

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Wprowadzenie do systemów baz danych

Autor: Ramez Elmasri, Shamkant B. Navathe

Tłumaczenie: Mikołaj Szczepaniak (rozdz. 0 – 9, 27 – 29),

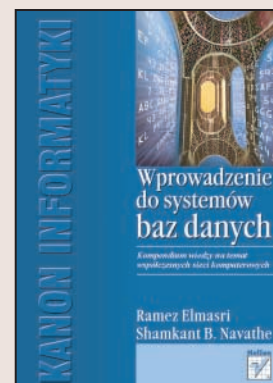
Bartłomiej Garbacz (rozdz. 10 – 19, dod. A – C),

Bartłomiej Moczulski (rozdz. 20 – 26)

ISBN: 83-7361-716-7

Tytuł oryginału: [Fundamentals of Database Systems, 4th Edition](#)

Format: B5, stron: 1056



Bazy danych to podstawa większości złożonych systemów informatycznych.

W oparciu o dane czerpane z tabel w bazie działają portale i sklepy internetowe, aplikacje biznesowe i informacyjne, a nawet multimedialne witryny, coraz częściej spotykane w urzędach, muzeach i innych budynkach użyteczności publicznej.

Na rynku dostępnych jest wiele systemów zarządzania bazami danych, oferowanych przez różnych producentów i na różnych zasadach licencjonowania. Pomimo istotnych różnic, wszystkie opierają się na podobnych założeniach, a projektowanie wydajnych baz danych odbywa się w niemal identyczny sposób, niezależnie od docelowego systemu zarządzania nimi. Opanowanie wiadomości leżących u podstaw projektowania i wykorzystywania baz danych jest więc niezbędne do stworzenia efektywnego i bezpiecznego zaplecza bazodanowego dla systemu informatycznego.

Książka „Wprowadzenie do systemów baz danych” to szczegółowe omówienie wszystkich aspektów projektowania i stosowania baz danych. Szczególny nacisk położono w niej na podstawy modelowania danych i definiowania tabel. Książka może pełnić rolę podręcznika pomocnego przy poznawaniu zagadnień związanych z bazami danych lub źródła informacji dla projektantów i administratorów systemów bazodanowych.

- Rozwiązania oparte na bazach danych
- Użytkownicy baz danych
- Architektury systemów zarządzania bazami danych
- Modelowanie danych oparte na związkach encji
- Zastosowanie języka UML w modelowaniu danych
- Relacyjny model danych
- Język SQL-99
- Normalizacja danych
- Składowanie danych na dysku
- Indeksy i klucze
- Algorytmy przetwarzania zapytań
- Mechanizmy transakcyjne
- Obiektowe bazy danych
- Bezpieczeństwo danych
- Język XML w bazach danych
- Technologie eksploracji danych
- Hurtownie danych, systemy GIS i bazy danych dla urządzeń mobilnych

Książka stanowi źródło wiedzy dla projektantów baz danych i oprogramowania bazodanowego.



Spis treści

| | |
|-----------------|----|
| Przedmowa | 13 |
|-----------------|----|

I

| | |
|--|----|
| WPROWADZENIE I MODELOWANIE KONCEPCYJNE | 19 |
|--|----|

1.

| | |
|--|----|
| Bazy danych i ich użytkownicy | 21 |
| 1.1. Wprowadzenie | 22 |
| 1.2. Przykład | 23 |
| 1.3. Właściwości rozwiązań opartych na bazach danych | 26 |
| 1.4. Aktorzy na scenie | 31 |
| 1.5. Pracownicy poza sceną | 33 |
| 1.6. Zalety stosowania rozwiązań opartych na systemach zarządzania bazami danych | 34 |
| 1.7. Krótka historia praktycznych zastosowań baz danych | 40 |
| 1.8. Kiedy nie należy używać systemów zarządzania bazami danych | 43 |
| 1.9. Podsumowanie | 44 |

2.

| | |
|---|----|
| Architektura systemów baz danych i związane z nimi pojęcia | 47 |
| 2.1. Modele danych, schematy i egzemplarze | 48 |
| 2.2. Trójwarstwowa architektura i niezależność danych | 51 |
| 2.3. Języki i interfejsy baz danych | 54 |
| 2.4. Środowisko systemu bazy danych | 57 |
| 2.5. Architektury systemów zarządzania bazami danych scentralizowane i typu klient-serwer | 61 |
| 2.6. Klasyfikacja systemów zarządzania bazami danych | 66 |
| 2.7. Podsumowanie | 68 |

3.

| | |
|--|----|
| Modelowanie danych zgodnie z modelem związków encji | 71 |
| 3.1. Stosowanie wysokopoziomowych, koncepcyjnych modeli danych podczas projektowania bazy danych | 72 |
| 3.2. Przykładowa aplikacja bazy danych | 74 |
| 3.3. Typy encji, zbiory encji, atrybuty i klucze | 75 |
| 3.4. Typy związków, zbiory związków, role i ograniczenia strukturalne | 82 |
| 3.5. Słabe typy encji | 89 |
| 3.6. Udoskonalanie projektu ER dla bazy danych FIRMA | 90 |

| | |
|--|----|
| 3.7. Diagramy ER, konwencje nazewnictwa oraz zagadnienia związane z projektowaniem | 91 |
| 3.8. Notacja dla diagramów klas UML | 95 |
| 3.9. Podsumowanie | 97 |

4.

| | |
|--|-----|
| Rozszerzony model związków encji oraz modelowanie UML | 105 |
| 4.1. Podklasy, nadklasy i dziedziczenie | 106 |
| 4.2. Specjalizacja i generalizacja | 108 |
| 4.3. Ograniczenia i właściwości związków specjalizacji i generalizacji | 111 |
| 4.4. Modelowanie typów UNII w oparciu o kategorie | 118 |
| 4.5. Przykład schematu EER dla bazy danych UNIWERSYTET oraz formalne definicje dla modelu EER | 120 |
| 4.6. Reprezentowanie specjalizacji-generalizacji oraz dziedziczenia na diagramach klas metodologii UML | 124 |
| 4.7. Typy związków stopnia wyższego niż drugi | 125 |
| 4.8. Abstrakcja danych, reprezentacja wiedzy oraz zagadnienia związane z ontologią | 129 |
| 4.9. Podsumowanie | 135 |

II

| | |
|--|-----|
| MODEL RELACYJNY: ELEMENTY SKŁADOWE, OGRANICZENIA, JĘZYKI, PROJEKTY I PROGRAMOWANIE | 143 |
|--|-----|

5.

| | |
|--|-----|
| Relacyjny model danych i ograniczenia relacyjnych baz danych | 145 |
| 5.1. Podstawowe pojęcia relacyjnego modelu danych | 146 |
| 5.2. Ograniczenia modelu relacyjnego i schematy relacyjnych baz danych | 152 |
| 5.3. Operacje aktualizacji i obsługa naruszeń więzów integralności | 160 |
| 5.4. Podsumowanie | 163 |

6.

| | |
|--|-----|
| Algebra relacyjna i rachunek relacji | 169 |
| 6.1. Relacyjne operacje unarne: selekcja i projekcja | 170 |
| 6.2. Operacje algebry relacyjnej pochodzące z teorii zbiorów | 175 |
| 6.3. Binarne operacje na relacjach: złączenie i dzielenie | 179 |
| 6.4. Dodatkowe operacje relacyjne | 186 |
| 6.5. Przykłady zapytań w algebrze relacyjnej | 192 |
| 6.6. Relacyjny rachunek krotek | 194 |
| 6.7. Relacyjny rachunek dziedzin | 203 |
| 6.8. Podsumowanie | 206 |

7.

| | |
|--|-----|
| Projektowanie relacyjnych baz danych przez odwzorowywanie modelu ER i EER w model relacyjny | 213 |
| 7.1. Projektowanie relacyjnych baz danych w oparciu o odwzorowywanie modelu ER w model relacyjny | 213 |
| 7.2. Odwzorowania konstrukcji modelu EER w relacje | 221 |
| 7.3. Podsumowanie | 225 |

8.

| | |
|--|-----|
| SQL-99: Definicja schematu, podstawowe ograniczenia oraz zapytania | 229 |
| 8.1. Definicje danych i typy danych języka SQL | 231 |
| 8.2. Określanie podstawowych ograniczeń w języku SQL | 236 |
| 8.3. Dostępne w języku SQL polecenia zmiany schematu | 240 |
| 8.4. Podstawowe zapytania języka SQL | 242 |
| 8.5. Bardziej skomplikowane zapytania języka SQL | 253 |
| 8.6. Dostępne w języku SQL polecenia INSERT, DELETE i UPDATE | 270 |
| 8.7. Dodatkowe własności języka SQL | 273 |
| 8.8. Podsumowanie | 274 |

9.

| | |
|---|-----|
| Więcej o języku SQL: asercje, perspektywy i techniki programowania | 279 |
| 9.1. Definiowanie ogólnych ograniczeń w postaci asercji | 280 |
| 9.2. Perspektywy (tabele wirtualne) w języku SQL | 281 |
| 9.3. Programowanie baz danych: najważniejsze zagadnienia i stosowane techniki | 286 |
| 9.4. Osadzony język SQL, dynamiczny język SQL oraz język SQLJ | 289 |
| 9.5. Programowanie baz danych z wywołaniami funkcji: SQL/CLI oraz JDBC | 301 |
| 9.6. Procedury składowane w bazie danych i technika SQL/PSM | 311 |
| 9.7. Podsumowanie | 314 |

III

| | |
|---|-----|
| TEORIA I METODOLOGIA PROJEKTOWANIA BAZ DANYCH | 317 |
|---|-----|

10.

| | |
|---|-----|
| Zależności funkcyjne i normalizacja w relacyjnych bazach danych | 319 |
| 10.1. Nieformalne wskazówki dotyczące projektowania schematów relacji | 321 |
| 10.2. Zależności funkcyjne | 329 |
| 10.3. Postaci normalne oparte na kluczach głównych | 337 |
| 10.4. Definicje ogólne drugiej i trzeciej postaci normalnej | 345 |
| 10.5. Postać normalna Boyce'a-Codda | 349 |
| 10.6. Podsumowanie | 351 |

11.

| | |
|---|-----|
| Algorytmy projektowania relacyjnych baz danych i dodatkowe zależności | 357 |
| 11.1. Właściwości dekompozycji relacyjnych | 358 |
| 11.2. Algorytmy projektowania schematów relacyjnych baz danych | 364 |
| 11.3. Zależności wielowartościowe i czwarta postać normalna | 370 |
| 11.4. Zależności złączeniowe i piąta postać normalna | 376 |
| 11.5. Zależności zawierania | 378 |
| 11.6. Inne zależności i postaci normalne | 379 |
| 11.7. Podsumowanie | 381 |

12.

| | |
|---|-----|
| Praktyczna metodologia projektowania baz danych i użycie diagramów UML | 385 |
| 12.1. Rola systemów informacyjnych w przedsiębiorstwach | 386 |
| 12.2. Projekt bazy danych i proces jej implementacji | 390 |
| 12.3. Użycie diagramów języka UML jako środka wspomagającego tworzenie specyfikacji | 408 |
| 12.4. Rational Rose — narzędzie projektowe oparte na języku UML | 417 |
| 12.5. Narzędzia zautomatyzowanego projektowania baz danych | 421 |
| 12.6. Podsumowanie | 425 |

IV

| | |
|---|-----|
| PRZECHOWYWANIE DANYCH, INDEKSOWANIE, PRZETWARZANIE ZAPYTAŃ ORAZ PROJEKTOWANIE FIZYCZNE | 429 |
|---|-----|

13.

| | |
|---|-----|
| Składowanie danych na dysku, podstawowe struktury plikowe i funkcje mieszające | 431 |
| 13.1. Wprowadzenie | 432 |
| 13.2. Drugorzędne urządzenia pamięciowe | 435 |
| 13.3. Buforowanie bloków | 441 |
| 13.4. Rozmieszczanie rekordów plików na dysku | 442 |
| 13.5. Operacje wykonywane na plikach | 446 |
| 13.6. Pliki nieuporządkowanych rekordów (pliki stertowe) | 449 |
| 13.7. Pliki uporządkowanych rekordów (pliki posortowane) | 450 |
| 13.8. Techniki mieszania | 453 |
| 13.9. Inne podstawowe metody organizacji plików | 461 |
| 13.10. Zapewnianie równoległego dostępu do dysku przy użyciu architektury RAID | 462 |
| 13.11. Sieci obszarów składowania danych | 467 |
| 13.12. Podsumowanie | 468 |

14.

| | |
|---|-----|
| Struktury indeksowe dla plików | 475 |
| 14.1. Rodzaje jednopoziomowych indeksów uporządkowanych | 476 |
| 14.2. Indeksy wielopoziomowe | 485 |
| 14.3. Dynamiczne indeksy wielopoziomowe z użyciem B-drzew i B ⁺ -drzew | 488 |
| 14.4. Indeksy na wielu kluczach | 501 |
| 14.5. Inne rodzaje indeksów | 505 |
| 14.6. Podsumowanie | 506 |

15.

| | |
|--|-----|
| Algorytmy przetwarzania i optymalizacji zapytań | 513 |
| 15.1. Translacja zapytań języka SQL do postaci wyrażeń algebry relacji | 515 |
| 15.2. Algorytmy sortowania zewnętrznego | 516 |
| 15.3. Algorytmy operacji wybierania i złączenia | 518 |
| 15.4. Algorytmy operacji rzutowania i teoriomnościowych | 528 |
| 15.5. Implementacja operacji agregujących oraz złączeń zewnętrznych | 529 |
| 15.6. Łączenie operacji poprzez mechanizm potokowy | 531 |
| 15.7. Wykorzystanie metod heurystycznych do optymalizacji zapytań | 532 |
| 15.8. Wykorzystanie oszacowań selektywności i kosztu w optymalizacji zapytań | 543 |
| 15.9. Przegląd technik optymalizacji zapytań w systemie Oracle | 552 |
| 15.10. Semantyka optymalizacji zapytań | 553 |
| 15.11. Podsumowanie | 554 |

16.

| | |
|---|-----|
| Praktyczne projektowanie i strojenie baz danych | 557 |
| 16.1. Fizyczne projektowanie baz danych w przypadku baz relacyjnych | 557 |
| 16.2. Przegląd technik strojenia baz danych w systemach relacyjnych | 560 |
| 16.3. Podsumowanie | 567 |

