Maria Chałon

Ochrona i bezpieczeństwo danych oraz tendencje rozwojowe baz danych

Oficyna Wydawnicza Politechniki Wrocławskiej
Wrocław 2007

Recenzenci
Piotr KOCIATKIEWICZ
Aleksander ZGRZYWA

Opracowanie redakcyjne i korekta
Aleksandra WAWRZYNKOWSKA

Projekt okładki
Zofia i Dariusz GODLEWSCY

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.wroc.pl
e-mail: oficwyd@pwr.wroc.pl

ISBN 978-83-7493-360-5

Drukarnia Oficyny Wydawniczej Politechniki Wrocławskiej. Zam. nr 834/2007.

Przedmowa

Jednym z priorytetowych zadań podczas projektowania systemów informatycznych jest zapewnienie bezpieczeństwa przechowywanych, przetwarzanych i przesyłanych danych. Szeroko rozumiany problem bezpieczeństwa danych polega na ich zabezpieczeniu przed umyślnym lub przypadkowym ujawnieniem, modyfikacją lub zniszczeniem. Najważniejsze atrybuty bezpiecznego systemu to: poufność, autentyczność, integralność, dostępność oraz niezawodność. Ich zapewnienie wymaga równoległego stosowania rozmaitych metod i środków ochrony danych. Na ogół odbywa się to kosztem funkcjonalności i efektywności systemu. Dlatego ważne jest określenie stopnia bezpieczeństwa, jaki powinno się zapewnić określonemu systemowi.

Istnieją dwa podejścia w zakresie polityki bezpieczeństwa. Pierwsze polega na przydziale systemowi tak zwanej klasy bezpieczeństwa, określonej w ramach szeroko stosowanego standardu TCSEC (Trusted Computer Systems Evaluation Criteria). Dokument ten, znany też pod nazwą Orange Book, zawiera opis kryteriów przydziału analizowanych systemów do odpowiednich klas bezpieczeństwa, informacje na temat sposobu wykonywania analiz bezpieczeństwa, a także zalecenia dotyczące zapewnienia bezpieczeństwa systemu informatycznego. Drugie podejście polega na wykonaniu ekspertyz, określanych mianem analizy ryzyka. Cechą charakterystyczną takiej analizy jest to, że bierze się w niej pod uwagę prawdopodobieństwo wystąpienia danego zagrożenia przy uwzględnieniu priorytetu zagrożeń.

Do klasycznych zadań każdego Systemu Zarządzania Bazą Danych (SZBD) należy zapewnienie ochrony integralności danych. Można wyróżnić integralność statyczną i transakcyjną. Statyczne więzy definiują poprawny stan bazy danych pomiędzy kolejnymi operacjami wykonywanymi w bazie, czyli w pewnym stopniu chronią przed niepoprawną modyfikacją danych. Mechanizmy integralności transakcyjnej chronią spójność bazy danych w przypadku awarii sprzętowej systemu, współbieżnie realizowanych operacji przez wielu użytkowników lub błędów oprogramowania.

Osobny problem stanowi archiwizacja i odtwarzanie danych. Jest to zbiór strategii i operacji, mających na celu zabezpieczenie bazy danych przed skutkami awarii i umożliwiających jej rekonstrukcję. Archiwizacja zabezpiecza nie tylko przed utratą danych w wyniku awarii nośnika, lecz również przed nieautoryzowanymi zmianami, dokonanymi przez nieupoważnionych użytkowników. Bezpieczeństwo bazy danych

zależy od poprawności i częstości wykonywania kopii. Kopia stanowi stan bazy w chwili archiwizacji. Zmiany w bazie, powstałe między ostatnią archiwizacją a awarią, giną bezpowrotnie. Odtworzenie stanu sprzed awarii jest możliwe tylko dzięki prowadzeniu dziennika transakcji. Niezawodność systemu komputerowego zwiększa się również przez duplikowanie komputerów, podzespołów lub całych systemów informatycznych. Im ważniejsza informacja, im większa musi być dostępność do systemu – tym bardziej rosną wymagania wobec nadmiarowości sprzętu i oprogramowania. System bazodanowy powinien być tak zorganizowany, aby zapewniać stałą gotowość operacyjną i mechanizmy pozwalające zminimalizować przestoje serwera związane z awariami sprzętu.

Ze względu na przedstawione problemy układ książki jest następujący.

Część pierwsza zawiera opis zagrożeń, na które jest narażony zarówno system, aplikacja, jak i sprzęt komputerowy, oraz opis modeli bezpieczeństwa i przyjęte standardy bezpieczeństwa. Część druga wprowadza w zagadnienia ochrony przed nieupoważnionym dostępem umyślnym i przypadkowym. Część trzecia dotyczy problemu integralności danych. Szczególny nacisk położono na integralność transakcyjną, a dokładnie na dzienniki transakcji, które stanowią jedyne źródło informacji zarówno w razie awarii systemu, jak i w przypadku nieprawidłowości występujących w wyniku współbieżnego dostępu wielu użytkowników. W części czwartej omówiono problemy archiwizacji programowej i sprzętowej. Na zakończenie, w części piątej, przedstawiono tendencje rozwojowe baz danych. Współczesne bazy danych rozwijają się w różnych kierunkach. Duża grupa zastosowań to integracja informacji. Korzysta się z wielu źródeł informacji i buduje z nich jedną dużą bazę danych; może to być również baza wirtualna. Użytkownik korzystający z informacji ma wrażenie, że pochodzą one z jednego źródła. Podstawowym problemem jest sposób interpretacji i przekształcenia danych po wyszukaniu ze źródła. Służą do tego między innymi wrapery i ekstraktory – programy, które umożliwiają dostosowanie źródeł informacji do schematów rozproszonych. Ciekawym rozwiązaniem jest mediator. Nie przechowuje on żadnych własnych danych, lecz przekształca zapytanie użytkownika w ciąg zapytań do różnych źródeł, a następnie zbiera odpowiedzi cząstkowe, syntetyzuje i przekazuje odpowiedź użytkownikowi.

Publikacja ta powstała jako rozszerzona wersja treści zajęć, prowadzonych dla studentów kierunku Informatyka. Jest kontynuacją wydanej kilka lat temu książki autorki *Systemy baz danych. Wprowadzenie*. Zawarty w niej materiał zainteresuje nie tylko studentów informatyki, zajmujących się teorią i praktycznym tworzeniem baz danych, lecz także wszystkich informatyków, którzy w swej pracy zawodowej zajmują się bezpieczeństwem i ochroną danych we współczesnych systemach zarządzania bazami danych.

CZĘŚĆ I

Ataki, awarie, modele bezpieczeństwa

Kto bez uprawnienia uzyskuje informacje dla niego nie przeznaczoną otwierając zamknięte pismo, podłączając się do przewodu służącego do przekazywania informacji lub przełamując elektroniczne, magnetyczne albo inne szczególne jej zabezpieczenie, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 2.

§ 267 p. 1 Kodeksu Karnego 1998 r.

1. Zagrożenia dla danych

Dane przechowywane w bazach danych mogą być narażone na wiele różnych zagrożeń. Zakres zagrożeń jest ogromny, a każde z nich w istotny sposób wpływa na bezpieczeństwo przechowywanych w systemie informacji. Zagrożenia możemy podzielić na:

- ataki na systemy,
- wirusy komputerowe,
- awarie sprzętowe,
- ujawnienie tajnych danych,
- usunięcie, zniszczenie lub niepożądana modyfikacja danych,
- błędy w oprogramowaniu,
- niedostateczne testowanie,
- kwestionowanie transakcji.

