https://doi.org/10.18778/8142-629-9.03

ROZDZIAŁ 2

Bazy danych

2.1. Wprowadzenie

Trudno przecenić znaczenie baz danych we współczesnym świecie. O ile w czasach, kiedy to pojęcie powstało (początek lat 70.), nawet wśród informatyków było ono słabo rozpoznawalne, o tyle dziś znane jest wszystkim, którzy funkcjonują w otoczeniu cyfrowym chociażby za sprawą korzystania z telefonu komórkowego. Z aplikacji baz danych korzysta wielu użytkowników zarówno w pracy, w domu, jak i w podróży. Nawet nie korzystając z komputera ani z komórki, otrzymujemy papierowe wydruki związane z używaniem mediów, trzymamy pieniądze w banku, dostajemy i wysyłamy przesyłki, otrzymujemy wynagrodzenia i płacimy podatki – a z każdą z wymienionych aktywności związane jest wykorzystanie baz danych, w których przechowywane są nasze dane osobowe, dane o naszych domach, samochodach, transakcjach czy zakupach. Tak więc można powiedzieć, że bazy danych we współczesnym świecie są wszechobecne jak elektryczność. Ich zastosowanie jest powszechne, a w niektórych dziedzinach są one niezastąpione, szczególnie tam, gdzie przechowywane są wielkie zbiory danych, na przykład w centralnych systemach, takich jak PESEL czy CEPIK.

Bazy danych stanowią podstawę systemów informatycznych zarządzania. To w nich przechowywane są wszelkie dane o osobach, obiektach, transakcjach, zdarzeniach w wymiarze ilościowym i wartościowym. W bazach danych przechowywane są zarówno niewyobrażalnie wielkie ilości danych w systemach społecznościowych typu Facebook, wyszukiwarkach, jak Google, zasobach multimedialnych muzycznych, filmowych czy fotograficznych, ale także w małych systemach, a właściwie „programach", przeznaczonych do obsługi małych sklepów czy punktów usługowych. Działalność takich organizacji jak giełda, uczelnia, bank, supermarket czy sklep internetowy jest praktycznie niemożliwa bez korzystania z bazy danych jako systemu, w którym

przechowywane są dane. Właściwie trudno znaleźć dziedzinę działalności, w której wykorzystuje się komputery, a w której nie miałyby zastosowania bazy danych.

W okresie od powstania pierwszych koncepcji baz danych obserwujemy gwałtowny rozwój teorii baz danych, przy czym nauce o bazach danych przypisuje się niezwykle wysoką efektywność przejawiającą się w szybkiej implementacji w praktycznych rozwiązaniach wszelkich nowych koncepcji. Jest to także ogromny, wielomiliardowy rynek, dowodem czego może być chociażby bardzo wysoka pozycja pod względem wartości sprzedaży firmy Oracle dominującej na rynku baz danych (ok. 50% rynku).

Celem tego rozdziału jest zaprezentowanie głównych zagadnień teorii baz danych, w szczególności związanych z projektowaniem bazy danych, oraz poglądowych przykładów wykorzystania języka SQL do tworzenia bazy danych i formułowania zapytań. W końcowej części pracy zamieszczono przypadek ilustrujący proces projektowania prostej bazy danych zarówno na etapie projektu konceptualnego, jak i jego translacji do modelu relacyjnego. Ostatni fragment zawiera przykładowe zapytania SELECT do tak utworzonej bazy danych.

2.2. Podstawowe pojęcia baz danych

Pojęcie „baza danych" jest obecnie dość często używane nie tylko przez informatyków, ale także w przestrzeni publicznej. Potoczne jego rozumienie jako zbioru danych jest poprawne i wystarczające dla potrzeb komunikacji społecznej, jednak literatura przedmiotu oferuje bardziej precyzyjną definicję. I rzeczywiście, definicję bazy danych można znaleźć niemal w każdym podręczniku, ale są one bardzo zróżnicowane. Najprościej bazę danych definiują Elmasri i Navathe [Elmasri, 2005] jako „zbiór powiązanych ze sobą danych". Znacznie obszerniejszą definicję podają Ullman i Widom [Ulman, 2000] jako „zbiór danych o określonej strukturze zapisany na zewnętrznym nośniku pamięci komputera, mogący zaspokoić potrzeby wielu użytkowników korzystających z niego w sposób selektywny w dogodnym dla siebie czasie". Dziś można byłoby jeszcze dodać „… i miejscu". Do naszych celów wystarczające będzie rozumienie bazy danych jako zbioru danych zapisanego „w komputerze", posiadającego dokładnie zdefiniowaną strukturę i obsługiwanego za pomocą wysoce skomplikowanego

oprogramowania określanego jako System Zarządzania Bazą Danych (SZBD). W literaturze przedmiotu, także polskiej, używana bywa oryginalna angielskojęzyczna nazwa *Database Management System* (DBMS). Nie namawiamy do używania jej w tekstach czy wypowiedziach w języku polskim, ale warto ją znać.

W mowie potocznej „bazą danych" czasem nazywa się komercyjne oprogramowanie do obsługi baz danych, na przykład mówimy, że w szkole pracowaliśmy z bazą danych Access albo że za najlepszą bazę danych uważany jest Oracle. Chcąc jednak zachować precyzję, Access czy Oracle określimy jako właśnie Systemy Zarządzania Bazą Danych.

Dla użytkownika istotne są przede wszystkim dane zawarte w bazie danych, użyteczne z jego punktu widzenia, na przykład dane o wyrobach, klientach czy dokumentach zakupu i sprzedaży. Oprócz tych danych, wykorzystywanych bezpośrednio przez użytkowników, baza danych zawiera szereg innych treści istotnych dla funkcjonowania systemu bazodanowego, z którymi użytkownicy końcowi mają do czynienia tylko pośrednio, korzystając z ich udogodnień czy zderzając się z ograniczeniami.

Te treści w bazie danych, niebędące danymi – to metadane, zwane też katalogiem systemowym. Obejmują one następujące kategorie:

- dane o strukturze bazy danych,
- dane o wymogach integralności,
- definicje tabel wirtualnych,
- indeksy,
- dane o użytkownikach i ich uprawnieniach,
- programy, tzw. procedury wbudowane,
- dane o procesach przetwarzania, dziennik (LOG),
- kopie danych.

Dane o strukturze bazy danych

Uprzedzając dokładniejsze definicje, baza danych to zbiór tabel. Każda tabela posiada określoną strukturę: składa się z pewnej liczby kolumn, każda kolumna ma nazwę, definicję typu danych, często rozmiar. Wszystkie te własności tabel są zdefiniowane precyzyjnie w metadanych.

Dane o wymogach integralności

Dla każdej tabeli oprócz struktury definiowane są różne wymogi poprawności dotyczące danych, zwane wymogami integralności. Na przykład nie można dopuścić, aby w tabeli Pracownicy znalazły się dwie osoby posiadające ten sam numer PESEL. Szczegółowy opis rodzajów wymogów integralności znajduje się w dalszej części tekstu.

Definicje tabel wirtualnych

Na podstawie oryginalnych tabel fizycznych można tworzyć różne tzw. tabele wirtualne. Na przykład na podstawie tabeli Zawodnicy można utworzyć tabele wirtualne Seniorzy i Juniorzy, dalej Dziewczęta i Chłopcy itp. Tabele wirtualne sporządza się także ze względów technologicznych na etapie programowania aplikacji bazodanowych, czyli programów użytkowych korzystających z baz danych.

Indeksy

Indeksy w bazie danych można porównać do katalogów w bibliotece. Umożliwiają one szybkie wyszukiwanie danych oraz równie szybką prezentację w sekwencji klucza indeksu, czyli zgodnie z kolejnością wartości klucza, na przykład według nazwiska czy ceny. Indeksy mogą być tworzone automatycznie przez system lub na żądanie użytkownika według jego specyfikacji.

Dane o użytkownikach i ich uprawnieniach

Każdy użytkownik bazy danych musi posiadać założone w bazie konto. Oprócz standardowych danych (nazwy i hasła) użytkownicy muszą mieć zdefiniowane uprawnienia systemowe (co mogą robić) oraz obiektowe (jakich obiektów i w jaki sposób mogą używać).

Programy, tzw. procedury wbudowane

W bazie danych, szczególnie w wersjach zgodnych z SQL99, mogą być zapisywane moduły programowe, takie jak procedury, funkcje czy wyzwalacze. Są to moduły napisane w języku SQL, które realizują różne

obliczenia, znajdują obiekty spełniające określone warunki, tworzą zestawienia według zadanych parametrów itp. Niektóre z nich (wyzwalacze) są uruchamiane automatycznie w momencie wystąpienia określonego zdarzenia, inne są wywoływane jawnie przez inne programy.

Dane o procesach przetwarzania, dziennik (LOG)

Dziennik jest kluczowym dla bezpieczeństwa bazy danych elementem. Zapisywane są w nim wszystkie operacje wykonywane w bazie danych. W przypadku awarii dziennik wykorzystywany jest przez menedżera rekonstrukcji do automatycznego podniesienia (przywrócenia spójności) bazy danych. Jest to jedyny element bazy danych, który zaleca się zapisywać w dwóch egzemplarzach na oddzielnych dyskach.

Kopie danych

Dla zabezpieczenia bazy danych przed utratą danych oraz z powodów formalnych (przepisy prawa nakazujące przechowywanie danych przez określony czas) sporządza się kopie zapasowe oraz kopie archiwalne. O ile kopie archiwalne sporządza się raczej na nośnikach zewnętrznych (poza bazą danych), o tyle kopie zapasowe tworzone są również w samej bazie danych. Mogą one być ustanawiane automatycznie oraz na żądanie.

Szerszym pojęciem jest „System baz danych". Obejmuje ono wszystkie składniki sprzętowe, programowe i kadrowe tworzące złożony system pozwalający tworzyć i eksploatować bazę danych. Na system baz danych składają się:

- baza danych, czyli dane i metadane;
- System Zarządzania Bazą Danych, czyli pakiet oprogramowania umożliwiający tworzenie i eksploatację bazy danych;
- sprzęt informatyczny wraz z niezbędnym oprogramowaniem systemowym (system operacyjny, oprogramowanie sieciowe, antywirusowe itp.):
  - serwer bazy danych, czyli wyróżniony komputer, na którym zainstalowany jest system zarządzania bazą danych i sama baza danych,
  - inne serwery, np. serwer WWW umożliwiający pracę z bazą danych w sieci za pomocą przeglądarki,
  - serwery aplikacji, na których zainstalowane są aplikacje bazodanowe; rozwiązanie to stosowane jest w dużych systemach;

- klienty [to nie jest błąd!], czyli komputery użytkowników; serwery nigdy nie są wykorzystywane przez użytkowników do bieżącej pracy. Zwykle znajdują się one w wyizolowanym pomieszczeniu zwanym serwerownią, posiadającą wydzielone zasilanie i klimatyzację. Są one niedostępne dla użytkowników poza administratorami. Większość użytkowników zwykle nawet nie wie, gdzie znajduje się serwer wykorzystywanej przez nich bazy danych;
- sieć komputerowa, na którą składają się, poza serwerami i klientami, wszystkie inne elementy sprzętowe i programowe, takie jak okablowanie, specjalistyczne urządzenia sieciowe (routery, koncentratory, huby, firewalle itp.) oraz niezbędne oprogramowanie sieciowe;
- narzędzia wspomagające, na przykład używane do projektowania baz danych, generatory raportów, a także bardziej rozbudowane narzędzia programistyczne typu CASE (*Computer Aided System Engineering*);
- aplikacje, czyli oprogramowanie przeznaczone do określonych celów użytkowych. W obszarze zarządzania są to takie systemy, jak system sprzedaży, zaopatrzenia, kadrowo-płacowy czy finansowo-księgowy. Często tworzą one bardzo rozbudowane systemy zintegrowane zawierające wszystkie niezbędne podsystemy;
- użytkownicy, czyli osoby pracujące w taki czy inny sposób z bazą danych bezpośrednio lub – najczęściej – za pośrednictwem aplikacji. W zasadzie dziś wszyscy korzystający z komputerów, a nawet tylko ze smartfonów, są użytkownikami baz danych, chociaż nie muszą być tego świadomi. W literaturze przedmiotu czasem pod pojęciem użytkowników rozumie się jedynie użytkowników końcowych; część autorów definiuje szerszą kategorię użytkowników, zaliczając do nich także informatyków i osoby korzystające z bazy danych incydentalnie.