V**ZAGADNIENIA Z ZAKRESU PRZETWARZANIA TRANSAKCJI569****17.**

| | |
|---|-----|
| Wprowadzenie do problematyki i teorii przetwarzania transakcji | 571 |
| 17.1. Wprowadzenie do problematyki przetwarzania transakcji | 572 |
| 17.2. Pojęcia dotyczące transakcji i systemu | 578 |
| 17.3. Pożądane właściwości transakcji | 581 |
| 17.4. Charakteryzowanie harmonogramów na podstawie możliwości odtwarzania | 582 |
| 17.5. Charakterystyka harmonogramów według ich szeregowalności | 586 |
| 17.6. Obsługa transakcji w języku SQL | 596 |
| 17.7. Podsumowanie | 598 |

18.

| | |
|--|-----|
| Techniki sterowania współbieżnego | 601 |
| 18.1. Techniki blokowania dwufazowego dla celów sterowania współbieżnego | 602 |
| 18.2. Sterowanie współbieżne w oparciu o uporządkowanie według znaczników czasu | 612 |
| 18.3. Techniki wielowersyjnego sterowania współbieżnego | 614 |
| 18.4. Techniki walidacyjnego (optymistycznego) sterowania współbieżnego | 617 |
| 18.5. Ziarnistość elementów danych i blokowanie z wieloma poziomami ziarnistości | 619 |
| 18.6. Użycie blokad dla celów sterowania współbieżnego w przypadku indeksów | 623 |
| 18.7. Inne kwestie związane ze sterowaniem współbieżnym | 624 |
| 18.8. Podsumowanie | 625 |

19.

| | |
|--|-----|
| Techniki odtwarzania baz danych | 629 |
| 19.1. Pojęcia związane z odtwarzaniem | 630 |
| 19.2. Techniki odtwarzania oparte na aktualizacjach odroczonej | 636 |
| 19.3. Techniki odtwarzania oparte na aktualizacjach natychmiastowych | 641 |
| 19.4. Stronicowanie z przesłaniem | 642 |
| 19.5. Algorytm odtwarzania ARIES | 644 |
| 19.6. Odtwarzanie w systemach wielu baz danych | 647 |
| 19.7. Tworzenie kopii bezpieczeństwa bazy danych i odtwarzanie po awariach | 648 |
| 19.8. Podsumowanie | 649 |

VI**OBIEKTOWE I OBIEKTOWO-RELACYJNE BAZY DANYCH655****20.**

| | |
|--|-----|
| Idea obiektowych baz danych | 657 |
| 20.1. Przegląd pojęć zorientowanych obiektowo | 658 |
| 20.2. Tożsamość i struktura obiektów oraz konstruktory typów | 661 |
| 20.3. Enkapsulacja operacji, metody i trwałość obiektów | 666 |
| 20.4. Hierarchia typów i klas oraz dziedziczenie | 670 |
| 20.5. Obiekty złożone | 674 |
| 20.6. Inne pojęcia zorientowane obiektowo | 676 |
| 20.7. Podsumowanie | 678 |

21.

| | |
|--|-----|
| Standardy, języki i projektowanie obiektowych baz danych | 681 |
| 21.1. Wstęp do modelu obiektowego ODMG | 682 |
| 21.2. Język definicji obiektów ODL | 693 |
| 21.3. Obiektowy język zapytań OQL | 699 |
| 21.4. Przegląd wiązania z językiem C++ | 707 |
| 21.5. Projektowanie pojęciowe obiektowej bazy danych | 708 |
| 21.6. Podsumowanie | 711 |

22.

| | |
|--|-----|
| Systemy obiektowo-relacyjne i rozszerzone relacyjne | 713 |
| 22.1. Przegląd SQL i jego cech obiektowo-relacyjnych | 714 |
| 22.2. Obecne tendencje i ewolucja w technologiach baz danych | 721 |
| 22.3. Informix Universal Server | 722 |
| 22.4. Obiektowo-relacyjne cechy systemu Oracle 8 | 732 |
| 22.5. Problemy implementacyjne w rozszerzonych systemach typów | 735 |
| 22.6. Zagnieżdżony model relacyjny | 736 |
| 22.7. Podsumowanie | 739 |

VII

| | |
|-------------------|-----|
| INNE TEMATY | 741 |
|-------------------|-----|

23.

| | |
|---|-----|
| Bezpieczeństwo baz danych i mechanizmy uwierzytelniania | 743 |
| 23.1. Wprowadzenie do bezpieczeństwa baz danych | 743 |
| 23.2. Dyspozycyjna kontrola dostępu polegająca na nadawaniu i odbieraniu uprawnień | 747 |
| 23.3. Realizacja zabezpieczeń wielopoziomowych za pomocą obowiązkowej kontroli dostępu i zabezpieczeń opartych na rolach | 752 |
| 23.4. Wprowadzenie do bezpieczeństwa statystycznych baz danych | 756 |
| 23.5. Wprowadzenie do kontroli przepływu | 758 |
| 23.6. Szyfrowanie i infrastruktura klucza publicznego | 759 |
| 23.7. Podsumowanie | 761 |

24.

| | |
|--|-----|
| Rozszerzone modele danych stosowane w zaawansowanych aplikacjach | 765 |
| 24.1. Wyzwalacze i inne pojęcia związane z aktywnymi bazami danych | 766 |
| 24.2. Koncepcja czasowych baz danych | 776 |
| 24.3. Przestrzenne i multimedialne bazy danych | 787 |
| 24.4. Wprowadzenie do dedukcyjnych baz danych | 790 |
| 24.5. Podsumowanie | 803 |

25.

| | |
|--|-----|
| Rozproszone bazy danych i architektury klient-serwer | 809 |
| 25.1. Koncepcja rozproszonej bazy danych | 810 |
| 25.2. Techniki fragmentacji, replikacji i alokacji danych w projekcie rozproszonej bazy danych | 815 |
| 25.3. Rodzaje rozproszonych systemów baz danych | 820 |
| 25.4. Przetwarzanie zapytań w rozproszonych bazach danych | 823 |
| 25.5. Techniki sterowania współbieżnego i odzyskiwania danych w rozproszonych bazach danych | 828 |
| 25.6. Przegląd trójwarstwowej architektury klient-serwer | 831 |
| 25.7. Rozproszone bazy danych w Oracle | 833 |
| 25.8. Podsumowanie | 836 |

VIII**NOWE TECHNOLOGIE841****26.**

| | |
|---|------------|
| XML i internetowe bazy danych | 843 |
| 26.1. Dane strukturalne, półstrukturalne i niestukturalne | 843 |
| 26.2. Hierarchiczny (drzewiasty) model danych w dokumentach XML | 846 |
| 26.3. Dokumenty XML, DTD i schematy | 848 |
| 26.4. Dokumenty XML a bazy danych | 855 |
| 26.5. Zapytania XML | 862 |
| 26.6. Podsumowanie | 864 |

27.

| | |
|---|------------|
| Elementy drażenia danych | 867 |
| 27.1. Przegląd technologii drażenia danych | 868 |
| 27.2. Reguły asocjacyjne | 872 |
| 27.3. Klasyfikacja | 884 |
| 27.4. Grupowanie | 888 |
| 27.5. Strategie rozwiązywania pozostałych problemów związanych z drażeniem danych | 891 |
| 27.6. Zastosowania technik drażenia danych | 894 |
| 27.7. Komercyjne narzędzia drażenia danych | 894 |
| 27.8. Podsumowanie | 897 |

28.

| | |
|--|------------|
| Przegląd hurtowni danych i rozwiązań OLAP | 901 |
| 28.1. Wprowadzenie, definicje i terminologia | 901 |
| 28.2. Właściwości hurtowni danych | 903 |
| 28.3. Modelowanie danych dla hurtowni danych | 904 |
| 28.4. Budowanie hurtowni danych | 910 |
| 28.5. Typowa funkcjonalność hurtowni danych | 913 |
| 28.6. Hurtownie danych kontra perspektywy | 914 |
| 28.7. Problemy i nierozwiązane zagadnienia związane z hurtowniami danych | 914 |
| 28.8. Podsumowanie | 916 |

29.

| | |
|---|------------|
| Przegląd najnowszych technologii i zastosowań baz danych | 919 |
| 29.1. Mobilne bazy danych | 920 |
| 29.2. Multimedialne bazy danych | 928 |
| 29.3. Systemy informacji geograficznej | 936 |
| 29.4. Zarządzanie danymi kodu genetycznego | 943 |

DODATKI953

| | |
|---|------------|
| A Alternatywne notacje modeli związków encji | 955 |
| B Parametry dysków | 959 |
| C Omówienie języka QBE | 963 |
| Bibliografia | 971 |
| Skorowidz | 1007 |

Bazy danych i ich użytkownicy

Bazy danych i systemy baz danych stały się kluczowym narzędziem w życiu codziennym współczesnego społeczeństwa. Niemał codziennie każdy z nas wykonuje wiele czynności, które w taki czy inny sposób wiążą się z wykorzystywaniem bazy danych. Przykładowo, kiedy idziemy do banku celem wpłacenia lub wypłacenia pieniędzy z naszego konta, kiedy rezerwujemy pokój w hotelu lub bilet lotniczy, kiedy szukamy interesujących nas pozycji w skomputeryzowanym katalogu biblioteki lub kiedy kupujemy cokolwiek (książkę, zabawkę czy choćby komputer) w sklepie internetowym obsługiwany za pośrednictwem strony WWW, jest bardzo prawdopodobne, że tego typu czynności wymagają od kogoś lub od jakiegoś programu komputerowego właśnie dostępu do bazy danych. Nawet zwykłe zakupy w supermarkecie w wielu przypadkach wiążą się obecnie z automatyczną aktualizacją bazy danych reprezentującej stan zapasów poszczególnych artykułów.

Wymienione powyżej działania są przykładami tego, co możemy nazwać **tradycyjnymi zastosowaniami baz danych**, w których większość przechowywanych i uzyskiwanych informacji ma postać albo danych tekstowych, albo danych numerycznych. Przez ostatnie kilka lat stały postęp technologiczny doprowadził do wynalezienia zupełnie nowych, pasjonujących zastosowań dla systemów baz danych. Multimedialne bazy danych mogą teraz przechowywać obrazy, klipy wideo czy komunikaty głosowe. **Geograficzne systemy informacyjne** (ang. *Geographic Information Systems — GIS*) mogą przechowywać i analizować mapy, dane o pogodzie, a nawet zdjęcia satelitarne. **Hurtownie danych i systemy OLAP** (od ang. *Online Analytical Processing*) są wykorzystywane do wydzielania z ogromnych baz danych przydatnych informacji i ich analizy, która ma ostatecznie ułatwić podjęcie właściwych decyzji. Technologie **baz danych czasu rzeczywistego i aktywnych baz danych** są przydatne podczas sterowania procesami w produkcji przemysłowej. Okazuje się także, że techniki przeszukiwania baz danych znalazły zastosowanie w świecie witryn i stron internetowych, gdzie w ostatnim czasie znacznie usprawniono mechanizmy wyszukiwania przez użytkowników potrzebnych informacji w internecie.

Aby zrozumieć podstawy tych technologii, musimy jednak rozpocząć nasze rozważania od analizy bardziej tradycyjnych zastosowań baz danych. W podrozdziale 1.1 zdefiniujemy więc to, czym faktycznie jest baza danych, i dopiero potem przystąpimy do omawiania innych podstawowych pojęć. W podrozdziale 1.2 przedstawimy prosty przykład bazy danych UNIwersytet, który będzie stanowił dobrą ilustrację dla naszych rozważań. W podrozdziale 1.3 opiszemy niektóre spośród najważniejszych własności systemów baz danych, natomiast w podrozdziałach 1.4 i 1.5 spróbujemy odpowiednio skategoryzować typy pracowników, których praca wiąże się z wykorzystywaniem systemów baz danych. Podrozdziały 1.6, 1.7 oraz 1.8 zawierają bardziej szczegółową analizę możliwości oferowanych przez systemy baz danych oraz ich typowych zastosowań. Treść podrozdziału 1.9 będzie podsumowaniem całego rozdziału.

Czytelnicy, którzy chcieliby bardzo szybko przebrnąć przez teoretyczne wprowadzenie do systemów baz danych, mogą przeczytać jedynie podrozdziały od 1.1 do 1.5, pomijając podrozdziały od 1.6 do 1.8 i od razu przejść do rozdziału 2.

1.1. Wprowadzenie

Bazy danych i technologie baz danych w ogromnym stopniu przyczyniają się do wzrostu liczby zastosowań komputerów. Możemy z pełnym przekonaniem stwierdzić, że właśnie bazy danych odgrywają główną rolę niemal we wszystkich obszarach, w których w ogóle wykorzystuje się komputery — wystarczy wymienić takie gałęzie jak biznes, handel elektroniczny, inżynieria, medycyna, prawo, edukacja czy bibliotekarstwo. Określenie *baza danych* jest dziś na tyle popularne, że powinniśmy rozpocząć nasze rozważania od zdefiniowania, czym faktycznie jest baza danych. Przedstawiona poniżej definicja będzie na razie dosyć ogólna.

Baza danych jest zbiorem powiązanych ze sobą danych. Przez **dane** rozumiemy w tym przypadku znane fakty, które można jakoś zarejestrować, i które mają konkretne znaczenie. Przykładowo, rozważmy nazwiska, numery telefonów oraz adresy osób, które znamy — możemy mieć te dane zarejestrowane w odpowiednio indeksowanej książce adresowej lub przechowywać je na twardym dysku swojego komputera z wykorzystaniem oprogramowania (choćby aplikacji Microsoft Access lub Microsoft Excel). Mamy więc zbiór powiązanych ze sobą danych o konkretnym znaczeniu, co oznacza, że dysponujemy bazą danych.