1.1. Ataki na systemy

Ataki mogą być przeprowadzane z systemów znajdujących się poza atakowaną siecią – określane jako zdalne – gdzie atakujący nie posiada jeszcze żadnych uprawnień w systemie atakowanym, oraz z systemów znajdujących się w atakowanej sieci – zwane lokalnymi – gdzie atakujący ma już dostęp do systemu i próbuje zwiększyć swoje uprawnienia. Atak może być świadomy – atakujący zdaje sobie sprawę z tego, co robi i jakie konsekwencje mogą z tego wyniknąć, na przykład atak w celu uzyskania konkretnych informacji. Atak może być nieświadomy – atakujący przypadkowo dokonuje ataku i przez błąd programu obchodzi system autoryzacji, uzyskując prawa administratora. Ataki mogą być aktywne, jeśli w ich wyniku system komputerowy traci integralność. Atakiem aktywnym może być także modyfikowanie strumienia danych lub tworzenie danych fałszywych. Taki atak popularnie nazywa się „człowiek w środku" (ang. man in the middle). Ataki mogą być pasywne – atak ten polega na wejściu do systemu bez dokonywania żadnych zmian, na przykład kopiowanie pewnej ilości ważnych danych niepowodujące zmian w działaniu programów. Atakiem pa-

sywnym jest również podsłuchiwanie lub monitorowanie przesyłanych danych. Ataki pasywne są bardzo trudne do wykrycia, ponieważ nie wiążą się z modyfikacjami jakichkolwiek danych.

Ataki można również podzielić ze względu na przepływ informacji. Przykładem może być przechwycenie informacji. Jest to atak na poufność. Występuje wtedy, gdy ktoś niepowołany uzyskuje dostęp do zasobów systemu komputerowego. Przykładowo jest nim podsłuch pakietów w sieci w celu przechwycenia danych, nielegalne kopiowanie plików lub programów. Inny przykład ataku to fizyczne zniszczenie fragmentu komputera lub sieci, np. uszkodzenie dysku, przecięcie linii łączności między komputerem a drugim obiektem lub uniemożliwienie działania systemu zarządzania plikami. Zdarzają się również ataki polegające na zdobyciu dostępu do zasobów przez niepowołaną osobę, która wprowadza do nich jakieś zmiany w celu uzyskania wyższych praw, lub dąży do utrzymania dostępu do danego systemu. Podrobienie informacji jest atakiem opierającym się na autentyczności. Podczas przesyłania danych z jednego do drugiego komputera trzeci komputer blokuje tę czynność, uniemożliwiając dalsze przesyłanie, a sam wprowadza do systemu drugiego komputera fałszywe obiekty. Zdarzają się również ataki na samą metodę uwierzytelniania.

Do najbardziej znanych ataków należą [6, 7, 10, 11, 14, 15]:

- Podsłuch (ang. sniffing) – atak polegający na podsłuchiwaniu informacji przesyłanych siecią. Technika ta jest stosunkowo prosta, jednak wymaga dostępu do medium transmisyjnego, po którym płyną dane. Podsłuch jest metodą pasywną, to znaczy nie modyfikuje żadnych danych.
- Zgadywanie haseł (ang. brute force attack) poprzez zgadywanie inteligentne, ataki słownikowe czy wykorzystanie przez włamywacza specjalnych programów zawierających ogromny zbiór słów, które kolejno sprawdzają wszystkie możliwe kombinacje, w celu uzyskania właściwego hasła.
- Podgląd sieciowy (ang. snooping), czyli namierzanie sieci z wykorzystaniem zaawansowanych analizatorów sieci, aby następnie wybrać najefektywniejszą metodę ataku na sieć.
- Blokowanie działania (ang. denial of service), polegające na wysyłaniu dużej partii tych samych informacji żądających wykonania usług, które doprowadzają do przeładowania pamięci komputera, a w efekcie sparaliżowania jego działania i uniemożliwienia korzystania z wybranych usług, uruchomionych na tym komputerze. Bardziej niebezpieczną odmianą tego ataku jest rozproszona odmowa usługi (ang. distributed denial of service), przeprowadzana nie z jednego komputera, którego łącza i zasoby sprzętowe są najczęściej dużo słabsze w stosunku do zasobów atakowanego urządzenia, ale z wielu komputerów równocześnie.
- Szukanie luk (ang. seeking vulnerabilities), polegające na wyszukiwaniu słabych punktów, będących wynikiem błędów popełnianych przez projektantów i programistów tworzących daną strukturę bazową lub błędów w językach skryptowych (ang. SQL injection).

• Maskarada (ang. spoofing) to szereg technik, zmierzających do podszywania się pod inny komputer w sieci czy innego użytkownika wcześniej uwierzytelnionego. Jeżeli podszywający się zdoła wysłać pakiety w imieniu autoryzowanego wcześniej klienta, to może wyrządzić wiele szkód, takich jak zerwanie połączenia, modyfikację sesji czy przejęcie całej sesji.

• Przepełnienie bufora (ang. buffer overflow) przez błędnie napisane aplikacje czy niepoprawne instrukcje w kodzie źródłowym. Ataki przepełnienia bufora mają na celu przejęcie praw superużytkownika.

• Terroryzm sieciowy (ang. cyber terrorism) – szantażowanie firm polegające na wykorzystywaniu słabych zabezpieczeń systemów.

• Socjotechnika (ang. socjal engineering) – manipulowanie ludźmi w taki sposób, aby zmusić ich do ujawnienia poufnych informacji o systemie.

Ataki ewoluują wraz z rynkiem informatycznym. Początkowo atakowano trywialne hasła i słabe punkty systemu. Lata dziewięćdziesiąte to przechwytywanie haseł, ataki na rutery. Od 2000 roku pojawił się pocztowy spam (ang. e-mail spamming) aż do ataków na komunikatory (ang. spimming) w 2005 roku.

W trakcie przeprowadzania ataków można wyróżnić pewne charakterystyczne fazy:

• wyszukiwanie słabości systemu – skanowanie,
• wyznaczenie celu, czyli określenie niezabezpieczonych usług,
• atak na system, modyfikacja danych,
• usuwanie śladów i propagacja ataku.

1.2. Wirusy komputerowe

Wirusy są to programy komputerowe, które zakłócają poprawną pracę systemów i aplikacji. Utrudniają, a nawet czasami uniemożliwiają pracę na komputerze. Mogą niszczyć i zmieniać dane oraz oprogramowanie zawarte na dysku, jak również wykradać dane, przenosić i instalować inne typy złośliwych programów. Wirusy [2, 7] potrafią powielać się wewnątrz plików lub w sektorach dysku. Istnieją również wirusy, które udają pożyteczne programy i zdobywają zaufanie użytkowników. Są to tak zwane konie trojańskie. Koń trojański nie powiela się jak wirus, ale jego działanie jest równie szkodliwe. Ukrywa się pod nazwą, która użytkownikowi wydaje się przydatna w pracy. Oprócz tego wykonuje szkodliwe operacje w tle, na przykład zbiera informacje na temat haseł i danych strategicznych. Znane są makrowirusy, programy tworzone za pomocą makrorozkazów, rozpowszechniające się z takimi popularnymi dokumentami jak na przykład edytory tekstu. Inny typ to grupa programów samopowielających się, zwanych czasami bakteriami, które zużywają takie zasoby komputera jak procesor czy pamięć. Podobne do bakterii są robaki, ale polem ich działania jest sieć komputerowa. Robak tak jak wirus kopiuje się z jednego komputera na drugi.

Robi to automatycznie, przejmując kontrolę nad funkcjami komputera odpowiedzialnymi za przesyłanie informacji. Robak rozpowszechnia się samodzielnie i powiela wielokrotnie. Robaki obciążają i blokują serwery, zmniejszając przepustowość sieci. Szczególna grupa wirusów, które aktywują się w konkretnych okolicznościach, znane są pod nazwą bomby logicznej. Bomba logiczna to kod sprawdzający istnienie w systemie określonego zbioru warunków. W przypadku ich spełnienia jest wykonywana funkcja, która ma za zadanie na przykład skasowanie danych, zmianę haseł, czy zawieszenie systemu. Najczęściej stosowanym typem bomb logicznych są bomby czasowe (ang. time bomb), które zaczynają działać w określonym czasie.