2.3. Użytkownicy baz danych

R. Ramakrishnan [2003] dzieli użytkowników na trzy kategorie:
- administratorów,
- użytkowników profesjonalnych,
- użytkowników końcowych.

Administratorzy baz danych są wysoko kwalifikowanymi informatykami odpowiedzialnymi za instalację i poprawne funkcjonowanie systemu zarządzania bazą danych. W szczególności do zadań administratorów należą:

- instalacja SZBD:
  - instalacja serwera i jego bieżąca konserwacja,
  - aktualizacja oprogramowania,
  - instalacja oprogramowania klientów na komputerach użytkowników,
  - instalacja narzędzi dodatkowych;
- konfiguracja serwera:
  - alokacja pamięci RAM systemu i pamięci dyskowej,
  - planowanie przyszłych potrzeb w tym zakresie;
- tworzenie i modyfikacja bazy danych:
  - alokacja folderów i plików,
  - tworzenie wyjściowych obiektów bazy danych,
  - tworzenie tabel i perspektyw wirtualnych oraz indeksów;
    - ► w rzeczywistości administrator rzadko wykonuje ostatnią z wymienionych grup prac osobiście – twórcami wymienionych obiektów są najczęściej projektanci baz danych i aplikacji oraz programiści;
- modyfikacja struktur bazy danych,
  - zadanie to należy do właścicieli obiektów, którymi z definicji są ich twórcy, niekoniecznie będący administratorami;
- zarządzanie dostępem:
  - tworzenie i usuwanie użytkowników,
  - nadawanie (także odbieranie, modyfikacja) uprawnień użytkownikom,
  - monitorowanie dostępu użytkowników do danych;
- monitorowanie i optymalizacja wydajności bazy danych,
  - rozwiązywanie problemów, takich jak np. zbyt wolne działanie systemu czy brak niezbędnych do realizacji określonych zadań uprawnień użytkowników;
- planowanie bezpieczeństwa danych:
  - sporządzanie kopii zapasowych i kopii archiwalnych,
  - postępowanie w razie awarii (podnoszenie bazy danych po awarii);
- kontakty z producentem SZBD dla zapewnienia wsparcia,
- zapewnienie zgodności instalacji z licencją.

Powyższa obszerna lista nie wyczerpuje zakresu obowiązków administratora, które obejmują rozwiązywanie wszelkich problemów pozostałych użytkowników, włącznie z najczęściej zgłaszanym zapomnieniem hasła.

Niemal wszystkie operacje w systemie bazy danych administrator może wykonać za pomocą poleceń SQL i ich znajomość jest absolutnie niezbędna na tym stanowisku. Systemy komercyjne często oferują dodatkowe narzędzia posiadające interfejs graficzny i działające w przeglądarce internetowej (np. *Oracle Enterprise Manager*). Są one bardzo pomocne, ułatwiając bieżącą pracę i wzbogacając możliwości monitorowania systemu o wykresy, diagnozy i alarmy, jednak nie wyeliminują one całkowicie konieczności znajomości języka SQL.

Grupę użytkowników profesjonalnych tworzą pozostali informatycy pracujący z bazą danych, a w szczególności:
- projektanci bazy danych,
- analitycy systemowi, projektanci i programiści aplikacji,
- twórcy SZBD i narzędzi bazodanowych,
- operatorzy systemów,
- personel pomocniczy.

Do grupy tej nie zalicza się jednak opisanych w następnym punkcie użytkowników zaawansowanych, którzy mogą być informatykami.

Ostatnią – najliczniejszą – grupę tworzą użytkownicy końcowi. R. Ramakrishnan [2003] dzieli ich znów na trzy kategorie:
- użytkownicy zaawansowani,
- użytkownicy okazjonalni,
- użytkownicy sparametryzowani.

Użytkownikami zaawansowanymi można nazwać tych użytkowników, którzy mają większą wiedzę i umiejętności niż „zwykły" użytkownik końcowy oraz posługują się zaawansowanymi narzędziami, niekiedy w wyszukany sposób. Do tej kategorii mogą być zaliczeni:
- informatycy,
- inżynierowie,
- naukowcy,
- analitycy finansowi,
- kontrolerzy skarbowi i bankowi.

Oczywiście, nie każdy inżynier czy naukowiec z definicji jest użytkownikiem zaawansowanym. Obecność na liście oznacza, że tego rodzaju użytkownicy, jeśli już pracują z bazą danych, to często korzystają z zaawansowanych narzędzi i metod.

Użytkownicy okazjonalni stanowią zróżnicowaną grupę użytkowników, którzy rzadko korzystają z konkretnej bazy danych, pojawiają się niespodziewanie, mogą posiadać szerokie uprawnienia, korzystać z bazy danych w sposób niestandardowy, posługując się zaawansowanymi narzędziami i stosując wyszukane metody pracy, aczkolwiek wszystkie te pozytywne cechy nie są żelazną regułą. Użytkownikami tej kategorii

mogą być pracownicy służb policyjnych i specjalnych, wysłannicy centrali czy spółek-matek pojawiający się np. w celu zdiagnozowania niezadowalających wyników placówki czy doraźnie zatrudniani specjaliści.

Ostatnią, lecz najliczniejszą, grupą użytkowników końcowych są użytkownicy sparametryzowani, którzy pracują z bazą danych najczęściej za pośrednictwem aplikacji, wykorzystując przyjazne narzędzia w postaci interfejsu graficznego, posługując się gotowymi schematami i procedurami postępowania. Do tej kategorii zaliczymy pracowników okienek bankowych, biur podróży, poczty, urzędników administracji, ale też kasjerów w marketach i sklepach.

Klasyfikacja Ramakrishnana powstała jakiś czas temu, kiedy komputer nie był codziennym narzędziem większości ludzi. Dziś kategorię użytkowników baz danych należy rozszerzyć o użytkowników niekorporacyjnych, korzystających z dobrodziejstw powszechnej komputeryzacji w życiu prywatnym oraz zawodowym na własny rachunek. Można więc przyjąć, że wszyscy należymy do grupy użytkowników indywidualnych, chyba że ktoś w ogóle nie korzysta z komputera ani w pracy, ani w domu. Jesteśmy też zróżnicowaną pod względem zaawansowania grupą. Niektórzy z nas korzystają czasem z poczty elektronicznej, mediów społecznościowych, dokonują zakupów przez Internet, inni tworzą własne sklepy internetowe albo dokonują inwestycji na rynkach kapitałowych.

2.4. Modele danych

W pierwszym okresie rozwoju baz danych wykorzystywane były modele hierarchiczny i sieciowy przejęte z systemów tradycyjnych. Obecnie w bazach danych nie są one już stosowane.

2.4.1. Model relacyjny

W roku 1974 brytyjski matematyk Edgar F. Codd stworzył model relacyjny, który zrewolucjonizował teorię i praktykę baz danych, stając się powszechnie akceptowanym i stosowanym do dziś modelem w przytłaczającej większości systemów bazodanowych. Centralnym pojęciem modelu relacyjnego jest relacja, którą – abstrahując

od matematycznych definicji – możemy utożsamiać z tabelą[1]. Baza danych może zawierać wiele tabel (system USOS przechowuje dane w ok. 700 tabelach), a każda z nich musi być oznaczona niepowtarzalną w bazie danych nazwą.

Relacja (tabela) posiada szereg atrybutów, które w praktyce nazywamy kolumnami lub polami. Każdy atrybut musi posiadać unikatową dla danej relacji nazwę oraz zdefiniowaną na stałe domenę (typ danych). W przeciwieństwie do powszechnie używanego arkusza kalkulacyjnego w jednej kolumnie nie można umieszczać danych różnego typu, na przykład dat i liczb czy tekstów. Maksymalna liczba kolumn jest teoretycznie nieograniczona, ale w praktyce komercyjne systemy baz danych ograniczają tę liczbę do kilkudziesięciu lub kilkuset kolumn. Nowsze systemy pozwalają zdefiniować kilka, a nawet kilkadziesiąt tysięcy kolumn, ale w bazach danych wykorzystywanych w zarządzaniu takie ekstensywne rozwiązania nie mają racji bytu.

Domena to zbiór wartości, jakie mogą być umieszczane w danej kolumnie. Domeny posiadają nazwy, które mogą się różnić w poszczególnych systemach komercyjnych. Najczęściej wykorzystywane są następujące domeny:

- CHARACTER lub CHAR – domena pozwalająca umieszczać w kolumnie ciągi dowolnych znaków – liter, cyfr i znaków specjalnych (przestankowych i innych),
- VARCHAR – domena znakowa zmiennej długości,
- NUMERIC, NUMBER, DECIMAL – domena numeryczna pozwalająca przechowywać liczby całkowite i ułamkowe,
- INTEGER, INT – domena numeryczna pozwalająca przechowywać tylko liczby całkowite,
- REAL, FLOAT – domena numeryczna przeznaczona do przechowywania liczb o zmiennej precyzji (bardzo dużych i bardzo małych); ten typ danych znajduje zastosowanie np. w astronomii czy fizyce atomowej, ale w zarządzaniu jest praktycznie bezużyteczny,
- DATA – w kolumnie tego typu można umieszczać daty,
- LOGICAL, BOOLEAN – domena obejmująca wartości logiczne PRAWDA i FAŁSZ, a także wartość nieznaną NULL w przypadku domeny BOOLEAN.

1 W wielu tekstach relacja w modelu relacyjnych bywa rozumiana mylnie jako związek pomiędzy tabelami, co jest wynikiem nieporozumienia na tle przeciążenia semantycznego pojęcia „relacja" w języku polskim. W języku angielskim występują dwa pojęcia: *relation* używane w modelu relacyjnym, które należy rozumieć jako tabelę, i *relationship* oznaczające zależność między encjami w późniejszym modelu encyjno-relacyjnym.

W większości przypadków dla kolumny definiuje się jej rozmiar, ale sposób definicji zależy od domeny. I tak dla kolumn znakowych (typu CHARACTER) szerokość kolumny podaje się w znakach, np. 10 znaków oznacza, że do tej kolumny będzie można wprowadzić najwyżej 10 znaków. Kolumna taka zawsze zajmować będzie zdefiniowaną liczbę znaków niezależnie od wprowadzonej treści. Odmianą domeny znakowej jest VARCHAR, dla której również definiuje się maksymalną szerokość, ale po wprowadzeniu danej pole zajmuje tylko tyle znaków, ile faktycznie wprowadzono. Typ ten, chętnie stosowany przez praktyków, ułatwia projektowanie baz danych oraz pozwala oszczędzić przestrzeń pamięci zajmowaną przez dane, ale zmniejsza wydajność przetwarzania.

