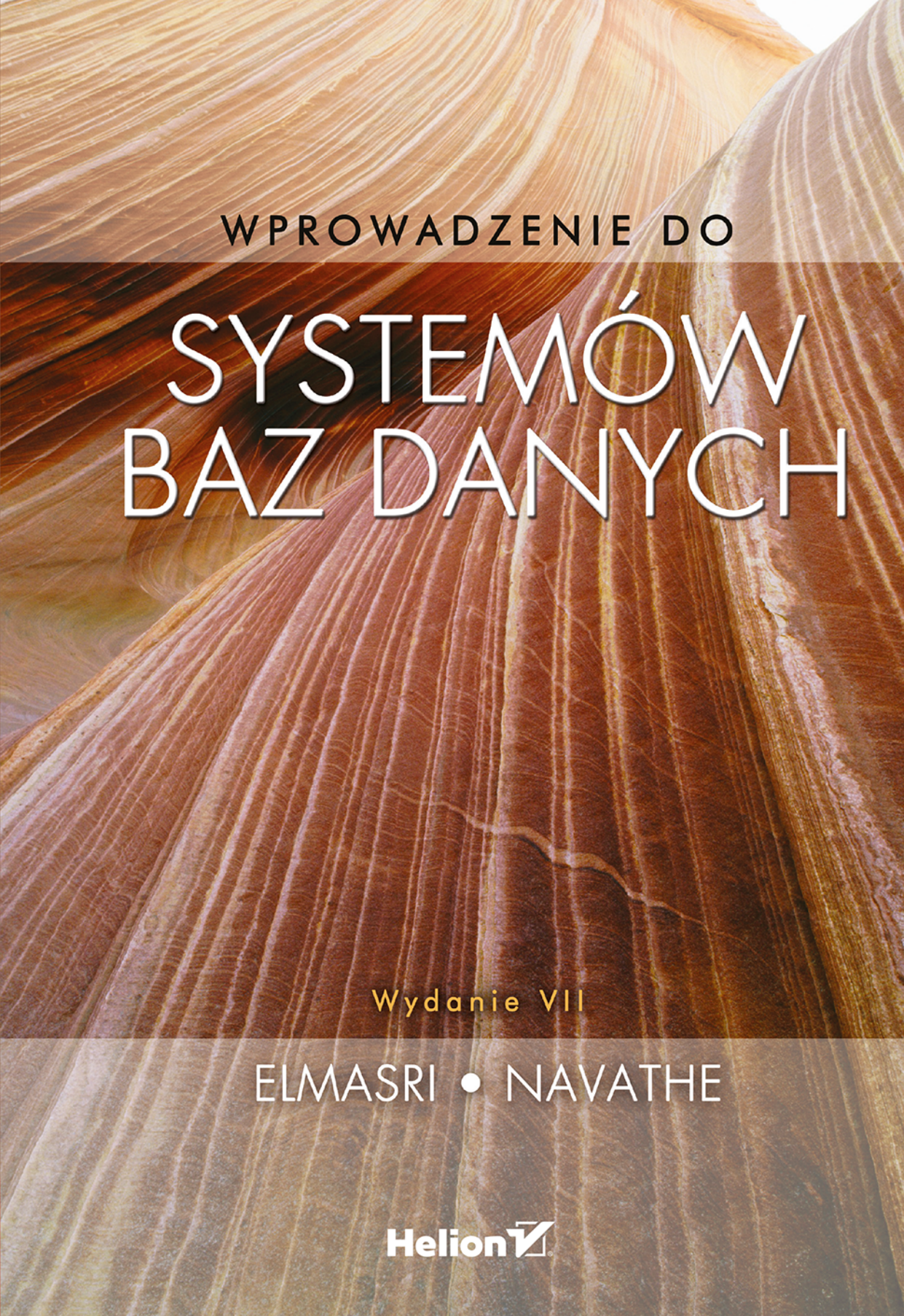

WPROWADZENIE DO
SYSTEMÓW BAZ DANYCH
Wydanie VII
ELMASRI • NAVATHE
Helion

Tytuł oryginału: Fundamentals of Database Systems (7th Edition)

Tłumaczenie: Tomasz Walczak
z wykorzystaniem fragmentów „Wprowadzenie do systemów baz danych” w przekładzie Bartłomieja Garbacza, Bartłomieja Moczulskiego i Mikołaja Szczepaniaka

ISBN: 978-83-283-4695-6

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/wprsy7
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

II. Koncepcyjne modelowanie danych i projektowanie baz danych 99

IV. Techniki programowania baz danych 367

2

Architektura systemów baz danych i związane z nimi pojęcia

Architektura pakietów systemów zarządzania bazami danych ewoluowała od wczesnych systemów monolitycznych, w których cały pakiet oprogramowania SZBD był ściśle zintegrowanym systemem, do współczesnych pakietów tego typu, które składają się z modułów i są konstruowane w oparciu o model klient-serwer. Widoczny ostatnio wzrost ilości danych wymagających składowania doprowadził do powstania systemów baz danych o architekturze rozproszonej, składających się z tysięcy komputerów zarządzających magazynami danych. Ewolucja systemów zarządzania bazami danych odzwierciedla pojawiające się w technice komputerowej trendy, które sprawiły, że wielkie scentralizowane komputery zostały zastąpione setkami rozproszonych stacji roboczych i komputerów osobistych połączonych za pośrednictwem sieci komunikacyjnej z serwerami rozmaitych typów — serwerami WWW, serwerami baz danych, serwerami plików, serwerami aplikacji itp. Nowe środowiska **chmur obliczeniowych** obejmują tysiące dużych serwerów zarządzających tzw. **big data** na potrzeby użytkowników internetu.

W podstawowej architekturze systemu zarządzania bazą danych typu klient-serwer funkcjonalność tego systemu jest dzielona pomiędzy dwa typy modułów[1]. **Moduł klienta** jest zwykle projektowany w taki sposób, aby możliwe było jego uruchamianie w urządzeniach mobilnych, na stacjach roboczych lub komputerach osobistych. Moduły klienta są zwykle używane do obsługi aplikacji wykorzystujących bazę danych oraz do udostępniania odpowiednich interfejsów. Oznacza to, że tak naprawdę to moduł klienta obsługuje działania użytkownika na bazie danych i udostępnia mu przyjazny interfejs — np. aplikację na urządzenia mobilne albo oparty na formularzach lub menu graficzny interfejs użytkownika (ang. *Graphical User Interface — GUI*) w komputerach osobistych. Drugi rodzaj modułów — tzw. **moduły serwera** — obsługuje zazwyczaj przechowywanie danych, operacje dostępu do danych, operacje przeszukiwania danych i inne, podobne funkcje. Architekturę typu klient-serwer omówimy bardziej szczegółowo w podrozdziale 2.5. Wcześniej musimy poświęcić chwilę na przestudiowanie podstawowych zagadnień, które umożliwią nam lepsze zrozumienie architektur współczesnych baz danych.

W tym rozdziale zaprezentujemy terminologię i podstawowe pojęcia, które będą wykorzystywane w książce. Rozpoczniemy (w podrozdziale 2.1) od omówienia modeli danych oraz zdefiniowania pojęć schematów i egzemplarzy, które mają zasadnicze znaczenie podczas rozważań na temat systemów baz danych. Następnie, w podrozdziale 2.2 omówimy

[1] W podrozdziale 2.5 przekonamy się, że istnieją różne odmiany tej prostej, *dwuwarstwowej* architektury klient-serwer.

trójwarstwową architekturę systemów zarządzania bazami danych (ang. *three-schema DBMS*) oraz niezależność danych — w ten sposób dojdziemy do problemu oczekiwanych działań systemu zarządzania bazą danych z perspektywy jej użytkownika. W podrozdziale 2.3 opiszemy typy interfejsów i języków oferowanych przez większość współczesnych systemów zarządzania bazami danych. Podrozdział 2.4 poświęcono omówieniu środowiska oprogramowania systemu bazy danych. W podrozdziale 2.5 zawarto przegląd rozmaitych typów architektur typu klient-serwer. I wreszcie w podrozdziale 2.6 zaprezentowaliśmy klasyfikację typów pakietów systemów zarządzania bazami danych. Podrozdział 2.7 zawiera podsumowanie tego rozdziału.

Materiał zawarty w podrozdziałach od 2.4 do 2.6 dotyczy bardziej szczegółowych zagadnień, które nie muszą być omawiane w trakcie podstawowego kursu wprowadzającego w świat baz danych.

2.1. Modele danych, schematy i egzemplarze

Jedną z podstawowych właściwości rozwiązań opartych na bazach danych jest to, że ich użytkownicy korzystają z abstrakcji danych. **Abstrakcja danych** oznacza ukrywanie szczegółów struktury i przechowywania danych oraz podkreślanie najważniejszych ich aspektów, co pozwala lepiej zrozumieć informacje. Jedną z głównych cech podejścia wykorzystującego bazy jest zapewnianie abstrakcji danych, dzięki czemu różni użytkownicy mogą korzystać z nich na preferowanym poziomie szczegółowości. **Model danych** — czyli zbiór pojęć, które można wykorzystywać do opisywania struktury bazy danych — zapewnia środki niezbędne do osiągnięcia oczekiwanego poziomu abstrakcji[2]. Przez *strukturę bazy danych* rozumiemy typy danych, związki i ograniczenia dotyczące danych. Większość modeli danych zawiera także zbiór **podstawowych operacji**, które umożliwiają otrzymywanie i aktualizowanie informacji zawartych w bazie danych.

Poza podstawowymi operacjami zdefiniowanymi w modelu danych, coraz bardziej popularne staje się dołączanie do tego modelu także elementów określających **aspekty dynamiczne** i **zachowania** aplikacji bazodanowych. Dzięki temu projektanci baz danych mogą stosunkowo łatwo określać zbiory poprawnych **operacji definiowanych przez użytkowników**, które będą bez problemów wykonywane na obiektach bazy danych[3]. Przykładem takich operacji definiowanych przez użytkowników może być procedura `OBLICZ_ŚROC`, którą można stosować dla obiektów przechowywanych w pliku `STUDENT`. Z drugiej strony, uniwersalne operacje wstawiania, usuwania, modyfikowania i odczytywania dowolnych rodzajów obiektów są zwykle dołączane do *podstawowych operacji modelu danych*. Elementy definiujące zachowania są nie tylko zasadniczym elementem obiektowych

[2] Słowo *model* jest niekiedy wykorzystywane w odniesieniu do konkretnego opisu bazy danych lub do schematu — przykładowo, można się spotkać z wyrażeniem *model danych marketingowych*. W tej książce nie będziemy używać słowa *model* w tym znaczeniu.

[3] Dołączanie elementów opisujących zachowanie baz danych odzwierciedla kierunek rozwoju systemów tego typu, który przewiduje, że działania związane z projektowaniem baz danych i projektowaniem oprogramowania z czasem zostaną połączone w zbiór tych samych czynności. Określanie zachowania oprogramowania tradycyjnie było elementem projektowania oprogramowania.

modeli danych (patrz rozdział 12.), ale także zostały z powodzeniem wprowadzone do bardziej tradycyjnych modeli danych. Przykładowo, modele obiektowo-relacyjne (patrz rozdział 12.) rozszerzają tradycyjny model relacyjny między innymi o elementy tego typu. W podstawowym modelu relacyjnym możliwe jest dodawanie zachowań do relacji za pomocą trwałych modułów składowanych, nazywanych zwykle procedurami składowanymi (patrz rozdział 10.).

2.1.1. Kategorie modeli danych

Istnieje wiele różnych modeli danych, które możemy skategoryzować np. według elementów wykorzystywanych do opisywania struktury bazy danych. Przykładowo, **wysokopoziomowe** i **koncepcyjne modele danych** opierają się na rozwiązaniach, które są zbliżone do abstrakcyjnego sposobu postrzegania danych przez wielu użytkowników, natomiast **niskopoziomowe** i **fizyczne modele danych** opierają się na szczegółowych opisach metod przechowywania danych na dysku komputera (zwykle na dyskach magnetycznych). Niskopoziomowe modele danych są przeznaczone przede wszystkim dla specjalistów w zakresie sprzętu komputerowego, nie dla typowych użytkowników końcowych systemów baz danych. Pomiędzy wspomnianą parą skrajnych rozwiązań istnieje klasa **reprezentacyjnych** (lub **implementacyjnych**[4]) **modeli danych**, które z jednej strony opierają się na pojęciach zrozumiałych dla przeciętnego użytkownika końcowego, ale jednocześnie nie odbiegają znacząco od sposobu organizowania danych na fizycznym nośniku w komputerze. Reprezentacyjne modele danych ukrywają co prawda niektóre szczegóły związane z przechowywaniem danych, ale mogą być implementowane w systemach komputerowych w sposób bezpośredni.

Koncepcyjne modele danych opierają się na elementach takich jak encje, atrybuty czy związki. Pojedyncza **encja** reprezentuje obiekt lub pojęcie z mini-świata (np. pracownika lub projekt), które zostało opisane w bazie danych. **Atrybut** reprezentuje pewną interesującą własność, która dodatkowo opisuje daną encję — może to być np. nazwisko lub pensja pracownika. **Związek** pomiędzy dwoma (lub więcej) encjami reprezentuje łączące je powiązanie — przykładowo, może to być związek *pracuje nad*, który łączy encje reprezentujące pracownika i projekt. W rozdziale 3. zaprezentujemy model związków encji (ang. *Entity-Relationship*, w skrócie *ER*), który jest popularnym wysokopoziomowym koncepcyjnym modelem danych. W rozdziale 4. omówimy dodatkowe elementy charakterystyczne dla koncepcyjnego modelowania danych, w tym generalizację, specjalizację oraz kategorie (typy unii).