1.3. Awarie sprzętowe

Nie istnieje bezawaryjnie pracujący system komputerowy. Ciągłe ulepszanie sprzętu komputerowego zmniejsza jego awaryjność, ale całkowicie awariom nie zapobiega. Można wyróżnić następujące awarie [4, 6, 9, 14]:

- drobne i pojedyncze awarie podzespołów, takich jak: płyty główne, procesory, pamięci operacyjne, monitory,
- uszkodzenia dotyczące nośników danych: dysków twardych, optycznych itp.,
- awarie sieci komputerowych,
- trwałe uszkodzenia całego sprzętu (pożar, kradzież itp.).

Wymienione awarie powodują częściową lub całkowitą utratę danych. Dlatego każdy system musi posiadać odpowiednie mechanizmy, umożliwiające zarówno programową, jak i sprzętową ochronę danych. Jeśli na przykład następuje awaria przestrzeni dyskowej czy serwera, uniemożliwiająca całkowicie pracę systemu, to właściwą techniką zabezpieczeń jest: proces dublowania w czasie rzeczywistym zapisu danych na dowolnym urządzeniu (ang. mirroring), składowanie danych (ang. backup), replikacja, czy dodatkowy sprzęt w postaci macierzy RAID. Utrata zasilania zmusza do zabezpieczenia systemu przez dodatkowe urządzenia typu UPS, redundancje zasilaczy, odpowiedni rozdział obwodów zasilania. Wymienione mechanizmy opisano w czwartej części książki.

1.4. Ujawnienie tajnych danych

W celu zminimalizowania ryzyka ujawnienia utajnionych informacji przechowywanych w pamięci komputera, czy przesyłanych pomiędzy komputerami, należy przede wszystkim ograniczyć metody, które umożliwiają uzyskanie dostępu do tych danych, a także ograniczyć liczbę osób, które mają dostęp do tych danych. Systemy należy projektować ze szczególnym uwzględnieniem aspektu bezpieczeństwa, wła-

ściwej konfiguracji serwera i oprogramowania. Trzeba pozbyć się wszystkich zbędnych usług, co pozwoli na zredukowanie słabych punktów systemu. Należy zaimplementować mechanizm uwierzytelnienia i dokładnego testowania, a także zadbać o bezpieczne przesyłanie danych, dokonując wyboru odpowiedniego protokołu przesyłania danych.

1.5. Usunięcie, zniszczenie lub niepożądana modyfikacja danych

Utrata danych jest o wiele bardziej kosztowna niż jej ujawnienie. Należy zabezpieczyć dane przed umyślnym lub przypadkowym zniszczeniem. Tak samo zdarzające się od czasu do czasu awarie systemu mogą spowodować bezpowrotną utratę zgromadzonych danych. Okazuje się jednak, że o ile usunięcie lub zniszczenie danych na ogół jest szybko zauważane i często odbudowywane za pomocą kopii zapasowych, to może upłynąć dużo czasu zanim zostanie odkryta modyfikacja danych. Jeśli zorientujemy się, że naruszono zabezpieczenia systemowe, nie jesteśmy w stanie stwierdzić jednoznacznie, które pliki są zmodyfikowane. Szczególnie dotyczy to tych plików bazy danych, które zmieniają się przez cały czas funkcjonowania systemu. Nie wiemy wtedy, czy dane zostały zmodyfikowane w wyniku normalnych procesów systemowych, czy przez włamywacza. W trakcie modyfikacji zmianie ulega nie tylko zawartość plików, lecz również mogą być podmieniane całe pliki na ich sabotażowe wersje, umożliwiające włamywaczowi wejście do systemu.

1.6. Błędy w oprogramowaniu i niedostateczne testowanie

Pojawiające się błędy w oprogramowaniu mogą spowodować wiele przykrych następstw, takich jak: dziury w systemie bezpieczeństwa, dostarczanie miernej jakości usług, a nawet straty finansowe. Prawdopodobieństwo wystąpienia błędów w ostatecznej wersji oprogramowania jest tym większe, im bardziej niejednoznaczna i ogólna jest dokumentacja systemu. Również założenia przyjmowane przez projektantów powinny być bardzo dokładnie udokumentowane, aby legalny użytkownik miał jasno i przejrzyście sformułowane warunki wprowadzania i przechowywania danych i nie otrzymywał informacji o błędach w systemie w przypadku, gdy projektant nie przewidział pewnych sytuacji. Niemniej jednak przewidzenie i przetestowanie wszystkich możliwości i warunków, które mogą się pojawić przy okazji wprowadzania danych do różnych aplikacji pracujących pod kontrolą wielu systemów operacyjnych, wykorzystujących różne platformy sprzętowe, jest niemożliwe. Dlatego należy bardzo dokład-

nie i sumiennie zaplanować porządek przeprowadzenia testu, aby obejmował on analizę działania wszystkich funkcji zainstalowanych aplikacji na określonej grupie komputerów. Zaleca się, aby testy przeprowadzały osoby niezależne, niezwiązane bezpośrednio z programem. Istnieje wtedy większe prawdopodobieństwo wyszukania błędów, popełnionych przez programistów, których wcześniej nie wykryto.

1.7. Kwestionowanie transakcji

Bardzo często zdarzają się przypadki, w których jedna ze stron zaangażowanych w proces transakcji kwestionuje swój udział w niej. Rozwiązanie tego problemu sprowadza się do zapewnienia wiarygodności wysyłanych wiadomości, na przykład przez podpis elektroniczny, aby żadna ze stron nie mogła zaprzeczyć udziału w przeprowadzonej transakcji. Dodatkowo każda ze stron mogłaby niepodważalnie wykazać stronie trzeciej fakt uczestnictwa w zrealizowanej transakcji. Każdy uczestnik transakcji powinien być specjalnie przeszkolony do pracy z systemem i wyczulony na ataki. Często bowiem nie oprogramowanie czy sprzęt, lecz człowiek jest najsłabszym ogniwem narażającym system na niebezpieczeństwo.

2. Modele bezpieczeństwa

Istotną sprawą jest waga przechowywanej informacji. Im ważniejsza informacja, tym lepsze powinno być zabezpieczenie, a co za tym idzie – ponoszone są większe koszty. Nowoczesne SZBD realizują założenia jednego lub dwóch [2, 4, 15] podstawowych modeli bezpieczeństwa:

- modelu kontroli uznaniowej – model Lampsona (ang. Discretionary Access Control – DAC),
- modelu kontroli obowiązkowej – model Bella i La Paduli, opracowany w 1976 roku (ang. Mandatory Access Control – MAC).

Do opisu tych modeli służą takie pojęcia jak obiekt oraz podmiot. Obiekt oznacza jednostkę, która zawiera lub jest w stanie przyjąć informacje. Obiekt jest celem ochrony. Podmiotem jest użytkownik bazy danych, odpowiedzialny za przepływ informacji między obiektami i podmiotami, czyli użytkownikami. Modele te, wykorzystując teorię zbiorów, definiują pojęcia stanu bezpieczeństwa i elementarnych trybów dostępu do obiektu oraz określają reguły przyznawania podmiotom określonych trybów dostępu do obiektów. Zawierają także dowód twierdzenia, zwanego podstawowym twierdzeniem bezpieczeństwa (ang. Basic Security Theorem), mówiącego iż zastosowanie dowolnej sekwencji reguł do systemu, który jest w stanie bezpiecznym, spowoduje przejście systemu do innego stanu, który jest również bezpieczny.

Inna metoda oceny poziomu bezpieczeństwa opiera się na ekspertyzach, które – jak wspomniano w przedmowie – są określane mianem analizy ryzyka. Ocena ryzyka w silnym stopniu uwzględnia prawdopodobieństwo wystąpienia danego zagrożenia. Można przyjąć zasadę, że byłoby nierozsądne zajmować się stosunkowo mało prawdopodobnym ryzykiem wówczas, gdy nie zostały jeszcze odparte zagrożenia potencjalnie bardziej kosztowne w skutkach.

2.1. Model kontroli uznaniowej

W modelu kontroli uznaniowej DAC dla każdego podmiotu, czyli użytkownika, definiuje się reguły dostępu, które określają uprawnienia tego podmiotu do danego

obiektu, czyli jednostki chronionej. Dostęp do obiektu jest określany na podstawie wartości logicznej funkcji dostępu.