Powyższa definicja bazy danych jest dosyć ogólna; przykładowo, możemy traktować zbiór słów składających się na tę stronę tekstu jako zbiór wzajemnie powiązanych danych, a więc (zgodnie z przyjętymi przed chwilą warunkami) jako bazę danych. Pojęcie *bazy danych* w powszechnym rozumieniu jest jednak znacznie węższe. Każda taka baza danych musi mieć następujące własności:

- Baza danych reprezentuje jakiś wybrany aspekt świata rzeczywistego, nazywany niekiedy **mini-światem** lub **dziedziną problemu** (ang. *Universe of Discourse* — *UoD*). Zmiany w takim mini-świecie muszą być uwzględniane w bazie danych.
- Baza danych jest logicznie koherentnym zbiorem danych z jakimś spójnym znaczeniem. Przykładowy zbiór danych nie jest prawidłową bazą danych i nie należy dla niego stosować tego określenia.
- Baza danych jest projektowana, konstruowana i wypełniana danymi w określonym celu. Do bazy danych powinna być przypisana grupa docelowych użytkowników oraz z góry przyjęte zastosowania, które będą realizowane przez tych użytkowników.

Innymi słowy, każda baza danych powinna mieć pewne źródło, z którego będą pochodzić przechowywane dane, pewien stopień interakcji ze zdarzeniami mającymi miejsce w reprezentowanym przez nią wycinku świata rzeczywistego oraz użytkowników, którzy są aktywnie zainteresowani jej zawartością.

Bazy danych mogą mieć dowolne rozmiary i zróżnicowane poziomy złożoności. Przykładowo, wspomniana już lista nazwisk i adresów może się składać tylko z kilku tysięcy rekordów, z których każdy ma stosunkowo prostą strukturę. Z drugiej strony, skomputeryzowany katalog zbiorów wielkiej biblioteki może się składać z pół miliona wpisów zorganizowanych w różnych kategoriach (według nazwiska autora, tematu, tytułu itp.), które są z kolei uporządkowane w kolejności alfabetycznej. Jeszcze większa i bardziej skomplikowana baza danych jest utrzymywana przez amerykański urząd skarbowy (Internal Revenue Service), który przechowuje formularze podatkowe wypełniane przez podatników na terenie całych Stanów Zjednoczonych. Jeśli przyjmiemy, że takich

podatników jest 100 milionów i rekord dla każdego z nich zawiera średnio pięć formularzy po 400 znaków informacji, otrzymamy bazę danych zawierającą $100 \times 10^6 \times 400 \times 5$ znaków (bajtów) informacji. Gdyby okazało się, że amerykański urząd skarbowy przechowuje dodatkowo dla każdego podatnika po trzy formularze nadesłane w przeszłości (poza formularzem tegorocznym), otrzymalibyśmy bazę danych zawierającą aż 8×10^{11} bajtów (800 gigabajtów) informacji. Tak ogromna ilość informacji musi być zorganizowana i zarządzana w taki sposób, aby użytkownicy mogli tę bazę danych przeszukiwać, uzyskiwać interesujące ich informacje oraz — w razie potrzeby — aktualizować istniejące zapisy.

Baza danych może być generowana i utrzymywana ręcznie lub może zostać całkowicie skomputeryzowana. Przykładowo, katalog kart bibliotecznych jest tym rodzajem bazy danych, który można stworzyć i utrzymywać bez pomocy komputerów. Skomputeryzowane bazy danych mogą być tworzone i utrzymywane zarówno za pomocą grupy programów napisanych specjalnie z myślą o tym zadaniu, jak i przez uniwersalny system zarządzania bazą danych. W tej książce będziemy się oczywiście skupiali wyłącznie na skomputeryzowanych bazach danych.

System zarządzania bazą danych (w skrócie **SZBD**, ang. *Database Management System* — *DBMS*) jest zbiorem programów, które umożliwiają tworzenie i utrzymywanie bazy danych. SZBD jest więc *uniwersalnym systemem programowym*, który ułatwia *definiowanie, konstruowanie, manipulowanie* i *udostępnianie* baz danych różnym użytkownikom i aplikacjom. **Definiowanie** bazy danych wiąże się z określaniem typów i struktur danych oraz ograniczeń dla przechowywanych informacji. **Konstruowanie** bazy danych jest kontrolowanym przez SZBD procesem umieszczania w niej właściwych informacji na jakimś medium przechowywania. **Manipulowanie** bazą danych obejmuje takie działania jak wykonywanie zapytań wyciągających z bazy określone informacje, aktualizowanie bazy danych w taki sposób, aby zawarte w niej informacje odzwierciedlały rzeczywisty stan reprezentowanego mini-swiata, oraz generowanie raportów na podstawie informacji zawartych w bazie danych. **Udostępnianie** bazy danych umożliwia wielu użytkownikom i programom jednocześnie operowanie na zawartych w niej informacjach.

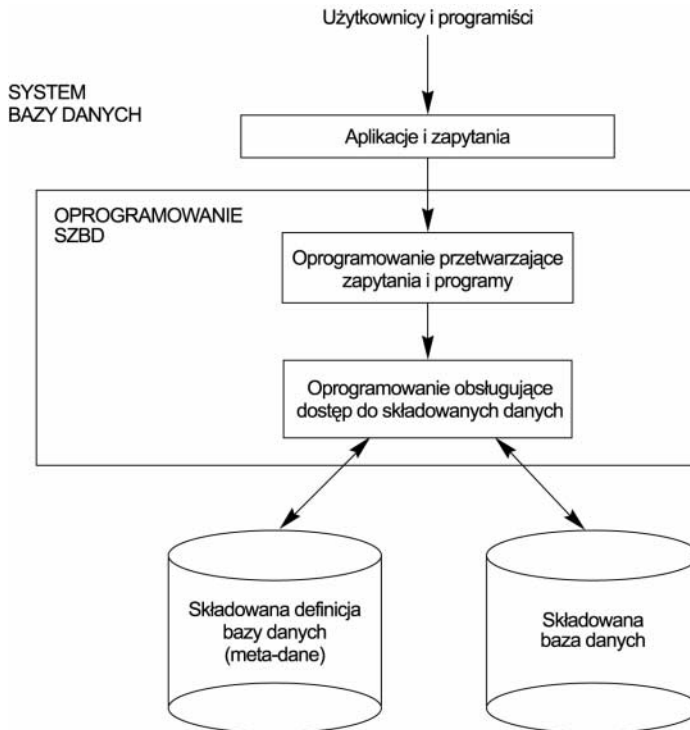
Do pozostałych funkcji oferowanych przez systemy zarządzania bazami danych należy *ochrona* bazy danych oraz *konserwowanie* jej przez długi okres czasu. Proces **ochrony** bazy danych obejmuje zarówno *ochronę systemową* przed niewłaściwym działaniem (lub awariami) sprzętu oraz oprogramowania, jak i *zabezpieczenie* przed nieuprawnionym dostępem. Cykl życia wielkich baz danych może trwać wiele lat, zatem systemy zarządzania takimi bazami danych muszą umożliwiać ich **konserwowanie**, tak aby możliwe było dostosowywanie ich do ewoluujących wymagań.

Implementowanie skomputeryzowanych baz danych wcale nie wymaga stosowania uniwersalnego oprogramowania typu SZBD. Możemy przecież napisać własny zbiór programów tworzących i utrzymujących naszą bazę danych, tworząc tym samym własne, specyficzne oprogramowanie typu SZBD. W obu przypadkach (a więc niezależnie od tego, czy zdecydowaliśmy się wykorzystać uniwersalny SZBD) musimy zwykle wdrożyć w docelowym systemie komputerowym dość skomplikowany pakiet oprogramowania. W rzeczywistości większość systemów zarządzania bazami danych jest bardzo skomplikowanymi systemami oprogramowania.

Aby prezentowany w tym podrozdziale zbiór definicji wprowadzających był kompletny, musimy jeszcze określić, czym jest **system bazy danych** — otóż jest to połączenie samej bazy danych z oprogramowaniem SZBD. Rysunek 1.1 ilustruje niektóre spośród omówionych do tej pory zagadnień.

1.2. Przykład

Przeanalizujmy teraz prosty przykład, który większości czytelników tej książki może wydać się znajomy — chodzi o bazę danych UNIWERSYTET reprezentującą informacje dotyczące studentów, przedmiotów i ocen w środowisku wyższej uczelni. Na rysunku 1.2 przedstawiono strukturę bazy



RYSUNEK 1.1.
Uproszczone
środowisko systemu
bazy danych

danych wraz z kilkoma przykładowymi informacjami wypełniającymi poszczególne tabele. Prezentowana baza danych składa się z pięciu plików, z których każdy zawiera rekordy danych tego samego typu¹. Plik *STUDENT* zawiera dane na temat poszczególnych studentów, plik *PRZEDMIOT* — na temat poszczególnych przedmiotów, plik *KURS* — na temat poszczególnych kursów z odpowiednich przedmiotów, plik *RAPORT_OCEN* zawiera dane na temat ocen uzyskanych przez studentów podczas zaliczania poszczególnych kursów, natomiast plik *WYMAGANIE_WSTEPNE* — informacje o wymaganiach odnośnie do zaliczonych zajęć przed uzyskaniem możliwości uczestnictwa w zajęciach z poszczególnych przedmiotów.

Aby zdefiniować taką bazę danych, musimy określić strukturę rekordów przechowywanych w każdym z plików przez sprecyzowanie różnych typów **elementów danych**, które mają być przechowywane w kolejnych rekordach. Na rysunku 1.2 każdy rekord pliku *STUDENT* zawiera dane reprezentujące nazwisko studenta (element danych *Nazwisko*), numer indeksu (element danych *NumerIndeksu*), rok studiów (element danych *RokSt*, który może zawierać takie wartości jak *pierwszy* lub *1*, *Drugi* lub *2*, ...) oraz kierunek (element danych *Kierunek*, który może zawierać takie wartości jak *MAT*, *Informatyka* lub *INF*, ...). Każdy rekord pliku *PRZEDMIOT* zawiera dane reprezentujące nazwę przedmiotu (element danych *NazwaPrzedmiotu*), identyfikator przedmiotu (element danych *NumerPrzedmiotu*), liczbę godzin tygodniowo (element danych *Godziny*) oraz wydział, na którym dany przedmiot jest wykładany (element danych *Wydzial*), itd. Dla każdego takiego elementu danych wewnątrz rekordu musimy także określić **typ danych**. Przykładowo, możemy określić, że element danych *Nazwisko* w pliku *STUDENT* jest ciągiem znaków alfabety, element danych *NumerIndeksu* w tym

¹ Wykorzystujemy w tym miejscu pojęcie *pliku* w sposób nieformalny. Na poziomie koncepcyjnym *plik* jest *zbiorem* rekordów, które mogą (choć nie muszą) być uporządkowane.

| STUDENT | Nazwisko | NumerIndeksu | Rok | Kierunek |
|---------|----------|--------------|-----|----------|
| | Nowak | 17 | 1 | INF |
| | Kowalski | 8 | 2 | INF |

| PRZEDMIOT | NazwaPrzedmiotu | NumerPrzedmiotu | Godziny | Wydział |
|-----------|-----------------------------|-----------------|---------|---------|
| | Wprowadzenie do informatyki | INF1310 | 4 | INF |
| | Struktury danych | INF3320 | 4 | INF |
| | Matematyka dyskretna | MAT2410 | 3 | MAT |
| | Bazy danych | INF3380 | 3 | INF |

| DZIAŁ | IdentyfikatorDziału | NumerPrzedmiotu | Semestr | Rok | Prowadzący |
|-------|---------------------|-----------------|----------|-----|-------------|
| | 85 | MAT2410 | Jesienny | 98 | Kaczmarek |
| | 92 | INF1310 | Jesienny | 98 | Asnyk |
| | 102 | INF3320 | Wiosenny | 99 | Kowalik |
| | 112 | MAT2410 | Jesienny | 99 | Chłopicki |
| | 119 | INF1310 | Jesienny | 99 | Asnyk |
| | 135 | INF3380 | Jesienny | 99 | Szamotulski |

| RAPORT_OCEN | NumerIndeksu | IdentyfikatorDziału | Ocena |
|-------------|--------------|---------------------|-------|
| | 17 | 112 | 4 |
| | 17 | 119 | 3 |
| | 8 | 85 | 5 |
| | 8 | 92 | 5 |
| | 8 | 102 | 4 |
| | 8 | 135 | 5 |

| WYMAGANIE_WSTĘPNE | NumerPrzedmiotu | NumerWymagania |
|-------------------|-----------------|----------------|
| | INF3380 | INF3320 |
| | INF3380 | MAT2410 |
| | INF3380 | INF1310 |

RYSUNEK 1.2.
Baza danych
zawierająca
informacje
o studentach
i przedmiotach

samym pliku jest liczbą całkowitą, natomiast element danych *Ocena* w pliku `RAPORT_OCEN` jest pojedynczą wartością ze zbioru $\{5, 4, 3, 2\}$. Do reprezentowania wartości poszczególnych elementów danych możemy także użyć odpowiedniego schematu kodowania. Przykładowo, w widocznym na rysunku 1.2 elemencie danych *RokSt* w pliku `STUDENT` wartość *1* lub *Pierwszy* reprezentuje studenta pierwszego roku, wartość *2* lub *Drugi* studenta drugiego roku, wartość *3* lub *Trzeci* studenta trzeciego roku, wartość *4* lub *Czwarty* studenta czwartego roku, a wartość *5* lub *Piąty* — studenta piątego roku.

Aby *skonstruować* bazę danych `UNIwersytet`, zapisujemy dane, które mają reprezentować poszczególnych studentów, przedmioty, kursy, oceny i wymagania w postaci rekordu w odpowiednim pliku. Łatwo zauważyć, że rekordy w różnych plikach mogą być ze sobą powiązane. Przykładowo, rekord *Nowak* w pliku `STUDENT` może być powiązany z dwoma rekordami w pliku `RAPORT_OCEN`, który przechowuje oceny, jakie student o takim nazwisku uzyskał z dwóch zaliczanych przezeń kursów. Podobnie, każdy rekord w pliku `WYMAGANIE_WSTĘPNE` jest związany z dwoma rekordami

reprezentującymi dwa różne przedmioty: przedmiot, dla którego zdefiniowano dane wymagane, oraz przedmiot, którego zaliczenie stanowi warunek uczestnictwa w zajęciach z pierwszego przedmiotu. Większość średnich i wielkich baz danych składa się z wielu typów rekordów i zawiera *wiele związków* pomiędzy tymi rekordami.