W tradycyjnych, komercyjnych systemach zarządzania bazami danych wykorzystuje się najczęściej reprezentacyjne lub implementacyjne modele danych. Takie modele obejmują zarówno popularny obecnie **relacyjny model danych**, jak i bardziej tradycyjne, **sieciowe** i **hierarchiczne modele danych**, które były powszechnie stosowane w przeszłości. Część 3. tej książki poświęcono relacyjnemu modelowi danych, charakterystycznym dla niego operacjom i językom, a także niektórym technikom programowania aplikacji operujących na relacyjnych bazach danych. Standard języka SQL dla relacyjnych baz danych opisano w rozdziałach 6. i 7. Reprezentacyjne modele danych reprezentują dane za

[4] Określenie *implementacyjny model danych* nie jest standardowym pojęciem. Wprowadziliśmy je tutaj na potrzeby opisu modeli danych dostępnych w komercyjnych systemach bazodanowych.

pomocą struktur rekordów, stąd czasami są nazywane **modelami danych opartymi na rekordach**.

Obiektowe modele danych należy traktować jak zupełnie nową rodzinę wysokopoziomowych implementacyjnych modeli danych, które są nieco bliższe koncepcyjnym modelom danych. Ogólne właściwości obiektowych baz danych i standardu *ODMG* (ang. *Object Data Management Group*) zaproponowanego przez grupę o tej samej nazwie omówimy w rozdziale 12. Obiektowe modele danych są często wykorzystywane także w roli wysokopoziomowych modeli koncepcyjnych (w szczególności dotyczy to dziedziny inżynierii oprogramowania).

Fizyczne modele danych opisują sposób przechowywania danych na komputerze w postaci plików, zatem reprezentują takie informacje jak formaty rekordów, uporządkowanie rekordów czy ścieżki dostępu. **Ścieżka dostępu** jest strukturą, która zwiększa efektywność operacji przeglądania bazy danych w poszukiwaniu konkretnych rekordów (np. dzięki indeksowaniu lub mieszaniu). Techniki fizycznego przechowywania danych i struktury dostępowe omówimy w rozdziałach 16. i 17. **Indeks** to przykładowa ścieżka umożliwiająca bezpośredni dostęp do danych za pomocą pojęcia z indeksu lub słowa kluczowego. Taka struktura przypomina indeks z końcowej części tej książki, przy czym może być uporządkowana liniowo, hierarchicznie (za pomocą struktury drzewiastej) lub w jeszcze inny sposób.

Inną kategorią są **samoopisujące się modele danych**. W systemach opartych na takich modelach opis danych jest połączony z ich wartościami. W tradycyjnych systemach SZBD opis (schemat) jest oddzielony od danych. Modele z tej kategorii to **XML** (rozdział 12.), a także liczne **magazyny danych typu klucz-wartość** i **systemy NOSQL** (rozdział 24.), opracowane ostatnio na potrzeby zarządzania big data.

2.1.2. Schematy, egzemplarze i stany baz danych

W każdym modelu danych bardzo ważne jest rozróżnienie *opisu* bazy danych od *samej bazy danych*. Opis bazy danych jest najczęściej nazywany **schematem bazy danych** — w założeniu taki schemat powstaje podczas projektowania bazy danych i nie powinien ulegać częstym zmianom[5]. W większości modeli danych istnieją pewne konwencje prezentowania schematów w postaci diagramów. Graficzna prezentacja schematu jest nazywana **diagramem schematu**. Na rysunku 2.1 zademonstrowano diagram schematu dla bazy danych zaprezentowanej w poprzednim rozdziale na rysunku 1.2 — diagram przedstawia jedynie strukturę poszczególnych typów rekordów, zatem nie obejmuje rzeczywistych egzemplarzy tych rekordów. Każdy obiekt w tego typu schemacie (np. `STUDENT` lub `PRZEDMIOT`) nazywamy **konstrukcją schematu**.

Diagram schematu przedstawia tylko *niektóre aspekty* reprezentowanego schematu bazy danych, w tym nazwy typów rekordów i elementów danych oraz niektóre rodzaje ograniczeń. Pozostałe aspekty schematu bazy danych w ogóle nie są wyrażane na tego

[5] Zmiany schematów baz danych są często nieuniknione w przypadku zmian wymagań leżących u podstaw danego rozwiązania. Nowsze systemy baz danych oferują nawet specjalne operacje stworzone z myślą o modyfikowaniu schematów.

typu diagramach; przykładowo, na rysunku 2.1 nie są widoczne ani typy danych poszczególnych elementów danych, ani związki łączące poszczególne pliki. Na diagramach schematów nie jest reprezentowanych wiele typów ograniczeń. Warto pamiętać, że ograniczenia w postaci „studenci informatyki muszą zdać egzamin z przedmiotu *INF1310* zanim zostanie im zaliczony drugi rok" są dosyć trudne do reprezentowania w postaci graficznej.

RYSUNEK 2.1. Diagram schematu dla bazy danych przedstawionej na rysunku 1.2

Rzeczywiste informacje przechowywane w bazie danych mogą się dosyć często zmieniać. Przykładowo, zawartość bazy danych zaprezentowanej na rysunku 1.2 zmienia się za każdym razem, gdy dodajemy nowego studenta lub wpisujemy nową ocenę dla istniejącego studenta. Informacje przechowywane w bazie danych w konkretnym momencie czasu są nazywane **stanem bazy danych** lub **migawką bazy danych** (**snapshotem**). Stan bazy danych jest także nazywany *bieżącym* zbiorem zawartych w niej **wystąpień** lub **egzemplarzy**. W danym stanie bazy danych każda konstrukcja schematu musi zawierać własny bieżący *zbiór wystąpień*; przykładowo, konstrukcja `STUDENT` będzie zawierała zbiór encji (rekordów) reprezentujących pojedynczych studentów, który będzie stanowił jej egzemplarze. Wiele stanów baz danych można konstruować w taki sposób, aby odpowiadały konkretnym schematom baz danych. Za każdym razem, gdy dodajemy lub usuwamy rekord albo przynajmniej zmieniamy wartość elementu danych w istniejącym rekordzie, zmieniamy stan bazy danych — a więc wprowadzamy bazę danych w nowy stan.

Rozróżnienie pomiędzy schematem bazy danych a stanem bazy danych jest bardzo ważne. Kiedy **definiujemy** nową bazę danych, określamy w systemie zarządzania bazą danych jedynie jej schemat. Na tym etapie baza danych znajduje się w tzw. *pustym stanie* (nie zawiera żadnych informacji). Bazę danych wprowadzamy w *stan początkowy* w momencie, w którym po raz pierwszy **wypełniamy** ją początkowymi danymi. Od tej pory każda zastosowana dla naszej bazy danych operacja aktualizacji powoduje, że otrzymujemy inny stan bazy danych. Warto pamiętać, że w każdym momencie można wyznaczyć dla bazy danych jej *stan bieżący*[6]. Za zapewnienie, aby każdy stan bazy danych był **stanem pra-**

[6] Stan bieżący bazy danych jest także nazywany *bieżącą migawką bazy danych*. Nazywano go też egzemplarzem bazy danych, jednak określenie *egzemplarz* wolimy stosować do poszczególnych rekordów.

widłowym (zgodnym z ograniczeniami i strukturą ze schematu), odpowiada po części system zarządzania bazą danych. Oznacza to, że szczególnie ważne jest zdefiniowanie na potrzeby systemu zarządzania bazą danych jej prawidłowego schematu — schemat bazy danych musi więc być projektowany z wyjątkową ostrożnością. System zarządzania bazą danych przechowuje konstrukcje zdefiniowanego schematu wraz z ograniczeniami (nazywanymi łącznie **metadanymi**) w swoim katalogu, dzięki czemu oprogramowanie systemu zarządzania bazą danych może się do takiego schematu odwoływać za każdym razem, gdy jest to konieczne. Schemat bazy danych jest niekiedy nazywany **intensją**, natomiast stan bazy danych jest nazywany ekstensją (**rozwinięciem**) określonego schematu.

Chociaż (jak już wspominaliśmy) schemat bazy danych nie powinien ulegać częstym zmianom, nierzadko okazuje się, że modyfikacje wprowadzone w wymaganiach wymuszają zastosowanie zmian także w schemacie. Przykładowo, możemy zdecydować, że każdy z rekordów danego pliku musi dodatkowo zawierać jeszcze jeden element danych — taka zmiana mogłaby polegać np. na dodaniu do schematu pliku STUDENT (patrz rysunek 2.1) elementu danych *DataUrodzenia*. Takie działania są niekiedy nazywane **ewolucją schematu**. Większość współczesnych systemów zarządzania bazami danych oferuje pewne operacje ułatwiające przeprowadzanie ewolucji danych, gdy baza już działa.

2.2. Trójwarstwowa architektura i niezależność danych

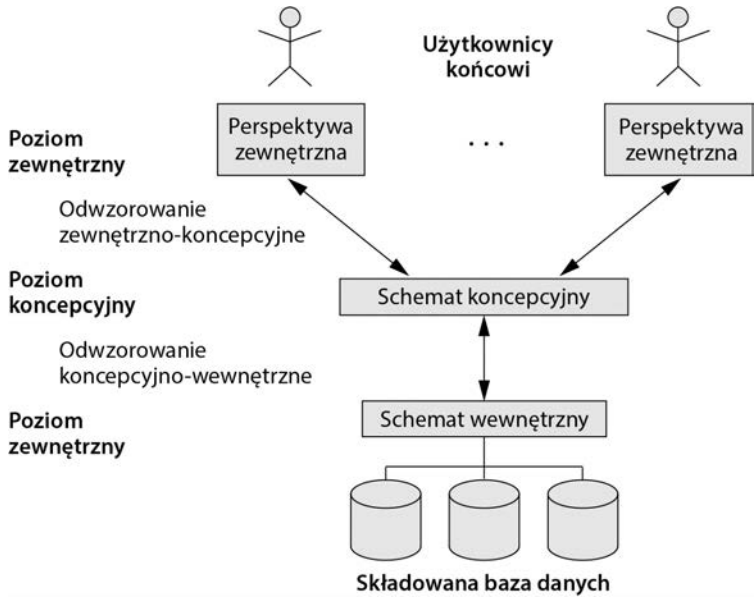
Oto trzy z czterech wymienionych w podrozdziale 1.3 ważnych cech rozwiązań opartych na bazach danych: (1) wykorzystywanie katalogu do przechowywania opisu (schematu) bazy danych, (2) oddzielenie programów od danych (tzw. niezależność programu od danych i niezależność programu od operacji) oraz (3) obsługa wielu perspektyw użytkowników. W tym podrozdziale omówimy jedną z popularnych architektur systemów baz danych, powszechnie znaną jako **architektura trójwarstwowa**[7], która została zaproponowana z myślą o ułatwieniu uzyskiwania (i wizualnego przedstawiania) tych własności w praktyce. W dalszej części tego podrozdziału omówimy pojęcia związane z niezależnością danych.

2.2.1. Architektura trójwarstwowa

Celem opracowania architektury trójwarstwowej (patrz rysunek 2.2) było oddzielenie aplikacji użytkownika od fizycznej bazy danych. Architektura trójwarstwowa przewiduje możliwość definiowania schematów na następujących trzech poziomach:

(1) **Poziom wewnętrzny** zawiera **wewnętrzny schemat**, który opisuje fizyczną strukturę przechowywania bazy danych. Wewnętrzny schemat wykorzystuje fizyczny model danych i opisuje wszystkie szczegóły związane z przechowywaniem informacji zawartej w bazie danych oraz z jej ścieżkami dostępu.

[7] Architektura trójwarstwowajest także nazywana *architekturą ANSI/SPARC* od skrótów organizacji, która to rozwiązanie zaproponowała (Tsichritzis i Klug, 1978).



RYSUNEK 2.2. Architektura trójwarstwowa

(2) **Poziom koncepcyjny** zawiera **schemat koncepcyjny**, który opisuje strukturę całej bazy danych na potrzeby społeczności końcowych użytkowników bazy danych. Schemat koncepcyjny ukrywa szczegóły fizycznych struktur przechowywania danych i koncentruje się na opisywaniu encji, typów danych, związków, operacji użytkownika oraz ograniczeń. Do opisywania schematów koncepcyjnych podczas implementowania systemu bazy danych wykorzystuje się zwykle reprezentacyjny model danych. *Implementacyjny schemat koncepcyjny* opiera się często na *koncepcyjnym projekcie schematu* w wysokopoziomowym modelu danych.

(3) **Poziom zewnętrzny** (nazywany także **poziomem perspektyw**) zawiera wiele **schematów zewnętrznych** lub **schematów użytkownika**. Każdy taki schemat zewnętrzny opisuje tylko tę część bazy danych, która znajduje się w obszarze zainteresowań konkretnej grupy użytkowników, i ukrywa przed tą grupą użytkowników pozostałe części bazy danych. Podobnie jak w poprzednim przypadku, każdy schemat zewnętrzny jest zazwyczaj implementowany w oparciu o reprezentacyjny model danych, często bazujący na zewnętrznym projekcie schematu w wysokopoziomowym modelu danych.

