zmiana klucza kandydującego w tabeli głównej,
- usunięcie związanego rekordu z tabeli głównej.**Błąd! Nie można odnaleźć źródła odwołania.**

Wszystkie te przypadki musza być obsługiwane, aby móc zachować ***integralność powiązania***.

5. Integralność bazy danych

Najbardziej ogólną postacią więzów integralności są więzy bazy danych. Więzy bazy danych odnoszą się do więcej niż jednej relacji, a tego typu postać ma większość więzów bazy danych.

Przy definiowaniu integralności bazy danych trzeba zachować ostrożność i nie pomylić więzów bazy danych ze specyfikacją procesu działania. ***Proces działania*** dotyczy operacji wykonywanych na bazach danych, takich jak na przykład dodawanie zamówienia, podczas gdy więzy integralności to reguły odnoszące się do zawartości bazy danych. Reguły definiujące zadania wykonywane z użyciem bazy danych są ***więzami procesów działania,*** a nie więzami bazy danych.

Czasami pojawia się problem, czy dana reguła firmy kwalifikuje się jako więzy integralności czy proces działania. Różnica może nie być wcale tak istotna. Tego typu reguły należy implementować tam gdzie jest to najwygodniejsze. Jeśli naturalne jest umieszczenie reguły w więzach bazy danych, należy tak zrobić. Jeśli sprawia to problemy, należy przenieść ją do oprogramowania interfejsu, gdzie można zaimplementować ja w formie procedury.

6. Integralność transakcji

Więzy integralności transakcji decydują o sposobach manipulowania bazą danych. W odróżnieniu od wielu innych więzów, więzy transakcji mają charakter proceduralny i w związku z tym nie stanowią części modelu danych.

Transakcja wiąże się ściśle z procesami działania. Dany proces działania może składać się w rzeczywistości z jednej lub wielu transakcji i na odwrót. ***Transakcja*** to grupa działań, a wszystkie muszą być wykonane w całości (albo w ogóle). Baza danych musi spełniać wszystkie zdefiniowane więzy integralności zarówno przed rozpoczęciem transakcji, jaki i po jej ukończeniu.

Transakcje mogą dotyczyć wielu rekordów, wielu relacji, a nawet wielu baz danych. Ściśle biorąc wszystkie działania na bazie danych są transakcjami. Nawet aktualizacja jednego rekordu jest transakcją. Na szczęście transakcje na tak niskim poziomie są wykonywane w sposób niewidoczny przez aparat bazy danych i można je ignorować.**Błąd! Nie można odnaleźć źródła odwołania.**

Reasumując terminem "integralność danych" określa się poprawność, spójność oraz dokładność danych przechowywanych w bazie. Nie jest przesadą stwierdzenie, że dokładność informacji odczytywanych z bazy jest wprost proporcjonalna do stopnia integralności zawartych w niej danych. Zapewnienie integralności danych jest jednym z najważniejszych aspektów procesu projektowania i problemu tego nie wolno ominąć, zignorować czy nawet częściowo zaniedbać. Spowodowałoby to wystąpienie niezauważonych błędów, co z kolei zmusiłoby użytkowników bazy danych do korzystania z niedokładnych lub wręcz niepoprawnych informacji.

Przyjmując schematyczną (skróconą) wersję procesu projektowania bazy danych, zakłada ona wprowadzenie jedynie czterech rodzajów integralności danych (nie sześciu). Trzy z nich są oparte na strukturze samej bazy i oznacza się je zgodnie z poziomem, na którym funkcjonują. Czwarty zależy wyłącznie od sposobu, w jaki dane są wykorzystywane przez użytkowników. Oto typy integralności danych oraz ich opisy:

- ***Integralność na poziomie tabel*** gwarantuje, że pole identyfikujące każdy rekord w danej tabeli ma zawsze unikatową wartość i nigdy nie jest puste.
- ***Integralność na poziomie pól*** gwarantuje, że struktura każdego pola jest poprawna, a zawarte w nim wartości - logiczne, oraz wszystkie pola tego samego typu są zdefiniowane w identyczny sposób w całej bazie danych.
- ***Integralność na poziomie relacji*** (tradycyjnie nazywana ***integralnością referencyjną***) gwarantuje, że wszystkie relacje są poprawnie zdefiniowane i dane w powiązanych tabelach są ze sobą zsynchronizowane.
- ***Reguły integralności*** nakładają ograniczenia na niektóre aspekty bazy danych ze względu na sposób jej użytkowania przez organizację. Ograniczenia te mają wpływ na projektowanie bazy - definiują na przykład typ i zakres wartości przechowywanych w poszczególnych polach; typ i stopień uczestnictwa tabel w relacjach oraz rodzaje synchronizacji wykorzystywanej we wprowadzaniu integralności na poziomie relacji.**Błąd! Nie można odnaleźć źródła odwołania.**

1.7. Projektowanie relacyjnych baz danych

Zrozumienie sposobu funkcjonowania relacyjnej bazy danych nie tylko nie jest tak trudne, jak zrozumienie wszechświata; ale jest nawet o wiele łatwiejsze. Należy jednak przykładać wagę do całościowego ogarnięcia procesu projektowania oraz etapów, z których się on składa.**Błąd! Nie można odnaleźć źródła odwołania.**

1.7.1. Formułowanie definicji celu i założeń wstępnych

Formułowanie ***definicji celu*** i ***założeń wstępnych*** jest pierwszą faza tworzenia projektu bazy danych i porusza temat określenia ***zadań głównych***. Definicja celu mówi, po co projektujemy bazę danych oraz zakreśla granicę w przestrzeni zagadnienia.

Każda baza danych powstaje w określonym celu. Może to być baza służąca jakiejś firmie lub organizacji do codziennego działania, bądź też służąca do przechowywania danych, które systematycznie się zmieniają, bądź baza będąca częścią systemu informacyjnego, itp. Określając cel, jakiemu ma służyć baza danych i na jakie pytania powinna "odpowiadać", łatwiej i poprawniej będzie skonstruować zadania główne.

Przez określenie zadania główne rozumiemy funkcje aplikacji, które będą ostatecznie realizowane za pośrednictwem formularzy i raportów w bazie danych. Natomiast założenia wstępne są to wymagania, jakie powinny spełniać dane przechowywane w bazie.

Definicja celu powinna być uzupełniana przez zadania główne oraz założenia wstępne. Warunki te powinny również pomagać w tworzeniu poszczególnych elementów bazy.

Należy pamiętać, że nie sam twórca bazy odpowiada za definicje celu, założenia wstępne oraz za zadania główne. Odpowiedzialność za nie należy, bowiem również do osób, którym ma służyć baza danych lub, którzy mają z niej korzystać.

1.7.2. Analiza istniejącej bazy danych

Analiza istniejącej bazy danych, jeżeli taka istnieje, jest drugą fazą procesu projektowania. Wcześniej istniejące bazy dane możemy podzielić na dwa typy:

- spadkowa baza danych (odziedziczona),
- tradycyjna baza danych.

Ten pierwszy, jest to stara baza danych używana od wielu lat, a ten drugi to luźny zbiór formularzy, teczek, notatek i tym podobnych. Niezależnie od typu i stanu istniejącej bazy danych, dokładne przyjrzenie się jej strukturze udzieli cennych informacji na temat sposobu wykorzystywania danych przez organizację. Co więcej, analiza ta pozwoli na zrozumienie sposobu, w jaki dane są gromadzone i pokazywane użytkownikom. Każdy twórca nowej bazy danych powinien przyjrzeć się roli pełnionej przez papier w gromadzeniu danych (formularze) oraz ich prezentacji (raporty). Analogicznie, jeżeli organizacja czy firma, dla której opracowywana zostaje baza danych, wykorzystuje już jakieś oprogramowanie komputerowe, należy przestudiować jego działanie oraz metody przetwarzania przez nie danych.

Kolejnym bardzo ważnym elementem analizy istniejącej bazy danych jest przeprowadzenie wywiadu z użytkownikami w celu ustalenia sposobu, w jaki istniejąca baza jest wykorzystywana do codziennym funkcjonowaniu firmy. Przy projektowaniu należy uzyskać informacje, jaką rolę spełniała dotychczas baza danych dotychczas a jakie są aktualne wymagania.