Manipulowanie bazą danych wiąże się z wykonywaniem zapytań i aktualizacją zawartych w niej informacji. Przykładowo, zapytania mogą mieć następującą postać: „wygeneruj listę wszystkich przedmiotów, w których uczestniczył student o nazwisku *Nowak*, wraz z uzyskanymi ocenami”, „wygeneruj listę nazwisk studentów, którzy brali udział w zajęciach z danego kursu w ramach przedmiotu *Bazy Danych* w semestrze jesiennym 1999 roku wraz z uzyskanymi przez nich ocenami z tego kursu” oraz „jakie wymagania należy spełnić przed przystąpieniem do zajęć z przedmiotu *Bazy danych*?”. Przykładowe instrukcje aktualizujące mogą natomiast mieć postać: „zmień rok studiów studenta *Nowak* na drugi (*Drugi*)”, „stwórz nowy kurs dla przedmiotu *Bazy danych* prowadzonego w bieżącym semestrze” oraz „zapisz ocenę 5 dla studenta *Nowak* za zaliczenie w ostatnim semestrze wskazanego kursu z przedmiotu *Bazy danych*”. Przetworzenie wymienionych przed chwilą nieformalnych zapytań i instrukcji aktualizujących wymaga oczywiście nadania im postaci precyzyjnych poleceń zapisanych w języku zapytań, który jest obsługiwany w wykorzystywanym systemie zarządzania bazą danych.

1.3. Właściwości rozwiązań opartych na bazach danych

Wiele unikatowych właściwości odróżnia rozwiązania oparte na bazach danych od tradycyjnych metod programowania aplikacji wykorzystujących pliki z danymi. W tradycyjnym **przetwarzaniu plików** każdy użytkownik definiuje i implementuje pliki, które są mu potrzebne do konkretnego zastosowania danego programu, i czynność ta stanowi jeden z elementów programowania tego zastosowania. Przykładowo, jeden użytkownik — przyjmijmy, że jest to *dziekanat* — może utrzymywać plik z informacjami o studentach i uzyskanych przez nich ocenach. Programy drukujące listy studentów i umożliwiające wpisywanie do pliku nowych ocen są implementowane w postaci jednego z elementów tego zastosowania. Drugi użytkownik — niech to będzie tym razem *dział księgowości* — może zarządzać przyznawaniem i wypłacaniem stypendiów. Chociaż oba podmioty są zainteresowane danymi na temat studentów, każdy utrzymuje własne pliki (wraz z programami operującymi na tych plikach), ponieważ każdy z tych podmiotów wymaga pewnych danych, które nie są dostępne w plikach utrzymywanych przez pozostałe podmioty. Efektem tej nadmiarowości w definiowaniu i przechowywaniu danych jest niepotrzebna strata wykorzystywanej przestrzeni oraz zwiększony koszt utrzymywania aktualnych informacji.

Typowe rozwiązanie oparte na bazie danych przewiduje, że utrzymywane jest tylko jedno repozytorium danych, które, raz utworzone, jest udostępniane wielu różnym użytkownikom. Poniżej przedstawiono główne cechy odróżniające rozwiązania oparte na bazach danych od rozwiązań opartych na przetwarzaniu plików:

- samoopisująca natura systemów baz danych,
- oddzielenie programów od danych oraz abstrakcja danych,
- obsługa wielu perspektyw dla tych samych danych,
- współdzielenie danych oraz przetwarzanie transakcji.

W poniższych punktach opiszemy oddzielnie każdą z tych właściwości. Dodatkowe omówienie własności systemów baz danych można znaleźć w podrozdziałach 1.6 i 1.8.

1.3.1. Samoopisująca natura systemów baz danych

Podstawowa właściwość rozwiązań opartych na stosowaniu baz danych określa, że system bazy danych zawiera nie tylko samą bazę danych, ale także kompletną definicję lub opis jej struktury i ograniczeń. Wspomniana definicja jest przechowywana w **katalogu** systemu zarządzania bazą danych (SZBD), który zawiera informacje odnośnie do struktury poszczególnych plików, typu i formatu przechowywania poszczególnych elementów danych oraz rozmaitych ograniczeń nałożonych na te dane. Informacje przechowywane w katalogu są często nazywane **metadanymi**. Metadane zawsze opisują strukturę głównej bazy danych (patrz rysunek 1.1).

Katalog jest wykorzystywany nie tylko przez oprogramowanie systemu zarządzania bazą danych, ale także przez jej użytkowników, którzy potrzebują informacji na temat struktury bazy danych. Uniwersalne pakiety programowe SZBD nie są tworzone z myślą o konkretnych zastosowaniach baz danych, a więc muszą się odwoływać do katalogów celem uzyskania informacji o strukturze plików w konkretnych bazach danych (np. aby dysponować niezbędną wiedzą na temat typu i formatu danych, które mają zostać odczytane lub zapisane). Oprogramowanie systemu zarządzania bazą danych, które wykorzystuje katalog do przechowywania definicji baz danych, musi się sprawdzać tak samo dobrze *we wszystkich możliwych zastosowaniach obsługiwanych baz danych* — niezależnie od tego, czy jest to baza danych uniwersytetu, banku, czy jakiegokolwiek innego podmiotu.

W tradycyjnym modelu przetwarzania plików definicja danych jest zwykle częścią samych aplikacji wykorzystywanych do ich obsługi. Oznacza to, że możliwości tych programów ograniczają się do pracy *tylko z jedną bazą danych*, której struktura została w nich z góry zadeklarowana. Przykładowo, aplikacja napisana w języku programowania C++ może zawierać deklaracje struktur lub klas, natomiast program napisany w języku COBOL może wykorzystywać do definiowania swoich plików instrukcji *Data Division*. O ile oprogramowanie przetwarzające pliki może uzyskiwać dostęp wyłącznie do określonych baz danych, oprogramowanie systemów zarządzania bazami danych może obsługiwać rozmaite bazy danych przez odczytywanie i odpowiednie wykorzystywanie ich definicji zawartych w katalogu.

W przykładzie zaprezentowanym na rysunku 1.2 katalog wykorzystywanego systemu SZBD będzie przechowywał definicje wszystkich przedstawionych plików. Takie definicje są tworzone i umieszczane w katalogu przez projektanta bazy danych, zanim jeszcze przystąpi do jej tworzenia. Za każdym razem, gdy do systemu zarządzania bazą danych dociera żądanie dostępu, np. wartości elementu danych *Nazwisko* rekordu w pliku STUDENT, oprogramowanie tego systemu odwołuje się do katalogu, aby określić strukturę pliku STUDENT oraz pozycję i rozmiar elementu danych *Nazwisko* wewnątrz danego rekordu. Zupełnie inaczej byłoby w przypadku typowej aplikacji przetwarzającej pliki, gdzie struktura pliku i — w ekstremalnej sytuacji — także dokładne położenie elementu danych *Nazwisko* wewnątrz rekordu STUDENT są na stałe zakodowane wewnątrz każdego programu, który może uzyskać dostęp do składowanych w ten sposób informacji.

1.3.2. Oddzielenie programów od danych oraz abstrakcja danych

W tradycyjnej metodzie przetwarzania plików struktura plików z danymi jest umieszczana w wykorzystywanych programach, zatem wszystkie ewentualne zmiany tej struktury mogą wymagać odpowiedniego *zmodyfikowania wszystkich programów*, które uzyskują dostęp do danego pliku. Zupełnie inaczej jest w przypadku programów wykorzystujących pośrednictwo systemów zarządzania bazami danymi, które w większości podobnych sytuacji nie wymagają tego typu dodatkowych zmian. Struktura plików z danymi jest przechowywana w specjalnym katalogu systemu SZBD, dzięki czemu jest oddzielona od programów dostępowych. Taki model nazywamy **niezależnością programu od danych**. Przykładowo, pewien program uzyskujący dostęp do pliku mógłby zostać napisany w taki

sposób, że będzie mógł pracować wyłącznie z rekordami pliku STUDENT, których struktura jest identyczna jak ta przedstawiona na rysunku 1.3. Jeśli uznamy za konieczne dodanie kolejnego elementu danych do każdego rekordu w tym pliku, np. reprezentujący datę urodzenia element *DataUrodzenia*, tak skonstruowany program nie będzie mógł dłużej działać i najprawdopodobniej konieczna będzie ingerencja w jego kod źródłowy. Gdybyśmy zamiast tego programu użyli środowiska SZBD, musielibyśmy jedynie zmienić w katalogu opis rekordów pliku STUDENT w taki sposób, aby uwzględniły dodanie nowego elementu danych *DataUrodzenia* — zmiana struktury nie pociągałaby więc za sobą konieczności dodatkowych modyfikacji żadnego programu. Już podczas następnego odwołania programu SZBD do zmienionego katalogu zostałaby odczytana i wykorzystana nowa struktura rekordów pliku STUDENT.

| Nazwa elementu danych | Pozycja początkowa w rekordzie | Długość w znakach (bajtach) |
|-----------------------|--------------------------------|-----------------------------|
| Nazwisko | 1 | 30 |
| NumerIndeksu | 31 | 4 |
| Rok | 35 | 4 |
| Kierunek | 39 | 4 |

RYSUNEK 1.3. Wewnętrzny format przechowywania dla rekordu z pliku STUDENT

W niektórych typach systemów baz danych, np. w systemach obiektowych i obiektowo-relacyjnych (patrz rozdziały 20. i 22.), to użytkownicy mogą (w ramach definiowania baz danych) określać możliwe operacje na danych. Taka **operacja** (nazywana także *funkcją* lub *metodą*) składa się z dwóch elementów. Pierwszym z nich jest *interfejs* (nazywany często *sygnaturą*), który zawiera nazwę operacji oraz typy danych jej argumentów (nazywanych także parametrami). Drugim jest *implementacja* (*metoda*), która jest definiowana osobno i może być zmieniana bez konieczności modyfikowania odpowiadającego jej interfejsu. Aplikacje użytkownika mogą operować na danych przez wywoływanie tak zdefiniowanych operacji poprzez ich nazwy i argumenty, niezależnie od tego, w jaki sposób te operacje zostały zaimplementowane. Można taki model nazwać **niezależnością programu od operacji**.

Właściwość, która umożliwia zapewnienie niezależności programu od danych oraz niezależności programu od operacji, jest nazywana **abstrakcją danych**. Systemy zarządzania bazami danych udostępniają użytkownikom **konceptyjną reprezentację** danych, która nie zawiera wielu szczegółów związanych z wykorzystywanymi technikami przechowywania danych oraz implementacji operacji. Nieformalnie można przyjąć, że jednym z typów abstrakcji danych jest **model danych**, stosowany do zapewniania właśnie reprezentacji konceptyjnej. Model danych wykorzystuje takie logiczne pojęcia jak obiekty, ich własności oraz ich wewnętrzne relacje, które dla wielu użytkowników są bardziej zrozumiałe od pojęć związanych z przechowywaniem danych na współczesnych komputerach. Oznacza to, że model danych *ukrywa* szczegóły przechowywania i implementacji danych, które nie są przedmiotem zainteresowania dla większości użytkowników baz danych.

Przykładowo, przeanalizujmy ponownie strukturę bazy danych przedstawioną na rysunku 1.2. Wewnętrzna implementacja pliku może być zdefiniowana za pomocą długości jego rekordów — liczby znaków (bajtów) w każdym rekordzie — natomiast każdy z wykorzystywanych elementów danych może zostać określony za pomocą jego bajtu początkowego wewnątrz odpowiedniego rekordu oraz długości (także wyrażonej w bajtach). Rekord z pliku STUDENT byłby w takim przypadku reprezentowany w taki sam sposób, jak to przedstawiono na rysunku 1.3. Typowego użytkownika bazy danych nie interesują jednak ani dokładne miejsce przechowywania poszczególnych elementów danych wewnątrz rekordu, ani jego rzeczywista długość; zamiast tego, taki użytkownik oczekuje przede wszystkim tego, aby po odwołaniu się do elementu *Nazwisko* wybranego rekordu w pliku STUDENT została zwrócona prawidłowa wartość. Konceptyjną reprezentację rekordów w pliku STUDENT przedstawiono na rysunku 1.2. Wiele innych szczegółów organizacji plików z danymi (w tym zdefiniowanych

dla niego ścieżek dostępu) można ukryć przed użytkownikami baz danych dzięki odpowiednim mechanizmom systemu zarządzania bazami danych (szczegóły związane z przechowywaniem danych omówimy w rozdziałach 13. i 14.).

W typowym rozwiązaniu opartym na bazie danych szczegółowa struktura i organizacja poszczególnych plików jest przechowywana w katalogu. Użytkownicy bazy danych i obsługujących ją aplikacje odwołują się jedynie do koncepcyjnej reprezentacji tych plików, a za wydobywanie z katalogu szczegółów związanych z przechowywaniem plików odpowiadają stosowne moduły dostępu używanych systemów zarządzania bazami danych. Do zapewniania abstrakcji danych użytkownikom baz danych można wykorzystywać rozmaite modele danych. Znaczna część tej książki jest poświęcona prezentacji różnych modeli danych i stosowanym w nich mechanizmom abstrahowania reprezentacji danych.

W obiektowych i obiektowo-relacyjnych bazach danych proces abstrahowania obejmuje nie tylko struktury danych, ale także zdefiniowane dla nich operacje. Takie operacje stanowią abstrakcję dla czynności w reprezentowanym przez bazę danych mini-świecie, które zwykle są łatwiejsze do zrozumienia dla zwykłych użytkowników. Przykładowo, do obliczania średniej ocen wskazanego obiektu z pliku STUDENT może służyć operacja OBLICZ ŚROC. Operacje takie jak ta mogą być wywoływane przez użytkownika za pomocą odpowiednich zapytań (lub za pośrednictwem aplikacji obsługujących bazę danych) bez znajomości szczegółów dotyczących sposobu zaimplementowania wykonywanych działań. W tym sensie abstrakcja czynności z reprezentowanego mini-świata jest udostępniana użytkownikowi jako **operacja abstrakcyjna**.