Funkcja ta działa na następujących zbiorach:

O =$\{o_1, o_2, o_3, ... o_k\}$ zbiorze obiektów,

S = $\{s_1, s_2, s_3, ... s_l\}$ zbiorze podmiotów,

T = $\{t_1, t_2, t_3, ... t_m\}$ zbiorze dozwolonych operacji (praw dostępu),

P = $\{p_1, p_2, ... p_n\}$ opcjonalnym zbiorze predykatów, z których każdy definiuje zakres dostępu, czyli ogranicza uprawnienia pewnymi warunkami, np. prawo dostępu jest ważne w określone dni tygodnia.

Funkcja dostępu f jest definiowana jako f: O×S×T×P → {*True, False*}, natomiast mianem reguły dostępu określa się krotkę o postaci ‹*o,s,t,p*›. Jeżeli dla pewnej krotki o postaci ‹*o,s,t,p*› wartością funkcji dostępu f jest wartość *True*, to podmiot *s* ma uprawnienia *t* do dostępu do obiektu *o* w zakresie określonym przez predykat *p*.

Model DAC oferuje użytkownikom dużą elastyczność i swobodę współdzielenia zasobów. Właściciel zasobu może decydować o jego atrybutach i uprawnieniach innych użytkowników systemu względem tego zasobu. Powszechnym zagrożeniem DAC jest niefrasobliwość przydziału uprawnień i niewystarczająca ochrona zasobów.

2.2. Model kontroli obowiązkowej

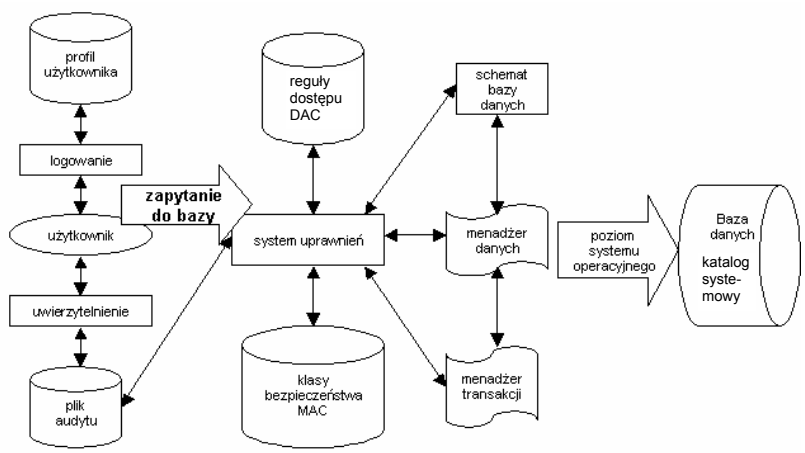
Obowiązkowy model bezpieczeństwa MAC, zwany również wielopoziomowym modelem ochrony danych lub systemem zaufanym, jest wykorzystywany w systemach wymagających szczególnie silnej ochrony informacji. W systemach takich klasyfikuje się przechowywane dane i oceny stopnia zaufania użytkowników. Każdemu obiektowi i podmiotowi przypisuje się pewne parametry ochrony, zwane klasami bezpieczeństwa lub poziomem zaufania. Klasę bezpieczeństwa Ψ definiujemy następująco: Ψ = (P, K), gdzie P oznacza poziom tajności obiektu (ściśle tajne, tajne, poufne, jawne), a K – kategorię zaufania podmiotu. Poziomy tajności są ściśle uporządkowane. Sterowanie dostępem przy wykorzystaniu modelu kontroli obowiązkowej jest oparte na następujących regułach:

- MAC1 – podmiot, czyli użytkownik, nie może uruchomić procesu o wyższym poziomie tajności niż jego aktualna kategoria zaufania,
- MAC2 – podmiot *s* może czytać daną *d*, gdy Ψ (*s*) ≥ Ψ (*d*), co oznacza, że użytkownik nie może odczytać informacji o wyższym poziomie tajności niż jego kategoria zaufania. Reguła ta nazywa się **zasadą czytania w dół**.
- MAC3 – podmiot *s* może zapisać daną *d*, gdy Ψ (*s*) ≤ Ψ (*d*), co oznacza, że użytkownik nie może zapisywać informacji na niższym poziomie ochrony, gdyż groziłoby to jej odtajnieniem. Reguła ta nazywa się **zasadą pisania w górę**.

MAC pozwala łatwiej zrealizować założenia polityki bezpieczeństwa i konsekwentnie stosować je do całości zasobów. Precyzyjnie ustalone reguły dostępu do

danych automatycznie wymuszają uprawnienia, ograniczając prawa dostępu właściciela do zasobu.

Ogólny schemat architektury SZBD z uwzględnieniem mechanizmów bezpieczeństwa przedstawiono na rysunku 1. Użytkownik przed uzyskaniem dostępu do bazy danych musi zostać zidentyfikowany i uwierzytelniony w systemie. Następnie należy sprawdzić, czy zapytania kierowane przez użytkownika do bazy danych spełniają zasady dostępu, przechowywane w plikach systemu uprawnień. Dla modelu bezpieczeństwa uznaniowego DAC są to reguły dostępu, a w przypadku modelu kontroli obowiązkowej MAC są to klasy bezpieczeństwa. Jeżeli zasady są spełnione, to użytkownik uzyskuje dostęp do danych. W przypadku naruszenia uprawnień wszystkie informacje zapisywane są w plikach audytu. Następnie zapytania użytkownika są przekształcane do postaci żądań dostępu do danych. Służy do tego menadżer danych i transakcji. Pozostałe operacje są wykonywane na poziomie systemu operacyjnego.

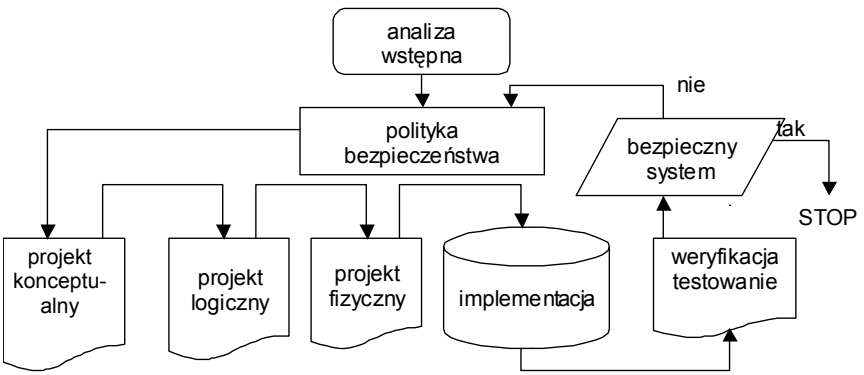


Rys. 1. Ogólny schemat architektury SZBD z uwzględnieniem bezpieczeństwa

2.3. Metodyka projektowania

Zgodnie z definicją Kricka [3, 4] proces projektowania obejmuje czynności i zdarzenia występujące pomiędzy pojawieniem się problemu, a powstaniem dokumentacji opisującej rozwiązanie problemu z punktu widzenia funkcjonalnego, ekonomicznego i innych wymagań. W każdym procesie projektowania, bez względu na klasę projektowanego obiektu i specyfikę zadania projektowego, można się dopatrzyć pewnej sekwencji działań, do której należy: formułowanie problemu, generacja i ocena rozwiązań. W przypadku projektowania bezpiecznych systemów dodatkowo należy opracować pewien zbiór procedur postępowania, określany niekiedy jako polityka bezpie-

czeństwa mająca na celu ochronę danych. Następne etapy projektowania [3] to model konceptualny, logiczny i fizyczny. Implementacja tego ostatniego wymusza na nas niejako weryfikację i testowanie otrzymanego produktu.