Architektura trójwarstwowa jest wygodnym narzędziem, za pomocą którego użytkownik może łatwo wizualizować poziomy schematów wykorzystywanego systemu bazy danych. Większość systemów zarządzania bazami danych nie oddziela tych trzech poziomów od siebie całkowicie — obsługuje architekturę tego typu tylko do pewnego stopnia. Niektóre starsze systemy zarządzania bazami danych mogą dołączać do schematów koncepcyjnych także szczegóły warstwy fizycznej. Trójwarstwowa architektura ANSI zajmuje ważne miejsce w rozwoju technologii bazodanowych, ponieważ wyraźnie odgranicza w kontekście projektowania baz poziomy zewnętrzny (użytkowników), koncepcyjny (bazy danych) i wewnętrzny (składowania danych). Nawet dziś ta architektura

znajduje zastosowania w projektowaniu SZBD. W większości systemów SZBD, które obsługują perspektywy użytkownika, schematy zewnętrzne są definiowane w oparciu o ten sam model danych, na bazie którego opisano schemat poziomu koncepcyjnego (w relacyjnych SZBD, takich jak Oracle lub SQL Server, używany jest do tego SQL).

Warto pamiętać, że wymienione trzy poziomy schematów stanowią wyłącznie *opisy* właściwych danych; informacje są zapisywane jedynie na poziomie fizycznym. W architekturze trójwarstwowej każda grupa użytkowników odwołuje się wyłącznie do własnego schematu zewnętrznego. Oznacza to, że system zarządzania bazą danych musi przekształcić żądanie zdefiniowane na schemacie zewnętrznym w odpowiednie żądanie na schemacie koncepcyjnym, po czym przekształcić otrzymany wynik w prawidłowe żądanie na schemacie wewnętrznym, które umożliwi przetwarzanie rzeczywistych informacji przechowywanych w bazie danych. Jeśli żądanie dotyczy odczytania wybranych informacji zawartych w bazie danych, dane wyciągnięte z bazy muszą zostać ponownie sformatowane do postaci odpowiadającej zewnętrznej perspektywie użytkownika. Zachodzące pomiędzy wspomnianymi trzema poziomami procesy przekształcania żądań i uzyskanych wyników są nazywane **odwzorowaniami**. Tego typu operacje mogą być kosztowne czasowo, co powoduje, że niektóre SZBD (w szczególności te, które w założeniu mają obsługiwać niewielkie bazy danych) w ogóle nie obsługują zewnętrznych perspektyw. Jednak nawet w takich systemach wykonywanie pewnych operacji odwzorowywania okazuje się niezbędne podczas przekształcania żądań pomiędzy poziomem koncepcyjnym a poziomem wewnętrznym.

2.2.2. Niezależność danych

Omówioną przed chwilą architekturę trójwarstwową możemy wykorzystać podczas omawiania pojęcia **niezależności danych**, które można zdefiniować jako możliwość modyfikowania schematu na jednym poziomie systemu bazy danych bez konieczności wprowadzania zmian w schemacie znajdującym się na kolejnym (wyższym) poziomie. Możemy zdefiniować dwa typy niezależności danych:

(1) **Logiczna niezależność danych** oznacza możliwość zmiany schematu koncepcyjnego bez konieczności modyfikowania schematów zewnętrznych ani aplikacji. Zmiana schematu koncepcyjnego może mieć na celu rozbudowę bazy danych (przez dodanie nowego typu rekordu lub nowego elementu danych), modyfikację ograniczeń lub odchudzenie bazy danych (przez usunięcie typu rekordu lub elementu danych). W tym ostatnim przypadku schematy zewnętrzne odwołujące się wyłącznie do danych, które nie są usuwane i pozostają w bazie danych, w ogóle nie powinny podlegać zmianom. Przykładowo, schemat zewnętrzny z rysunku 1.5(a) nie powinien ulec zmianie na skutek zaprezentowanych na rysunku 1.6(a) modyfikacji w pliku `RAPORT_OCEN` (względem stanu przedstawionego na rysunku 1.2). System zarządzania bazą danych, który zapewnia logiczną niezależność danych, musi jedynie odpowiednio zmodyfikować definicję perspektywy i właściwe reguły odwzorowywania. Po logicznej reorganizacji schematu koncepcyjnego programy aplikacji, które odwołują się do konstrukcji schematu zewnętrznego, mogą pracować tak samo jak wcześniej. Ewentualne zmiany ograniczeń mogą być stosowane dla schematu koncepcyjnego bez wpływu ani na schematy zewnętrzne, ani na aplikacje.

(2) **Fizyczna niezależność danych** oznacza możliwość zmiany schematu wewnętrznego bez konieczności modyfikowania schematu koncepcyjnego. Oznacza to, że konieczność zmian nie dotyczy także schematów zewnętrznych. Zmiany w schematach wewnętrznych mogą się natomiast okazać niezbędne z uwagi na potrzebę reorganizacji niektórych plików fizycznych (np. przez stworzenie dodatkowych struktur dostępu) w celu poprawienia wydajności operacji odczytywania lub aktualizacji danych. Jeśli po wprowadzeniu zmian baza danych zawiera dokładnie takie same dane jak przedtem, modyfikowanie schematu koncepcyjnego nie powinno być konieczne. Przykładowo, dodanie ścieżki dostępu, która ma poprawić szybkość wyszukiwania żądanych rekordów z pliku KURS (patrz rysunek 1.2) według wartości elementów danych *Semestr* i *Rok*, nie powinno wymagać modyfikowania takich zapytań jak „wymień wszystkie kursy omawiane w semestrze jesiennym roku 1998", chociaż dzięki nowej ścieżce dostępu wspomniane zapytanie będzie szybciej wykonywane przez system zarządzania bazą danych.

Fizyczna niezależność danych występuje w większości środowisk bazodanowych i plikowych, w których fizyczne szczegóły (takie jak dokładna lokalizacja danych na dysku oraz sprzętowe aspekty kodowania, rozmieszczania, kompresji, podziału i scalania rekordów) są ukryte przed użytkownikami. Aplikacje nie znają tych szczegółów. Trudniej jest osiągnąć logiczną niezależność danych, ponieważ ma ona umożliwiać wprowadzanie zmian struktury i ograniczeń bez wpływu na aplikacje, co jest dużo bardziej restrykcyjnym wymogiem.

Katalog każdego wielowarstwowego SZBD musi być stale rozszerzany w taki sposób, aby zawierał informacje niezbędne do odwzorowywania żądań i danych pomiędzy różnymi poziomami. Systemy zarządzania bazami danych wykorzystują do takiego odwzorowywania dodatkowe oprogramowanie, które odwołuje się do odpowiednich reguł w katalogach. Mamy więc do czynienia z niezależnością danych, ponieważ zmiana schematu na jednym poziomie nie wiąże się z koniecznością zmiany schematu na kolejnym, wyższym poziomie — niezbędne modyfikacje należy jedynie wprowadzić w regułach *odwzorowywania* na obu poziomach. Oznacza to, że aplikacje odwołujące się do schematu na wyższym poziomie w ogóle nie muszą być zmieniane.

2.3. Języki i interfejsy baz danych

W podrozdziale 1.4 omówiliśmy różne kategorie użytkowników końcowych wykorzystujących systemy zarządzania bazami danych. Takie systemy muszą oczywiście udostępniać odpowiednie języki i interfejsy dla każdej z tych kategorii. W tym podrozdziale omówimy rozmaite typy języków i interfejsów oferowanych poszczególnym kategoriom użytkowników przez systemy zarządzania bazami danych.

2.3.1. Języki systemów zarządzania bazami danych

Kiedy już zakończy się faza projektowania bazy danych i zostanie wybrany system zarządzania bazą danych, w którym nowa baza danych będzie implementowana, najważniejszym zadaniem jest opracowanie koncepcyjnego i wewnętrznego schematu tej bazy danych

oraz reguł odwzorowywania pomiędzy tymi poziomami. W wielu systemach zarządzania bazami danych, w których nie ma ścisłego rozgraniczenia pomiędzy utrzymywanymi poziomami, do definiowania obu schematów przez administratorów i projektantów baz danych wykorzystuje się jeden język, nazywany **językiem definicji danych** (ang. *Data Definition Language — DDL*). Takie systemy zarządzania bazami danych zawierają specjalne kompilatory języka DDL, które odpowiadają za przetwarzanie instrukcji opisujących konstrukcje schematu oraz za umieszczanie takiego przetworzonego opisu w katalogu SZBD.

W tych systemach zarządzania bazami danych, w których jest utrzymywany ścisły podział pomiędzy poziomem koncepcyjnym a poziomem wewnętrznym, język DDL wykorzystuje się wyłącznie do określania schematu koncepcyjnego. Do określania schematu wewnętrznego wykorzystywany jest wówczas inny język, nazywany **językiem definicji składowania** (ang. *Storage Definition Language — SDL*). Reguły odwzorowywania pomiędzy tymi dwoma schematami mogą być definiowane za pomocą odpowiednich instrukcji jednego z tych języków. Obecnie w większości relacyjnych SZBD *nie występuje konkretny język* używany jako SDL. Schemat wewnętrzny jest definiowany za pomocą funkcji, parametrów i specyfikacji dotyczących składowania plików. Administratorzy baz danych mogą dzięki temu wybierać sposoby indeksowania i odwzorowywania danych na mechanizmy ich przechowywania. Prawdziwa architektura trójwarstwowa będzie oczywiście wymagała jeszcze jednego języka — **języka definicji perspektyw** (ang. *View Definition Language — VDL*), za pomocą którego projektant lub administrator bazy danych będzie mógł określać perspektywy użytkownika wraz z ich odwzorowaniami do schematu koncepcyjnego — jednak w większości systemów zarządzania bazami danych *do definiowania schematu koncepcyjnego i zewnętrznego wykorzystuje się język DDL*. W relacyjnych SZBD jako VDL stosowany jest SQL, używany do definiowania **perspektyw** użytkowników lub aplikacji za pomocą wcześniej przygotowanych zapytań (patrz rozdziały 6. i 7.).

Kiedy już schematy bazy danych zostaną prawidłowo skompilowane, a sama baza danych zostanie wypełniona właściwymi danymi, użytkownicy muszą jeszcze otrzymać jakieś mechanizmy, które umożliwią im operowanie na gotowej bazie danych. Typowe działania w tym zakresie obejmują odczytywanie, wstawianie, usuwanie i modyfikowanie danych. W tym celu systemy zarządzania bazami danych udostępniają odpowiedni zbiór operacji lub tzw. **język manipulowania danymi** (ang. *Data Manipulation Language — DML*).

We współczesnych systemach zarządzania bazami danych wymienione przed chwilą rodzaje języków zwykle *nie są od siebie ściśle oddzielane* — zamiast tego, stosuje się jeden uniwersalny, zintegrowany język, który zawiera konstrukcje niezbędne do definiowania schematu koncepcyjnego, do definiowania perspektyw oraz do operowania na danych. Jedynie język definicji składowania jest zwykle wyodrębniany, ponieważ wykorzystuje się go do definiowania fizycznych struktur przechowywania danych głównie po to, aby poprawić wydajność systemu bazy danych, za co zwykle odpowiada administrator bazy danych. Typowym przykładem wspomnianego wszechstronnego języka bazy danych jest stosowany w relacyjnych bazach danych język SQL (patrz rozdziały 6. i 7.), który nie tylko stanowi połączenie języków DDL, VDL i DML, ale także obejmuje instrukcje dla specyfikacji ograniczeń, modyfikowania schematu i wielu innych zadań. Wczesne wersje SQL-a zawierały język SDL, jednak w pewnym momencie zarzucono pomysł umieszczania tego języka w tej formie w SQL-u i pozostawiono wyłącznie obsługę poziomu koncepcyjnego i zewnętrznego.

Istnieją dwa główne typy języków manipulowania danymi (DML). **Wysokopoziomowy** (**nieproceduralny**) język DML może być wykorzystywany do stosunkowo zwięzłego definiowania skomplikowanych operacji na bazie danych. Wiele systemów zarządzania bazami danych obsługuje zarówno te konstrukcje wysokopoziomowego języka DML, które zostaną wpisane przez użytkownika w odpowiednim terminalu, jak i te, które programista osadzi w kodzie uniwersalnego języka programowania. W tym drugim przypadku konstrukcje języka DML muszą być tak oznaczone w kodzie programu, aby prekompilator mógł je wyodrębnić i przekazać do przetworzenia przez system zarządzania bazą danych. **Niskopoziomowy** (**proceduralny**) język DML *musi* być osadzany w kodzie napisanym w uniwersalnych językach programowania. Język DML tego typu jest zwykle wykorzystywany do odczytywania z bazy danych pojedynczych rekordów lub obiektów i ich odrębnego przetwarzania. Oznacza to, że odczytywanie i przetwarzanie poszczególnych rekordów należących do całego zbioru wymaga użycia konstrukcji (np. pętli) wykorzystywanego języka programowania.Z tego powodu niskopoziomowe języki DML są często nazywane językami typu **record-at-a-time**. Wysokopoziomowe języki DML, w tym SQL, mogą określać i odczytywać wiele rekordów za pomocą pojedynczej konstrukcji i w związku z tym są nazywane językami typu **set-at-a-time** lub **set-oriented**. Zapytanie zdefiniowane w wysokopoziomowym języku DML zwykle określa tylko to, *które* dane mają zostać odczytane, nie określa natomiast *sposobu* ich odczytania — takie języki są w związku z tym często nazywane językami **deklaratywnymi**.

W sytuacji, gdy polecenia języka DML (niezależnie od tego, czy jest to język wysokopoziomowy, czy niskopoziomowy) są osadzane w uniwersalnym języku programowania, język ten jest nazywany **językiem nadrzędnym**, natomiast język DML jest nazywany **podjęzykiem danych**[8]. Z drugiej strony, wysokopoziomowy język DML wykorzystywany przez użytkownika w sposób odrębny jest nazywany **językiem zapytań**. Generalnie, zarówno polecenia odczytywania, jak i polecenia aktualizacji wysokopoziomowego języka DML mogą być wykorzystywane bezpośrednio przez użytkownika, zatem są traktowane jak części języka zapytań[9].