Dla pól numerycznych typu NUMBER/DECIMAL definiuje się liczbę cyfr w części całkowitej oraz ewentualnie liczbę pozycji dziesiętnych. Także w tym przypadku szerokość kolumny jest stała.

Inne domeny, takie jak DATE, LOGICAL czy INTEGER, nie wymagają deklaracji rozmiaru – ich wielkość i sposób kodowania wartości są predefiniowane.

2.4.2. Model encyjno-relacyjny

Model relacyjny, pomimo swojej prostoty, na etapie analizy i projektowania konceptualnego może być zbyt szczegółowy i nazbyt techniczny dla użytkowników jako uczestników analizy i procesu projektowego. Wychodząc naprzeciw potrzebom i oczekiwaniom, Peter Chen stworzył model semantyczny nazywany modelem encyjno-relacyjnym (*Entity-Relationship Model*); w języku polskim nazywany jest też modelem związków encji. Jest to model graficzny przeznaczony do projektowania baz danych na wyższym poziomie abstrakcji niż ten stosowany w modelu relacyjnym. Model ten stał się podstawą większości podręczników i kursów projektowania baz danych, ponieważ nadaje się do celów dydaktycznych i prezentacyjnych. W praktyce, gdzie często mamy do czynienia z dużą liczbą atrybutów, nie zdaje on egzaminu ze względu na kłopotliwość ich przedstawienia na rysunku i szybko malejącą przejrzystość projektu. Z tego powodu zastosowanie oryginalnego modelu Chena ogranicza się do celów dydaktycznych i publicystycznych, zaś do celów praktycznych używane są zmodyfikowane wersje pozwalające uwzględnić wiele atrybutów na niewielkiej przestrzeni rysunku.

Model encyjno-relacyjny abstrahuje od pojęcia tabeli, posługując się w zamian bardziej ogólnym pojęciem encji, które oznacza każdy byt, o którym będą przechowywane dane w bazie danych. Tak więc encją

może być osoba, produkt, miasto, ale też np. przedmiot na studiach czy język obcy. Dla każdej encji definiuje się szereg opisujących ją atrybutów, na przykład nazwisko, imię, datę urodzenia dla osoby czy symbol, nazwę, cenę itd. dla towaru.

Drugim centralnym pojęciem w modelu encyjno-relacyjnym jest relacja (*relationship*), która oznacza związek pomiędzy encjami. Nie należy mylić tego pojęcia z pojęciem relacji (*relation*) używanym w modelu relacyjnym. Relacja określa związek pomiędzy encjami – na przykład osoba posiada samochód, pacjent ma przypisane łóżko w szpitalu, student zalicza przedmiot itd. Relacje opisywane są dokładniej za pomocą odpowiednich symboli graficznych, które definiują obowiązkowość albo dobrowolność związku oraz dopuszczalną liczbę egzemplarzy encji wchodzących w relację. Na przykład można mieć jednego małżonka (w cywilizacji europejskiej), ale wiele dzieci czy samochodów. Podstawowy model encyjno-relacyjny został rozszerzony o dodatkowe możliwości stosowania specjalizacji, czyli podziału encji na podklasy, dla których można definiować różne zbiory atrybutów i różne relacje. Tę odmianę modelu nazywamy rozszerzonym modelem encyjno-relacyjnym (*Enhanced Entity-Relational Model, EE-R*).

2.5. Funkcje Systemu Zarządzania Bazą Danych (SZBD)

Intuicyjnie łatwo jest określić podstawowe funkcje systemu baz danych, takie jak zapewnienie obsługi użytkownika w zakresie wprowadzania i aktualizacji danych, przechowywanie danych czy wyszukiwanie danych. SZBD realizuje jednak o wiele szersze spektrum funkcji. Poniżej wymieniamy najważniejsze z nich:

- obsługa dostępu do danych,
- wykonywanie manipulacji na danych,
- definiowanie i modyfikacja struktury bazy danych,
- kontrola integralności danych,
- kontrola dostępu do danych,
- kontrola dostępu do programów,
- zapewnienie interfejsu użytkownika,
- wbudowany język SQL,
- narzędzia administratora,
- zabezpieczenia przed skutkami awarii (nie zawsze),

- narzędzia projektowe (nie zawsze),
- narzędzia programistyczne (nie zawsze),
- inne mechanizmy.

Obsługa dostępu do danych obejmuje:

- sekwencyjne udostępnianie danych,
- wyszukiwanie danych według zadanych kryteriów,
- tworzenie, utrzymanie i wykorzystanie indeksów,
- obsługa wielodostępu.

Sekwencyjne udostępnianie danych

Dane w bazie danych przechowywane są w jakiejś kolejności fizycznej, która zwykle nie jest znana użytkownikowi i w zasadzie nie jest istotna. Jednakże użytkownik często wymaga, aby dane były udostępniane (wyświetlane, drukowane czy przetwarzane) w określonej kolejności, np. alfabetycznie według nazwiska czy malejąco według średniej oceny w semestrze. SZBD zapewnia uzyskanie dowolnej, zdefiniowanej przez użytkownika, kolejności danych.

Wyszukiwanie danych według zadanych kryteriów

Użytkownik czy aplikacja może potrzebować danych dotyczących dowolnego obiektu, na przykład pojazdu posiadającego określony numer rejestracyjny czy listy albumów nagranych przez wybranego artystę. Odpowiednio wyszukiwania te nazywamy wyszukiwaniem punktowym i przedziałowym.

Tworzenie, utrzymanie i wykorzystanie indeksów

Aby obie wymienione funkcje, a także inne, niewymienione, były realizowane z oczekiwaną przez użytkownika szybkością, w zasadzie natychmiast, w bazie danych muszą być utworzone, aktualizowane automatycznie i wykorzystywane indeksy. Można je porównać do tradycyjnych katalogów w bibliotece. Systemy bazodanowe obsługują dwa podstawowe rodzaje indeksów: ISAM i drzewo B+.

Obsługa wielodostępu

Zwykle z bazy danych korzysta równocześnie wielu użytkowników, co rodzi rozliczne problemy i stwarza zagrożenia dla integralności danych i wydajności systemu. Odpowiednie mechanizmy SZBD muszą realizować obsługę wielodostępu tak, aby integralność danych w żadnej sytuacji nie była zagrożona, a wydajność systemu zapewniała odpowiedni komfort użytkowania.

Wykonywanie manipulacji na danych

Funkcja ta oznacza zapewnienie możliwości wykonywania codziennych operacji w bazie danych, takich jak dopisywanie nowych danych, modyfikacja danych, usuwanie danych oraz zliczanie danych. Ostatnia z wymienionych operacji oznacza dokonywanie różnych obliczeń i wyszukiwań, takich jak znalezienie najtańszego wyrobu, obliczenie przeciętnego wynagrodzenia czy policzenie, ilu klientów kupiło dany wyrób.

Definiowanie i modyfikacja struktury bazy danych

System zarządzania bazą danych musi oferować mechanizmy i narzędzia umożliwiające utworzenie bazy danych, a także jej modyfikacje w trakcie użytkowania, na przykład dodanie nowej kolumny czy zmianę jej szerokości.

Kontrola integralności danych

Kontrola integralności danych jest jedną z fundamentalnych i absolutnie niezbędnych funkcji systemu bazodanowego; musi ją zawierać każdy, nawet najprostszy SZBD. Integralność bazy danych definiuje się w fazie projektowania bazy danych w postaci wymogów integralności. Jest kilka rodzajów tych wymogów, które szczegółowo omówione zostaną w dalszej części tego rozdziału.

Kontrola integralności polega między innymi na niedopuszczeniu do sytuacji, w której np. dwie osoby posiadają ten sam numer PESEL, faktura nie posiada przypisanego klienta albo liczba ludności jest ujemna. Użytkownik ma prawo się pomylić, ale baza danych musi blokować próby pogwałcenia wymogów integralności, na przykład polegające

na usunięciu klienta, dla którego wystawiono fakturę albo przypisaniu nowemu studentowi numeru PESEL innego studenta. Baza danych nie zapobiegnie jednak wprowadzeniu błędnej wartości tego numeru czy jakiejkolwiek innej danej. Kontrola integralności dotyczy tylko wymogów zdefiniowanych w bazie danych oraz poprawności formalnej typów predefiniowanych, takich jak data.

Kontrola dostępu do danych

Dostęp do danych zawartych w bazie danych regulowany jest za pomocą skomplikowanych mechanizmów. Najczęściej stosowany wykorzystuje metodę uznaniową (*Discretionary Access Control*), która pozwala indywidualnie dla każdego użytkownika określić jego uprawnienia do wykonywania określonych operacji na każdym obiekcie bazy danych z osobna. Na przykład użytkownik Jan może mieć prawo przeglądania cennika, ale nie może w nim niczego zmienić, zaś użytkownik Ewa może zmieniać cenę, ale nie ma prawa usunąć żadnej pozycji z cennika. Uprawnienia użytkowników zwykle definiuje się w momencie zakładania dla nich kont w bazie danych, ale mogą one być później zmieniane w dowolnym momencie.

W zaawansowanych technologicznie bazach danych może być też stosowana metoda obligatoryjna (*Mandatory Access Control*), w której każdy obiekt oraz każdy użytkownik posiada określoną klasę poufności, co stanowi podstawę do umożliwienia lub zablokowania dostępu odczytu i zapisu obiektu przez danego użytkownika. W metodzie tej najczęściej stosowany jest model Bell-LaPadula wyróżniający 4 klasy poufności: ściśle tajne, tajne, poufne i nieposiadające klauzuli poufności. Jak można się domyślać, metoda ta jest stosowana głównie w systemach o podwyższonych wymaganiach dotyczących kontroli dostępu do danych.

Kontrola dostępu do programów

Chodzi tu o kontrolę dostępu do tzw. modułów wbudowanych, czyli modułów programowych zapisanych w bazie danych. Funkcji tej nie realizuje metoda uznaniowa, jest ona dostępna tylko w bazach danych posiadających zaimplementowaną metodę obligatoryjną.

Zapewnienie interfejsu użytkownika

SZBD musi zapewnić użytkownikom możliwość wykonywania operacji na bazie danych. Dotyczy to zarówno operacji użytkowych, jak wprowadzanie czy wyszukiwanie danych, jak i projektowania bazy danych czy operacji administracyjnych. W zasadzie wyróżniamy dwa rodzaje interfejsu użytkownika: graficzny i tekstowy.

Interfejs graficzny w postaci okien dialogowych zawiera różne tzw. kontrolki (rozwijane listy, znaczniki, przyciski) umożliwiające interakcję za pomocą klawiatury oraz myszki i innych urządzeń. Interfejs ten oferowany jest zarówno użytkownikom końcowym, zwłaszcza w popularnych bazach danych, jak Access czy starszy Visual FoxPro, jak również zaawansowanym, zwłaszcza projektantom i administratorom.

Inna metoda pracy z bazą danych polega na pisaniu instrukcji w oknie tekstowym. W metodzie tej wykorzystywany jest przede wszystkim standardowy język baz danych SQL (opisany poniżej); w przeszłości popularnym językiem baz danych użytkowanych na komputerach osobistych PC był język Xbase. Metoda ta wykorzystywana jest przede wszystkim przez profesjonalnych użytkowników baz danych, ale także przez niektórych użytkowników zaawansowanych niebędących informatykami.