Rys. 2. Fazy procesu projektowania bezpiecznych systemów

Metodyka projektowania bezpiecznych systemów baz danych, przedstawiona na rysunku 2, powinna obejmować wymienione fazy:

- Analizę wstępną

Analiza wstępna jest swego rodzaju analizą kosztów i zysków. Polega na ustaleniu zakresu wykonalności techniczno-ekonomicznej systemu. Na wstępie ustala się wszelkie zagrożenia, jakie mogą wystąpić w systemie i ich potencjalne skutki. W tej fazie należy również określić, czy w systemie będzie realizowany model bezpieczeństwa uznaniowego, czy obowiązkowego, a więc czy mamy do czynienia z systemem komercyjnym, czy systemem ochrony wielopoziomowej, odpowiednim dla zastosowań wojskowych. W trakcie analizy wstępnej należy również zwrócić uwagę na wydajność projektowanego systemu, a także rozważyć wady i zalety wyboru między wykorzystaniem dostępnych mechanizmów bezpieczeństwa, a tworzeniem ich od podstaw.

- Określenie wymagań dotyczących bezpieczeństwa oraz polityki bezpieczeństwa

Polityka bezpieczeństwa jest to zbiór zasad, norm i procedur postępowania, określony w celu ochrony przed ujawnieniem, nieuprawnioną modyfikacją oraz zniszczeniem informacji. Powinna ona przyjąć postać dokumentu, w którym określono: prawa i obowiązki wszystkich użytkowników systemu, odpowiedzialność za ewentualne straty, a także postępowanie w sytuacjach awaryjnych. Określenie wymagań dotyczących bezpieczeństwa należy poprzedzić analizą wszystkich możliwych zagrożeń, na jakie jest narażony system. Bezpieczeństwo w dużym stopniu zależy też od wybranych technologii. Zaleca się stosowanie sprzętu i oprogramowania certyfikowanego. Polityka powinna być niezależna od konkretnej implementacji mechanizmów bezpieczeństwa.

• Projektowanie konceptualne

Projektowanie konceptualne jest to faza uściślenia i sformalizowania wymagań dotyczących polityki bezpieczeństwa, a więc podmiotów i obiektów ważnych z punktu widzenia zabezpieczeń, praw dostępu i reguł dostępu. Etap projektowania konceptualnego jest jednym z najtrudniejszych i najważniejszych zadań. Od poprawnie zaprojektowanego schematu zależy właściwe działanie systemu. Poprawny schemat [5, 8] to taki, który ma ścisły związek z faktami świata rzeczywistego, jest podatny na zmiany, zawiera kompletne dla danej aplikacji informacje oraz pozwala na tworzenie różnych obrazów danych, czyli różnych logicznych modeli baz danych. Niżej podano przykład fragmentu konceptualnego modelu bezpieczeństwa dla pewnego systemu. W modelu tym zostały sprecyzowane wymagania dotyczące bezpieczeństwa. Wymagania sformalizowano za pomocą pseudojęzyka wykorzystującego notację języka SQL [3].

1. Identyfikacja podmiotów:
podmiot {Bartoszek} *nazwa* kier_dziekanatu;
podmiot {Jakow, Martan, Nowak} *nazwa* kier_płac;
podmiot {Ada, Ewa} *nazwa* prac_płac;
podmiot {Nowak} *nazwa* ABD;
podmiot {*}ABD *nazwa* użytkownicy BD;
2. Identyfikacja obiektów:
obiekt{Pracownicy(NrPrac, Nazwisko, Adres, Wiek, Oddział, Stanowisko, Płaca)}*nazwa* dane1;
obiekt {*SELECT* NrPrac, Nazwisko, Adres, Oddział Stanowisko, Płaca *from* Pracownicy} *nazwa* dane2;
obiekt {*SELECT* NrPrac, Płaca *from* Pracownicy} *nazwa* dane3;
3. Definicja praw dostępu:
prawo_dostępu {SELECT}*nazwa* READ;
prawo_dostępu {INSERT, UPDATE}*nazwa* WRITE;
prawo_dostępu {DELETE, DROP}*nazwa* DELETE;
prawo_ABD {GRANT}*nazwa* GRANT;
prawo_ABD {REVOKE}*nazwa* REVOKE;
4. Definicja reguł dostępu:
ABD *can* GRANT, REVOKE użytkownicy BD*,*;
użytkownicy BD *cannot* GRANT, REVOKE;
kier_dziekanatu *can* READ,WRITE, DELETE dane1;
kier_płac *can* READ,WRITE dane2;
prac_płac *can* READ dane3;
kier_dziekanatu *can* GRANT kier_płac READ dane1;

Konceptualny model bezpieczeństwa umożliwia wyraźne przedstawienie wymagań i polityki bezpieczeństwa oraz weryfikację niektórych właściwości systemu.

• Projektowanie logiczne

W tej fazie model konceptualny jest przekształcany w logiczny model zabezpieczeń. Model logiczny z punktu widzenia architektury danych jest zbiorem ogólnych zasad posługiwania się danymi. Modelowanie logiczne wiąże się z określeniem zawartości bazy danych, niezależnie od wymagań konkretnej implementacji fizycznej. Na przykład relacyjne systemy baz danych, realizujące założenia uznaniowego modelu bezpieczeństwa, wykorzystują mechanizmy kontroli dostępu oparte na instrukcjach GRANT i REVOKE, perspektywach i grupach uprawnień. Projektowanie logiczne to przede wszystkim tworzenie bazy danych z zachowaniem integralności. Ustala się ponadto, czy wszystkie wymagania i ograniczenia dotyczące bezpieczeństwa, przedstawione w fazie poprzedniej mogą być zrealizowane przez dostępne mechanizmy na poziomie systemu zarządzania bazą danych lub systemu operacyjnego. Jeśli nie, to konieczne jest wówczas zaprojektowanie tego typu mechanizmów.

• Projektowanie fizyczne

Projektowanie fizyczne jest procesem przekształcającym logiczny model zabezpieczeń w fizyczny projekt zarówno dla konkretnego systemu zarządzania bazą danych, jak i określonej konfiguracji sprzętowej. W tej fazie zajmujemy się projektowaniem fizycznej struktury reguł dostępu oraz ich powiązań z fizyczną strukturą danych w celu spełnienia określonych wcześniej założeń polityki bezpieczeństwa. Należy uwzględnić tu funkcje bezpieczeństwa dostępne w systemach zarządzania bazą danych i systemach operacyjnych. Czasami konieczny jest powrót do fazy projektowania konceptualnego lub logicznego w celu modyfikacji pewnych elementów. Konieczne może się okazać zaprojektowanie nowych mechanizmów, pozwalających w pełni realizować przyjętą politykę bezpieczeństwa.

• Implementacja

W trakcie projektowania i implementacji nowych mechanizmów bezpieczeństwa należy pamiętać, aby były one proste, wydajne i implementowane zarówno przy użyciu sprzętu, jak i oprogramowania. Trzeba zadbać o wzajemną niezależność pomiędzy kilkoma mechanizmami. Wszelkie opcje w aplikacjach związane z bezpieczeństwem powinny być domyślnie ustawione na korzyść bezpieczeństwa. Element tajności wykorzystywanych technik zabezpieczeń powinien polegać raczej na kluczach i hasłach niż na algorytmach. Użytkownicy, którym odmawia się dostępu do danego obiektu, nie powinni otrzymywać informacji o istnieniu określonego obiektu, a tym bardziej o jego strukturze. Aby nie naruszać poufności danych, należy usuwać z pamięci systemu pozostawione po zakończeniu procesu niepotrzebne fragmenty informacji.

• Weryfikacja i testowanie

Celem tego etapu jest zweryfikowanie, czy wymagania dotyczące bezpieczeństwa zostały poprawnie zaimplementowane. Wykorzystuje się do tego metody formalne i nieformalne. Za pomocą metod nieformalnych można zweryfikować zachowanie oprogramowania tylko w pewnych określonych sytuacjach. Metody formalne,

oparte na dowodach matematycznych, mają za zadanie dać odpowiedź na pytanie, czy zaimplementowane mechanizmy bezpieczeństwa właściwie realizują przyjętą politykę.