Dorywczy użytkownicy końcowi wykorzystują zwykle wysokopoziomowe języki zapytań do określania swoich żądań, natomiast programiści korzystają najczęściej z języków DML w formie instrukcji osadzonych. Niedoświadczeni użytkownicy parametryczni wykorzystują zwykle specjalnie przygotowane **przyjazne interfejsy użytkownika** bazy danych — takie interfejsy mogą również być stosowane przez użytkowników doraźnych i inne osoby, które nie chcą się uczyć szczegółów związanych z używaniem wysokopoziomowych języków zapytań. Omówimy te typy interfejsów w kolejnym punkcie.

[8] W obiektowych bazach danych język nadrzędny i podjęzyk danych tworzą zwykle jeden zintegrowany język — przykładowo, język programowania C++ z pewnymi rozszerzeniami obsługuje funkcje bazy danych. Niektóre relacyjne systemy zarządzania bazami danych dodatkowo udostępniają zintegrowane języki — przykładowo, system Oracle oferuje język PL/SQL.

[9] Zgodnie z ogólnie przyjętym znaczeniem słowa *zapytanie* należałoby go używać wyłącznie w odniesieniu do operacji odczytywania danych, a nie do aktualizacji.

2.3.2. Interfejsy systemów zarządzania bazami danych

Poniżej wymieniono przyjazne interfejsy użytkownika oferowane przez systemy zarządzania bazami danych.

Interfejsy oparte na menu dla klientów WWW lub przeglądarek. Interfejsy tego typu prezentują użytkownikom listy dostępnych opcji, nazywane **menu**, które przeprowadzają użytkowników przez proces formułowania żądań. Wykorzystywane w ten sposób menu eliminują konieczność zapamiętywania konkretnych poleceń i składni języka zapytań; zamiast tego zapytanie jest tworzone krok po kroku przez wybieranie kolejnych opcji z menu wyświetlanych przez oprogramowanie systemu bazy danych. W przypadku **interfejsów internetowych** (opartych na stronach WWW) bardzo popularne są tzw. menu rozwijane. Takie menu są także popularne w **interfejsach przeglądania**, które umożliwiają użytkownikom przeszukiwanie zawartości bazy danych (choćby w celach orientacyjnych).

Aplikacje dla urządzeń mobilnych. Takie interfejsy umożliwiają użytkownikom urządzeń mobilnych dostęp do danych. Przykładowo, banki, systemy rezerwacji i firmy ubezpieczeniowe udostępniają aplikacje umożliwiające użytkownikom dostęp do danych za pomocą telefonów komórkowych i innych urządzeń mobilnych. Takie aplikacje mają wbudowane zaprogramowane interfejsy, które zwykle pozwalają użytkownikom zalogować się za pomocą nazwy i hasła, a następnie udostępniają ograniczone menu z opcjami mobilnego dostępu do danych, a także z operacjami takimi jak opłacanie rachunków (w bankach) lub dokonywanie rezerwacji (w serwisach internetowych obsługujących rezerwacje).

Interfejsy oparte na formularzach. Interfejsy oparte na formularzach wyświetlają każdemu użytkownikowi formularz. Użytkownicy mogą wypełniać wszystkie pola takiego **formularza**, aby wstawić nowe dane, lub tylko niektóre z nich, aby system zarządzania bazą danych zwrócił właściwe dane wypełniające pozostałe pola. Formularze są zwykle projektowane i programowane z myślą o niedoświadczonych użytkownikach i pełnią rolę interfejsu dla wykonywanych przez nich podstawowych transakcji. Wiele systemów zarządzania bazami danych oferuje specjalne **języki definiowania formularzy**, ułatwiające programistom tworzenie tego typu struktur. SQL*Forms to oparty na formularzach język, w którym zapytania są tworzone za pomocą formularza projektowanego na podstawie schematu relacyjnej bazy danych. Oracle Forms to komponent pakietu produktów firmy Oracle udostępniający bogaty zestaw funkcji do projektowania i budowania aplikacji z użyciem formularzy. Niektóre systemy udostępniają narzędzia, za pomocą których użytkownicy końcowi mogą interaktywnie budować proste formularze w środowisku graficznym.

Graficzne interfejsy użytkownika. Graficzny interfejs użytkownika (ang. *Graphical User Interface — GUI*) przeważnie wyświetla użytkownikowi schemat bazy danych w formie graficznej. Użytkownik może wówczas tworzyć zapytania przez operowanie diagramem. W wielu przypadkach graficzne interfejsy użytkownika opierają się zarówno na menu, jak i na formularzach.

Interfejsy języka naturalnego. Interfejsy tego typu akceptują żądania pisane w języku naturalnym (najczęściej angielskim) i próbują je *zrozumieć* i odpowiednio przetworzyć. Interfejsy języka naturalnego zawierają zwykle nie tylko własne *schematy*, które są zbliżone do koncepcyjnego schematu bazy danych, ale także słowniki ważnych słów. Podczas interpretowania otrzymanego żądania interfejs języka naturalnego odwołuje się zarówno do słów zdefiniowanych w tym schemacie, jak i do zbioru standardowych słów zawartych w jego słowniku. Jeśli proces interpretacji żądania zakończy się powodzeniem, interfejs języka naturalnego generuje odpowiednie zapytanie wysokopoziomowe i przekazuje je do przetworzenia do systemu zarządzania bazą danych; w przeciwnym przypadku nawiązywany jest dialog z użytkownikiem, mający na celu uszczegółowienie żądania.

Przeszukiwanie baz na podstawie słów kluczowych. Te interfejsy przypominają nieco wyszukiwarki internetowe, które przyjmują słowa w języku naturalnym (np. angielskim lub polskim) i dopasowują je do dokumentów z konkretnej witryny (w wyszukiwarkach lokalnych) lub stron WWW z całego internetu (w wyszukiwarkach takich jak Google lub Ask). Stosuje się tu predefiniowane indeksy słów i funkcje rankingowe do pobierania i wyświetlania dokumentów w kolejności wyznaczanej przez poziom dopasowania. W ustrukturyzowanych bazach relacyjnych tego rodzaju dające pełną swobodę interfejsy tekstowe do tworzenia zapytań nie są na razie częste, choć niedawno powstała dziedzina badań dotycząca **zapytań opartych na słowach kluczowych** w kontekście baz relacyjnych.

Głosowe przekazywanie danych wejściowych i wyjściowych. Coraz popularniejsze staje się ograniczone wykorzystanie mowy do podawania zapytań oraz zwracania odpowiedzi lub wyników. Aplikacje z ograniczonym słownikiem (np.: informacji z książki telefonicznej, o odlotach i przylotach czy o kontach do kart kredytowych) pozwalają na wejściu i na wyjściu stosować mowę, aby umożliwić klientom dostęp do danych. Wejściowe dane głosowe są wykrywane na podstawie biblioteki predefiniowanych słów i używane do określenia parametrów przekazywanych do zapytania. Na wyjściu wykonywana jest podobna konwersja tekstu lub liczb na mowę.

Interfejsy dla użytkowników parametrycznych. Użytkownicy parametryczni, tacy jak kasjerzy bankowi, często wielokrotnie wykonują stosunkowo niewielki i stały zbiór operacji. Przykładowo, kasjer może za pomocą jednego przycisku funkcyjnego uruchamiać rutynowe i powtarzalne transakcje, takie jak wpłaty lub wypłaty, a także odpowiadać na zapytania o stan konta. Analitycy systemowi i programiści odpowiednio projektują i implementują specjalny interfejs dla każdej znanej klasy takich niedoświadczonych użytkowników. Tworzone w ten sposób interfejsy obsługują zwykle niewielki zbiór skróconych poleceń, ponieważ celem jest zminimalizowanie liczby naciskanych klawiszy niezbędnej do wywołania poszczególnych żądań.

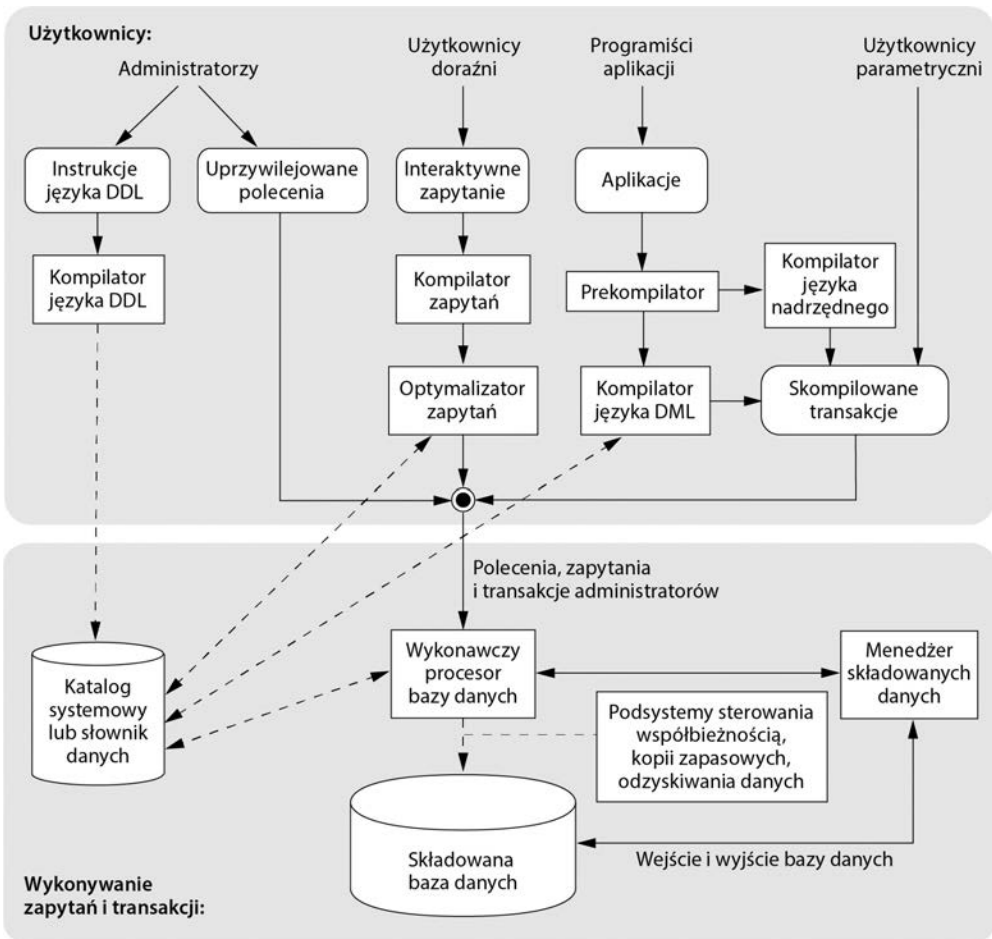
Interfejsy dla administratorów baz danych. Większość systemów baz danych zawiera uprzywilejowane polecenia, które mogą być wykorzystywane wyłącznie przez administratorów. Należą do nich polecenia tworzenia kont użytkowników, ustawiania parametrów systemowych, przypisywania uprawnień do kont użytkowników, modyfikowania schematu oraz reorganizowania struktur przechowywania bazy danych.

2.4. Środowisko systemu bazy danych

System zarządzania bazą danych jest skomplikowanym systemem oprogramowania. W tym podrozdziale omówimy typy składników programowych, które połączone tworzą SZBD, wraz z rodzajami oprogramowania systemu komputerowego, z którymi SZBD współpracuje.

2.4.1. Moduły składające się na system zarządzania bazą danych

Na rysunku 2.3 przedstawiono w uproszczonej formie typowe składniki systemu zarządzania bazą danych. Rysunek jest podzielony na dwie części. Górna przedstawia różnych użytkowników środowiska bazy danych i ich interfejsy. W dolnej widoczne są wewnętrzne moduły SZBD odpowiedzialne za składowanie danych i przetwarzanie transakcji.



RYSUNEK 2.3. Moduły składające się na system zarządzania bazą danych i związane z nimi interakcje

Sama baza danych i katalog systemu zarządzania bazą danych są zwykle przechowywane na dysku. Dostęp do dysku jest kontrolowany głównie przez **system operacyjny** (ang. *Operating System — OS*), który ustala kolejność wykonywania operacji odczytu-zapisu. Liczne systemy zarządzania bazami danych zawierają własne **moduły zarządzania buforami**, służące do szeregowania operacji odczytu i zapisu na dysku. Te moduły są używane, ponieważ zarządzanie zawartością bufora ma istotny wpływ na wydajność. Przyspieszenie odczytów i zapisów znacząco zwiększa wydajność. Znajdujący się na wyższym poziomie i należący do systemu zarządzania bazą danych moduł **menadżera składowanych danych** kontroluje dostęp do informacji umieszczonych na dysku niezależnie od tego, czy są to informacje znajdujące się w bazie danych, czy w katalogu.

Przyjrzyj się najpierw górnej części rysunku 2.3. Widoczne są tam interfejsy dla administratorów bazy danych, użytkowników doraźnych (którzy formułują zapytania za pomocą interaktywnych interfejsów), programistów aplikacji (tworzących programy za pomocą nadrzędnych języków programowania) i użytkowników parametrycznych (wprowadzających dane przez podawanie parametrów predefiniowanych transakcji). Administratorzy bazy danych definiują i dostrajają bazę, wprowadzając zmiany w jej definicji za pomocą języka DDL i poleceń wymagających wysokich uprawnień.