Wbudowany język SQL

Język SQL (*Structured Query Language*) powstał w latach 70. XX w., a następnie był rozszerzany i udoskonalany w kolejnych odsłonach. Jest to język standaryzowany, to znaczy definiowany w sposób precyzyjny w dokumentach opracowywanych przez Międzynarodową Organizację Normalizacyjną (ISO) i Amerykański Narodowy Instytut Normalizacji (ANSI). W okresie ponad 30 lat ustanowiono w ten sposób szereg standardów, spośród których najważniejszymi były SQL92 i SQL99. Pierwszy z nich stanowi podstawę języka i jest powszechnie akceptowany i stosowany praktycznie we wszystkich systemach baz danych. Pozwala on wykonywać wszystkie operacje w bazach danych, jednakże nie daje możliwości pisania programów. Standard SQL99 zawiera istotne rozszerzenia programistyczne umożliwiające pisanie programów, ale nie zawiera on praktycznie narzędzi do tworzenia samodzielnych aplikacji, nie dając możliwości programowania interfejsu użytkownika.

Język SQL używany jest zarówno jako narzędzie pracy bezpośredniej z bazą danych, jak i w programowaniu aplikacji. Trzeba podkreślić, że komunikacja z bazą danych może być realizowana wyłącznie za pomocą

instrukcji SQL, tak więc muszą one być umieszczane w każdej aplikacji korzystającej z bazy danych. Dla języka SQL praktycznie nie ma alternatywy i jest on wbudowany w zasadzie we wszystkie systemy baz danych oraz stosowany na całym świecie.

Narzędzia administratora

Naturalnym i podstawowym narzędziem administratora jest język SQL, ale systemy komercyjne oferują często dodatkowe narzędzia w postaci programów oferujących interfejs graficzny, a w nim nie tylko możliwość wykonywania wielu operacji bez konieczności pisania instrukcji SQL, ale także dające możliwości dodatkowego monitoringu i analiz w postaci różnych wykresów obciążenia bazy danych i innych parametrów. Narzędzia te jednak zwykle nie są udostępniane innym niż administratorzy użytkownikom.

Zabezpieczenia przed skutkami awarii (nie zawsze)

Dane, które mają zostać zapisane na dysku, przechowywane są w buforze. Awaria sprzętu lub zapaść softwareowa mogą spowodować utratę tych danych. W systemach tradycyjnych po takim incydencie musi być przeprowadzona żmudna analiza ostatnich zapisów w konfrontacji z ostatnimi aktualizacjami danych. Jest to zabieg czasochłonny, uciążliwy i niegwarantujący pełnego przywrócenia integralności danych. Zaawansowane systemy bazodanowe wyposażone są w moduł rekonstrukcji bazy danych po awarii. Jest to proces w pełni zautomatyzowany i niezawodny, gwarantujący przywrócenie pełnej integralności danych.

Narzędzia projektowe (nie zawsze)

Systemy zarządzania bazą danych mogą zawierać narzędzia wspomagające projektowanie bazy danych, takie jak edytor graficzny do sporządzania projektu encyjno-relacyjnego czy generator kodu SQL dokonujący translacji tego projektu do modelu relacyjnego, czyli tworzący komplet instrukcji CREATE TABLE tworzących tabele bazy danych wraz z wymogami integralności. Narzędzia te często zawarte są w popularnych systemach bazodanowych „z niższej półki", co nie jest regułą dla profesjonalnych serwerowych systemów zarządzania bazą danych. Producenci systemów

oferują czasami darmowe narzędzia tego typu, np. Oracle SQL Developer, ale ich jakość w przeszłości bywała niezadowalająca. Na rynku oferowane są także odpłatne i darmowe narzędzia innych firm. Jednym z lepszych narzędzi komercyjnych jest produkt holenderskiej firmy DeZign.

Narzędzia programistyczne (nie zawsze)

Baza danych jest fundamentem systemów informatycznych. Przechowuje ona dane, zapewniając ich integralność, ale dostęp do tych danych realizowany jest w zasadzie wyłącznie za pośrednictwem języka SQL, który nie daje możliwości tworzenia przyjaznych interfejsów użytkownika. Te tworzone są za pomocą innych narzędzi projektowo-programistycznych. Popularne systemy bazodanowe firmy Microsoft, takie jak Access czy wcześniejszy Visual FoxPro, posiadają wbudowane narzędzia do tworzenia aplikacji zarówno typu *single user*, jak i sieciowych. Profesjonalne serwery baz danych niekoniecznie oferują takie narzędzia, np. starsze wersje Oracle zawierały generator raportów oraz interfejsu użytkownika, ale w późniejszych wersjach zrezygnowano z nich. Przyczyną były dominujące wymagania, aby aplikacje działały w Internecie w środowisku przeglądarek. Aplikacje takie są o wiele bardziej skomplikowane i obecnie w zasadzie do ich projektowania i programowania wykorzystywane są oprócz języka SQL zewnętrzne narzędzia projektowo-programistyczne.

Inne mechanizmy

Zaawansowane systemy bazodanowe mogą zawierać szereg innych narzędzi, na przykład umożliwiających zaawansowane analizy danych (*data mining*), tworzenie hurtowni danych czy rozproszonych baz danych.

2.6. Wymogi integralności

W stosunku do danych przechowywanych w bazach danych stawia się restrykcyjne wymagania dotyczące integralności. Wymagania te definiuje się w momencie tworzenia tabeli bazy danych, ale mogą one być później zmieniane.

Tworząc tabelę, trzeba zdefiniować jej poszczególne kolumny, a dla każdej z nich – domenę (typ danych). Definicja tabeli implikuje więc natychmiastowo wymogi domeny. Oznacza to, że jeżeli cenę wyrobu zdefiniujemy jako pole numeryczne o określonej liczbie cyfr i pozycji dziesiętnych, to nie będzie możliwe wprowadzenie do tego pola ciągu znaków nietworzących poprawnej liczby (np. uwagi 'nieustalona') ani liczby zawierającej więcej cyfr, niż zadeklarowano.

Dla innych wymogów integralności centralnym pojęciem jest klucz, który identyfikuje obiekt, a czasem jego własność. Na przykład kluczem może być nazwisko, numer faktury czy PESEL.

Wyróżnia się następujące rodzaje kluczy:

- klucz główny,
- klucz kandydujący,
- klucz obcy.

Klucz główny (*Primary Key*) jest jednoznacznym identyfikatorem obiektu, co oznacza, że w tabeli nie mogą istnieć dwa obiekty o tej samej wartości klucza głównego. Klucz główny nie może być pusty – musi być wypełniony i nie można go później usunąć, aczkolwiek można go zmieniać. Wynika z tego, że na przykład nazwisko może być kluczem, ale nie głównym, chociaż formalnie można je zadeklarować jako klucz główny. W konsekwencji jednak nie można byłoby wprowadzić dwóch osób o tym samym nazwisku.

Dla danej tabeli można zadeklarować tylko jeden klucz główny. Tabela może jednak zawierać więcej kolumn spełniających wymogi klucza głównego. Na przykład tabela Studenci może zawierać kolumny „pesel" oraz „numer albumu". Wartości obu tych kolumn muszą być unikatowe, więc spełniają one wymogi klucza głównego. Kolumnę „numer albumu" można wówczas zdefiniować jako klucz kandydujący (*Candidate Key*). W odróżnieniu od klucza głównego tabela może posiadać wiele kluczy kandydujących. Klucze kandydujące, tak jak klucz główny, chronią tabelę przed błędami polegającymi na wprowadzeniu więcej niż jednego wiersza o tej samej wartości klucza oraz przed pozostawieniem pustego pola kluczowego.

Klucz obcy (*Foreign Key*) deklaruje się w sytuacji, gdy wymagamy, aby wartości w danej kolumnie pochodziły ze zbioru wartości istniejących w kolumnie innej (najczęściej) tabeli, w której kolumna ta powinna być kluczem głównym albo kandydującym. Na przykład w kolumnie symbol_klienta tabeli Faktury mogą znajdować się tylko symbole klienta występujące w tabeli Klienci zawierającej dane wszystkich klientów. Klucz obcy definiuje wymogi integralności referencyjnej i uniemożliwia nie tylko wystawienie faktury dla nieistniejącego klienta, ale również

zapobiega usunięciu klienta, dla którego została wystawiona faktura oraz zmianie symbolu klienta w tabeli Klienci. W rzeczywistości język SQL umożliwia określenie za pomocą odpowiednich klauzul zachowania bazy danych w przypadku próby pogwałcenia wymogów integralności referencyjnej, ale w tekście tym konsekwentnie pomijamy niuanse, pozostawiając je dociekliwości czytelnika.

Tabela może zawierać wiele kluczy obcych, co zdarza się dość często. Poza wymienionymi wymogami w tabeli można definiować wymogi integralności semantycznej, które ograniczają zakres domeny dla danych. Na przykład cenę czy ilość definiujemy jako pola numeryczne, ale nakładamy dodatkowo warunek, aby były to wartości nieujemne. Z kolei podatek VAT może przyjmować tylko predefiniowane wartości.

Jak widzimy, wymogi integralności chronią spójność bazy danych w wielu aspektach, ale nieprawdą jest, że uniemożliwiają pogwałcenie tej spójności. Owszem, zapobiegają większości możliwych błędów, ale nie eliminują całkowicie niebezpieczeństwa ich popełnienia przez użytkowników. Należy pamiętać, że we wszystkich rodzajach systemów informatycznych najczęstszym zagrożeniem bezpieczeństwa danych są błędy użytkownika, których nie da się uniknąć; można jedynie zmniejszać prawdopodobieństwo ich popełnienia, głównie drogą szkoleń.

2.7. Struktura SZBD

System Zarządzania Bazą Danych jest bardzo złożonym, obszernym i wielowarstwowym zintegrowanym systemem. Powyżej omówiono jego funkcje, w tym podrozdziale przedstawiona zostanie jego struktura.

SZBD składa się z 5 warstw oprogramowania realizujących różne kategorie zadań oraz z trzech modułów działających na różnych poziomach. Część główną, posiadającą strukturę warstwową, tworzą następujące moduły funkcjonalne:

- zarządzanie przestrzenią dyskową,
- menedżer bufora,
- pliki i metody dostępu,
- operatory relacyjne,
- optymalizacja i wykonywanie zapytań.

Zarządzanie przestrzenią dyskową

Funkcja ta dotyczy alokacji i dealokacji obszarów na dysku oraz wszelkich operacji plikowych. Jest ona realizowana przez system operacyjny, więc można byłoby oczekiwać, że SZBD ma zapewnioną pełną obsługę w tym zakresie. Tak rzeczywiście jest i w zasadzie systemy baz danych korzystają z serwisu systemu operacyjnego. W wielkich systemach bazodanowych istnieje jednak czasem konieczność dodatkowych funkcjonalności nieoferowanych przez system operacyjny, na przykład obsługi gigantycznych plików danych. Systemy bazodanowe oferują takie możliwości, pozwalając tworzyć bazy danych nawet na surowych, niesformatowanych dyskach. Obsługa danych z pominięciem usług systemu operacyjnego stanowi wówczas dodatkowe zabezpieczenie przed atakami hakerskimi wykorzystującymi luki bezpieczeństwa w systemach operacyjnych.