1.3.3. Obsługa wielu perspektyw dla tych samych danych

Typowa baza danych ma wielu użytkowników, z których każdy może potrzebować dostępu do innej **perspektywy** (ang. *view*) dla tej bazy danych. Perspektywa może być albo podzbiorem bazy danych, albo może zawierać dane wirtualne, które zostały wywiedzione z plików bazy danych, gdzie są przechowywane w nieco innej postaci (dane wirtualne same w sobie nie są więc trwale składowane w bazie danych). Niektórzy użytkownicy w ogóle nie muszą wiedzieć, czy dane, do których się odwołują, faktycznie są przechowywane w takiej postaci w bazie danych, czy są z niej jedynie w odpowiedni sposób wywiedzione (w procesie tworzenia perspektywy). Wielodostępny system zarządzania bazą danych, który jest wykorzystywany przez użytkowników do odmiennych celów, musi zapewniać mechanizmy ułatwiające definiowanie wielu różnych perspektyw. Przykładowo, jeden z użytkowników bazy danych przedstawionej na rysunku 1.2 może być zainteresowany wyłącznie dostępem z możliwością drukowania do listy studentów wraz z opisującymi ich informacjami; perspektywę przygotowaną dla takiego użytkownika przedstawiono na rysunku 1.4(a). Drugi użytkownik, którego interesuje wyłącznie sprawdzanie, czy poszczególni studenci zaliczyli wszystkie przedmioty wymagane do ich udziału w zajęciach, na które się zapisali, może potrzebować perspektywy przedstawionej na rysunku 1.4(b).

1.3.4. Współdzielenie danych oraz wielodostępne przetwarzanie transakcji

Wielodostępny system zarządzania bazą danych, jak sama nazwa wskazuje, musi umożliwiać wielu użytkownikom jednoczesny dostęp do bazy danych. Ta właściwość ma zasadnicze znaczenie, jeśli z pojedynczą bazą danych chcemy zintegrować wiele aplikacji. System zarządzania bazą danych

a)

| LISTA | NazwiskoStudenta | Lista studentów | | | | |
|----------|------------------|-----------------|-------|----------|-----|---------------------|
| | | NumerPrzedmiotu | Ocena | Semestr | Rok | IdentyfikatorDziału |
| Nowak | | INF1310 | 3 | Jesienny | 99 | 119 |
| | | MAT2410 | 4 | Jesienny | 99 | 112 |
| Kowalski | | MAT2410 | 5 | Jesienny | 98 | 85 |
| | | INF1310 | 5 | Jesienny | 98 | 92 |
| | | INF3320 | 4 | Wiosenny | 99 | 102 |
| | | INF3380 | 5 | Jesienny | 99 | 135 |

b)

| WYMAGANIA_WSTĘPNE | NazwaPrzedmiotu | NumerPrzedmiotu | WymaganiaWstępne |
|-------------------|-----------------|-----------------|------------------|
| Bazy danych | | INF3380 | INF3320 |
| | | | MAT2410 |
| Struktury danych | | INF3320 | INF1310 |

RYSUNEK 1.4. Dwie perspektywy dla bazy danych z rysunku 1.2. (a) Perspektywa REJESTR_OCEN_STUDENT. (b) Perspektywa PRZEDMIOT_WYMAGANIA_WSTĘPNE

musi wówczas zawierać oprogramowanie **sterowania współbieżnego** (ang. *concurrency control*), które zapewni — w sposób kontrolowany — wielu użytkownikom możliwość podejmowania prób aktualizacji tych samych danych i zagwarantuje, że efekt tych działań będzie spójny logicznie. Przykładowo, kiedy wielu pracowników linii lotniczych spróbuje jednocześnie zarezerwować miejsce w samolocie, system zarządzania bazą danych powinien zagwarantować dostępność każdego miejsca tylko dla jednego pasażera obsługiwane przez jednego pracownika. Tego typu zastosowania są określane ogólnym terminem rozwiązań z **przetwarzaniem transakcji na bieżąco** (ang. *Online Transaction Processing — OLTP*). Zasadniczą rolą oprogramowania wielodostępnego systemu zarządzania bazą danych jest wówczas zapewnienie właściwego przetwarzania współbieżnych transakcji.

Pojęcie **transakcji** stało się centralnym elementem w wielu współczesnych zastosowaniach baz danych. Transakcja jest *wykonywanym programem* lub *procesem*, na który składa się jeden lub więcej operacji dostępu do bazy danych, takich jak odczytywanie czy aktualizacja jej rekordów. Każda transakcja, która zostanie zrealizowana w całości, ma w założeniu wykonywać logicznie poprawne operacje dostępu do bazy danych i nie może wpływać na przebieg pozostałych transakcji. System zarządzania bazą danych musi wymuszać wiele właściwości transakcji. Własność **izolacji** daje nam gwarancję, że każda transakcja będzie wykonywana w warunkach separacji od pozostałych transakcji, nawet jeśli jednocześnie będą wykonywane setki transakcji. Własność **niepodzielności** gwarantuje, że albo zostaną wykonane wszystkie operacje na bazie danych składające się na daną transakcję, albo nie zostanie wykonana żadna z tych operacji. Szczegółowe omówienie transakcji można znaleźć w części 5. tej książki.

Wymienione w tym podrozdziale właściwości są głównymi elementami odróżniającymi systemy zarządzania bazami danych od tradycyjnego oprogramowania przetwarzającego pliki. W podrozdziale 1.6 omówimy dodatkowe cechy charakterystyczne dla systemów zarządzania bazami danych; najpierw jednak spróbujemy skategoryzować różne typy osób pracujących w środowisku systemów baz danych.

1.4. Aktorzy na scenie

W przypadku niedużej, osobistej bazy danych (np. wspomianej w podrozdziale 1.1 listy adresów) zwykle jedna osoba definiuje, konstruuje i operuje na bazie danych, zatem nie ma potrzeby zapewnienia mechanizmów jej współdzielenia pomiędzy wielu użytkowników. Wiele osób jest jednak zaangażowanych w przedsięwzięcia zupełnie innego typu i innej skali — w projektowanie, wykorzystywanie i konserwację ogromnych baz danych wykorzystywanych przez setki użytkowników. W tym podrozdziale spróbujemy zidentyfikować osoby, których praca wiąże się z codziennym wykorzystywaniem ogromnych baz danych — nazwiemy ich *aktorami na scenie*. W podrozdziale 1.5 przeanalizujemy role osób, które można nazwać *pracownikami poza sceną* — czyli osób, które co prawda pracują nad utrzymaniem środowiska systemu bazy danych, ale nie są aktywnie zainteresowane samą zawartością tej bazy danych.

1.4.1. Administratorzy bazy danych

W każdej organizacji, gdzie wiele osób wykorzystuje te same zasoby, istnieje konieczność zatrudnienia głównego administratora, który będzie te zasoby nadzorował i nimi zarządzał. W środowisku bazy danych głównym zasobem jest oczywiście sama baza danych, drugim zasobem jest natomiast system zarządzania bazą danych lub oprogramowanie pokrewne. Obowiązki związane z administrowaniem tymi zasobami spadają na **administratora bazy danych** (ang. *database administrator* — *DBA*). Osoba na tym stanowisku odpowiada za uwierzytelnianie dostępu do bazy danych, koordynowanie i monitorowanie jej wykorzystania oraz za pozyskiwanie niezbędnych zasobów programowych i sprzętowych. Administrator bazy danych jest także odpowiedzialny za wszystkie problemy związane z administrowanym środowiskiem, w tym ewentualne naruszenia bezpieczeństwa lub zbyt długi czas odpowiedzi. W wielkich organizacjach administratorzy baz danych dysponują zwykle całym sztabem ludzi, którzy pomagają im w realizacji wymienionych zadań.

1.4.2. Projektanci bazy danych

Projektanci bazy danych odpowiadają za identyfikację danych, które docelowo mają być przechowywane w bazie danych, oraz za wybór właściwych struktur, które będą te dane reprezentowały i przechowywały. Większość tych zadań musi być wykonana zanim jeszcze baza danych zostanie faktycznie zaimplementowana i wypełniona danymi. Projektanci baz danych odpowiadają za zapewnienie odpowiedniej komunikacji ze wszystkimi potencjalnymi użytkownikami projektowanych rozwiązań, aby dobrze zrozumieć definiowane przez nich wymagania — tylko w ten sposób mogą zapewnić zgodność efektów swojej pracy z tymi wymaganiami. W wielu przypadkach projektanci należą do zespołu kierowanego przez administratora bazy danych i mogą mieć przydzielane zupełnie inne obszary odpowiedzialności już po zakończeniu procesu projektowania bazy danych. Projektanci bazy danych zazwyczaj uzgadniają oczekiwany kształt opracowywanego rozwiązania z każdą grupą potencjalnych użytkowników i tworzą takie **perspektywy** bazy danych, które w jak największym stopniu odpowiadają zdefiniowanym przez te grupy wymaganiom dotyczącym dostępnych danych i funkcji przetwarzających. Każda perspektywa jest następnie analizowana i *integrowana* z perspektywami opracowanymi dla pozostałych grup przyszłych użytkowników. Ostateczna wersja projektu bazy danych musi być zgodna z wymaganiami zgłoszonymi przez wszystkie takie grupy.

1.4.3. Użytkownicy końcowi

Użytkownicy końcowi to ci pracownicy organizacji, których codzienna praca wymaga dostępu do takich operacji na bazie danych jak przetwarzanie zapytań, aktualizowanie informacji oraz generowanie raportów — baza danych jest więc tworzona przede wszystkim dla nich. Istnieje kilka kategorii, do których można przypisać użytkowników końcowych:

- **Użytkownicy dorywczy** korzystają z bazy danych tylko od czasu do czasu, ale mogą za każdym razem potrzebować innych informacji. Tacy użytkownicy końcowi często używają do definiowania swoich żądań wyszukanych języków zapytań do bazy danych i zwykle pełnią stanowiska menadżerów średniego lub wysokiego szczebla.
- **Naiwni lub parametryczni użytkownicy końcowi** stanowią znaczącą część wszystkich użytkowników bazy danych. Ich praca polega przeważnie na wielokrotnym wykonywaniu zapytań i aktualizowaniu informacji zawartych w bazie danych. Użytkownicy należący do tej grupy wykorzystują zwykle standardowe typy zapytań i operacji aktualizacji, które zostały bardzo uważnie zaprogramowane i przetestowane (są to tzw. **transakcje zapuszkowane** — ang. *canned transactions*). Zadania realizowane przez tego typu użytkowników mogą się od siebie bardzo różnić:

Kasjerzy bankowi mogą sprawdzać salda kont i wykonywać operacje wpłat oraz wypłat.

Pracownicy biur rezerwacji linii lotniczych, hoteli oraz firm wynajmujących samochody sprawdzają dostępność żadanego zasobu i dokonują rezerwacji.

Pracownicy magazynu firmy kurierskiej rejestrują identyfikatory i opisy przyjętych paczek za pomocą kodów kreskowych, aktualizując tym samym centralną bazę danych otrzymanych i transportowanych przesyłek.

- Do grupy **doświadczonych użytkowników końcowych** należą najczęściej inżynierowie, naukowcy, analitycy biznesowi i wszyscy inni pracownicy organizacji, którzy mieli już do czynienia z mechanizmami systemów zarządzania bazami danych oraz stosowali już rozmaite rozwiązania odpowiadające ich skomplikowanym wymaganiom.
- **Samodzielni użytkownicy** utrzymują zwykle osobiste bazy danych w oparciu o gotowe pakiety oprogramowania, które oferują łatwe w użyciu interfejsy, często oparte na wygodnych menu lub na komponentach graficznych. Przykładem może być użytkownik pakietu ułatwiającego zarządzanie podatkami, który przechowuje w swoim oprogramowaniu osobiste dane finansowe.

Typowe systemy zarządzania bazami danych oferują wiele mechanizmów ułatwiających dostęp do obsługiwanych baz danych. Niedoświadczeni użytkownicy końcowi nie muszą wówczas poświęcać zbyt dużo czasu na naukę obsługi udostępnianej w ten sposób funkcjonalności — muszą jedynie zrozumieć, jak należy korzystać z interfejsu użytkownika dla standardowych transakcji, które zostały zaprojektowane i zaimplementowane specjalnie z myślą o nich. Użytkownicy korzystający z bazy danych dorywczo muszą się nauczyć tylko kilku funkcji, które będą być może wielokrotnie stosowali w swojej pracy. Doświadczeni użytkownicy zapewne spróbują poznać większość mechanizmów wykorzystywanego systemu zarządzania bazą danych, aby móc w przyszłości realizować funkcje zdefiniowane w ich zawiłych wymaganiach. Samodzielni użytkownicy zwykle nabierają ogromnej wprawy w wykorzystywaniu konkretnych pakietów oprogramowania.

1.4.4. Analitycy systemowi i programiści aplikacji (inżynierowie oprogramowania)

Analitycy systemowi określają wymagania użytkowników końcowych, szczególnie tych należących do grupy naiwnych lub parametrycznych, i opracowują specyfikacje wielokrotnie wykonywanych transakcji, które powinny w jak największym stopniu odpowiadać tym wymaganiom. **Programiści aplikacji** w pierwszej kolejności implementują w postaci programów specyfikacje otrzymane od analityków systemowych, po czym poddają gotowe rozwiązania testom, przygotowują niezbędną dokumentację i konserwują przygotowane w ten sposób transakcje. Analitycy systemowi i programiści aplikacji (nazywani często **inżynierami oprogramowania**) powinni mieć odpowiednią wiedzę na temat wszystkich tych możliwości oferowanych przez wykorzystywany system zarządzania bazą danych, które mogą się przydać podczas realizacji tego zadania.

1.5. Pracownicy poza sceną

Poza osobami, które projektują, wykorzystują i administrują bazą danych, działania związane z projektowaniem, tworzeniem i operowaniem na *oprogramowaniu i środowisku systemowym SZBD* wymagają udziału także innych osób. Pracownicy należący do tej grupy zwykle nie są zainteresowani samą bazą danych ani zawartymi w niej informacjami — nazywamy ich *pracownikami poza sceną*. Takie osoby można podzielić na trzy główne grupy:

- **Projektanci i programiści systemu zarządzania bazą danych** są autorami projektów i (mających postać pakietów oprogramowania) implementacji modułów i interfejsów systemu zarządzania bazą danych. Systemy zarządzania bazami danych są bardzo skomplikowanymi pakietami oprogramowania, które składają się z wielu **modułów** (komponentów), włącznie z modułami implementującymi katalog, język przetwarzania zapytań, interfejs przetwarzania, mechanizmy dostępu i buforowania, sterowanie współbieżne oraz obsługę odzyskiwania danych i bezpieczeństwa. Każdy system zarządzania bazą danych musi oferować interfejsy dla innych systemów oprogramowania, takich jak systemy operacyjne czy kompilatory różnych języków programowania.
- Do grupy **programistów narzędzi** należą wszystkie osoby projektujące i implementujące **narzędzia**, czyli pakiety oprogramowania, które nie tylko ułatwiają projektowanie i wykorzystywanie systemu bazy danych, ale także pomagają poprawić jego wydajność. Narzędzia są opcjonalnymi pakietami, które w wielu przypadkach można dokupić osobno do już posiadanego systemu zarządzania bazą danych. Takie pakiety mogą służyć łatwiejszemu projektowaniu baz danych, monitorowaniu wydajności, obsłudze interfejsów języka naturalnego lub interfejsów graficznych, modelowaniu, przeprowadzaniu symulacji oraz testowemu generowaniu danych. W wielu przypadkach narzędzia tego typu są opracowywane i wprowadzane na rynek przez niezależnych producentów.
- Kategoria **operatorów i personelu pomocniczego** obejmuje administratorów systemów, którzy odpowiadają za bieżące funkcjonowanie i konserwację środowiska sprzętowego i programowego dla wykorzystywanego systemu bazy danych.