Kompilator języka DDL przetwarza zdefiniowane w tym języku specyfikacje schematów i zapisuje opisy przetworzonych schematów (metadane) w katalogu systemu zarządzania bazą danych. Katalog zawiera takie informacje jak nazwy i rozmiary poszczególnych plików, nazwy i typy danych poszczególnych elementów danych, szczegóły przechowywania każdego z plików, informacje o regułach odwzorowywania pomiędzy schematami, specyfikacje ograniczeń oraz wiele dodatkowych rodzajów informacji, które są niezbędne do pracy rozmaitych modułów systemu zarządzania bazą danych.

Użytkownicy doraźni i osoby, które sporadycznie potrzebują danych z bazy, komunikują się z nią za pomocą interfejsu do zgłaszania **interaktywnych zapytań** (patrz rysunek 2.3). *Nie przedstawiliśmy tu bezpośrednio* żadnych opartych na menu, formularzu lub rozwiązaniach mobilnych interakcji, które zwykle są używane do automatycznego generowania interaktywnych zapytań lub dostępu do transakcji zapuszkowanych. Takie zapytania są parsowane i sprawdzane pod kątem poprawności składni oraz nazw plików i elementów danych przez **kompilator zapytań**, który kompiluje je do wewnętrznej postaci. Zapytania w wewnętrznej postaci są optymalizowane (patrz rozdziały 18. i 19.). **Optymalizator zapytań** odpowiada m.in. za zmianę układu operacji, eliminowanie nadmiarowości i wykorzystanie wydajnych algorytmów wyszukiwania w trakcie przetwarzania zapytania. Optymalizator sprawdza w katalogu systemu statystyczne i fizyczne informacje o składowanych danych oraz generuje kod wykonywalny, który przeprowadza operacje potrzebne do obsługi zapytania i kieruje wywołania do procesora wykonawczego.

Programiści aplikacji piszą programy w językach nadrzędnych takich jak Java, C i C++. Te programy są przekazywane do prekompilatora. **Prekompilator** wyodrębnia polecenia języka DML z kodu aplikacji napisanego w języku nadrzędnym. Takie polecenia są następnie przekazywane do kompilatora języka DML, który kompiluje je do postaci kodu obiektów wykorzystywanych w operacjach dostępu do bazy danych. Reszta programu jest przesyłana do kompilatora języka nadrzędnego. Kody obiektów dla poleceń języka DML pozostają jednak połączone ze skompilowanym programem języka nadrzędnego i tworzą tzw. zapuszkowane transakcje, których kod zawiera wywołania operacji wy-

konawczego procesora bazy danych. Do pisania programów bazodanowych coraz częściej stosuje się języki skryptowe takie jak PHP i Python. Zapuszkowane transakcje są wykonywane wielokrotnie przez użytkowników parametrycznych za pomocą komputerów osobistych lub aplikacji mobilnych. Tacy użytkownicy określają tylko parametry transakcji. Każde wywołanie jest wtedy uznawane za odrębną transakcję. Przykładem jest tu transakcja płatności w banku, gdzie jako parametry można podać numer konta, odbiorcę i kwotę.

W dolnej części rysunku 2.3 **wykonawczy procesor bazy danych** uruchamia (1) uprzywilejowane polecenia, (2) wykonywalne plany zapytań i (3) zapuszkowane transakcje z parametrami wykonawczymi. Ten procesor korzysta z **katalogu systemu** i może aktualizować zapisane w nim statystyki. Używa też **menedżera danych składowanych**, który z kolei korzysta z podstawowych usług systemu operacyjnego do wykonywania niskopoziomowych operacji wejścia-wyjścia (odczytu i zapisu) z wykorzystaniem dysku i pamięci głównej. Wykonawczy procesor bazy obsługuje też inne aspekty przenoszenia danych, takie jak zarządzanie buforami w pamięci głównej. Niektóre SZBD mają własny moduł zarządzania buforami, natomiast inne korzystają w tym obszarze z systemu operacyjnego. Na rysunku systemy **sterowania współbieżnego** oraz **tworzenia kopii bezpieczeństwa i odtwarzania bazy** są przedstawione jako odrębny moduł. Na potrzeby transakcji te komponenty są integrowane z działaniem wykonawczego procesora bazy danych.

Popularnym rozwiązaniem jest wykorzystywanie **programu klienta**, który uzyskuje dostęp do systemu zarządzania bazą danych, działając na innym komputerze niż ten, na którym znajduje się baza danych. Pierwszy komputer jest wówczas nazywanym **komputerem klienta**, natomiast drugi komputer nosi nazwę **serwera bazy danych**. W niektórych przypadkach klient wykorzystuje dodatkowo komputer pośredniczący, nazywany **serwerem aplikacji**, który w imieniu klienta uzyskuje bezpośredni dostęp do serwera bazy danych. Omówimy to zagadnienie bardziej szczegółowo w podrozdziale 2.5.

Rysunek 2.3 nie ma ilustrować żadnego konkretnego systemu zarządzania bazą danych, a jedynie przedstawiać typowe związki pomiędzy modułami SZBD. System zarządzania bazą danych wykorzystuje usługi systemu operacyjnego w momencie uzyskiwania dostępu do informacji zapisanych na dysku — w bazie danych lub w katalogu. Jeśli dany system komputerowy jest współużytkowany przez wielu użytkowników, system operacyjny odpowiada za właściwe zaplanowanie kolejności obsługi otrzymywanych żądań dostępu oraz prawidłowe zarządzanie czasem przyznawanym procesowi systemu zarządzania bazą danych i innym działającym procesom. Z drugiej strony, jeśli dany system komputerowy jest przeznaczony przede wszystkim do zapewnienia odpowiedniego środowiska działania dla serwera bazy danych, system zarządzania bazą danych będzie kontrolował buforowanie stron dyskowych w głównej pamięci. System zarządzania bazą danych odpowiada także za dostarczanie niezbędnych interfejsów nie tylko dla kompilatorów uniwersalnych, nadrzędnych języków programowania, ale także dla serwerów aplikacji i programów klienckich działających na innych komputerach i komunikujących się za pośrednictwem interfejsu sieciowego.

2.4.2. Narzędzia systemu bazy danych

Poza opisanymi przed chwilą modułami programowymi większość systemów zarządzania bazami danych dodatkowo udostępnia **narzędzia baz danych**, które ułatwiają ich administratorom zarządzanie systemami baz danych. Do najczęściej stosowanych typów i funkcji oferowanych przez narzędzia tego typu należą:

- **Wczytywanie***:* Narzędzia wczytujące są wykorzystywane do kopiowania zawartości zewnętrznych plików z danymi (np. plików tekstowych lub plików sekwencyjnych) do bazy danych. Narzędzia tego typu otrzymują na wejściu bieżący (źródłowy) format pliku z danymi oraz oczekiwaną (docelową) strukturę pliku bazy danych, i na tej podstawie automatycznie formatują i zapisują odpowiednie informacje w bazie danych. Wraz z rosnącą popularnością systemów zarządzania bazami danych, konieczność przenoszenia danych pomiędzy różnymi pakietami tego typu staje się w wielu organizacjach codziennością. Niektórzy producenci oferują nawet **narzędziami konwersji** generujące — w oparciu o opisy (schematy wewnętrzne) istniejącego formatu składowania informacji i docelowego formatu przechowywania informacji w bazie danych — odpowiednie programy wczytujące.
- **Tworzenie kopii zapasowej***:* Narzędzia tego typu tworzą kopię zapasową bazy danych (takie działania polegają zwykle na wykonaniu zrzutu całej bazy danych i zapisaniu go na taśmie lub innym nośniku). Kopia zapasowa może być następnie wykorzystana do przywrócenia bazy danych w przypadku katastrofalnej awarii dysku. Często stosuje się technikę tworzenia przyrostowych kopii zapasowych, w której rejestrowane są wyłącznie zmiany, jakie miały miejsce od ostatniej operacji utworzenia takiej kopii. Wykonywanie przyrostowej kopii zapasowej jest co prawda bardziej skomplikowane, ale pozwala oszczędzić przestrzeń na wykorzystywanym nośniku.
- **Reorganizacja plików:** Narzędzia tego typu mogą być wykorzystywane do reorganizowania plików baz danych z wykorzystaniem innej struktury i nowych ścieżek dostępu, najczęściej z myślą o poprawie wydajności.
- **Monitorowanie wydajności:** Takie narzędzia monitorują obciążenie bazy danych i na tej podstawie generują odpowiednie dane statystyczne dla administratora tej bazy. Administrator bazy danych może wykorzystać te statystyki podczas podejmowania decyzji np. w kwestii ewentualnej reorganizacji plików albo dodanie lub usunięcia indeksów celem poprawy wydajności.

Systemy zarządzania bazami danych mogą także oferować funkcje sortowania plików, obsługi kompresji danych, monitorowania operacji dostępu wykonywanych przez użytkowników, obsługi interfejsów sieciowych oraz wykonywania innych przydatnych zadań.

2.4.3. Narzędzia, środowiska aplikacji oraz mechanizmy komunikacji

Projektanci, użytkownicy i SZBD mogą korzystać także z innych przydatnych narzędzi. W fazie projektowania systemów baz danych można wykorzystywać narzędzia typu CASE[10]. Do narzędzi, które mogą być przydatne w wielkich organizacjach, należy także rozszerzony **system słownika danych**, nazywany także **systemem repozytorium danych**. Poza danymi katalogowymi na temat schematów i ograniczeń słownik danych może zawierać także inne informacje, takie jak decyzje projektowe, standardy wykorzystywania bazy danych, opisy aplikacji i dane o użytkownikach. Systemy tego typu są także nazywane **repozytoriami danych**. W razie potrzeby użytkownicy lub administrator bazy danych mogą mieć *bezpośredni* dostęp do informacji zawartych w takim repozytorium. Rozwiązania obsługujące słowniki danych przypominają co prawda katalogi systemów zarządzania bazami danych, jednak zawierają więcej różnych informacji i są udostępniane przede wszystkim użytkownikom, nie oprogramowaniu SZBD (jak w przypadku katalogów).

Środowiska wytwarzania aplikacji, takie jak systemy PowerBuilder (firmy Sybase) czy JBuilder (Borland), są dość popularne. Takie systemy oferują środowisko ułatwiające szybkie wytwarzanie aplikacji baz danych, a także upraszczają wiele aspektów opracowywania systemów baz danych, w tym projektowania baz danych, tworzenia graficznych interfejsów użytkownika, przygotowywania zapytań i operacji aktualizujących, oraz budowania aplikacji.

Systemy zarządzania bazami danych potrzebują także interfejsów do **oprogramowania komunikacyjnego**, które ma umożliwiać zdalnym użytkownikom korzystanie z systemu bazy danych i dostęp do zawartych tam danych za pośrednictwem komputerowych terminali, stacji roboczych lub ich lokalnych komputerów osobistych. Komputery takich użytkowników są połączone z serwerem bazy danych za pomocą odpowiedniego sprzętu komunikacyjnego — routerów internetowych, linii telefonicznych, sieci rozległych, sieci lokalnych lub urządzeń komunikacji satelitarnej. Wiele komercyjnych systemów baz danych oferuje specjalne pakiety komunikacyjne, które współpracują z systemami zarządzania bazami danych. Systemy tego typu zintegrowane z systemami komunikacyjnymi są nazywane systemami **DB/DC**. Niektóre rozproszone systemy zarządzania bazami danych są dodatkowo fizycznie instalowane na wielu komputerach. W takim przypadku do łączenia komputerów wykorzystuje się sieci komunikacyjne. Najczęściej są to **sieci lokalne** (ang. *Local Area Network — LAN*), jednak mogą to być także inne typy sieci komputerowych.

[10] Chociaż CASE oznacza wspomaganą komputerowo inżynierię oprogramowania (ang. *Computer-Aided Software Engineer*), wiele narzędzi typu CASE jest wykorzystywanych przede wszystkim do projektowania baz danych.

2.5. Architektury systemów zarządzania bazami danych — scentralizowane i typu klient-serwer

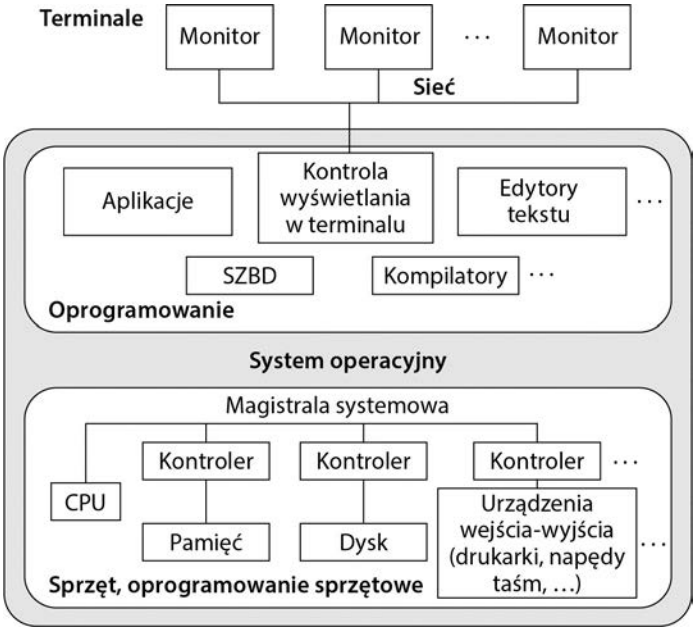
2.5.1. Scentralizowane architektury systemów zarządzania bazami danych

Architektury systemów zarządzania bazami danych ewoluowały zgodnie z ogólnymi trendami panującymi w świecie systemów komputerowych. Wcześniejsze architektury opierały się na centralnych komputerach, które obsługiwały przetwarzanie dla wszystkich funkcji systemu, włącznie z udostępnianiem aplikacji użytkowników i programów pełniących rolę interfejsów użytkowników, a także całą funkcjonalność systemu zarządzania bazą danych. Stosowanie takich rozwiązań wynikało z faktu, że większość użytkowników uzyskiwała dostęp do tych systemów za pośrednictwem komputerowych terminali, które nie oferowały wystarczającej mocy obliczeniowej, i które w związku z tym pełniły wyłącznie rolę sprzętu wyświetlającego otrzymywane dane. Wszystkie operacje związane z przetwarzaniem danych były wówczas zdalnie wykonywane w centralnym systemie komputerowym, a terminale odpowiadały jedynie za wyświetlanie otrzymanych z tego komputera informacji i instrukcji sterujących — terminale były połączone z centralnym komputerem za pośrednictwem rozmaitych typów sieci komunikacyjnych.