Menedżer bufora

Bufor jest zarezerwowanym obszarem w pamięci operacyjnej (RAM), w którym przechowywane są dane po ich wczytaniu z dysku oraz przed fizycznym zapisem na dysku. Zastosowanie bufora znakomicie przyspiesza przetwarzanie danych, ponieważ czas dostępu do pamięci buforowej wynosi jedynie ok. 100 ns (nanosekund) wobec 10 μs (mikrosekund) dla pamięci dyskowej, czyli jest około 100 000 razy krótszy. W czasie potrzebnym do wczytania jednej strony tekstu z dysku można wczytać 100 000 stron z bufora.

Rezygnacja z bufora skutkowałaby dramatycznym spowolnieniem pracy całego systemu, więc wszystkie systemy operacyjne obsługują funkcję buforowania. Niestety, menedżery bufora zawarte w systemach operacyjnych są stosunkowo proste, nieelastyczne, niedające możliwości różnicowania obsługi w zależności od charakteru danych. Z tego powodu systemy bazodanowe posiadają własne, dedykowane menedżery bufora, dające możliwości różnicowania tzw. polityki nadpisywania stron oraz działania w trybie katastroficznym (natychmiastowe wymuszenie zapisu) i niekatastroficznym (zapis ze zwłoką). Menedżery te umożliwiają ponadto tworzenie wielu obszarów buforowych, lepiej też potrafią przewidzieć, jakie dane za chwilę będą potrzebne.

Pliki i metody dostępu

Systemy operacyjne komputerów *mainframe* (wielkie, stacjonarne komputery działające w latach 60–80. ubiegłego wieku) posiadały wbudowaną obsługę plików danych o różnej organizacji, zapewniając szybki dostęp do danych. W komputerach osobistych zrezygnowano z tych możliwości i współczesne systemy operacyjne obsługują jedynie pliki seryjne, to znaczy takie, w których dane zapisywane są w zasadzie w kolejności wprowadzania, a dostęp możliwy jest też tylko seryjnie, czyli poprzez czytanie kolejnych zapisów aż do napotkania szukanych treści. Posługując się tą metodą, szukalibyśmy na przykład określonej książki w bibliotece uniwersyteckiej, przechowującej zapewne ok. 10 000 000 (dziesięć milionów) książek, przeglądając kolejno zawartość wszystkich półek we wszystkich szafach znajdujących się we wszystkich pomieszczeniach biblioteki. Być może rok wystarczyłby do odnalezienia szukanej książki. Taki tryb pracy jest jednak niezadowalający w przypadku baz danych mieszczących nawet miliardy rekordów, dodatkowo przeszukiwanych równocześnie przez wielu użytkowników. Z tego powodu nawet najprostsze systemy baz danych obsługują tzw. organizacje o bezpośrednim dostępie, pozwalające zredukować czas wyszukiwania danych z np. kilku godzin do ułamka sekundy. Istnieje kilka odmian organizacji tego typu, a najczęściej stosowaną w bazach danych jest organizacja sekwencyjno-indeksowa o nazwie drzewo B+. Systemy Zarządzania Bazą Danych oferują obsługę tej organizacji w ramach omawianej warstwy, a także innych, jeszcze szybszych organizacji.

Operatory relacyjne

Wyszukiwanie danych w bazach danych opiera się na wykorzystaniu kilku powszechnie znanych operatorów zbiorowych, mianowicie sumy, różnicy i iloczynu zbiorów oraz iloczynu kartezjańskiego. Warstwa ta realizuje wymienione operacje na tabelach danych, tworząc zbiory wynikowe według prostych i przejrzystych reguł. Dla ich wyjaśnienia przyjmijmy, że mamy dwa zbiory krotek (wierszy) R i S. I tak suma zbiorów R+S jest zbiorem krotek występujących w jednym ze zbiorów wejściowych R i S albo w obu. Różnicę zbiorów R – S stanowią krotki występujące w zbiorze R, lecz niewystępujące w zbiorze S. Iloczyn zbiorów R ∩ S zostanie utworzony przez krotki występujące w obu zbiorach.

Wymienione operatory wymagają tzw. zgodności unijnej (*union compatibility*), czyli zgodnej liczby i domen atrybutów tabel biorących udział w złączeniu. Wymóg ten nie musi być i na ogół nie jest spełniony w odniesieniu do operatora iloczynu kartezjańskiego (*Cross Product*), który tworzą połączenia każdej krotki ze zbioru R z każdą krotką ze zbioru S. Taki surowy zbiór na ogół jest bezużyteczny i podlega selekcji, która jest realizowana w praktyce przez tzw. operatory pochodne, które filtrują zbiór danych uzyskanych jako iloczyn kartezjański, odrzucając te krotki, które nie mają w danym przypadku sensu albo są bezużyteczne. Istnieją trzy rodzaje operatorów pochodnych: złączenia oznaczone (*Condition Join*), złączenia równościowe (*Equijoin*) i złączenia naturalne (*Natural Join*). Różnią się one sposobem specyfikacji warunków filtrujących. Szczegóły te znów wykraczają poza ramy materiału skróconego, a zainteresowani odszukają bez trudu stosowne opisy w Internecie, posługując się polskimi lub angielskimi nazwami operatorów.

Optymalizacja i wykonywanie zapytań

Zapytania SQL kierowane bezpośrednio przez użytkowników lub zawarte albo generowane przez aplikacje są tłumaczone na operacje zbiorowe, ale przedtem podlegają optymalizacji. Po pierwsze, bez tego zabiegu niektóre zapytania byłyby niemożliwe do wykonania z powodu wielkości zbiorów pośrednich tworzonych przez zapytania. Po drugie, czas wykonania zapytań jest niezwykle istotny zarówno dla użytkownika kierującego zapytanie, jak i innych użytkowników współdzielących zasoby tego samego serwera dla wykonywania własnych zapytań.

Poza strukturą warstwową funkcjonują moduły aktywne w wielu warstwach:

- menedżer transakcji,
- menedżer blokad,
- menedżer rekonstrukcji bazy danych po awarii.

Menedżer transakcji

W bazach danych dominuje przetwarzanie transakcyjne, w którym transakcja jest definiowana jako szereg pojedynczych operacji. W odniesieniu do przetwarzania transakcyjnego zdefiniowano tzw. własności transakcji ACID. Skrót ten pochodzi od pierwszych liter nazw poszczególnych własności:

- A jak *Atomicity* – niepodzielność[2] transakcji,
- C jak *Consistancy* – spójność,
- I jak *Isolation* – izolacja, separacja,
- D jak *Durability* – trwałość.

Własność *Atomicity* mówi, że transakcja musi być wykonana w całości albo musi być wycofana. Na przykład nie do przyjęcia jest, aby transakcja sprzedaży wystawiła fakturę, a nie skorygowała np. stanu magazynowego o ilość sprzedanego produktu.

Własność *Consistancy* jest nieco bardziej skomplikowana[3]. Jej definicja jest następująca: jeżeli mamy szereg transakcji i żadna z nich nie narusza spójności bazy danych, to ich wykonanie w dowolnej kolejności, także z przeplotem[4], nie może naruszyć spójności bazy danych.

Własność *Isolation* oznacza, że każda transakcja jest wykonywana tak, jak gdyby była jedyną wykonywaną transakcją w bazie danych, czyli że poszczególne transakcje nie mogą na siebie wpływać.

Własność *Durability* mówi, że jeżeli wykonanie transakcji zostało potwierdzone, to musi być ona bezwzględnie wykonana[5]. Jak opisano wcześniej, wyniki przetwarzania zapisywane są najpierw w buforze, a następnie poszczególne tzw. ramki bufora zapisywane są selektywnie na dysku. Niespodziewana awaria systemu może spowodować utratę wyników nawet potwierdzonych transakcji. Mimo to wymaga się, aby baza danych nie utraciła tych wyników. Jest to możliwe dzięki zapisom okresowym stanu bazy danych (*Checkpointing*) oraz wymuszonym (polityka *force*) zapisom w dzienniku WAL (*Write Ahead Log*) zawartości bufora katastroficznego. W procesie rekonstrukcji bazy danych po awarii potwierdzona transakcja zostanie dokończona (zapisana).

Własności ACID muszą być zapewnione przez producentów systemów bazodanowych, ale warunkiem ich dochowania jest nienaruszalność zapisów punktów kontrolnych (*checkpoints*) oraz dziennika WAL. Dlatego te dwa elementy powinny być zapisywane na niezawodnych nośnikach, a jako dodatkowe zabezpieczenie stosowany jest zapis równoległy na dwóch oddzielnych dyskach (*mirroring*).

---

2 W niektórych tłumaczeniach używana jest nazwa „atomowość", co niekoniecznie dobrze oddaje istotę tej własności.

3 Niektóre teksty internetowe, a nawet podręczniki, określają tę własność jako konieczność pozostawienia bazy danych przez transakcję w stanie spójnym, co jest oczywistym błędem. SZBD nie chroni bazy danych przed naruszeniem spójności przez użytkownika.

4 Przeplot oznacza naprzemienne wykonywanie operacji należących do różnych transakcji.

5 I znów, w niektórych tekstach własność ta wyjaśniana jest błędnie jako fakt, że dane są zapisywane w sposób trwały na dysku. Ten i inne przypisy do własności transakcji dowodzą, jak nierzetelnym źródłem wiedzy może być Internet, a nawet niektóre papierowe podręczniki.

O zachowanie własności transakcji troszczy się w czasie tzw. wykonywania normalnego menedżer transakcji. Bierze on także udział w procesie podnoszenia (rekonstrukcji) bazy danych po awarii.

Menedżer blokad

Przetwarzanie danych w bazie danych charakteryzuje wielozadaniowość – równoczesne wykonywanie wielu transakcji, zwykle z przeplotem – oraz wielodostęp polegający na równoczesnym dostępie do danych przez wielu użytkowników. Brak odpowiedniego systemowego zarządzania tymi procesami bardzo szybko doprowadziłby do chaosu, zakleszczeń, przeciążenia systemu i błędów. Dla zachowania bezpieczeństwa, spójności danych oraz odpowiedniej wydajności systemu stosowane są blokady różnych obiektów bazy danych, na przykład wierszy czy całych tabel. Istnieje kilka tzw. protokołów blokowania wykorzystujących różne mechanizmy. Jednym z najczęściej stosowanych jest protokół o nazwie 2PL (*Two Phase Locking*). W protokole tym wyróżnia się dwa rodzaje blokad: wyłączną i dzieloną. Blokada wyłączna wymagana jest do przeprowadzenia operacji aktualizacji danych – nikt inny w tym czasie nie może danych aktualizować ani nawet czytać. Blokada dzielona zakładana jest do czytania. Inni użytkownicy mogą w tym samym czasie czytać ten sam obiekt, ale nie mogą go aktualizować.

Blokady mogą powodować negatywne zjawiska, takie jak klincz i konwój. Z klinczem mamy do czynienia, gdy dwie (lub więcej) transakcje nawzajem blokują sobie obiekty niezbędne do wykonania transakcji. Konwój powstaje, gdy w oczekiwaniu na dostęp do obiektu blokowanego przez jakąś transakcję tworzy się kolejka transakcji. Do przeciwdziałania klinczom wykorzystywane są dwie metody o egzotycznych nazwach: *Wait-die* i *Wound-wait*. Nie wdając się w szczegóły, można powiedzieć, że usuwają one z kolejki, a nawet zabijają (wycofują) transakcje będące w trakcie wykonywania, stosując kryteria priorytetu. Dla złagodzenia tego okrucieństwa okresowo uaktywnia się procedura wyławiania ofiar, która umożliwia łaskawie wykonanie pechowych, wielokrotnie usuwanych z kolejki albo zabijanych transakcji.