Po przeanalizowaniu zagrożeń, na które narażony jest system bazodanowy, można postawić tezę, że dobrze zaprojektowany system to system bezpieczny. Bezpieczny system to taki, który został zaprojektowany zgodnie z zaproponowaną metodyką projektowania bezpiecznych systemów baz danych. System taki powinien spełniać standardy bezpieczeństwa. Standardy te służą do oceny poziomu bezpieczeństwa i umożliwiają zakwalifikowanie systemu do tzw. klas bezpieczeństwa w zależności od aplikacji. Innego typu zabezpieczeń wymagają systemy komercyjne, a innego te, które znajdują zastosowanie na przykład do celów wojskowych.

3. Standardy bezpieczeństwa

Do najbardziej znanych organizacji zajmujących się zagadnieniami standaryzacji w dziedzinie bezpieczeństwa komputerowego należą [10, 13]:

- ANSI – American National Standards Institute,
- ISO – International Organization for Standarization,
- NBS – National Bureau of Standards, Department of Commerce,
- NCSC – National Computer Security Center, Department of Defence.

Funkcjonujące obecnie dokumenty i standardy to [1, 9, 10, 12, 14]:

- Trusted Computer Systems Evaluation Criteria, DoD, USA, *The Orange Book*, 1985,
- Department of Trade and Industry Commercial Computer Security Centre, *The Green Books*, Wielka Brytania, 1989,
- Zerntralstelle fuer Sicherheit in der Informationstechnik, *The Green Book*, Niemcy, 1989,
- Information Technology Security Evaluation Criteria (Harmonised Criteria of France, Germany, the Netherlands, the UK), 1991.

Najbardziej znane są dwa standardy bezpieczeństwa:

TCSEC (ang. Trusted Computer System Evaluation Criteria) – standard amerykański, tzw. pomarańczowa księga (ang. Orange Book),

ITSEC (ang. Information Technology Security Evaluation Criteria) – standard europejski.

3.1. Standard TCSEC

TCSEC stał się pierwszym standardem w skali światowej. Na jego podstawie opracowano podobne normy w Europie i na całym świecie.

Podstawowe pojęcia i koncepcje to:

- Monitor referencyjny, czyli mechanizm wymuszania autoryzowanego dostępu podmiotów systemu do jego obiektów.
- Jądro ochrony, które jest kombinacją sprzętu, oprogramowania i realizuje kon-

cepcję monitorowania odwołań.

Aby rozszerzyć kryteria oceny bezpieczeństwa również na systemy niezawierające jądra ochrony, wprowadzono pojęcie TCB (ang. Trusted Computing Base). TCB jest bazą bezpiecznego systemu komputerowego, zawierającą wszystkie elementy odpowiedzialne za realizację polityki bezpieczeństwa i wspieranie izolacji obiektów systemu objętych ochroną. TCB zawiera sprzęt i oprogramowanie krytyczne dla ochrony systemu i musi być zaprojektowane i zaimplementowane tak, aby zapewniać założony poziom ochrony. TCB powinna mieć na tyle prostą strukturę, aby możliwe było wykonanie testów i analiz sprawdzających wiarygodność systemu. Ocena poziomu bezpieczeństwa systemu polega na zakwalifikowaniu go do jednej z poniższych klas:

klasa D – ochrona minimalna,
klasa C – ochrona uznaniowa,
klasa B – ochrona obowiązkowa,
klasa A – ochrona sprawdzona.

Klasa D – ochrona minimalna

Do klasy tej włączane są systemy, które były ocenione, ale nie zostały zakwalifikowane do żadnej z pozostałych klas.

Klasa C – ochrona uznaniowa, dzieli się na dwie podklasy:

- Podklasa C1: zabezpieczenie uznaniowe. Jądro ochrony (TCB) tej klasy zapewnia separację użytkowników i danych. Uzyskany poziom bezpieczeństwa pozwala użytkownikom chronić dane związane z projektami, nad którymi pracują, czy dane prywatne, uniemożliwiając innym użytkownikom ich odczyt, modyfikowanie lub usuwanie.
- Podklasa C2: zabezpieczenie kontrolowanego dostępu. Systemy tej podklasy wymuszają silniejszy poziom ochrony niż dla C1 poprzez wprowadzanie procedur logowania, mechanizmów audytu i izolacji zasobów.

Klasa B – ochrona obowiązkowa, dzieli się na trzy podklasy:

- Podklasa B1: zabezpieczenie etykietowane (określono poziomy tajności obiektu). Systemy należące do tej podklasy posiadają wszystkie właściwości systemów C2. Dodatkowo wprowadzony jest element etykietowania podmiotów i obiektów do opisywania ich właściwości w systemie bezpieczeństwa.
- Podklasa B2: zabezpieczenie strukturalne. TCB tej klasy jest oparte na jasno zdefiniowanej i udokumentowanej polityce bezpieczeństwa. Ponadto TCB musi być podzielone na część krytyczną pod względem ochrony (ang. Protection-Critical) i resztę. TCB posiada dobrze zdefiniowany interfejs i jest łatwy w testowaniu. W podklasie B2 wzmocnione są mechanizmy uwierzytelniania oraz narzędzia administrowania bezpieczeństwem systemu. System musi być w miarę odporny na penetrację.
- Podklasa B3: dziedziny bezpieczeństwa. W klasie tej zminimalizowana jest zło-

żoność TCB w celu umożliwienia wykonania dokładniejszych analiz. System posiada silne wsparcie dla administrowania bezpieczeństwem, poprzez mechanizm audytu rozszerzony do reagowania na sygnały związane z bezpieczeństwem. Wymagane jest opracowanie procedur odtwarzania stanu systemu. System jest wysoce odporny na penetrację.

Klasa A, a właściwie A1, to formalne procedury analizy i weryfikacji projektu oraz implementacji systemu. Systemy tej klasy są funkcjonalnie równoważne z systemami podklasy B3. Różnica polega na tym, że istnieje możliwość weryfikacji, czy TCB jest poprawnie zaimplementowana.

3.2. Standard ITSEC

Kryteria oceny bezpieczeństwa technologii informacyjnej opracowano w 1991 roku W porównaniu z Orange Book bezpieczeństwo systemu oceniane jest dodatkowo w wymiarach integralności, niezawodności i bezpieczeństwa transmisji. Wyróżniono następujące klasy funkcjonalności bezpieczeństwa systemu.

klasa F-C1 – odpowiednik podklasy C1 z Orange Book,
klasa F-C2 – odpowiednik podklasy C2 z Orange Book,
klasa F-B1 – odpowiednik podklasy B1 z Orange Book,
klasa F-B2 – odpowiednik podklasy B2 z Orange Book,
klasa F-B3 – odpowiednik podklasy B3 i A1 z Orange Book.

Ponadto określono dodatkowe klasy zawierające wymagania, które nie zostały uwzględnione w Orange Book. Wymagania te dotyczą między innymi: integralności, niezawodności, poufności i tajności danych. Są to:

klasa F-IN – związana z integralnością danych,
klasa F-AV – związana z niezawodnością systemu,
klasa F-DI – związana z integralnością danych w sieciach telekomunikacyjnych,
klasa F-DC – związana z tajnością danych,
klasa F-DX – związana z poufnością i integralnością danych.

Oprócz dziesięciu klas funkcjonalności bezpieczeństwa zdefiniowano również siedem klas pewności – od poziomu E0 najmniej pewnego do poziomu E6 najbardziej pewnego.

E0 – brak odpowiedniej pewności wiąże się z przerwaniem procedury oceniania,

E1 – istnienie nieformalnego opisu architektury bezpieczeństwa,

E2 – istnienie nieformalnego opisu koncepcji szczegółowej architektury bezpieczeństwa, dostarczenie dowodów testów, kontrola konfiguracji i proces nadzoru dystrybucji,

E3 – dostarczenie szczegółowej koncepcji architektury bezpieczeństwa i kodu źródłowego odpowiadającego funkcjom bezpieczeństwa,

E4 – istnienie formalnego modelu polityki bezpieczeństwa, rygorystyczne podejście przy opisie koncepcji ogólnej i szczegółowej architektury bezpieczeństwa,

E5 – istnienie ścisłej relacji między opisem formalnym koncepcji szczegółowej architektury bezpieczeństwa a kodem źródłowym,

E6 – istnienie formalnego opisu architektury bezpieczeństwa kompatybilnej z modelem formalnym polityki bezpieczeństwa.