Wraz z postępującym spadkiem cen sprzętu komputerowego większość użytkowników zastąpiła swoje terminale komputerami osobistymi (ang. *Personal Computer — PC*) oraz stacjami roboczymi, a później także urządzeniami mobilnymi. Systemy baz danych początkowo wykorzystywały te komputery w taki sam sposób, w jaki wcześniej używały terminali wyświetlających, zatem nadal mieliśmy do czynienia ze **scentralizowanymi** systemami zarządzania bazami danych, w których obsługa wszystkich funkcji, wykonywanie programów aplikacji oraz przetwarzanie interfejsu użytkownika odbywało się na jednym komputerze. Na rysunku 2.4 przedstawiono fizyczne składniki architektury scentralizowanej. Systemy zarządzania bazami danych z czasem zaczęły wykorzystywać moce przetwarzania dostępne po stronie użytkownika, co doprowadziło ostatecznie do opracowania architektury systemów zarządzania bazami danych typu klient-serwer.

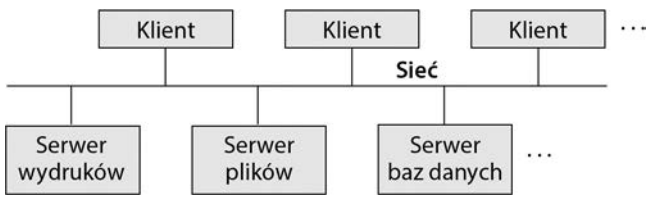
2.5.2. Podstawowe architektury typu klient-serwer

W pierwszej kolejności omówimy ogólną architekturę typu klient-serwer, dopiero potem przystąpimy do analizy jej zastosowań w systemach zarządzania bazami danych. **Architektura typu klient-serwer** została opracowana z myślą o środowiskach komputerowych z dużą liczbą komputerów osobistych, stacji roboczych, serwerów plików, drukarek, serwerów baz danych, serwerów WWW, serwerów poczty elektronicznej i innego sprzętu połączonego ze sobą za pomocą sieci komputerowej. Zasadniczym elementem tej koncepcji jest definiowanie **wyspecjalizowanych serwerów** realizujących tylko określone funkcje. Przykładowo, istnieje możliwość połączenia wielu komputerów osobistych lub niewielkich stacji roboczych w roli klientów **serwera plików**, który utrzymuje pliki wykorzystywane na komputerach klienckich. W tym samym środowisku możemy przydzielić innemu komputerowi rolę **serwera wydruków** i połączyć go z różnymi drukarkami — wszystkie generowane przez klientów żądania drukowania będą wówczas przekazywane właśnie do

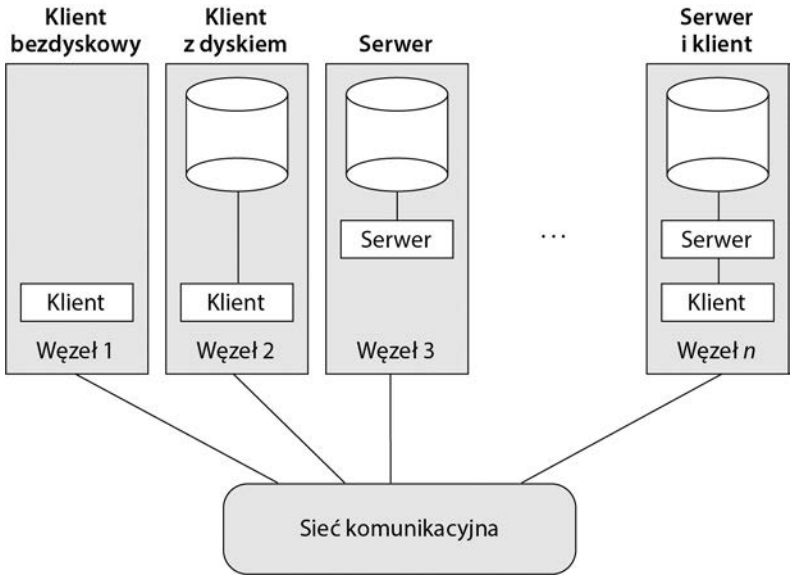


RYSUNEK 2.4. Fizycznie scentralizowana architektura

tego komputera. Także **serwery WWW** lub **serwery poczty elektronicznej** należą do kategorii serwerów wyspecjalizowanych. W ten sposób zasoby oferowane przez wyspecjalizowane serwery mogą być jednocześnie udostępniane wielu **komputerom klienckim**. Takie komputery oferują swoim użytkownikom nie tylko odpowiednie interfejsy umożliwiające wygodne i efektywne wykorzystywanie funkcjonalności wyspecjalizowanych serwerów, ale także lokalną moc przetwarzania umożliwiającą pracę z lokalnymi aplikacjami. Koncepcję stosowania wyspecjalizowanych serwerów można przenieść także na grunt samego oprogramowania, gdzie wyspecjalizowane pakiety — takie jak oprogramowanie typu *CAD* (ang. *Computer-Aided Design*) — może się znajdować na określonych serwerach i być udostępniane jednocześnie wielu klientom. Na rysunku 2.5 przedstawiono architekturę typu klient-serwer na poziomie logicznym, natomiast na rysunku 2.6 zademonstrowano uproszczony diagram pokazujący, jak wyglądałaby fizyczna architektura takiego środowiska. Niektóre komputery pełniłyby w takim systemie jedynie rolę urządzeń klienckich (przykładowo, może to dotyczyć urządzeń mobilnych albo też komputerów osobistych lub stacji roboczych z zainstalowanym wyłącznie oprogramowaniem klienckim). Inne komputery w tak skonstruowanym środowisku byłyby dedykowanymi serwerami, a jeszcze inne oferowałyby zarówno funkcjonalność klienta, jak i serwera.



RYSUNEK 2.5. Logiczna dwuwarstwowa architektura typu klient-serwer



RYSUNEK 2.6. Fizyczna dwuwarstwowa architektura typu klient-serwer

Koncepcja architektury typu klient-serwer przewiduje istnienie całej infrastruktury sieciowej obejmującej wiele komputerów osobistych, stacji roboczych i urządzeń przenośnych oraz znacznie mniej liczną grupę serwerów połączonych ze sobą za pomocą sieci bezprzewodowej, lokalnej lub innego typu. **Klient** w takiej infrastrukturze ma zwykle postać takiego komputera użytkownika, który oferuje pewne możliwości zarówno w zakresie obsługi interfejsu użytkownika, jak i w kwestii lokalnego przetwarzania. Kiedy klient zażąda dostępu do dodatkowych funkcji lub zasobów (np. do bazy), które nie są utrzymywane na jego lokalnym komputerze, komputer kliencki nawiąże połączenie z odpowiednim serwerem oferującym środki niezbędne do zrealizowania tego żądania. **Serwer** jest systemem obejmującym sprzęt i oprogramowanie, które pozwalają udostępniać komputerom klienckim takie usługi jak dostęp do plików, drukowanie, archiwizowanie czy dostęp do bazy danych. Niektóre komputery mogą mieć zainstalowane wyłącznie oprogramowanie klienckie, inne wyłącznie oprogramowanie serwera, jeszcze inne mogą natomiast zawierać zarówno oprogramowanie klienta, jak i oprogramowanie serwera (patrz rysunek 2.6). Znacznie bardziej popularnym rozwiązaniem jest jednak wykorzystywanie oprogramowania klienta i serwera na różnych komputerach. Na podstawie architektury klient-serwer stworzono dwa główne typy architektur systemów zarządzania bazami danych: **dwuwarstwową** i **trójwarstwową**[11]. Omówimy je w kolejnych punktach.

[11] Istnieje jeszcze wiele innych odmian architektury klient-serwer. W tym miejscu omówimy tylko wspomnianą parę podstawowych architektur tego typu.

2.5.3. Dwuwarstwowe architektury typu klient-serwer dla systemów zarządzania bazami danych

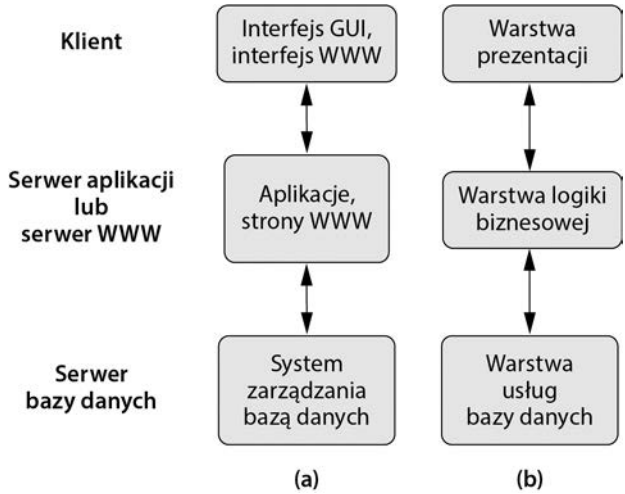
W relacyjnych systemach zarządzania bazami danych, których większość była początkowo systemami scentralizowanymi, do warstwy klienta w pierwszej kolejności przeniesiono takie elementy jak interfejs użytkownika oraz aplikacje. Ponieważ SQL (patrz rozdziały 6. i 7.) z czasem zyskał pozycję standardowego języka relacyjnych systemów zarządzania bazami danych, takie posunięcie stworzyło logiczny punkt podziału pomiędzy klientem a serwerem. Oznacza to, że obsługa zapytań i transakcji pozostały po stronie serwera. W związku z tym w rozwiązaniach zbudowanych zgodnie z tą architekturą serwer jest często nazywany **serwerem zapytań** lub **serwerem transakcji**. W relacyjnych systemach zarządzania bazami danych taki serwer jest równie często nazywany **serwerem SQL**.

W takich architekturach programy interfejsu użytkownika i aplikacje mogą działać po stronie klienta. Kiedy okaże się, że niezbędny jest dostęp do systemu zarządzania bazą danych, odpowiedni program nawiąże połączenie z takim systemem (działającym po stronie serwera); kiedy odpowiednie połączenie zostanie utworzone, program klienta będzie mógł się komunikować z systemem zarządzania bazą danych. Standard nazywany **otwartym łączem baz danych** (ang. *Open Database Connectivity — ODBC*) oferuje **interfejs programowy aplikacji** (ang. *Application Programming Interface — API*), który umożliwia programom działającym po stronie klienta wywoływanie systemu zarządzania bazą danych — jedynym wymaganiem jest zainstalowanie na komputerach klienta i serwera odpowiedniego oprogramowania. Większość producentów systemów zarządzania bazami danych oferuje w swoich pakietach specjalne sterowniki ODBC. Oznacza to, że pojedynczy program klienta może się łączyć z wieloma relacyjnymi systemami zarządzania bazami danych i wysyłać — za pośrednictwem interfejsu API standardu ODBC — zapytania i żądania transakcji, które będą następnie przetwarzane po stronie serwerów baz danych. Uzyskane w ten sposób wyniki zapytań zostaną odesłane do programu klienta, który może je dodatkowo przetworzyć lub od razu zaprezentować użytkownikowi. Istnieje także powiązany standard (nazwany **JDBC**) dla języka programowania Java. Zastosowanie takiego rozwiązania umożliwia dostęp do systemów zarządzania bazami danych z poziomu programu klienta napisanego w języku Java za pośrednictwem standardowego interfejsu.

Opisane w tym punkcie architektury są nazywane **architekturami dwuwarstwowymi**, ponieważ składniki oprogramowania są dzielone pomiędzy dwa systemy: klienta i serwera. Zaletą tego typu architektur jest ich prostota i bezproblemowa zgodność z istniejącymi systemami informatycznymi. Rozwój internetu (w szczególności popularność stron WWW) odmienił nieco role klientów i serwera, doprowadzając ostatecznie do powstania architektury trójwarstwowej.

2.5.4. Trójwarstwowe i n-warstwowe architektury typu klient-serwer dla aplikacji internetowych

Wiele aplikacji internetowych wykorzystuje model nazywany **architekturą trójwarstwową**, która dodaje pomiędzy klientem a serwerem bazy danych jeszcze jedną warstwę pośredniczącą (patrz rysunek 2.7(a)).