Inną metodą walki z klinczami jest stosowanie protokołów blokowania, które nie powodują klinczów. Jednym z nich jest dwufazowy protokół konserwatywny (*Conservative 2PL*), zgodnie z którym transakcja otrzymuje od razu wszystkie blokady, jakich może potrzebować, lub żadnej. Protokół optymistyczny, zakładający brak klinczów, jest rzadziej stosowany.

Menedżer rekonstrukcji (*Recovery Manager*)

Moduł ten – według niektórych autorów podręczników baz danych – jest najtrudniejszym do zaprojektowania i oprogramowania. Jednym z powodów jest wymóg absolutnej bezbłędności działania w warunkach awaryjnych. Menedżer rekonstrukcji wkracza do akcji po wystąpieniu awarii sprzętowej lub zapaści systemowej. W takiej sytuacji zazwyczaj następuje utrata danych znajdujących się w niekatastroficznym buforze. Mogą to być zarówno dane związane z już zakończonymi transakcjami, jak i będącymi w trakcie realizacji. Podczas rekonstrukcji stosowana jest prosta reguła: wyniki transakcji potwierdzonych muszą być zapisane, a transakcje niepotwierdzone (niezakończone) muszą być wycofane. Cały proces realizowany jest najczęściej z zastosowaniem algorytmu ARIES, który obejmuje trzy kroki*: Analysis*, *Redo*, *Undo*. Krok *Analysis* polega na identyfikacji punktu restartu zapisanego w obrazie migawkowym (*Checkpoint*), tzw. brudnych stron w buforze (stron, których zawartość uległa zmianie od ostatniego zapisu) oraz transakcji aktywnych w momencie awarii. Krok *Redo* stanowi odtworzenie wszystkich operacji w bazie danych wykonanych od momentu restartu do awarii. Po zakończeniu tego kroku baza danych jest w stanie dokładnie takim samym jak w momencie awarii. W trzecim kroku, *Undo*, następuje wycofanie wszystkich niezakończonych transakcji. Po zrealizowaniu tego kroku baza danych jest w stanie spójnym, a niedokończone i wycofane transakcje mogą być zrealizowane od początku w normalnym trybie.

Kroki *Redo* i *Undo* realizowane są na podstawie dziennika WAL, w którym zapisy wykonywane są z wykorzystaniem bufora katastroficznego stosującego politykę wymuszania zapisu (*force*) natychmiast po umieszczeniu zawartości w buforze. Konieczność stosowania do tego celu niezawodnego dysku jest oczywista.

2.8. Organizacja danych w bazach danych

System operacyjny zapewnia jedynie podstawowy rodzaj organizacji pliku, jakim jest organizacja seryjna (*Heap*). W organizacji tej rekordy danych zapisywane są kolejno, jeden za drugim, a dodatkowo zmiana kolejności danych może być skutkiem przesunięć stron pomiędzy kolejkami stron pełnych i niepełnych (kolejkowanie stron jest jedną z metod zarzą-

dzania plikami o organizacji seryjnej). W efekcie nie ma żadnych przesłanek do nawet przybliżonej identyfikacji miejsca, w którym może być zapisany dany rekord. Ta organizacja zapisu jest wystarczająca do celów przetwarzania małych zbiorów danych, szczególnie przy pracy w trybie *single user*. W bazach danych czas dostępu do danych jest niezwykle istotny, dlatego pliki danych w bazach danych posiadają organizację oferującą tzw. bezpośredni dostęp, co zapewnia bardzo krótki czas niezbędny do odszukania rekordu o wskazanej wartości klucza (identyfikatora).

W zasadzie do tego celu wykorzystywana jest organizacja sekwencyjno-indeksowa, która posiada dwie odmiany: ISAM (*Indexed Sequential Access Method*) i drzewo B+. Organizacja ISAM została stworzona przez IBM dla komputerów *mainframe* w latach 60. ubiegłego wieku. Jej cechą charakterystyczną jest statyczne drzewo indeksowe tworzone w procesie załadowczym na podstawie posortowanego zbioru danych. Drzewo to nie podlega modyfikacjom, co stanowi z jednej strony zaletę, a z drugiej – wadę. Zaletą statyczności jest szybkość. Strony drzewa statycznego nie wymagają blokowania ani aktualizacji i mogą być w dużej części umieszczone w pamięci wewnętrznej, co radykalnie zwiększa szybkość przetwarzania. Brak możliwości aktualizacji drzewa indeksowego z kolei powoduje, że w przypadku istotnego przyrostu danych powstają kłopotliwe w obsłudze łańcuchy stron nadmiarowych, co bardzo spowalnia przetwarzanie. W takich przypadkach wcześniej czy później baza danych musi być zatrzymana, dane – przekopiowane do pliku seryjnego, posortowane i użyte w nowym przebiegu załadowczym, w którym utworzone zostanie nowe, większe drzewo indeksowe.

W organizacji drzewo B+ drzewo indeksowe jest zbilansowane i elastyczne, rozrasta się w miarę przyrostu danych, a kurczy się, jeśli danych ubywa (co, generalnie, rzadko ma miejsce). Elastyczność drzewa indeksowego stanowi jednak spore obciążenie dla systemu, skutkiem czego organizacja ta jest wolniejsza niż ISAM. Pomimo to jest ona znacznie częściej stosowana i zazwyczaj jest organizacją domyślną. Organizację ISAM warto natomiast stosować w przypadkach, gdy ilość danych nie zwiększa się w stopniu generującym strony nadmiarowe, to znaczy nie więcej niż o 20%.

Jeszcze szybszy dostęp do danych zapewnia organizacja losowa (*Hashed Files*)[6], a raczej organizacje losowe, bowiem istnieją trzy odmiany tej organizacji.

6 W części tekstów polskojęzycznych używane jest określenie „pliki haszowane", co dowodzi nieznajomości nomenklatury informatycznej; pojęcie „organizacja losowa" używane jest w polskiej literaturze przedmiotu przynajmniej od lat 70. XX w.

Najstarsza z nich – organizacja statyczna (*Static Hashing*) – jest równocześnie nie tylko najszybszą organizacją losową, ale w ogóle najszybszą organizacją pliku, jaka może istnieć. W organizacji tej czas dostępu wynosi 1 proces I/O (wejścia/wyjścia) i z definicji nie może on być krótszy. Niestety organizacja ta wymaga stworzenia wzoru matematycznego stanowiącego algorytm obliczający miejsce rekordu na dysku na podstawie wartości klucza. W zasadzie organizacja przystosowana jest do pracy z kluczami numerycznymi, co ogranicza jej funkcjonalność. Ponadto jest to organizacja statyczna – istotny przyrost danych powoduje konieczność realokacji obszarów pliku oraz konieczność utworzenia zupełnie nowego algorytmu randomizacji mapującego rekordy do nowych – większych obszarów.

Organizacje rozszerzalna i liniowa stanowią uelastycznione wersje organizacji losowej. Obie wykorzystują końcowe fragmenty obrazu binarnego klucza, dzięki czemu są one niezależne od typu pola kluczowego (numeryczny lub znakowy). Pełna elastyczność obu organizacji umożliwia obsługę dowolnie rosnących zbiorów danych bez potrzeby przerywania pracy, modyfikacji algorytmów czy reorganizacji pliku. Organizacje te, aczkolwiek wolniejsze od organizacji statycznej (ich szybkość można oszacować na 1–3 procesów I/O), oferują jednak pełną automatykę funkcjonowania i elastyczność.

Wspólną wadą wszystkich organizacji losowych jest niemożność stosowania więcej niż jednego klucza dla danego pliku. Powoduje to, że w praktyce są one rzadko dziś wykorzystywane jako organizacje plików danych użytkowych, natomiast są stosowane do celów wewnętrznych w bazach danych, do obsługi plików zawierających dane robocze, takie jak listy blokad czy transakcji oczekujących na blokady. Użytkownik nie ma jednak z nimi styczności ani nie może wpływać na ich działanie.

2.9. Kontrola dostępu

W systemach operacyjnych Windows jesteśmy przyzwyczajeni do tego, że dostęp do danych jest chroniony na zasadzie „wszystko albo nic", to znaczy, że logując się jako użytkownik, mamy dostęp albo do wszystkich zasobów systemu, albo do pewnej ich części, bez różnicowania rodzajów plików danych czy nawet programów. Owszem, w systemach sieciowych dostęp do zasobów może być różnicowany według użytkowników,

typów plików czy trybu dostępu, ale wciąż jest to regulacja dostępu dość zgrubna. W bazach danych istnieje konieczność bardziej precyzyjnych ograniczeń, z dokładnością do pojedynczego pola, z rozróżnieniem rodzaju aktualizacji (dopisanie, usunięcie, modyfikacja) oraz możliwością przekazywania uprawnień pomiędzy użytkownikami.

Istnieją dwa podejścia do kontroli dostępu w bazach danych: metoda uznaniowa i metoda obligatoryjna. Metoda uznaniowa (*Discretionary Accecss Control*, DAC) przeznaczona jest do zastosowań w systemach przetwarzania transakcji. Została ona zaimplementowana w standardzie SQL92 i jest w pełni obsługiwana przez wszystkie systemy bazodanowe. W metodzie tej definiuje się następujące przywileje:
SELECT – prawo czytania danych,
INSERT – prawo dopisywania nowych danych,
DELETE – prawo usuwania danych,
UPDATE – prawo modyfikacji danych,
REFERENCES – prawo tworzenia w innych tabelach kluczy obcych z wykorzystaniem określonego pola jako klucza obcego.

Przywileje nadaje się do obiektów, którymi mogą być tabele i perspektywy wirtualne z możliwością specyfikacji kolumn tam, gdzie to ma zastosowanie. Beneficjentami przywilejów są użytkownicy identyfikowani przez nazwy *login*. Dodatkowo można upoważnić beneficjenta(ów) do przekazywania otrzymanych przywilejów innym użytkownikom. W efekcie może to powodować bardzo rozbudowane i zawikłane zależności, trudne do opanowania przez administratora. Z tego względu metodę rozszerzono w późniejszym okresie o możliwość definiowania ról, które przypisuje się użytkownikom, a następnie przywileje nadaje się rolom. Ułatwia to i upraszcza znacznie pracę administratora, pozwalając jedną instrukcją zmienić przywileje całej klasy użytkowników.

Instrukcja nadania przywilejów ma postać:

GRANT ALL PRIVILEGES | *przywileje* ON *obiekty* TO *użytkownicy* [WITH GRANT OPTION][7]

Przywileje można odbierać selektywnie, posługując się instrukcją REVOKE o podobnej składni:

REVOKE [GRANT OPTION FOR] *przywileje* ON *obiekty* FROM *użytkownicy* { RESTRICT | CASCADE }

7 Składnia instrukcji SQL w tym opracowaniu ma charakter wyłącznie poglądowy; jego celem nie jest nauczenie czytelnika posługiwania się językiem SQL.