Wymienione powyżej kategorie pracowników poza sceną są co prawda bardzo pomocne podczas udostępniania systemów baz danych użytkownikom końcowym, jednak pracownicy zaliczani do tej grupy raczej nie wykorzystują tych samych baz danych do własnych celów.

1.6. Zalety stosowania rozwiązań opartych na systemach zarządzania bazami danych

W tym podrozdziale omówimy niektóre z zalet stosowania systemów zarządzania bazami danych oraz cechy, które powinien posiadać dobry system tego typu. Prezentowane tutaj możliwości systemów zarządzania bazami danych należy traktować jako elementy uzupełniające cztery podstawowe właściwości omówione w podrozdziale 1.3, zwiększające atrakcyjność tego typu rozwiązań. Administratorzy baz danych wykorzystują taką dodatkową funkcjonalność do osiągania rozmaitych celów związanych z projektowaniem, administrowaniem i wykorzystywaniem ogromnych, wielodostępnych baz danych.

1.6.1. Kontrola nadmiarowości

W tradycyjnych pakietach oprogramowania tworzonych z myślą o przetwarzaniu plików z danymi, każda grupa użytkowników utrzymuje własne pliki obsługiwane przez aplikacje operujące na zawartych w nich danych. Przykładowo, przeanalizujmy raz jeszcze przykład bazy danych UNIWERSYTET z rysunku 1.2; przyjmijmy, że dwie grupy użytkowników mogą stanowić personel dziekanatu oraz działu księgowości. W podejściu tradycyjnym każda z tych grup niezależnie od siebie utrzymywałaby pliki reprezentujące informacje o studentach. Dział księgowości obsługiwałby zarówno ogólne dane o przebiegu studiów, jak i charakterystyczne dla zadań tego działu informacje o przyznanych stypendiach; natomiast dziekanat śledziłby przedmioty, w których poszczególni studenci uczestniczą, oraz dane o uzyskanych ocenach. Duża część wykorzystywanych przez obie grupy danych byłaby więc przechowywana dwukrotnie — w osobnych plikach, z których każdy zawierałby wszystkie informacje przetwarzane przez odpowiednią jednostkę uczelni. Kolejne grupy użytkowników mogłyby dodatkowo powielić niektóre lub wszystkie te dane we własnych plikach.

Taka **nadmiarowość** (redundancja), polegająca na wielokrotnym przechowywaniu tych samych danych, prowadzi do wielu niepotrzebnych problemów. Po pierwsze, każda logiczna aktualizacja reprezentowanych w ten sposób informacji (jak choćby wpisanie danych nowego studenta) musi być przeprowadzana wielokrotnie — przynajmniej raz w każdym pliku, w którym rejestrowane są dane studentów. To zaś prowadzi do *zwielokrotnienia wysiłku*. Po drugie, w sytuacji, gdy te same dane są przechowywane w wielu miejscach, mamy do czynienia z *marnotrawstwem przestrzeni przechowywania*. Po trzecie, pliki reprezentujące te same dane mogą się stać *niespójne*. Może to być efekt np. zastosowania operacji aktualizacji tylko dla niektórych danych, z pominięciem pozostałych. Okazuje się, że nawet jeśli aktualizacja (np. wspomniana operacja dodania nowego studenta) zostanie zastosowana dla wszystkich wymagających tego plików, dane dotyczące danego studenta nadal mogą być narażone na *niespójność* spowodowaną niezależnymi aktualizacjami przeprowadzonymi przez poszczególne grupy użytkowników. Przykładowo, jedna z grup użytkowników może błędnie podać datę urodzenia studenta, natomiast pozostałe grupy mogą wpisać poprawną datę — wartości reprezentujące tę samą informację będą wówczas różne.

Podobne niekorzystne zjawiska nie byłyby możliwe w przypadku zastosowania techniki opartej na bazie danych, która przewiduje tworzenie i integrację perspektyw dla różnych grup użytkowników bazy danych już w fazie jej projektowania. Idealnym rozwiązaniem jest oczywiście posiadanie jednego projektu bazy danych, który uwzględni wszystkie logiczne elementy danych (np. nazwisko i datę urodzenia studenta) umieszczone w *jednym miejscu*. W ten sposób można jednocześnie zapewnić spójność danych oraz oszczędzić przestrzeń przechowywania. W praktyce jednak często konieczne jest zastosowanie techniki **kontrolowanej nadmiarowości**, która poprawi wydajność przetwarzania zapytań. Przykładowo, możemy nadmiarowo przechowywać elementy danych

NazwiskoStudenta i *NumerPrzedmiotu* w pliku RAPORT_OCEN (patrz rysunek 1.5(a)), ponieważ za każdym razem, gdy wskutek wykonania zapytania uzyskamy rekord z tego pliku, będziemy chcieli otrzymać jednocześnie nazwisko studenta i numer (identyfikator) przedmiotu wraz z wystawioną oceną, numerem studenta oraz identyfikatorem kursu. Jeśli umieścimy wszystkie te dane w jednym miejscu, skompletowanie potrzebnych informacji nie będzie wymagało przeszukiwania wielu plików. W takich przypadkach system zarządzania bazą danych powinien jednak zapewniać możliwość *kontrolowania* nadmiarowości, aby zapobiec ewentualnym niespójnościom pomiędzy różnymi plikami. Taki mechanizm może to robić automatycznie przez sprawdzanie, czy pary wartości *NazwiskoStudenta-NumerIndeksu* we wszystkich rekordach pliku RAPORT_OCEN (patrz rysunek 1.5(a)) odpowiadają parom wartości *Nazwisko-NumerIndeksu* w odpowiednich rekordach pliku STUDENT (patrz rysunek 1.2). Podobnie, pary wartości *IdentyfikatorKursu-NumerPrzedmiotu* w poszczególnych rekordach pliku RAPORT_OCEN muszą odpowiadać odpowiednim parom wartości w rekordach pliku KURS. Takie testy można zdefiniować w systemie zarządzania bazą danych już podczas projektowania bazy danych — SZBD będzie wówczas wymuszał odpowiednie działania kontrolne za każdym razem, gdy zostanie podjęta próba zaktualizowania pliku RAPORT_OCEN. Na rysunku 1.5(b) przedstawiono rekord pliku RAPORT_OCEN, który jest niespójny z przedstawioną na rysunku 1.2 zawartością pliku STUDENT — jest to więc przykład sytuacji, w której *nie zastosowano kontroli* nadmiarowości i zapisano w pliku RAPORT_OCEN błędne dane.

| a) | RAPORT_OCEN | NumerIndeksu | NazwiskoStudenta | IdentyfikatorDziału | NumerPrzedmiotu | Ocena |
|----|-------------|--------------|------------------|---------------------|-----------------|-------|
| | | 17 | Nowak | 112 | MAT2410 | 4 |
| | | 17 | Nowak | 119 | INF1310 | 3 |
| | | 8 | Kowalski | 85 | MAT2410 | 5 |
| | | 8 | Kowalski | 92 | INF1310 | 5 |
| | | 8 | Kowalski | 102 | INF3320 | 4 |
| | | 8 | Kowalski | 135 | INF3380 | 5 |

| b) | RAPORT_OCEN | NumerIndeksu | NazwiskoStudenta | IdentyfikatorDziału | NumerPrzedmiotu | Ocena |
|----|-------------|--------------|------------------|---------------------|-----------------|-------|
| | | 17 | Kowalski | 112 | MAT2410 | 4 |

RYSUNEK 1.5. Nadmiarowe przechowywanie elementów danych *NazwiskoStudenta* i *NumerPrzedmiotu* w pliku RAPORT_OCEN. (a) Spójne dane. (b) Niespójny rekord

1.6.2. Ograniczanie możliwości uzyskania nieautoryzowanego dostępu

Kiedy jedną wielką bazę danych współdzielili wielu użytkowników, jest mało prawdopodobne, by większość użytkowników mogła uzyskać autoryzowany dostęp do wszystkich informacji przechowywanych w tej bazie danych. Przykładowo, dane finansowe są często uznawane za poufne i powinny w związku z tym być dostępne tylko dla niektórych użytkowników. Niektórzy użytkownicy mogą dodatkowo uzyskać prawo wyłącznie do odczytywania pewnych danych, inni natomiast mogą dysponować zarówno uprawnieniami odczytu, jak i możliwością aktualizacji. Oznacza to, że typ operacji dostępu (odczytu lub aktualizacji) także musi być odpowiednio kontrolowany. Użytkownicy lub grupy użytkowników mają zwykle przydzielane numery (identyfikatory) kont zabezpieczonych hasłem — dostęp do bazy danych jest wówczas możliwy wyłącznie po podaniu tych informacji

uwierzytelniających. System zarządzania bazą danych powinien oferować **podsystem bezpieczeństwa i autoryzacji**, za pomocą którego administrator bazy danych będzie mógł tworzyć konta użytkowników (lub grup użytkowników) i nakładać na nie odpowiednie ograniczenia. System zarządzania bazą danych powinien następnie automatycznie wymuszać przestrzeganie tych ograniczeń. Warto pamiętać, że podobne elementy zabezpieczające można spotkać w samym oprogramowaniu systemu zarządzania bazą danych. Przykładowo, tylko osoby z uprawnieniami administratora bazy danych mogą uruchamiać pewne **uprzywilejowane elementy tego oprogramowania**, np. moduły przeznaczone do tworzenia nowych kont użytkowników. Podobnie, użytkownicy parametryczni mogą mieć możliwość uzyskiwania dostępu do bazy danych wyłącznie za pośrednictwem transakcji zaprojektowanych specjalnie z myślą o realizowanych przez nich zadaniach.

1.6.3. Zapewnianie miejsca trwałego przechowywania dla obiektów stosowanych w programach

Bazy danych mogą być wykorzystywane do zapewniania **miejsca trwałego przechowywania** dla obiektów i struktur danych stosowanych w programach. Jest to jedna z przyczyn tworzenia **obiektowych systemów baz danych**. Języki programowania wykorzystują zwykle skomplikowane struktury danych, jak typy rekordów w języku Pascal czy definicje klas w językach C++ i Java. Wartości reprezentowane przez zmienne wykorzystywane w tak napisanych programach są porzucane z chwilą zakończenia ich pracy, chyba że programista napisze kod, który wprost będzie je umieszczał w trwałych plikach dyskowych (co zwykle wiąże się z koniecznością konwertowania tych skomplikowanych struktur do odpowiedniego formatu). Kiedy zaistnieje potrzeba ponownego odczytania tak utrwalonych danych, programista musi je przekonwertować z formatu pliku do struktury danych, która może być przetwarzana w jego programie. Obiektowe systemy baz danych są zgodne z takimi językami programowania jak C++ czy Java, a oprogramowanie systemów zarządzania bazami danych może automatycznie wykonywać niezbędne konwersje. Oznacza to, że skomplikowany obiekt w C++ może być trwale przechowywany w obiektowym systemie zarządzania bazą danych. O takiej strukturze mówimy, że jest obiektem **trwałym**, ponieważ zakończenie pracy jego programu nie powoduje jego usunięcia — przeciwnie, taka utrwalona struktura może zostać ponownie odczytana nawet przez inny program napisany w języku C++.

Trwale przechowywanie obiektów i struktur danych stosowanych w programach jest ważnym elementem funkcjonalności systemów baz danych. Tradycyjne systemy tego typu często były narażone na **problem niezgodności impedancji**, ponieważ struktury danych oferowane przez systemy zarządzania bazami danych były niezgodne ze strukturami danych wykorzystywanymi w językach programowania. Obiektowe systemy baz danych oferują zwykle **zgodność** obsługiwanych struktur danych ze strukturami stosowanymi w jednym lub większej ilości obiektowych języków programowania.

1.6.4. Zapewnianie struktur przechowywania dla efektywnego przetwarzania zapytań

Systemy baz danych muszą zapewniać możliwość *efektywnego wykonywania zapytań i operacji aktualizacji danych*. Ponieważ baza danych jest zwykle przechowywana na dysku, system zarządzania bazą danych musi udostępniać wyspecjalizowane struktury danych, które przyspieszą proces przeglądania dysku w poszukiwaniu żądanych rekordów. Do tego celu wykorzystywane są pliki

zewewnętrzne nazywane **indeksami**. Indeksy opierają się zwykle na drzewiastych strukturach danych lub strukturach mieszających — niezależnie od wybranej metody reprezentowania indeksów, każda z tych struktur jest odpowiednio przystosowana do zadania przeszukiwania danych dyskowych. Aby utworzyć rekordy bazy danych żądane przez konkretne zapytanie, należy je najpierw skopiować z dysku do pamięci operacyjnej. Oznacza to, że systemy zarządzania bazami danych często zawierają specjalne moduły **buforujące**, które utrzymują fragmenty baz danych w wydasygnowanych do tego celu buforach w pamięci operacyjnej. Istnieją także przypadki, w których systemy zarządzania bazami danych wykorzystują do buforowania danych dyskowych odpowiednie mechanizmy systemów operacyjnych.

Należący do systemu zarządzania bazą danych moduł **przetwarzania zapytań i optymalizacji** odpowiada za wybór (w oparciu o istniejące struktury przechowywania danych) efektywnego planu wykonania dla każdego zapytania. Wybór rodzaju tworzonych i utrzymywanych indeksów jest częścią procesów *fizycznego projektowania i strojenia bazy danych*, za które zawsze odpowiedzialny jest zespół administratorów bazy danych.