RYSUNEK 2.7. Logiczna trójwarstwowa architektura typu klient-serwer i wybrane standardowe pojęcia

Taka **środkowa** warstwa pośrednicząca jest niekiedy nazywana **warstwą aplikacji** lub **serwerem WWW**, w zależności od jej rzeczywistego zastosowania. Taki serwer pełni rolę pośrednika przechowującego reguły biznesowe (procedury lub ograniczenia), które są wykorzystywane do udostępniania danych z serwera bazy danych. Dodatkowa warstwa może także zwiększyć bezpieczeństwo bazy danych przez sprawdzanie danych uwierzytelniających klienta jeszcze przed przekazaniem otrzymanych żądań do serwera bazy danych. Komputery warstwy klienta zawierają interfejsy użytkownika i przeglądarki internetowe. Serwer pośredniczący przyjmuje i przetwarza żądania nadesłane przez klienta, po czym przesyła je dalej do serwera bazy danych (w postaci zapytań i poleceń). Następnie przekazuje (częściowo) przetworzone dane z serwera bazy danych do klientów, gdzie dane te mogą być dalej przetwarzane i filtrowane w celu ich zaprezentowania użytkownikom. Zatem rolę trzech warstw pełnią: *interfejs użytkownika*, *reguły aplikacji* oraz *dostęp do danych*. Na rysunku 2.7(b) pokazane jest inne ujęcie architektury trójwarstwowej używanej przez producentów baz danych i innych pakietów aplikacji. Warstwa prezentacji wyświetla tu informacje użytkownikowi i umożliwia wprowadzanie danych. Warstwa logiki biznesowej obsługuje reguły i ograniczenia pośrednie, stosowane przed przekazaniem danych w górę (do użytkownika) lub w dół (do SZBD). Dolna warstwa obejmuje wszystkie usługi zarządzania danymi. Warstwa pośrednia może też pełnić funkcję serwera WWW, który pobiera wyniki zapytań z serwera bazy danych i formatuje je do postaci dynamicznych stron WWW wyświetlanych za pomocą przeglądarki po stronie klienta. Maszyną klienta jest zwykle komputer osobisty lub urządzenie mobilne podłączone do internetu.

Zaproponowano także inne architektury. Warstwy pomiędzy użytkownikiem a składowanymi danymi można podzielić na bardziej szczegółowe komponenty, co skutkuje powstaniem architektur *n*-warstwowych (gdzie *n* to cztery lub pięć). Warstwa logiki biznesowej jest zwykle podzielona na wiele warstw. Oprócz możliwości udostępniania kodu i danych w sieci aplikacje *n*-warstwowe mają tę zaletę, że każdą warstwę można obsługiwać niezależnie oraz zastosować dla niej odpowiedni procesor lub system opera-

cyjny. Producenci pakietów ERP (ang. *enterprise resource planning*) i CRM (ang. *customer relationship management*) często stosują *warstwę pośrednią*, aby uwzględnić moduły frontonu (klientów) komunikujące się z licznymi bazami zaplecza (serwerami).

Postęp w obszarze technologii szyfrowania i deszyfrowania sprawił, że przesyłanie w postaci zaszyfrowanej poufnych danych z serwera do klienta (gdzie zostaną one odszyfrowane) jest teraz znacznie bezpieczniejsze. Operacja deszyfrowania może się opierać na odpowiednim sprzęcie lub na zaawansowanym oprogramowaniu. Wspomniana technologia zapewnia wyższy poziom bezpieczeństwa danych, problemem pozostaje jednak bezpieczeństwo komunikacji w sieci. Warto też pamiętać, że podczas przesyłania ogromnych ilości danych z serwerów do klientów za pośrednictwem sieci przewodowych lub bezprzewodowych pomocne są rozmaite technologie kompresji danych.

2.6. Klasyfikacja systemów zarządzania bazami danych

Podczas klasyfikowania systemów zarządzania bazami danych bierze się zwykle pod uwagę wiele kryteriów. Pierwszym jest **model danych**, na którym opiera się dany SZBD. Najczęściej spotykanym modelem tego typu wykorzystywanym w wielu współczesnych komercyjnych systemach zarządzania bazami danych jest **relacyjny model danych**. Oparte na nim systemy są nazywane **systemami SQL-owymi**. **Obiektowy model danych** co prawda został już zaimplementowany w kilku komercyjnych systemach, jednak nie jest na razie popularny. W nowszych **systemach typu big data** (inne nazwy to **systemy składowania danych typu klucz-wartość** lub **systemy NOSQL**) używane są różne modele danych: **oparte na dokumentach**, **oparte na grafach**, **oparte na kolumnach** i **typu klucz-wartość**. Wiele starszych rozwiązań nadal opiera się na **hierarchicznych** lub **sieciowych modelach danych**.

Relacyjne systemy zarządzania bazami danych stale ewoluują i są wzbogacane o nowe rozwiązania — dotyczy to zwłaszcza wielu cech charakterystycznych dla obiektowych baz danych. Taki kierunek rozwoju relacyjnych baz danych zaowocował powstaniem nowej klasy systemów zarządzania bazami danych, nazwanych **obiektowo-relacyjnymi** SZBD. Systemy zarządzania bazami danych możemy więc podzielić na kategorie według zastosowanego modelu danych: relacyjnego, obiektowego, obiektowo-relacyjnego, NOSQL, klucz-wartość, hierarchicznego, sieciowego itp.

Niektóre eksperymentalne SZBD są oparte na modelu XML. Jest to **drzewiasty model danych**. Są to tzw. **natywne XML-owe SZBD**. W kilku komercyjnych relacyjnych SZBD dodano XML-owe interfejsy i magazyny danych.

Drugim istotnym kryterium podziału systemów zarządzania bazami danych jest obsługiwana przez nie **liczba użytkowników**. Systemy **jednodostępne** obsługują w tym samym czasie tylko jednego użytkownika i są wykorzystywane przede wszystkim na komputerach osobistych. **Systemy wielodostępne**, do których należy większość współczesnych systemów zarządzania bazami danych, umożliwiają jednoczesną pracę wielu użytkownikom.

Trzecim kryterium jest **liczba węzłów**, pomiędzy które baza danych jest rozpraszana. System zarządzania bazą danych jest **scentralizowany**, jeśli dane są przechowywane na jednym komputerze. Scentralizowany system zarządzania bazą danych może co prawda

obsługiwać wielu użytkowników pracujących przy wielu komputerach, jednak sam system tego typu wraz z udostępnianą bazą danych musi się znajdować na jednym komputerze. **Rozproszony** system zarządzania bazą danych (ang. *Distributed Database Management System — DDBMS*) może wykorzystywać bazę danych i moduły oprogramowania SZBD działające na wielu różnych węzłach połączonych za pomocą sieci. Systemy big data są często wysoce rozproszone i działają w setkach węzłów. Dane są nieraz replikowane w wielu węzłach, dlatego awaria jednego z nich nie powoduje niedostępności danych.

Homogeniczne rozproszone systemy zarządzania bazami danych wykorzystują to samo oprogramowanie na wielu komputerach. W systemach **heterogenicznych** w każdym węźle używane może być inne oprogramowanie. Można też opracować **oprogramowanie warstwy pośredniej**, aby uzyskać dostęp do wielu autonomicznych, już istniejących baz danych zarządzanych przez heterogeniczne systemy SZBD. Kolejnym krokiem w tym kierunku są **federacyjne** systemy zarządzania bazami danych (tzw. **systemy wielo-bazodanowe**), w których wykorzystuje się wiele luźno powiązanych ze sobą systemów zachowujących w całej tej strukturze część lokalnej autonomii. W wielu rozproszonych systemach zarządzania bazami danych stosuje się architekturę typu klient-serwer, co opisano w podrozdziale 2.5.

Czwartym kryterium jest **koszt** systemu zarządzania bazą danych. Trudno jest zaproponować klasyfikację SZBD opartą na kosztach. Obecnie dostępne są otwarte (bezpłatne) produkty tego typu, np. MySQL i PostgreSQL, wzbogacane przez niezależnych producentów o dodatkowe usługi. Popularne relacyjne SZBD są dostępne w postaci bezpłatnych 30-dniowych wersji próbnych i wersji osobistych, które mogą kosztować tylko kilkaset złotych i udostępniać wiele funkcji. Bardzo rozbudowane systemy są sprzedawane w postaci modułowej z komponentami do obsługi dystrybucji danych, replikacji, przetwarzania równoległego, mechanizmów mobilnych itd. Konfiguracja takich produktów wymaga zdefiniowania licznych parametrów. Ponadto omawiane systemy są sprzedawane w formie licencji. Licencje na lokalizacje umożliwiają nieograniczone użytkowanie systemu bazy danych i uruchamianie go w dowolnej liczbie egzemplarzy w siedzibie klienta. Inny rodzaj licencji ogranicza liczbę jednoczesnych użytkowników lub stanowisk w danej lokalizacji. Niezależne wersje dla pojedynczych użytkowników (np. systemu Microsoft Access) są sprzedawane na sztuki albo dodawane do konfiguracji komputera stacjonarnego lub laptopa. Za dodatkową opłatą udostępniane są funkcje hurtowni i drążenia danych, a także obsługa innych typów danych. Instalacja i utrzymanie dużego systemu bazy danych może kosztować miliony złotych rocznie.

Systemy zarządzania bazami danych możemy sklasyfikować także według dostępnych opcji w zakresie **typów ścieżek dostępu** do przechowywanych plików. Przykładowo, jedna z dobrze znanych rodzin systemów tego typu opiera się na odwróconych strukturach plików. Wreszcie, system zarządzania bazą danych może być rozwiązaniem **uniwersalnym** lub **wyspecjalizowanym**. W sytuacjach, gdy kluczowe wymaganie dotyczy wydajności systemu, warto rozważyć zaprojektowanie i budowę wyspecjalizowanego SZBD dla konkretnego zastosowania, który bez wprowadzenia istotnych zmian nie będzie mógł być wykorzystywany do innych zastosowań. Wiele opracowanych w przeszłości systemów zarządzania bazami danych wykorzystywanych w biurach rezerwacji biletów lotniczych lub przez operatorów telefonicznych było właśnie rozwiązaniami wyspecjalizowanymi. Tak powstały systemy **przetwarzania transakcji na bieżąco** (ang. *Online Transaction Processing — OLTP*), które muszą obsługiwać ogromną liczbę współbieżnych transakcji, nie powodując przy tym zbyt dużych opóźnień.

Spróbujmy teraz krótko omówić model danych — główne kryterium klasyfikacji systemów zarządzania bazami danych. Podstawowy **relacyjny model danych** reprezentuje bazę danych w postaci zbioru tabel, z których każda może być składowana w osobnym pliku. Baza danych zademonstrowana na rysunku 1.2 jest bardzo zbliżona do reprezentacji relacyjnej. Większość relacyjnych baz danych wykorzystuje wysokopoziomowy język zapytań SQL i obsługuje (choć w ograniczonym wymiarze) perspektywy użytkownika. Model relacyjny wraz z jego językami i operacjami techniki programowania relacyjnych aplikacji omówimy w rozdziałach od 5. do 8.; techniki programowania relacyjnych aplikacji omówimy w rozdziałach 10. i 11.

Obiektowy model danych definiuje bazę danych w oparciu o obiekty, ich własności oraz ich operacje. Obiekty z taką samą strukturą i zachowaniem należą do jednej **klasy**, natomiast same klasy są organizowane w **hierarchie** (nazywane także **grafami acyklicznymi**). Operacje dla poszczególnych klas są konstruowane na bazie predefiniowanych procedur nazywanych **metodami**. Modele wykorzystywane w wielu relacyjnych systemach zarządzania bazami danych zostały przystosowane do obsługi pojęć typowych dla obiektowych baz danych oraz funkcji innych systemów i są w związku z tym nazywane systemami **obiektowo-relacyjnymi** lub **rozszerzonymi systemami relacyjnymi**. Obiektowe bazy danych oraz systemy obiektowo-relacyjne omówimy w rozdziale 12.

Systemy big data są oparte na różnych modelach danych. Najczęściej stosowane są cztery z nich. **Model danych typu klucz-wartość** łączy z każdą wartością (może nią być rekord lub obiekt) unikatowy klucz i zapewnia bardzo szybki dostęp do wartości na podstawie klucza. **Model danych oparty na dokumentach** wykorzystuje format JSON (ang. *Java Script Object Notation*) i przechowuje dane jako dokumenty przypominające nieco złożone obiekty. **Model danych oparty na grafie** przechowuje obiekty jako węzły grafu, a relacje między obiektami jako krawędzie grafu skierowanego. **Kolumnowe modele danych** przechowują kolumny wierszy pogrupowane na stronach dysku, aby umożliwić szybki dostęp do danych i przechowywanie ich w wielu wersjach. Niektóre z tych modeli są opisane szczegółowo w rozdziale 24.

Model XML-owy powstał jako standard wymiany danych w internecie i był używany do zaimplementowania kilku prototypowych natywnych systemów XML-owych. W XML-u używane są hierarchiczne struktury drzewiaste. Ten model łączy mechanizmy baz danych z rozwiązaniami z modeli reprezentacji dokumentów. Dane są tu reprezentowane jako elementy. Za pomocą znaczników dane można zagnieżdżać i tworzyć w ten sposób złożone struktury drzewiaste. Ten model koncepcyjnie przypomina model obiektowy, choć stosowana jest tu inna terminologia. Mechanizmy XML-owe zostały dodane do wielu komercyjnych SZBD. Omówienie XML-a prezentujemy w rozdziale 13.