Metoda uznaniowa posiada jedną istotną wadę: nie pozwala regulować dostępu do programów, a dokładniej do modułów wbudowanych zapisywanych w bazie danych. Wady tej pozbawiona jest metoda obligatoryjna (*Mandatory Access Control*, MAC), zwana też w polskich tekstach metodą mandatową. W metodzie tej zarówno użytkownicy, jak i obiekty otrzymują określone klasy poufności (*Top Secret*, *Secret*, *Confidential*, *Unclasified*). Prawa czytania i pisania do obiektów regulują dość proste zasady:

- prawo czytania obiektu mają użytkownicy posiadający klasę poufności wyższą lub równą klasie poufności obiektu;
- prawo pisania do obiektu otrzymują użytkownicy posiadający klasę poufności równą lub niższą (!) od klasy poufności obiektu.

Celem ograniczenia jest uniemożliwienie przenikania informacji o wysokiej klasie poufności do obiektów o niższej klasie poufności. Metoda obligatoryjna stosowana jest głównie w systemach o podwyższonych wymaganiach dotyczących dostępu do danych, szczególnie wojskowych, policyjnych i w służbach specjalnych.

Funkcję kontroli dostępu do danych spełniają też perspektywy wirtualne, zwane także widokami. Są to wirtualne tabele tworzone na podstawie tabel fizycznych, a także innych tabel wirtualnych, na potrzeby konkretnych użytkowników, klas użytkowników, aplikacji czy procesów. Perspektywy wirtualne umożliwiają ograniczenie dostępu do określonych danych (np. tylko do niektórych kolumn) w określonym trybie (np. tylko do odczytu). Inną funkcją perspektyw wirtualnych jest dostarczenie danych z bazy danych dla aplikacji, która dzięki temu nie musi w czasie swojej aktywności angażować mocy serwera bazy danych; jest to jeden z często wykorzystywanych mechanizmów w aplikacjach bazodanowych.

2.10. Język SQL

Język SQL (*Structured Query Language*) powstał na początku lat 70. XX w. w firmie IBM, lecz dopiero pod koniec dekady firma Oracle skojarzyła możliwość wykorzystania tego języka w świeżo powstałym modelu relacyjnym baz danych, implementując go w swoich komercyjnych produktach. Początkowo język SQL powoli przebijał się jako podstawowy język komunikacji w bazach danych. Pierwszy, niedoskonały standard opublikowano w 1986 r., kolejna wersja ukazała się w 1989 r., ale dopiero wersja z 1992 r.

stała się biblią języka, standardem powszechnie uznawanym i stosowanym na całym świecie. W 1999 r. udostępniono standard języka SQL poszerzony o bardzo istotne elementy, umożliwiające programowanie (procedury, funkcje, wyzwalacze, instrukcje pętli i instrukcje warunkowe). Dzięki temu mógł on być stosowany jako język programowania aplikacji, z wyłączeniem warstwy interfejsu użytkownika, ponieważ nie obejmował niezbędnych do tego narzędzi. Tym niemniej standard SQL99 jest powszechnie przyjętym wzorcem, zastosowanym w wielu systemach zarządzania bazą danych. W następnych latach ukazywały się kolejne rozszerzenia standardu, niewnoszące już jednak rewolucyjnych zmian do składni języka.

Język SQL zgodny ze standardem SQL92 jest narzędziem całkowicie wystarczającym do pracy bezpośredniej (konwersacyjnej) z bazą danych, dlatego też jest on podstawą większości podręczników oraz kursów uczelnianych i komercyjnych podstaw baz danych. Standard SQL92 wyróżnia następujące podjęzyki SQL, grupujące instrukcje przeznaczone do realizacji poszczególnych klas funkcji:

- *Data Definition Language* (DDL),
- *Data Manipulation Language* (DML),
- *Embedded And Dynamic SQL*,
- *Security*,
- *Transaction Management*,
- *Client-Server Execution and Remote Data Access*.

Podjęzyk *Data Definition Language* (DDL) obejmuje instrukcje umożliwiające tworzenie, usuwanie i modyfikacje tabel baz danych wraz z wymogami integralności. Główną instrukcją tego podjęzyka jest instrukcja CREATE TABLE, pozwalająca zdefiniować tabelę baz danych. Poniżej przedstawiono przykładową instrukcję tworzącą tabelę Stypendia[8].

```
CREATE TABLE Stypendia
        (pesel          CHAR(11),
        nazwisko        CHAR(20),
        imię            CHAR(20),
        data_ur         DATE,
        kwota           NUMBER(6,2));
```

Jeżeli chcemy wyspecyfikować wymogi integralności, definicja tabeli będzie bardziej skomplikowana.

8 Tak jak poprzednio instrukcje SQL zamieszczono jedynie w celach poglądowych. Nie jest zamiarem autora nauczenie czytelnika projektowania baz danych, a jedynie pokazanie na przykładach, jak się to robi.

```
CREATE TABLE Stypendia
        (student_id     CHAR(8) PRIMARY KEY,
        pesel           CHAR(11) UNIQUE,
        nazwisko        CHAR(20) NOT NULL,
        imię            CHAR(20) NOT NULL,
        data_ur         DATE,
        kwota           NUMBER(6,2)
                        CHECK (kwota >= 0));
```

Więcej przykładów znajdziemy w dalszej części rozdziału, w której przedstawiono projekt przykładowej bazy danych.

Strukturę tabeli baz danych można modyfikować za pomocą instrukcji ALTER TABLE, co może być w praktyce dość skomplikowane, gdy tabela zawiera już jakieś dane. Natomiast usunięcie tabeli jest proste:

```
DROP TABLE nazwa_tabeli;
```

Podjęzyk definicji danych jest używany głównie przez projektantów i programistów baz danych, a także przez nieprofesjonalnych, ale zaawansowanych użytkowników. Jego instrukcje są jednak z reguły objęte podstawowym kursem z zakresu baz danych.

Podjęzyk *Data Manipulation Language* (DML) jest najczęściej używanym podjęzykiem SQL92. Nic dziwnego, raz bowiem utworzona baza danych czasem jest modyfikowana, ale używana jest na co dzień przez długi okres i przez szerokie rzesze użytkowników. Podjęzyk DML obejmuje instrukcje czytania (wyszukiwania), zliczania oraz aktualizacji danych.

Instrukcja SELECT jest najczęściej używaną instrukcją SQL, posiada ona najszersze zastosowanie zarówno w pracy bezpośredniej, jak i w aplikacjach bazodanowych. Instrukcja posiada tylko kilka klauzul, ale wzbogacona o operatory zbiorowe i możliwości zagnieżdżania pozwala konstruować niezwykle złożone zapytania.

Składnia instrukcji SELECT jest następująca:

```
SELECT * | lista_pozycji
FROM lista_tabel
WHERE kwalifikacja
[GROUP BY lista grupowania]
[HAVING kwalifikacja_grup]
[ORDER BY lista_uporządkowania]
```

W części rozdziału poświęconej przykładowej bazie danych zamieszczono kilka zapytań SELECT o różnym stopniu złożoności.

Drugą grupę instrukcji podjęzyka DML stanowią instrukcje aktualizacji danych, zaprezentowane w poniższych przykładach.

Instrukcja wprowadzania danych:

```
INSERT INTO Kraje (symbol, nazwa, stolica, ludność)
      VALUES('PL','Polska','Warszawa',38000000);
```

Instrukcja modyfikacji danych:

```
UPDATE Kraje
        SET (ludność=39000000)
  WHERE symbol='PL';
```

Instrukcja usuwania danych:

```
DELETE FROM Kraje
       WHERE nazwa='Jugosławia';
```

Jak widzimy, instrukcje te są bardzo proste i zrozumiałe intuicyjnie.

Pozostałe podjęzyki używane są przez znacznie mniejsze grono użytkowników zaawansowanych i profesjonalnych. Przykłady niektórych poznaliśmy w części dotyczącej kontroli dostępu do danych. Tu więc ograniczymy się do wyliczenia i krótkich opisów reszty podjęzyków:

- *Embedded And Dynamic SQL* – podjęzyk obejmujący instrukcje umożliwiające umieszczanie instrukcji SQL w programach pisanych w innych językach programowania oraz dynamiczne generowanie instrukcji SQL w trakcie wykonywania programów (aplikacji); podjęzyk ten w zasadzie wykorzystywany jest jedynie przez programistów aplikacji bazodanowych;
- *Security* – podjęzyk implementujący metodę uznaniową kontroli dostępu do danych, obejmujący poznane wcześniej instrukcje GRANT i REVOKE;
- *Transaction Management* – nieliczna grupa instrukcji do zarządzania transakcjami; najważniejsze to instrukcje COMMIT (zatwierdzenie transakcji) i ROLLBACK (wycofanie transakcji);
- *Client-Server Execution and Remote Data Access* – podjęzyk umożliwiający komunikację z serwerami baz danych; używany przez zawodowych informatyków.

Późniejsze standardy nieco inaczej dzielą język SQL na podjęzyki, więc w literaturze można znaleźć różne podziały w zależności od standardu.

2.11. Podsumowanie

W rozdziale przedstawiono selektywnie i w dużym skrócie najważniejsze zagadnienia z obszaru baz danych zarówno od strony teoretycznej, jak i praktycznej. W części teoretycznej znalazły się treści niezbędne do zrozumienia zasad budowy i funkcjonowania baz danych. Wiedza ta potrzebna jest zarówno w projektowaniu bazy danych, jak i w późniejszym jej wykorzystaniu z uwzględnieniem zasad kontroli dostępu do danych i zachowania ich integralności. Rozdział uwzględnia także wybrane zagadnienia technologii baz danych pozwalające zrozumieć i stosować w praktyce rozwiązania adekwatne do obszaru zastosowań, na przykład różne typy organizacji plików determinujące ich przydatność i koszty przetwarzania. We fragmentach praktycznych – częściowo zagnieżdżonych w tekstach je objaśniających – zaprezentowano szereg przykładowych instrukcji języka SQL. Należy podkreślić przy tym, że celem nie było dostarczenie pełnej i systematycznej wiedzy na temat składni i wykorzystania tego języka, co wymagałoby rozbudowania tekstu rozdziału.

Pytania kontrolne

1. Co to jest baza danych?
2. Co to jest System Zarządzania Bazą Danych?
3. Zdefiniuj podstawowe pojęcia modelu relacyjnego.
4. Co to jest model encyjno-relacyjny i w jaki sposób jest on wykorzystywany?
5. Wymień i omów najważniejsze funkcje Systemu Zarządzania Bazą Danych.
6. Jakie znasz rodzaje wymogów integralności?

7. Wymień i scharakteryzuj operatory relacyjne.
8. Wymień i opisz własności transakcji (ACID).
9. Scharakteryzuj organizacje sekwencyjno-indeksowe stosowane w bazach danych.
10. Scharakteryzuj metodę uznaniową kontroli dostępu do danych w bazach danych.
11. Z jakich podjęzyków składa się język standard języka SQL92?

Studium przypadku

Przedstawiona poniżej przykładowa baza danych została zaprojektowana dla hipotetycznej organizacji handlowej prowadzącej sprzedaż różnych wyrobów na terenie Europy i posiadającej jeden centralny magazyn w Polsce.

Firma posiada oddziały w różnych krajach Europy. W dużych krajach może istnieć kilka oddziałów, w innych po jednym, a w niektórych może ich nie być w ogóle. Klienci firmy są rezydentami różnych krajów europejskich, zarówno należących do Unii Europejskiej, jak i niebędących jej członkami. Dla uproszczenia przyjmujemy, że sprzedaż prowadzona jest wyłącznie dla klientów korporacyjnych. Z tego samego powodu pomijamy podatki, cła czy koszty wysyłki, a wszystkie wartości wyrażamy w walucie euro. Firma zatrudnia sprzedawców, którzy mogą mieszkać w tej samej miejscowości, w której jest biuro, ale może to być też inna miejscowość, a nawet inny kraj.