Największe znaczenie dla produktów komercyjnych mają klasy C2 (F-C2) oraz B1 (F-B1). Przykładowo Oracle 8 Server [6] jest zakwalifikowany do klasy C2 według TCSEC/TDI oraz do klasy F-C2/E3 European ITSEC. Wersje Trusted Oracle Server oraz Informix-OnLine/Secure spełniają natomiast wymagania klasy B1 według TCSEC/TDI oraz F-B1/E3 według ITSEC.

Dla problematyki bezpieczeństwa baz danych ważnym dokumentem jest TDI – Lawendowa księga (ang. Lavender Book). Podaje ona interpretacje wymagań określonych w TCSEC w odniesieniu do baz danych. Orange Book kładzie nacisk na systemy kontroli obowiązkowej, tzn. na poufność danych. Dla środowiska komercyjnego ważniejsza jest integralność i niezawodność systemu.

Aktualnie obowiązujący standard Common Criteria Assurance Levels (EAL) stanowi połączenie TCSEC, ITSEC oraz kanadyjskiego systemu CTCPEC. Od 1996 roku znany jest powszechnie jako Common Criteria for Information Technology Security Evaluation, w 1999 roku zatwierdzony jako norma ISO15408 (EAL v.2).

Podsumowanie

Osiągnięcie całkowitego bezpieczeństwa w dzisiejszych systemach komputerowych jest praktycznie niemożliwe. Wynika to przede wszystkim z tego, iż:

- systemy komputerowe często są systemami otwartymi i zaimplementowanie w nich pełnego, o ile to możliwe, bezpieczeństwa mogłoby spowodować, że straciłyby swój charakter,
- koszt wprowadzenia absolutnego bezpieczeństwa mógłby być tak wysoki, iż przerósłby wartość samego systemu.

Ocena bezpieczeństwa systemu komputerowego pozwala uświadomić sobie, jakie istnieją w nim luki i niedociągnięcia. Pomaga to w podniesieniu poziomu ochrony informacji, przechowywanych i przetwarzanych w tym systemie.

Bibliografia

[1] Baird S., Miler Ch., *SQL Server. Administracja*, Wydawnictwo Robomatic, Wrocław 2000.
[2] Castano S., Fugini M.G., *Database Security*, Addision-Wesley Publishing Company, 1995.
[3] Chałon M., *Systemy baz danych. Wprowadzenie*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2001.
[4] Chałon M., *Ochrona i bezpieczeństwo systemów baz danych*, *Inżynieria komputerowa*, praca zbiorowa pod redakcją W. Zamojskiego, WKŁ, Warszawa 2005.
[5] Date C.J., *Wprowadzenie do systemów baz danych*, WNT, Warszawa 2000.
[6] Denning D., *Wojna informacyjna i bezpieczeństwo informacji*, WNT, Warszawa 2002.
[7] Dudek A., *Nie tylko wirusy. Hacking, cracking, bezpieczeństwo Internetu*, Wydawnictwo Helion, Gliwice 2004.
[8] Elmasri R., Navathe S., *Wprowadzenie do systemów baz danych*, Wydawnictwo Helion, Gliwice 2005.
[9] Ericson J., *Hacking. Sztuka penetracji*, Wydawnictwo Helion, Gliwice 2004.
[10] Litchfield D., *The Database Hacker's Handbook*: *Defending Database Server*, Wiley & Sons, July 2005.
[11] NETFORM, *Bezpieczeństwo systemów komputerowych*, Wydanie specjalne 2000.
[12] Oracle 8 Server Administrator's Guide. Oracle Corporation.
[13] PN-I-13335-1, *Technika Informatyczna – wytyczne do zarządzania bezpieczeństwem systemów informatycznych – Pojęcia i modele bezpieczeństwa systemów informatycznych*, PKN, 1999.
[14] Potter B., Fleck B., *802.11. Bezpieczeństwo*, Wydawnictwo Helion, Gliwice 2004.
[15] Praca zbiorowa pod redakcją A. Grzywaka, *Bezpieczeństwo systemów komputerowych*, Wydawnictwo pracowni komputerowej J. Skalmierskiego, Gliwice 2000.
[16] Wrembel R., Jezierski J., Zakrzewicz M., *System zarządzania bazą danych Oracle 7 i Oracle 8*, Nakom, Poznań 1999.

CZĘŚĆ II

Nieupoważniony dostęp do danych

W dzisiejszych czasach najcenniejsza jest informacja. Dla wielu firm dane są wręcz bezcenne, a ich utrata lub kradzież może oznaczać koniec interesu, a nawet kłopoty natury prawnej.

Aleksander Martin

4. Mechanizmy uznaniowej kontroli dostępu do danych

Z punktu widzenia zapewnienia kontroli dostępu do danych można wyróżnić dwie kategorie ochrony [1, 2, 3]: ochronę systemu oraz ochronę danych.

Ochrona systemu powinna obejmować:

- uwierzytelnienie użytkownika,
- określenie rodzaju operacji systemowych, które może wykonywać użytkownik,
- możliwość monitorowania bazy danych,
- określenie ograniczeń dotyczących przydzielania zasobów systemowych dla użytkownika.

Wśród mechanizmów ochrony danych powinny się znaleźć określenia, dotyczące pytań:

Którzy użytkownicy i do jakich obiektów mają prawo dostępu?

Jakiego typu operacje są dozwolone dla danego użytkownika?

Które operacje będą monitorowane?

Zabezpieczenia w systemach kontroli dostępu mogą mieć charakter zapobiegawczy – chronią informację przed wystąpieniem incydentu, lub reaktywny – chronią informację po wystąpieniu zdarzenia.

Zabezpieczenia powinny charakteryzować się:

- spójnością, czyli działać na różnych platformach sprzętowych, w różnych systemach operacyjnych (nie ma znaczenia lokalizacja fizyczna),
- kompletnością, gdzie każdy system osiąga ten sam poziom ochrony,
- efektywnością kosztową, czyli nakłady poniesione na ochronę zależą od wartości informacji.

Projektując mechanizmy uznaniowej kontroli danych, należy zapewnić:

- poufność – ochronę informacji przed nieautoryzowanym jej ujawnieniem,
- identyfikację – możliwość rozróżnienia użytkowników,
- integralność – utrzymanie stanu, w którym dane nie zostały zmienione lub zniszczone w nieautoryzowany sposób,
- rozliczalność – zdolność jednoznacznej identyfikacji osoby odpowiedzialnej za określone działania,

- uwierzytelnienie – proces weryfikacji tożsamości użytkownika,
- autoryzację – proces przydzielenia praw dostępu do zasobów użytkownika,
- autentyczność – pewność co do pochodzenia danych,
- niezaprzeczalność – ochronę przed fałszywym zaprzeczeniem.

5. Uwierzytelnianie

Uwierzytelnianie jest to proces weryfikacji tożsamości użytkownika w trakcie wchodzenia do systemu. Uwierzytelnienie poprzedza proces autoryzacji [8, 13, 16], w którym sprawdza się, czy dany podmiot o ustalonej tożsamości ma prawo dostępu do zasobów, o które prosi. Wśród metod identyfikacji i uwierzytelnienia użytkowników można wyróżnić systemy oparte na:

- hasłach – proste, jednorazowe, wybrane znaki,
- metodzie pytań i odpowiedzi, np. zadawanie losowo wybranych pytań,
- uwierzytelnieniu tożsamości komputera, czyli dodatkowe hasła,
- procedurach przywitania – wykonanie przez użytkownika poprawnie jakiegoś algorytmu,
- procedurach użytkownika – wykonanie przed wejściem do systemu dostarczonych przez użytkownika procedur, po których zakończeniu system wywołuje własną procedurę bezpieczeństwa,
- fizycznych metodach uwierzytelniania, takich jak odciski palców, karty magnetyczne, pomiary cech biometrycznych człowieka, tokeny, a także podpisy cyfrowe.