1.6.5. Zapewnianie możliwości tworzenia kopii bezpieczeństwa i odzyskiwania danych

System zarządzania bazą danych musi także zapewniać mechanizmy ułatwiające przywracanie pierwotnego stanu w przypadku ewentualnych awarii sprzętowych lub programowych. Za przywracanie tego stanu odpowiada należący do systemu zarządzania bazą danych specjalny **podsystem tworzenia kopii bezpieczeństwa i odzyskiwania danych**. Przykładowo, jeśli wykorzystywany system komputerowy ulegnie awarii w czasie wykonywania skomplikowanej transakcji aktualizującej dane, podsystem odzyskiwania musi się upewnić, że baza danych zostanie przywrócona do stanu sprzed momentu, w którym rozpoczęło się wykonywanie przerwanej transakcji. Istnieje także rozwiązanie alternatywne — podsystem odzyskiwania może w takim przypadku zapewnić, że wykonywanie przerwanej transakcji zostanie wznowione od punktu, w którym nastąpiła awaria, co zagwarantuje zarejestrowanie w bazie danych pełnego (a więc spójnego) efektu przetwarzania.

1.6.6. Zapewnianie interfejsów dla wielu użytkowników

Ponieważ bazy danych mogą być wykorzystywane przez wiele typów użytkowników, którzy dysponują wiedzą techniczną na bardzo zróżnicowanych poziomach, systemy zarządzania bazami danych powinny udostępniać różne interfejsy użytkownika. Takim interfejsem jest język zapytań wykorzystywany przez użytkowników doraźnych, interfejs języka programowania dla programistów aplikacji, formularze i kody poleceń dla użytkowników parametrycznych oraz interfejsy oparte na menu i języku naturalnym dla samodzielnych użytkowników. Zarówno interfejsy oparte na formularzach, jak i te oparte na menu są często nazywane **graficznymi interfejsami użytkownika** (ang. *Graphical User Interface* — *GUI*). Istnieje wiele wyspecjalizowanych języków i środowisk do definiowania takich graficznych interfejsów użytkownika. Dostępnymi są także narzędzia tworzące interfejsy GUI dla baz danych oparte na stronach WWW.

1.6.7. Reprezentowanie skomplikowanych relacji pomiędzy danymi

Baza danych może zawierać wiele zróżnicowanych danych, które są wzajemnie powiązane na różne sposoby. Przykładowo, przeanalizujmy raz jeszcze bazę danych przedstawioną na rysunku 1.2. Rekord dla studenta o nazwisku *Kowalski* w pliku *STUDENT* jest związany z czterema rekordami w pliku *RAPORT_OCEN*. Podobnie, każdy rekord reprezentujący pojedynczy kurs jest związany nie tylko z pojedynczym rekordem reprezentującym przedmiot, ale także z kilkoma rekordami z pliku *RAPORT_OCEN* — po jednym dla każdego studenta, który zaliczył dany kurs. System zarządzania bazą danych musi oferować możliwość nie tylko samego reprezentowania wraz z danymi rozmaitych skomplikowanych związków, ale także łatwego i efektywnego generowania i aktualizowania powiązanych ze sobą informacji.

1.6.8. Wymuszanie ograniczeń integralnościowych

Większość baz danych ma zdefiniowane pewne **ograniczenia integralnościowe**, które muszą być spełnione przez przechowywane dane. System zarządzania bazą danych powinien oferować możliwość zarówno definiowania takich ograniczeń, jak i wymuszania ich przestrzegania. Najprostszym typem ograniczenia integralnościowego jest wymuszenie określonego typu danych dla poszczególnych elementów danych. Przykładowo, w zaprezentowanej na rysunku 1.2 bazie danych możemy określić, że element danych *RokSt* w każdym z rekordów pliku *STUDENT* musi być liczbą całkowitą z przedziału od 1 do 5, natomiast wartości elementu danych *Nazwisko* w rekordach tego samego pliku muszą być maksymalnie 30-znakowymi ciągami liter alfabetu. Bardziej skomplikowane, popularne typy ograniczeń integralnościowych polegają na określaniu wymaganych związków pomiędzy rekordami w jednym pliku a rekordami w innych plikach. Przykładowo, w bazie danych przedstawionej na rysunku 1.2 możemy określić, że „każdy rekord kursu musi być powiązany z odpowiednim rekordem przedmiotu”. Jeszcze inny rodzaj ograniczeń integralnościowych określa wyjątkowość wartości we wskazanym elemencie danych — takie ograniczenie może mieć postać: „każdy rekord przedmiotu musi mieć unikatową wartość elementu danych *NumerPrzedmiotu*”. Zaprezentowane przed chwilą przykładowe ograniczenia wynikają z **semantyki** (znaczenia) danych oraz reprezentowanego za ich pomocą mini-świata. Za identyfikację ograniczeń integralnościowych w fazie projektowania bazy danych zawsze odpowiada jej projektant. Niektóre ograniczenia mogą być definiowane w systemie zarządzania bazą danych i automatycznie przez ten system wymuszane; inne mogą wymagać przeprowadzania odpowiednich testów na poziomie programów aktualizujących dane lub już w momencie wpisywania informacji.

Element danych może być błędnie wpisany i nadal spełniać warunki wynikające z określonych ograniczeń integralnościowych. Przykładowo, jeśli na skutek błędu osoby wpisującej dane, dla studenta, który uzyskał w rzeczywistości ocenę 5, zostanie zarejestrowana w bazie danych ocena 3, system zarządzania bazą danych *nie będzie mógł* automatycznie wykryć tego błędu, ponieważ ocena 3 jest poprawną wartością dla elementu danych *Ocena*. Podobne błędy wynikające z nieuważnego wpisywania danych do systemu można wykrywać wyłącznie „ręcznie” (np. kiedy student poskarży się, że zarejestrowana ocena jest niezgodna z rzeczywistością) i poprawiać przez odpowiednią aktualizację bazy danych. Okazuje się jednak, że próba wpisania oceny 7 może być automatycznie odrzucana przez system zarządzania bazą danych, ponieważ 7 nie jest poprawną wartością dla elementu danych *Ocena*.

1.6.9. Zezwalanie na wnioskowanie i podejmowanie działań w oparciu o zdefiniowane reguły

Niektóre systemy baz danych oferują możliwość definiowania *reguł wnioskowania* (reguł dedukcyjnych), w oparciu o które można *wywodzić* nowe informacje na podstawie faktów reprezentowanych w bazie danych. Takie systemy są nazywane **dedukcyjnymi systemami baz danych**. Przykładowo, w świecie rzeczywistym (mini-świecie reprezentowanym przez bazę danych) mogą istnieć skomplikowane zasady określające, czy dany student odbywa staż. Można te zasady zdefiniować *deklaratywnie* w postaci **reguł**, które będą kompilowane i utrzymywane przez system zarządzania bazą danych, i które będą umożliwiały wyszukiwanie wszystkich studentów odbywających staż. W tradycyjnych systemach zarządzania bazami danych do obsługi tego typu zastosowań konieczny byłby odpowiedni *kod programu proceduralnego*. Jeśli jednak reguły obowiązujące w reprezentowanym mini-świecie ulegną zmianie, generalnie znacznie łatwiejszym zadaniem (od ponownego kodowania programów proceduralnych) jest zmiana zadeklarowanych wcześniej reguł wnioskowania. Mechanizmy zapewniające jeszcze większe możliwości są oferowane przez **aktywne systemy baz danych**, gdzie w przypadku wystąpienia określonych zdarzeń lub spełnienia określonych warunków aktywne reguły mogą automatycznie inicjować zdefiniowane wcześniej działania.

1.6.10. Dodatkowe własności wynikające ze stosowania rozwiązań opartych na bazach danych

W tym podrozdziale omówimy niektóre dodatkowe korzyści wynikające ze stosowania rozwiązań opartych na bazach danych i dotyczące większości organizacji.

Możliwość wymuszania stosowania standardów. Rozwiązania oparte na bazach danych umożliwiają administratorom baz danych wymuszanie na użytkownikach stosowania standardów przyjętych w danej organizacji. Standardy ułatwiają wzajemną komunikację i współpracę pomiędzy różnymi działami organizacji oraz pomiędzy użytkownikami przydzielonymi do różnych projektów. Standardy można definiować zarówno dla nazw i formatów elementów danych, jak i dla formatów wyświetlania, struktury raportów, terminologii itp. Administrator bazy danych może łatwiej wymuszać stosowanie standardów w środowisku scentralizowanej bazy danych niż w środowisku, w którym każda grupa użytkowników sama kontroluje swoje oprogramowanie i przetwarzane pliki.

Skrócony czas wytwarzania aplikacji. Szybkość wytwarzania nowych aplikacji (choćby takich, które pobierają z bazy danych pewne dane i drukują na ich podstawie nowy raport) jest zwykle wykorzystywana w roli kluczowego hasła marketingowego promującego wszelkie rozwiązania oparte na bazach danych. Projektowanie i implementowanie od zera nowych baz danych może w wielu przypadkach wymagać więcej czasu niż pisanie wyspecjalizowanych aplikacji operujących na plikach. Kiedy jednak taka baza danych będzie już gotowa, tworzenie nowych aplikacji w oparciu o odpowiednie mechanizmy systemu zarządzania bazą danych wymaga generalnie minimalnych nakładów czasowych. Tworzenie aplikacji z wykorzystaniem tych mechanizmów wymaga w przybliżeniu od jednej szóstej do jednej czwartej czasu, który zużylibyśmy na opracowanie odpowiedniego programu pracującego w tradycyjnym systemie plików.

Elastyczność. Każda ewentualna zmiana wymagań może się wiązać z koniecznością zmodyfikowania struktury bazy danych. Przykładowo, nowa grupa użytkowników może zgłosić konieczność przetwarzania informacji, które nie są przechowywane w bazie danych w obecnym kształcie. W efekcie, może zaistnieć potrzeba dodania do bazy danych nowego pliku lub rozszerzenia elementów danych w pliku już istniejącym. Współczesne systemy zarządzania bazami danych oferują na szczęście pewne możliwości w zakresie wprowadzania ewolucyjnych zmian do struktury bazy danych bez konieczności modyfikowania już przechowywanych danych ani istniejących aplikacji.

Dostępność aktualnych informacji. System zarządzania bazą danych udostępnia tę bazę wszystkim użytkownikom, którzy tylko dysponują odpowiednimi uprawnieniami. Oznacza to, że w momencie zastosowania w udostępnianej bazie danych operacji aktualizacji wykonanej przez jednego użytkownika, wszyscy pozostali użytkownicy mogą natychmiast zobaczyć efekt wykonania tej operacji. Dostępność najbardziej aktualnych informacji ma zasadnicze znaczenie w wielu aplikacjach przetwarzających transakcje (takich jak systemy rezerwacji czy bankowe bazy danych) — jest to możliwe dzięki mechanizmom sterowania współbieżnego i podsystemowi odzyskiwania danych systemu zarządzania bazą danych.

Korzyści ekonomiczne. Rozwiązania oparte na systemach sterowania bazami danych umożliwiają konsolidowanie danych i aplikacji celem ograniczenia niepotrzebnego pokrywania się działań personelu przetwarzającego dane przydzielonego do różnych projektów lub pracującego w różnych działach organizacji. Dzięki temu cała organizacja może inwestować większe środki w nowoczesne procesory, układy pamięci czy systemy komunikacji, zamiast pozwalać poszczególnym działom na dokonywanie samodzielnych zakupów (często gorszego) wyposażenia. Można w ten sposób zredukować łączne koszty operacyjne i zarządzania.

1.7. Krótka historia praktycznych zastosowań baz danych

Przedstawimy teraz krótki przegląd historii zastosowań systemów zarządzania bazami danych oraz ich wpływu na dynamikę rozwoju nowych rodzajów systemów baz danych.

1.7.1. Wczesne zastosowania baz danych oparte na systemach hierarchicznych i sieciowych

Wiele wczesnych rozwiązań opartych na bazach danych polegało na przechowywaniu rekordów utrzymywanych przez ogromne organizacje, takie jak korporacje, uniwersytety, szpitale czy banki. W wielu z tych rozwiązań trzeba było reprezentować wielkie liczby rekordów o podobnej strukturze. Przykładowo, w oprogramowaniu stosowanym przez uniwersytety podobne informacje były przechowywane dla każdego studenta, każdego przedmiotu, każdej oceny itp. Takie rozwiązania wiązały się z koniecznością utrzymywania wielu typów rekordów oraz obsługi wielu istniejących pomiędzy nimi relacji.

Jednym z głównych problemów występujących we wczesnych systemach baz danych było mieszanie relacji koncepcyjnych z fizycznym modelem przechowywania i rozmieszczania rekordów na dysku. Przykładowo, rekordy reprezentujące oceny uzyskane przez konkretnego studenta mogły być fizycznie przechowywane bezpośrednio obok odpowiedniego rekordu reprezentującego samego

studenta. Takie rozwiązanie zapewniałoby co prawda bardzo efektywny dostęp do oryginalnych zapytań i transakcji, których obsługa stanowiła jeden z celów projektowania całej bazy danych, jednak nie gwarantowałoby wystarczającej elastyczności w zakresie efektywności dostępu do danych za pomocą nowych zapytań i transakcji. Szczególnie trudna byłaby efektywna implementacja nowych zapytań, które wymagałyby do sprawnego przetwarzania danych zupełnie innej organizacji fizycznego przechowywania informacji. Okazuje się, że dosyć trudne byłoby także zreorganizowanie bazy danych w sytuacji, gdyby pojawiły się jakieś zmiany w oryginalnych wymaganiach dotyczących danego rozwiązania.

Innym niedociągnięciem wczesnych systemów baz danych było udostępnianie przez nie wyłącznie interfejsów dla języków programowania. Takie rozwiązanie niepotrzebnie wydłużało czas i zwiększało koszty implementowania nowych zapytań i transakcji, ponieważ wymagało pisania i testowania nowych programów. Większość tego typu systemów baz danych była implementowana na dużych i drogich komputerach centralnych (działa się tak począwszy od połowy lat 60-tych aż do końca lat 80-tych). Najbardziej popularne odmiany wczesnych systemów baz danych były konstruowane zgodnie z trzema paradygmatami — były to odpowiednio systemy hierarchiczne, systemy oparte na modelu sieciowym oraz odwrócone systemy plików.