Dwoma znacznie starszymi, historycznie ważnymi modelami danych, uważanymi obecnie za **przestarzałe**, są modele sieciowy i hierarchiczny. **Model sieciowy** reprezentuje dane w postaci typów rekordów i umożliwia (choć w ograniczonym zakresie) reprezentowanie relacji typu 1:N, tzw. **typu zbiorowego**. Relacja 1:N (jeden do wielu) łączy jeden egzemplarz rekordu z wieloma innymi za pomocą wskaźników. Model sieciowy, znany także jako model CODASYL DBTG[12], jest ściśle związany z językiem typu *record-at-a-time*, któ-

[12] Skrót CODASYL DBTG pochodzi od nazwy komitetu Conference of Data Systems Languages Data Base Task Group, który zdefiniował model sieciowy i jego język.

rego instrukcje muszą być osadzane w kodzie nadrzędnego języka programowania. Sieciowy język DML został zaproponowany w pracy 1971 Database Task Group (DBTG) Report jako rozszerzenie języka COBOL.

Model hierarchiczny reprezentuje dane w postaci hierarchicznej struktury drzewiastej. Każda taka hierarchia reprezentuje wiele powiązanych ze sobą rekordów. Dla modelu hierarchicznego nie istnieje standardowy język. Popularnym hierarchicznym językiem DML jest DL/1 z systemu IMS. Był to najpopularniejszy SZBD w latach 1965 – 1985. Używany w nim język DML (DL/1) był przez długi czas w zasadzie standardem w branży[13].

2.7. Podsumowanie

W tym rozdziale wprowadziliśmy podstawowe terminy wykorzystywane w systemach baz danych. Zdefiniowaliśmy pojęcie modelu danych i wyróżniliśmy trzy główne kategorie modeli tego typu:

- wysokopoziomowe modele danych (nazywane też koncepcyjnymi), które opierają się na encjach i związkach;
- niskopoziomowe modele danych (nazywane też fizycznymi);
- reprezentacyjne modele danych (nazywane też implementacyjnymi), które opierają się na rekordach lub obiektach.

W rozdziale zdefiniowaliśmy pojęcie schematu (opisu) bazy danych, które oddzieliliśmy od samej bazy. Schemat bazy danych nie jest zbyt często modyfikowany, natomiast stan bazy danych zmienia się za każdym razem, gdy wstawimy, usuniemy lub zmodyfikujemy dane. Następnie opisaliśmy architekturę trójwarstwową systemów zarządzania bazami danych, która umożliwia stosowanie aż trzech poziomów schematów bazy danych:

- schemat wewnętrzny, który opisuje fizyczną strukturę przechowywania bazy danych;
- schemat koncepcyjny, który jest wysokopoziomowym opisem całej bazy danych;
- schemat zewnętrzny, który opisuje perspektywy różnych grup użytkowników.

System zarządzania bazą danych, który ściśle oddziela te trzy poziomy, musi zawierać reguły odwzorowywania pomiędzy poszczególnymi schematami, aby właściwie przekształcać żądania i wyniki przekazywane z jednego poziomu do drugiego. Większość systemów zarządzania bazami danych nie oddziela tych trzech poziomów całkowicie. Wykorzystaliśmy w tym rozdziale architekturę trójwarstwową głównie po to, aby zdefiniować pojęcia logicznej i fizycznej niezależności danych.

W dalszej części tego rozdziału omówiliśmy najważniejsze typy języków i interfejsów obsługiwanych przez systemy zarządzania bazami danych. Język definicji danych (DDL) jest wykorzystywany do definiowania schematu koncepcyjnego bazy danych. W większości systemów tego typu język DDL jest ponadto wykorzystywany do definiowania per-

[13] Kompletne rozdziały na temat modeli sieciowych i hierarchicznych z drugiego wydania tej książki są dostępne w witrynie tej pozycji: *http://www.aw.com/elmasri*.

spektyw użytkownika, a niekiedy także struktur składowania; w innych systemach do określania struktur składowania danych mogą być używane odrębne języki lub funkcje. We współczesnych implementacjach systemów relacyjnych różnice między opisanymi funkcjami zanikają, ponieważ SQL działa jak uniwersalny język do wykonywania wielu zadań, w tym do definiowania perspektyw. We wczesnych wersjach SQL-a znajdował się język definiowania struktur składowania (język SDL), jednak obecnie w relacyjnych SZBD zwykle implementuje się go w formie specjalnych poleceń dla administratorów baz danych. System SZBD kompiluje wszystkie definicje schematów i umieszcza ich opisy w swoim katalogu.

Do określania operacji odczytywania i aktualizowania danych wykorzystuje się język manipulowania danymi (DML). Języki tego typu mogą być wysokopoziomowe (*set-oriented*, lub nieproceduralne) lub niskopoziomowe (*record-oriented*, lub proceduralne). Wysokopoziomowe języki DML można albo osadzać w kodzie nadrzędnego języka programowania, albo wykorzystywać w roli języków autonomicznych; w tym drugim przypadku języki tego typu są często nazywane językami zapytań.

Omówiliśmy także różne typy interfejsów udostępnianych przez systemy zarządzania bazami danych oraz rozmaite kategorie użytkowników systemów tego typu, dla których tworzone są poszczególne interfejsy. Dalej opisaliśmy środowisko systemu zarządzania bazą danych, typowe moduły programowe takiego systemu oraz narzędzia ułatwiające użytkownikom i administratorom wykonywanie ich zadań. W dalszej części tego rozdziału zaprezentowano przegląd architektur dwu- i trójwarstwowych używanych w aplikacjach bazodanowych.

W ostatnich fragmentach sklasyfikowaliśmy systemy zarządzania bazami danych według kilku kryteriów: modelu danych, liczby użytkowników, liczby węzłów, typów ścieżek dostępu oraz kosztów. Omówiliśmy dostępność SZBD i dodatkowych modułów — od rozwiązań darmowych w postaci oprogramowania otwartego po konfiguracje, których utrzymanie kosztuje miliony złotych rocznie. Wspomnieliśmy też o różnorodnych licencjach na SZBD i powiązane produkty. Najważniejszym kryterium podziału systemów zarządzania bazami danych jest oczywiście stosowany model danych. W tym rozdziale omówiliśmy krótko główne modele danych wykorzystywane we współczesnych komercyjnych systemach tego typu.

Pytania powtórkowe

2.1. Zdefiniuj następujące pojęcia: *model danych*, *schemat bazy danych*, *stan bazy danych*, *schemat wewnętrzny*, *schemat koncepcyjny*, *schemat zewnętrzny*, *niezależność danych, język DDL*, *język DML*, *język SDL*, *język VDL*, *język zapytań*, *język nadrzędny*, *podjęzyk danych*, *narzędzie bazy danych*, *katalog*, *architektura typu klient-serwer*, *architektura trójwarstwowa* i *architektura* n-*warstwowa*.

2.2. Przedstaw główne kategorie modelów danych. Jakie są podstawowe różnice między modelami relacyjnym, obiektowym a XML-owym?

2.3. Jaka jest różnica pomiędzy schematem bazy danych a stanem bazy danych?

2.4. Opisz architekturę trójwarstwową. Wyjaśnij, dlaczego niezbędne jest stosowanie reguł odwzorowywania pomiędzy poziomami schematu. W jaki sposób różne języki definiowania schematu wspomagają stosowanie tej architektury?

2.5. Jaka jest różnica pomiędzy logiczną niezależnością danych a fizyczną niezależnością danych? Którą z nich trudniej jest osiągnąć? Dlaczego?

2.6. Jaka jest różnica pomiędzy proceduralnymi a nieproceduralnymi językami DML?

2.7. Przedstaw różne typy przyjaznych dla użytkownika interfejsów oraz kategorie użytkowników, które zazwyczaj korzystają z poszczególnych typów.

2.8. Z jakimi innymi pakietami oprogramowania komputerowego współpracują systemy zarządzania bazami danych?

2.9. Jaka jest różnica pomiędzy dwu- i trójwarstwowymi architekturami typu klient-serwer?

2.10. Przedstaw niektóre typy przydatnych programów i narzędzi baz danych wraz z oferowanymi przez nie funkcjami.

2.11. Jakie dodatkowe funkcje obejmuje architektura *n*-warstwowa ($n > 3$)?

Ćwiczenia

2.12. Omów rodzaje potencjalnych użytkowników bazy danych z rysunku 1.2. Jakich typów aplikacji będą potrzebowali poszczególni użytkownicy? Do której z kategorii użytkowników należałoby ich przypisać i jakich typów interfejsów będą ci użytkownicy potrzebowali?

2.13. Wybierz znane Ci zastosowanie bazy danych. Zaprojektuj odpowiedni schemat i przedstaw przykładową bazę danych dla tego zastosowania (zgodnie z notacją użytą na rysunkach 2.1 i 1.2). Jakiego rodzaju dodatkowych informacji i ograniczeń chciałbyś użyć do reprezentowania tego schematu? Przemyśl role wielu różnych użytkowników swojej bazy danych i dla każdego z nich zaprojektuj osobną perspektywę.

2.14. Gdybyś miał zaprojektować internetowy system rezerwacji biletów lotniczych, którą architekturę SZBD z podrozdziału 2.5 byś wybrał? Dlaczego? Z jakiego powodu inne architektury nie będą dobrym wyborem?

2.15. Przyjrzyj się rysunkowi 2.1. Obok ograniczeń wiążących wartości kolumn z jednej tabeli z kolumnami innej tabeli występują też ograniczenia dotyczące wartości kolumn (pojedynczych lub ich kombinacji) tabeli. Jedno z takich ograniczeń powoduje, że kolumna lub grupa kolumn musi mieć unikatowe wartości w skali wszystkich wierszy tabeli. Na przykład w tabeli `STUDENT` unikatowa musi być wartość kolumny `NumerIndeksu` (aby zapobiec posiadaniu identycznego numeru przez dwóch studentów). Wskaż w innych tabelach kolumny lub grupy kolumn, które muszą być unikatowe w skali wszystkich wierszy danej tabeli.

Wybrane publikacje

Wiele książek poświęconych bazom danych, włącznie z publikacjami Date'a (2004), Ramakrishnana i Gehrke'a (2003), Garcia-Moliny i in. (2002, 2009) oraz Abiteboula i in. (1995), zawiera szczegółowe omówienie rozmaitych zaprezentowanych w tym rozdziale zagadnień związanych z bazami danych. Książka autorstwa Tsichritzisa i Lochovsky'ego (1982)

była jedną z pierwszych publikacji na temat modelów danych. Zarówno książka napisana przez Tsichritzisa i Kluga (1978), jak i książka autorstwa Jardine'a (1977) prezentowały architekturę trójwarstwową, której koncepcję zaproponowano po raz pierwszy w raporcie komisji DBTG CODASYL (1971) i później w raporcie Amerykańskiego Instytutu Normalizacyjnego — ANSI (1975). Dogłębną analizę relacyjnego modelu danych i niektórych z jego potencjalnych rozszerzeń można znaleźć w książce Codda (1990). Cattell i in. (2000) zawarli w swojej książce propozycję standardu dla obiektowych baz danych. Wiele materiałów opisujących język XML jest dostępnych w internecie, np. XML (2005).

Przykładami programów narzędziowych wykorzystywanych z bazami danych są: narzędzia Connect, Analyze i Transform firmy ETI (*http://www.eti.com*) oraz przeznaczone dla administratorów baz danych narzędzie DB Artisan firmy Embarcadero Technologies (*http://www.embarcadero.com*).

Baza danych? Stosuj tylko najskuteczniejsze rozwiązania!

Mijają lata, a bazy danych wciąż stanowią serce większości systemów informatycznych. Rozwój technologii jednak sprawia, że zaprojektowanie systemu baz danych, jego wdrożenie i administrowanie nim wymaga biegłości w wielu dziedzinach. Niezbędne są solidne podstawy modelowania i projektowania baz danych, umiejętność posługiwania się językami i modelami udostępnianymi przez systemy zarządzania bazami danych, a także znajomość technik implementacji samych systemów. Od profesjonalisty wymaga się również wiedzy o najnowszych technologiach, takich jak NoSQL i oczywiście big data. Ważnym uzupełnieniem tej szerokiej gamy jest też znajomość technologii powiązanych z systemami bazodanowymi.

Ta książka jest siódmym, zaktualizowanym wydaniem klasycznego podręcznika do nauki baz danych. Jest to szczegółowa prezentacja najważniejszych aspektów systemów i aplikacji bazodanowych oraz powiązanych technologii. To znakomity podręcznik dla studentów i świetne kompendium dla praktyków. Sporo miejsca poświęcono w nim systemom rozproszonym oraz technologiom opartym na systemie Hadoop i modelu MapReduce. Nie zabrakło opisu takich zagadnień jak model IR, wyszukiwanie z użyciem słów kluczowych, porównanie baz danych z modelem IR, modele wyszukiwania, ocena wyszukiwania i algorytmy rankingowe. Wykładowcom przyda się szereg ułatwiających pracę dydaktyczną diagramów, prezentacji i rysunków.

W książce między innymi:

- wprowadzenie do modeli, systemów i języków z obszaru baz danych
- model związków encji i programowanie baz danych
- bazy relacyjne, obiektowo-relacyjne, obiektowe i XML w bazach danych
- algorytmy przetwarzania zapytań i techniki optymalizacji
- bezpieczeństwo baz danych

Ramez Elmasri jest profesorem na Uniwersytecie Teksańskim. Posiada znaczący dorobek naukowy w dziedzinie baz danych, a jego książki są standardowymi podręcznikami na wielu uczelniach świata. Wypromował kilkunastu doktorów i ponad dwustu magistrów.

Shamkant Navathe jest profesorem w Instytucie Technologii w Georgii. Zajęcia dydaktyczne z wiedzy o bazach danych prowadzi od 1975 roku. Interesuje się również bioinformatyką. Ponadto jest niezależnym konsultantem, doradzającym wielu znaczącym przedsiębiorstwom.

ISBN 978-83-283-4695-6

PEARSON
ALWAYS LEARNING