Po wykonaniu tych czynności można zabrać się do tworzenia listy pól i usług, jakie będą musiały być udostępnione przez bazę danych. Lista ta powinna stanowić punkt wyjścia do stworzenia projektu logicznego bazy danych a powinna zawierać podstawowe wymagania informacyjne analizowanej firmy. Należy oczekiwać, że zawartość listy będzie się powiększać w miarę konstruowania tego projektu.

Gdy lista usług jest gotowa powinni ją zobaczyć użytkownicy, aby zweryfikować ewentualne punkty. Należy pamiętać o uważnym rozpatrywaniu każdej propozycji korekty. Jeżeli propozycja uważana jest za rozsądną trzeba wpisać ją na listę usług. Koniecznością jest wprowadzanie odpowiednich poprawek i uaktualnienie listy przed podejściem do następnego etapu projektowania.

1.7.3. Tworzenie struktur danych

Następnym etapem procesu projektowania jest tworzenie struktur danych. W tym momencie, gdy jest już określony cel bazy danych i założenie wstępne, przeanalizowana istniejąca baza danych i przeprowadzony wywiad w środowisku użytkowników, należy zdefiniować tabele i pola, wskazać pola kluczowe oraz określić domeny każdego z pól.

Pierwszą strukturą, którą należy się zająć, są ***tabele***. Tematami tych tabel powinny być encje widoczne wcześniej na liście pól i usług, które zostały zdefiniowane za pomocą wywiadu i istniejącej wcześniej bazy danych. Po utworzeniu tabeli, kolejnym krokiem jest utworzenie stosownych pól (na podstawie atrybutów). Pola te powinny być bezpośrednio powiązane logiczną więziom z encją W tej fazie projektowania nie powinny znajdować się tam pola z obcych tabel, które mogą zakłócić na tym etapie pracę całej bazy danych

Jeżeli zostały utworzone pola w tabelach, należy nadać im konkretne właściwości, czyli należy dobrać ***domeny***. Należy pamiętać, że domena to coś znacznie szerszego niż typ danych i korzystać z modelowania każdego pola i nadawać mu odpowiadające mu wartości. Przy definiowaniu domeny należy powtórnie zwrócić się do użytkowników, aby określili konkretnie informacje, które powinny znajdować się w odpowiednich polach.

Po wykonaniu tej czynności trzeba ***przeanalizować*** każda tabelę i upewnić się, że reprezentuje ona tylko jeden temat i nie zawiera powtarzających się pól. Trzeba teraz przyjrzeć się każdemu polu z osobna. Jeśli stwierdzone zostanie, że któreś z nich jest wielowartościowe lub segmentowe, trzeba je rozbić tak, aby wszystkie pola w tabeli zawierały pojedyncze wartości. Pole niereprezentujące tematu danej tabeli powinno być przesunięte do innej, bardziej odpowiedniej tabeli lub zostać usunięte z bazy. Na koniec należy zdefiniować ***klucze podstawowe***, które będą jednoznacznie identyfikować każdy nowy rekord w poszczególnych tabelach.

1.7.4. Definiowanie powiązań

W poprzedniej fazie procesu projektowania zostały utworzone tabele. Kolejnym krokiem jest zdefiniowanie ***relacji pomiędzy tymi tabelami***. Aby określić istniejące relacje, ustalić ich cechy oraz zagwarantować integralność na poziomie relacji należy po raz kolejny przeprowadzić rozmowy z użytkownikami bazy.

Współpraca z użytkownikami przy określaniu relacji jest bardzo pomocna, ponieważ skompletuje ona wiedze na temat aspektów pracy i funkcjonowania danej firmy. Większość pracowników ma jednak dobre wyobrażenie o danych, na których operuje, i może z łatwością wskazać ewentualne relacje.

Kiedy relacje zostaną już ustalone, należy określić ich ***cechy***. W zależności od typu danej relacji wchodzącej w jej skład tabele trzeba połączyć, korzystając albo z klucza podstawowego (przy relacjach: jeden-do-jednego, jeden-do-wielu), albo z tabeli łączących

(tabela sprzężona przy relacji wiele-do-wielu). Kolejnym krokiem jest zdefiniowanie typu i stopnia uczestnictwa tabel w poszczególnych relacjach. W większości przypadków cechy te będą wprost wynikały z rodzaju danych przechowywanych w konkretnych tabelach. Są jednak takie przypadki, w których należy odwołać się do reguł integralności.