1.7.2. Zapewnianie elastyczności w rozwiązaniach opartych na relacyjnych bazach danych

Początkowo, relacyjne bazy danych miały stanowić rozwiązanie, które pozwoli oddzielić mechanizmy fizycznego przechowywania danych od ich koncepcyjnej reprezentacji i jednocześnie wniesie odpowiedni model matematyczny dla technologii baz danych. Relacyjny model danych wprowadził także wysokopoziomowe języki zapytań, które stanowiły pierwszą ciekawą i efektywną alternatywę dla interfejsów języków programowania — ich wprowadzenie pozwoliło oczywiście na znaczne przyspieszenie procesu tworzenia nowych zapytań. Przykładem relacyjnej reprezentacji danych jest struktura bazy danych przedstawiona na rysunku 1.2. Systemy relacyjne były początkowo przeznaczone do tych samych zastosowań co wspomniane w poprzednim podrozdziale wczesne rozwiązania oparte na bazach danych, jednak szybko okazało się, że oferują nieporównanie większą elastyczność w obszarze szybkiego tworzenia nowych zapytań oraz reorganizacji struktury bazy danych (np. w odpowiedzi na zmiany pierwotnych wymagań).

Wczesne eksperymentalne systemy relacyjne były tworzone i rozwijane już w późnych latach 70-tych, natomiast komercyjne relacyjne systemy zarządzania bazami danych (ang. *Relational Database Management System* — *RDBMS*) zostały wprowadzone we wczesnych latach 80-tych — pierwsze rozwiązania tego typu były mało wydajne, ponieważ podczas operacji dostępu do rekordów wzajemnie powiązanych za pomocą relacji nie wykorzystywano jeszcze wskaźników do fizycznych obszarów pamięci. Wraz z rozwojem nowych technologii przechowywania danych, nowych technik indeksowania informacji, a także lepszych metod przetwarzania i optymalizacji zapytań, wydajność relacyjnych systemów zarządzania bazami danych stopniowo wzrastała. Relacyjne bazy danych stały się ostatecznie dominującym rodzajem systemów baz danych, przynajmniej w tradycyjnych zastosowaniach. Relacyjne bazy danych są obecnie dostępne dla niemal wszystkich typów komputerów, od niewielkich komputerów osobistych do dużych serwerów.

1.7.3. Aplikacje obiektowe i konieczność wprowadzenia nowych, bardziej skomplikowanych baz danych

Rozwój obiektowych języków programowania, który nastąpił w latach 80-tych, oraz potrzeba przechowywania i udostępniania obiektów o skomplikowanej strukturze doprowadziły ostatecznie do rozwoju obiektowych baz danych. Obiektowe bazy danych były początkowo uważane za rozwiązanie konkurencyjne względem relacyjnych baz danych, ponieważ oferowały bardziej ogólne struktury danych. Twórcy tego typu baz danych zastosowali także wiele przydatnych reguł znanych ze świata obiektowych języków programowania, a więc abstrakcyjne typy danych, hermetyzację operacji, dziedziczenie czy mechanizm identyfikacji obiektów. Poziom złożoności tego modelu i brak wcześniejszych standardów przyczynił się jednak do bardzo ograniczonej popularności tego typu rozwiązań. Obiektowe bazy danych są obecnie wykorzystywane do bardzo wyspecjalizowanych celów, takich jak projektowanie inżynierskie, publikowanie multimediów czy obsługa systemów produkcji przemysłowej.

1.7.4. Wykorzystywana w handlu elektronicznym wymiana danych za pośrednictwem stron WWW

Internet jest ogromną siecią połączonych ze sobą komputerów. Użytkownicy internetu mogą tworzyć dokumenty za pomocą specjalnych języków publikacji w sieci WWW, takich jak HTML (od ang. *HyperText Markup Language*), i umieszczać tak przygotowane materiały na serwerach WWW, gdzie pozostali użytkownicy (klienci) mogą uzyskiwać do nich dostęp. Dokumenty można ze sobą łączyć za pomocą tzw. **hiperłączy**, które są w rzeczywistości logicznymi wskaźnikami do innych dokumentów. W latach 90-tych ogromną popularność zdobyło zupełnie nowe zastosowanie internetu — handel elektroniczny (ang. *electronic commerce* lub *e-commerce*). Szybko okazało się, że część informacji wyświetlanych na stronach WWW sklepów internetowych może być dynamicznie pobierana z działającego w tle systemu zarządzania bazą danych. W odpowiedzi na rosnące zapotrzebowanie na rozwiązania tego typu opracowano wiele technik wymiany danych za pośrednictwem stron WWW. Za główny standard wymiany danych pomiędzy różnymi typami baz danych i stron internetowych uważa się obecnie język *XML* (ang. *eXtended Markup Language*). Język XML łączy w sobie pojęcia typowe dla systemów opartych na dokumentach z pojęciami charakterystycznymi dla modelowania baz danych.

1.7.5. Rozszerzanie możliwości współczesnych systemów baz danych z myślą o nowych zastosowaniach

Sukces systemów baz danych w tradycyjnych zastosowaniach zachęcił programistów do opracowania zupełnie innych rodzajów aplikacji, w których z powodzeniem będzie można wykorzystywać podobne technologie. Nowe rozwiązania miały być wykorzystywane w obszarach, w których do tej pory dominowały tradycyjne aplikacje operujące na wyspecjalizowanych plikach i strukturach danych. Poniżej przedstawiono przykłady takich obszarów:

- Rozwiązania **naukowe**, które wymagają przechowywania ogromnych ilości danych generowanych w trakcie eksperymentów naukowych w takich obszarach jak fizyka cząstek elementarnych czy odwzorowywanie ludzkiego materiału genetycznego.
- Przechowywanie i udostępnianie **obrazów**, począwszy od zeskanowanych artykułów prasowych i rodzinnych fotografii, a skończywszy na zdjęciach satelitarnych i obrazach powstających podczas takich procedur medycznych jak prześwietlenia rentgenowskie czy rezonans magnetyczny.
- Przechowywanie i udostępnianie obrazu **wideo**, np. filmów czy **klipów wideo** z wiadomościami telewizyjnymi lub zapisem z osobistych kamer cyfrowych.
- Rozwiązania w zakresie **drażenia danych**, w których ogromne ilości danych są poddawane analizie pod kątem wystąpienia konkretnych wzorców lub relacji.
- Techniki **przestrzenne**, w których przechowuje się przestrzenne lokalizacje takich danych jak informacje o pogodzie czy mapy wykorzystywane w systemach informacji geograficznej.
- Zastosowania związane z przechowywaniem **szeregów czasowych**, w tym danych ekonomicznych odpowiadających równomiernie rozłożonym punktom w czasie; mogą to być np. wykresy dziennej sprzedaży lub miesięcznego produktu krajowego brutto.

Bardzo szybko okazało się, że podstawowe relacyjne systemy baz danych nie nadają się do obsługi wielu z wymienionych przed chwilą zastosowań — zwykle powodem takiego stanu rzeczy jest jeden z poniższych problemów:

- Do modelowania istniejącego stanu mini-świata potrzebne są bardziej skomplikowane struktury danych niż prosta reprezentacja relacyjna.
- Poza podstawowymi typami numerycznymi i znakowymi niezbędne są nowe typy danych.
- Do operowania na nowych typach danych potrzebne są zupełnie nowe operacje i konstrukcje języka zapytań.
- Niezbędne są nowe struktury przechowywania i indeksowania danych.

Wymienione powyżej ograniczenia spowodowały, że twórcy systemów zarządzania bazami danych zaczęli wprowadzać do swoich pakietów dodatkowe elementy zwiększające ich funkcjonalność. Zastosowania niektórych z nowych funkcji są uniwersalne — dotyczy to np. przenoszenia pojęć z obiektowych baz danych do systemów relacyjnych. Inne elementy nowej funkcjonalności mają bardzo konkretne przeznaczenie i są zwykle oferowane w postaci opcjonalnych modułów, które można stosować tylko podczas realizacji specyficznych zadań. Przykładowo, użytkownicy mogą zakupić moduł obsługujący szeregi czasowe, który będą następnie wykorzystywali w swoim relacyjnym systemie zarządzania bazą danych do obsługi takich szeregów.

1.8. Kiedy nie należy używać systemów zarządzania bazami danych

Pomimo wymienionych w tym rozdziale zalet systemów zarządzania bazami danych, istnieje kilka sytuacji, w których stosowanie tego typu rozwiązań wiąże się z niepotrzebnymi kosztami, których z kolei można uniknąć stosując tradycyjne rozwiązania przetwarzające pliki. Nadmierne koszty stosowania systemów zarządzania bazami danych mogą wynikać z następujących uwarunkowań:

- Zbyt wysoki początkowy koszt inwestycji w sprzęt, oprogramowanie i szkolenia.
- Zbyt mały poziom szczegółowości oferowanej przez system zarządzania bazą danych w zakresie definiowania i przetwarzania danych.
- Negatywne skutki dla wydajności, będące efektem automatycznego stosowania mechanizmów bezpieczeństwa, sterowania współbieżnego, odzyskiwania i zapewniania spójności danych.

Dodatkowe problemy mogą wystąpić także w sytuacji, gdy programiści lub administratorzy baz danych popełnią błędy podczas projektowania, lub gdy aplikacje systemu bazy danych nie zostaną właściwie zaimplementowane. Zatem zastosowanie tradycyjnych rozwiązań opartych na przetwarzaniu zwykłych plików z danymi może być usprawiedliwione w następujących okolicznościach:

- Sama baza danych i aplikacje dodatkowe są proste, dobrze zdefiniowane i nie będą w przyszłości zmieniane.
- Zostały zdefiniowane ściśle wymagania dotyczące projektowanego pakietu oprogramowania, których nie można spełnić przy użyciu systemu zarządzania bazą danych (np. z powodu mniejszej wydajności).
- Reprezentowane dane nie muszą być dostępne dla wielu użytkowników.

1.9. Podsumowanie

W tym rozdziale zdefiniowaliśmy bazę danych jako zbiór wzajemnie powiązanych danych (informacji), gdzie przez same *dane* rozumiemy zarejestrowane fakty ze świata rzeczywistego. Typowa baza danych reprezentuje jakiś aspekt tego świata i jest wykorzystywana do określonych celów przez jedną lub więcej grup użytkowników. System zarządzania bazą danych jest uniwersalnym pakietem oprogramowania, który umożliwia implementowanie i utrzymywanie skomputeryzowanej bazy danych. Baza danych w połączeniu z tym oprogramowaniem tworzy system bazy danych. Wymieniliśmy w tym rozdziale szereg właściwości, które odróżniają rozwiązania oparte na bazach danych od tradycyjnych aplikacji przetwarzających pliki z danymi. W dalszej kolejności omówiliśmy najważniejsze kategorie użytkowników baz danych — nazwaliśmy ich *aktorami na scenie*. Stwierdziliśmy także, że poza wymienionymi rodzajami użytkowników baz danych istnieje także kilka kategorii personelu wspierającego tych użytkowników podczas ich pracy w środowisku bazy danych — nazwaliśmy ich *pracownikami poza sceną*.

Przedstawiliśmy także listę możliwości i funkcji, które powinny być dostępne na poziomie systemów zarządzania bazami danych dla administratorów, projektantów oraz użytkowników baz danych, i które powinny im ułatwiać projektowanie, administrowanie i wykorzystywanie tychże baz danych. Zaraz potem krótko omówiliśmy historię i ewolucję rozwiązań opartych na bazach danych. Wreszcie skupiliśmy się przez chwilę na kosztach wynikających ze stosowania systemów zarządzania bazami danych oraz przedstawiliśmy sytuacje, w których wykorzystywanie tego typu rozwiązań może powodować więcej strat niż korzyści.

Pytania powtórkowe

- 1.1. Zdefiniuj następujące pojęcia: *dane, baza danych, system zarządzania bazą danych, system bazy danych, katalog bazy danych, niezależność programu od danych, perspektywa użytkownika, administrator bazy danych, użytkownik końcowy, transakcja zapuszkowana, dedukcyjny system bazy danych, trwałe obiekty, metadane* oraz *aplikacja przetwarzająca transakcje*.
- 1.2. Jakie trzy główne typy działań wiążą się ze stosowaniem baz danych? Krótko omów każdy z wymienionych typów.
- 1.3. Omów główne właściwości rozwiązań opartych na bazach danych i przedstaw krótko różnice dzielące te rozwiązania od ich odpowiedników mających postać tradycyjnych systemów plików.
- 1.4. Jakie są zadania administratora bazy danych i projektanta bazy danych?
- 1.5. Wymień różne typy użytkowników końcowych bazy danych. Omów główne czynności podejmowane przez użytkowników należących do poszczególnych typów.
- 1.6. Przedstaw możliwości, jakie powinny być oferowane przez systemy zarządzania bazami danych.

Ćwiczenia

- 1.7. Przedstaw kilka nieformalnych zapytań i operacji aktualizujących, które chciałbyś zastosować dla bazy danych przedstawionej na rysunku 1.2.
- 1.8. Jaka jest różnica pomiędzy kontrolowaną a niekontrolowaną nadmiarowością? Odpowiedź zilustruj odpowiednimi przykładami.
- 1.9. Wymień wszystkie związki łączące rekordy bazy danych przedstawionej na rysunku 1.2.
- 1.10. Podaj kilka dodatkowych perspektyw, które mogą być przydatne dla innych grup użytkowników bazy danych zademonstrowanej na rysunku 1.2.
- 1.11. Przedstaw kilka przykładów ograniczeń integralnościowych, które Twoim zdaniem powinny zostać zdefiniowane dla bazy danych zaprezentowanej na rysunku 1.2.

Wybrane publikacje

Wydanie miesięcznika *Communications of the ACM* z października roku 1991 oraz książka Kima (1995) zawierają wiele artykułów i materiałów opisujących systemy zarządzania bazami danych następnej generacji; wiele spośród rozwiązań omówionych w tym pierwszym czasopiśmie jest już dostępnych w oferowanych obecnie komercyjnych pakietach oprogramowania. Wydanie miesięcznika *ACM Computing Surveys* z marca roku 1976 zawiera wczesne wprowadzenie do systemów baz danych, które dla zainteresowanych czytelników może stanowić ciekawy materiał historyczny na ten temat.