Dokumentem sprzedaży jest faktura; jest to dokument wielopozycyjny. Baza danych nie obejmuje dostawców ani dokumentów dostaw, lecz przechowywane są zapasy towarów oraz dane o cenach zakupu; można przyjąć, że są to ceny średnioważone.

Baza danych stanowi daleko uproszczony obraz rzeczywistości, a niektóre atrybuty uwzględniono w niej z powodów czysto dydaktycznych.

Pierwszym etapem projektowania bazy danych jest projektowanie konceptualne[9], które realizowane jest przy zastosowaniu modelu encyjno-relacyjnego.

9 W rzeczywistości projektowanie bazy danych poprzedzone zostaje analizą, która nie jest jednak przedmiotem teorii baz danych; to proces subiektywny oparty na wiedzy, intuicji i doświadczeniu analityków.

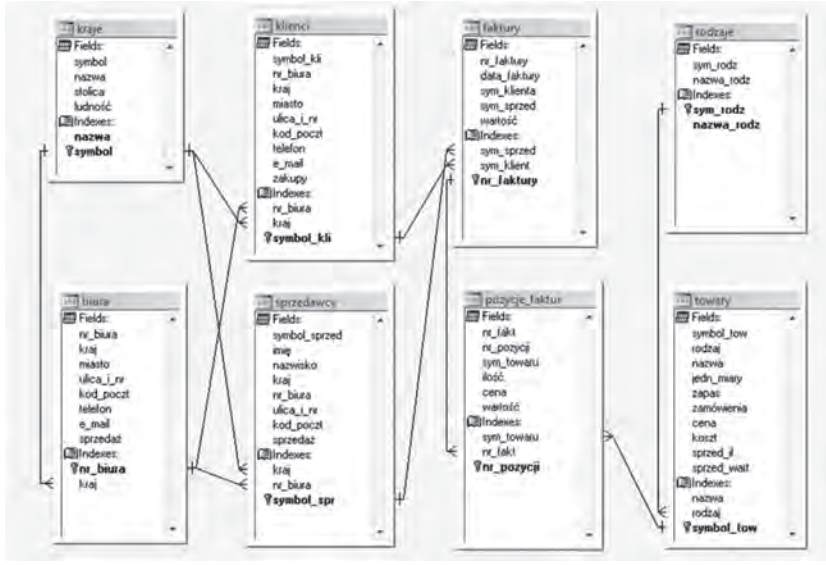
Z przedstawionych założeń wynika, że nasza baza danych powinna obejmować następujące encje:

KRAJE
BIURA
SPRZEDAWCY
KLIENCI
WYROBY
RODZAJE wyrobów
FAKTURY
POZYCJE faktur

Każda encja przekłada się najczęściej na jedną tabelę bazy danych, jednak w przypadku dokumentów wielopozycyjnych, jakimi są faktury, konieczne jest utworzenie dwóch oddzielnych tabel. Jedna z nich (w naszym projekcie FAKTURY) przechowuje dane globalne dokumentu – w tym przypadku np. numer faktury, data faktury, symbol klienta i wartość faktury. Druga zawierać będzie dane (pozycje) z poszczególnych wierszy dokumentu wielopozycyjnego, czyli w tym przypadku z faktury.

Graficznie projekt bazy danych można przedstawić następująco:

Rys. 2.1. Projekt przykładowej bazy danych wykorzystujący model encyjno-relacyjny



Źródło: opracowanie własne.

Dla każdej encji – docelowo tabeli – należy wyspecyfikować atrybuty, klucz główny (na schemacie oznaczony kluczykiem) oraz klucze drugorzędne (pozostałe pozycje indeksowe). Encje połączone są relacjami; na rysunku są to linie łączące poszczególne encje. Na końcach linii symbolizujących relacje znajdują się znaki poprzecznej kreski oraz „kurzej stopki". Są to relacje typu „1 do wielu", co oznacza, że każda encja znajdująca się na końcu linii z poprzeczną kreską może się łączyć z wieloma egzemplarzami encji na końcu linii z kurzą stopką. Istnieją też inne typy relacji, tu je pomijamy. Wszystkie te elementy zawiera prezentowany projekt.

Kolejnym etapem projektowania bazy danych jest translacja projektu encyjno-relacyjnego do modelu relacyjnego (nazywana też projektowaniem logicznym), czyli utworzenie bazy danych i poszczególnych tabel. Najczęściej każda encja stanowi podstawę do utworzenia jednej tabeli bazy danych, ale zdarza się, że na skutek zabiegu normalizacji pojawia się konieczność utworzenia dodatkowych tabel. W naszym projekcie nie ma takiej konieczności.

Aby utworzyć tabele bazy danych, należy dla każdego atrybutu każdej encji zdefiniować domenę (typ danych) oraz ewentualnie – w zależności od domeny – rozmiar. W praktyce domeny i rozmiary atrybutów często definiuje się już na etapie projektowania konceptualnego, zwłaszcza w przypadku mniejszych projektów.

Tabele bazy danych można tworzyć za pomocą instrukcji podjęzyka SQL DDL (*Data Definition Language*) CREATE TABLE. Mogą też być wykorzystywane wbudowane i zewnętrzne narzędzia wspierające projektowanie baz danych.

Poniżej zamieszczono kilka przykładowych instrukcji CREATE TABLE tworzących tabele bazy danych według utworzonego wcześniej projektu.

```
CREATE TABLE Kraje
    (symbol         CHAR(3) PRIMARY KEY,
    nazwa           CHAR(20) UNIQUE,
    stolica         CHAR(20) NOT NULL,
    ludność         NUMBER(4,1),
                    CHECK ludność>=0);

CREATE TABLE Biura
    (nr_biura       NUMBER(3) PRIMARY KEY,
    kraj            CHAR(3) REFERENCES Kraje,
```

```
miasto          CHAR(20) NOT NULL,
ulica_i_nr      CHAR(40) NOT NULL,
kod_poczt       CHAR(8) NOT NULL,
telefon         CHAR(15),
e_mail          CHAR(50));
```

```
CREATE TABLE Sprzedawcy
(id_sprzedawcy      CHAR(5) PRIMARY KEY,
nazwisko            CHAR(30) NOT NULL,
imię                CHAR(20) NOT NULL,
nr_biura            NUMBER(3) REFERENCES Biura,
kraj                CHAR(3) REFERENCES Kraje,
miasto              CHAR(20) NOT NULL,
ulica_i_nr          CHAR(40) NOT NULL,
kod_poczt           CHAR(8) NOT NULL,
telefon             CHAR(15),
e_mail              CHAR(50));
```

```
CREATE TABLE Klienci
(id_klienta         CHAR(5) PRIMARY KEY,
nazwa               CHAR(30) NOT NULL,
nr_biura            NUMBER(3) CONSTRAINT Klienci_FK_Biura
                    REFERENCES Biura,
kraj                CHAR(3) REFERENCES Kraje,
miasto              CHAR(20) NOT NULL,
ulica_i_nr          CHAR(40) NOT NULL,
kod_poczt           CHAR(6) NOT NULL,
telefon             CHAR(15));
```

```
CREATE TABLE Towary
(symbol             CHAR(7) PRIMARY KEY,
nazwa               CHAR(50) UNIQUE,
jedn_miary          CHAR(7) NOT NULL,
cena                 NUMBER(7,2),
zapas               NUMBER(9,3) DEFAULT 0,
zamówienia          NUMBER(9,3),
sprzedaż_opr*       NUMBER(10,3),
                    CHECK cena>=0);
```

*sprzedaż_opr – sprzedaż od początku roku

W praktyce wymogi integralności powinny być nazywane, tu dla uproszczenia umieszczono nazwę tylko jednego wymogu w definicji tabeli Klienci.

W przypadku konieczności zmian w projekcie można użyć instrukcji ALTER TABLE pozwalającej zmieniać struktury tabel oraz wymogi integralności. Na przykład chcielibyśmy rozszerzyć dane klientów o adres e-mail:

```
ALTER TABLE Klienci
ADD e_mail CHAR(50)
```

W rzeczywistym projektowaniu baz danych, szczególnie rozbudowanych, po etapie projektowania logicznego przeprowadzana jest normalizacja. Jest to proces obiektywny, realizowany za pomocą eleganckiej, jak się to określa w podręcznikach, teorii normalizacji. Tu pomijamy ten etap.

Kolejnym etapem projektowania dużych baz danych jest projektowanie fizyczne, w którym na podstawie analizy statystycznej spodziewanych zapytań projektowane są indeksy.

Wreszcie, niezależnie od wielkości projektu, baza danych wypełniana jest danymi próbnymi i następuje jej strojenie oraz testowanie.

Baza danych zwykle stanowi fundament aplikacji bazodanowej, czyli konkretnego systemu realizującego przewidziane w projekcie funkcje i dopiero w procesie rozwijania i testowania aplikacji następuje jej realna weryfikacja. Baza danych zawierająca dane rzeczywiste (zwana produkcyjną bazą danych) może być wykorzystywana do wyszukiwania danych za pomocą bezpośrednich zapytań pisanych przez użytkowników. Podstawowym narzędziem zapytań do profesjonalnych baz danych jest instrukcja SELECT języka SQL. Poniżej zamieszczono przykładowe zapytania do zaprojektowanej wcześniej bazy danych.

Proste zapytania do jednej tabeli:

- Wylistuj wszystkie dane wszystkich klientów

```
SELECT *
  FROM Klienci;
```

- Wylistuj symbol, nazwę i miasto klientów rezydujących w Holandii

```
SELECT symbol, nazwa i miasto
  FROM Klienci
  WHERE kraj='NL';
```

- Wylistuj kraje wraz z liczbą klientów

```
SELECT kraj, COUNT(*)
  FROM Klienci
  GROUP BY kraj;
```

Zapytania do wielu tabel:

- Wylistuj biura wraz z zatrudnionymi w nich sprzedawcami:

```
SELECT Biura.nr_biura, Biura.miasto, symbol_sprzedawcy, nazwisko_
sprzedawcy

  FROM Biura, Sprzedawcy
  WHERE Biura.nr_biura = Sprzedawcy.nr_biura;
```

Zapytania zagnieżdżone:

- Wyszukaj klienta, który dokonał największych zakupów od początku roku

```
SELECT symbol_klienta, nazwa_klienta, zakupy
  FROM Klienci
  WHERE zakupy = (SELECT MAX(zakupy) FROM Klienci);
```

- Wylistuj klientów (symbol, nazwa), którzy nic nie kupili:

```
SELECT symbol_klienta, nazwa
  FROM Klienci
  WHERE symbol_klienta NOT IN
  (SELECT symbol_klienta
  FROM Faktury);
```

Literatura

Elmasri R., Navathe S. (2005), *Wprowadzenie do systemów baz danych*, Wydawnictwo Helion, Gliwice.

Kurzyjamski R. (2003), *Zmierzch podejścia tradycyjnego*, Zeszyty Naukowe WSHE, Łódź.

Matosek W. (2006), *Język SQL w bazie danych Oracle 10g*, Wydawnictwo Uniwersytetu Gdańskiego, Gdańsk.

Ramakrishnan R., Gehrke J. (2003), *Database Management Systems*, 3rd edition, McGraw-Hill Higher Education, New York.

Ullman J.D., Widom J. (2000), *Podstawowy wykład z systemów baz danych*, WNT, Warszawa.