1.7.5. Wprowadzenie i kontrola reguł integralności

Kolejnym etapem procesu projektowania bazy danych jest wprowadzanie ***reguł integralności***. Przed rozpoczęciem piątego etapu projektowania po raz kolejny należy spotkać się z użytkownikami przyszłej bazy danych, ale powinno się również ustalić ograniczenia, jakim mają podlegać dane, zdefiniować reguły integralności oraz wprowadzić ***tabele walidacji***.

Tabela walidacji ma za zadanie przechowywanie danych zapewniających integralność. Dane w tabeli walidacji rzadko ulegają zmianom i równie rzadko wprowadza się doń nowe rekordy lub usuwa istniejące. Tabele walidacji zazwyczaj składają się z dwóch pól: jednemu przypisuje się rolę klucza podstawowego, a drugie jest zwykłym polem niekluczowym, zawierającym listę dopuszczalnych wartości dla jakiegoś innego pola w bazie danych.**Błąd! Nie można odnaleźć źródła odwołania.**

Podczas tworzenia bazy danych będą musiały zostać uwzględnione ograniczenia, które dyktuje dana organizacja w zależności od sposobu w jaki wykorzystuje ona dane. Poprzez rozmowy z pracownikami i kierownictwem ustalone zostaną wymagania, które powinny być spełnione przez dane oraz ich strukturę. Pełną listę tych wymagań należy traktować jako zestaw reguł integralności.

Dzięki rozmowom z pracownikami zostaną uzyskane informacje na temat konkretnych ograniczeń różnych aspektów bazy danych. Przykładowo, użytkownik bazy danych zamówień jest świadom faktu, ze "Data realizacji zamówienia" musi być późniejsza niż "Data złożenia zamówienia" oraz że zawsze należy podać "Telefon odbiorcy" i "Sposób wysyłki". Z drugiej strony, rozmowy prowadzone z kierownictwem dadzą wiedzę o ogólnych wymaganiach stawianych projektowej bazie. I tak, kierownik agencji powinien powiedzieć, że dany pośrednik nie może reprezentować więcej niż dwudziestu muzyków, a informacje o każdym muzyku należy uaktualniać przynajmniej raz w roku.**Błąd! Nie można odnaleźć źródła odwołania.**

Kolejnym krokiem na tym etapie jest zdefiniowanie i zaimplementowanie tabel walidacji, o ile tylko okażą się przydatne we wprowadzaniu reguł integralności. Na przykład,

jeśli ustalone zostanie, że poszczególne pola w tabeli mogą przyjmować ograniczoną ilość wartości zdefiniowanych na podstawie wymagań użytkowników, tabele walidacji powinny pomóc przy spełnieniu wszystkich wymagań.

W tym momencie ważne staje się wprowadzanie ***poziomu integralności danych***, opartego na regułach integralności, ponieważ będzie odnosił się on bezpośrednio do sposobu, w jaki dana organizacja wykorzystuje dane zawarte w bazie. Wraz z rozwojem firmy wymagania informacyjne mogą ulegać zmianie, a w ślad za nimi - również reguły integralności. Oznacza to, że ustalanie reguł integralności na danym poziomie jest procesem ciągłym i wymaga stałej uwagi i przezorności.

Na tym etapie kontroluje się również ***integralność danych***. Przede wszystkim należy przyjrzeć się każdej tabeli z osobna i upewnić się, że spełnia ona odpowiednie kryteria poprawności i oczywiście należy sprawdzić w ten sposób również wszystkie należące do niej pola. Wszystkie nieścisłości i problemy powinny teraz zostać skorygowane lub usunięte. Po wprowadzeniu odpowiednich poprawek należy sprawdzić ***integralność na poziomie tabel***.

Następnie trzeba przejrzeć domeny wszystkich pól w bazie, wprowadzając konieczne poprawki i sprawdzając integralność na poziomie pól. Dzięki temu elementowi uzyska się przekonanie, że integralność wprowadzona na wcześniejszych etapach projektowania bazy została zachowana.

Kolejnym krokiem jest sprawdzenie poprawności każdej z relacji oraz jej typ, a jak również rodzaju i stopnia uczestnictwa wszystkich tabel w relacjach, w których skład wchodzą. Należy poza tym przeanalizować integralność na poziomie relacji, upewniając się, że współdzielone pola zawierają odpowiednie wartości oraz że wprowadzanie, modyfikowanie i usuwanie danych z powiązanych tabel nie sprawia żadnych problemów.

Ostatnim krokiem w kontroli integralności danych jest ponowne przejrzenie listy reguł integralności, potwierdzając zgodność ograniczeń nakładanych przez nie na bazę danych z ustalonymi wcześniej wymaganiami. Jeśli podczas późniejszych faz procesu projektowania zauważono by konieczność wprowadzenia dodatkowych ograniczeń, należy je uwzględnić jako dodatkowe reguły integralności i dopisać je do listy.

1.7.6. Zapytania do bazy danych (kwerendy)

Kolejnym bardzo ważnym elementem bazy danych jest sprawdzenie ***zrealizowanych zagadnień głównych*** oraz ***listy pól*** i ***usług***. Wykonane do tej pory etapy pracy na bazie

danych pozwoliły w pewnym zakresie zrealizować zamierzone wcześniej punkty. Lecz większość z nich przedstawiona jest w szerszym zakresie niż jest to potrzebne. Ze względu na funkcjonalność bazy danych często informacje zawarte w tabelach mają znacznie szersze zastosowanie niż w niektórych przypadkach jest to potrzebne.

Przykładem może być tabela zawierająca dane o pracowniku. Znajdują się w niej pola określające tożsamość pracownika, adres zamieszkania, adres zameldowania, numer telefonu domowy, numer telefonu komórkowy, e-mail, numer NIP, PESEL, itp. Jeżeli w takiej ilości pól chcemy odnaleźć jedynie trzy interesujące nas pola; np.: Imię, Nazwisko, NrTelDom, może to zadanie być kłopotliwe, gdy jeszcze do tego tabela zawiera wiele rekordów. Rozwiązaniem takiego problemu może być zbudowanie kwerendy wybierającej z tabeli jedynie interesujące nas pola.**Błąd! Nie można odnaleźć źródła odwołania.**

Możemy wyróżnić wiele typów kwerend. Pierwszy typ kwerend to ***kwerendy wybierające***, które wybierają informacje z tabel i kwerend w bazie danych. Drugim typem są ***kwerendy funkcjonalne***, które wstawiają, uaktualniają i usuwają dane. Kwerendy funkcjonalne wywodzą się z kwerend wybierających. Należy stworzyć zawsze kwerendę wybierającą a na jej podstawie zbudować dopiero jedną z czterech rodzajów kwerendy funkcjonalnej. Kwerendę funkcjonalną można podzielić również na kilka podgrup: kwerenda aktualizującą, kwerenda tworząca tabelę, kwerenda dołączająca, kwerenda usuwająca.

Zdarza się, że potrzebne informacje z bazy danych są podsumowaniem różnych grup danych i wtedy należy wykorzystać ***kwerendy podsumowujące***. Natomiast kwerendy krzyżowe jest to specjalny typ kwerend podsumowujących. Pozwala ona na wyświetlenie wartości w formacie podobnym do arkusza danych i można użyć tego typu kwerendy, aby zobaczyć posegregowane dane w zależności od interesującej nas wartości, np. zarobki poszczególnych pracowników wg miesięcy, lub wg kwartału.

Kwerendy są elementem bazy danych, porównywalnym z tabelą. Zawierają, zatem również zbiór pól i rekordów. Ich wielkim atutem jest to, że służą do przechowywania danych, oraz do segregowania, podliczania, po prostu do modelowania danych i odpowiadania na zadane przez użytkownika pytania. Często służą jedynie do ograniczania ilości pól i rekordów, ale również zdarza się, ze łącza pola z tabel powiązanych ze sobą i wykorzystywane są do korespondencji seryjnej z innymi dokumentami.