5.1. Hasła

Najpopularniejsza, ze względu na niskie koszty i łatwość implementacji, jest metoda uwierzytelnienia oparta na hasłach. Metoda ta pozwala na zapewnienie:

- autentyczności, czyli pewności co do identyfikatora podmiotu systemu,
- integralności, czyli zapewnienia, że identyfikator i hasło to jednoznaczne cechy nadane podmiotowi,
- bezpieczeństwa przez ograniczenie dostępu osób nieupoważnionych.

Aby móc posługiwać się hasłami, nie wymaga się żadnych dodatkowych urządzeń. Dzięki hasłom uzyskuje się pewien stopień wiarygodności, jednak w przypadku systemów wymagających mocnych zabezpieczeń system haseł może się okazać niewystarczający. Hasła mogą być wielokrotne i jednorazowe. Wielokrotne są najpowszechniejszą metodą uwierzytelnienia. Tego samego hasła używa się za każdym razem, gdy

trzeba uwierzytelnić swoją tożsamość. Hasła wielokrotne, na ogół bardzo proste i łatwe do zapamiętania, są bardzo podatne na powszechne ataki. Strategie haseł jednorazowych oparte są na sekwencjach liczb pseudolosowych. Wartość kolejnych haseł jest obliczana na podstawie wartości początkowej tak zwanego ziarna. Bez znajomości ziarna, znając wcześniejsze sekwencje liczb, nie da się obliczyć wartości następnego hasła. Hasła jednorazowe są często realizowane w postaci listy, czyli sekwencji haseł, które muszą być kolejno wykorzystywane w procesie uwierzytelniania. Problemem takiego rozwiązania może być zaginięcie listy haseł. Innym rozwiązaniem realizacji haseł jednorazowych są urządzenia sprzętowe z synchronizacją czasową.

Hasła [5, 6, 11] są dobrym, lecz nieskutecznym zabezpieczeniem. Istnieje wiele sposobów ich łamania. Przykładem są specjalne listy słów kluczowych, zwane słownikami. Takie narzędzie, szukając hasła, sprawdza z zawartością słownika słowo po słowie, aż natrafi na to właściwe. Inny sposób odgadnięcia hasła to sprawdzenie każdej możliwej kombinacji liczb i cyfr, aż do uzyskania tej właściwej. Algorytm ten, zwany brutalnym atakiem (ang. brute force), jest nieoptymalny, lecz najprostszy do implementacji i w miarę skuteczny. Jeden z najbardziej popularnych programów do łamania oraz testowania haseł znany jest pod nazwą *John the Ripper*. Wykorzystuje on algorytm brute force. Inne zaawansowane narzędzie, które może być użyte do łamania haseł systemowych, to *Cain&Abel.* Istnieją również takie narzędzia, które pozwalają na uzyskanie dostępu do kont i haseł użytkowników w komunikatorach GG/TLEN.pl, na przykład *GG&Tlen Password Reader*. Program do odtwarzania zapomnianych haseł do archiwów utworzonych przez archiwizatory ZIP i WinZIP to *Advanced ZIP Password Recover.*

Do nadzoru i ochrony haseł w systemie wykorzystuje się: komunikaty systemowe, systemy z dwoma hasłami, blokowanie konta użytkownika, ochronę hasła administratora, odrzucenie haseł zbyt prostych czy generowanie hasła przez system. W niektórych systemach baz danych, w których uwierzytelnienie odbywa się za pomocą mechanizmu Basic Authentity oraz w protokołach sieciowych stosuje się pomocniczo kodowanie transportowe. Przykładem tego typu kodowania jest Base64, używane do przekształcania treści przesyłki lub jej części z postaci 8-bitowej na 7-bitową.

Bardzo często do weryfikacji haseł używa się funkcji składowanych. W przykładzie 1 pokazano taką procedurę, która jest uruchamiana w momencie, gdy użytkownik zmienia hasło. Niektóre SZBD (na przykład Oracle [5]) mają wbudowany system zarządzania hasłami.

Przykład 1

```
CREATE OR REPLACE FUNCTION Hasła_pracowników;
(nazwa konta VARCHAR2;
hasło VARCHAR2;
stare hasło VARCHAR2);
RETURN BOOLEAN IS;
```

```
BEGIN
IF hasło=nazwa konta THEN;
RAISE_APPLICATION_ERROR(-2001,”hasło nie może być takie samo jak konto”);
END IF;
IF hasło=stare hasło THEN
RAISE_APPLICATION_ERROR(-2002,”nowe hasło nie może być takie samo jak stare hasło”);
END IF;
IF LENGTH (hasło) < =7 THEN
RAISE_APPLICATION_ERROR(-2003,”hasło musi mieć więcej niż 7 znaków”);
END IF;
RETURN(TRUE);
END;
```

Począwszy od systemu Oracle 8 istnieje możliwość zarządzania hasłami przez profile.

Przykład 2

```
CREATE PROFIL profil limit;
FAILED LOGIN ATTEMPTS 3;
PASSWORD LIFE TIME 60;
PASSWORD LOCK TIME 1;
ALTER USER Nowak PROFILE profil;
```

W czasie przechowywania, przetwarzania i przesyłania hasła powinny mieć postać zaszyfrowaną. Hasła mogą być przechowywane w plikach w postaci otrzymanej po przekształceniu jednokierunkową funkcją skrótu. Zastosowanie tej funkcji uniemożliwia rozszyfrowanie hasła nawet przez system. Każde wejście do pliku haseł jest parą ID użytkownika i f(H), gdzie H jest jego hasłem. Aby uzyskać dostęp do systemu, użytkownik musi wprowadzić ID oraz H. System wyznacza zaszyfrowaną postać hasła f(H) i porównuje ją z przechowywaną w odpowiednim pliku hasłowym. Zgodny wynik porównania oznacza zezwolenie na dostęp do systemu.

W systemie ORACLE hasła są szyfrowane między innymi z wykorzystaniem popularnego w kryptografii algorytmu symetrycznego DES.

5.2. Autoryzacja

Autoryzacja jest to ograniczenie praw dostępu do różnych części bazy danych różnym użytkownikom. Uniemożliwia ono użytkowanie poufnych danych przez niepowo-

łanych użytkowników. Specyfikacja autoryzacji [4, 5, 11] składa się z trzech elementów:

- użytkownika,
- obiektu,
- prawa dostępu.

Wyróżnia się co najmniej dwóch użytkowników: administratora oraz właściciela obiektów. Administrator ma nieograniczone prawa dostępu do wszystkich obiektów bazy, możliwość rejestrowania nowych użytkowników bazy oraz możliwość zmiany praw dostępu określonych użytkowników do określonych obiektów. Właściciel danego obiektu jest na ogół automatycznie uprawniony do wykonywania wszystkich czynności, związanych jedynie ze swoim obiektem.

W systemach można wyróżnić dwa typy obiektów: własne, utworzone przez danego użytkownika, do których ma on nieograniczone prawa dostępu, i obiekty obce.

Rozróżnia się też dwa rodzaje praw dostępu: systemowe, o charakterze ogólnym, odnoszące się do całej bazy danych oraz obiektowe, określające prawa do wykonywania danej operacji na danym obiekcie bazy.

Przykład uprawnień ogólnych w ORACLE
CREATE SESSION – prawo do nawiązywania połączeń z bazą danych,
SELECT ANY TABLE – prawo do przeglądania dowolnej tabeli.

Przykład uprawnień obiektowych w ORACLE
SELECT – odczyt danych,
INSERT – wstawianie danych,
DELETE – usuwanie danych.

Niżej podano przykłady nadawania i odbierania uprawnień.

Przykład 3

```
GRANT, SELECT, UPDATE
ON Pracownicy
TO Kowalski
WITH GRANT OPTION;
```

przekazywanie (GRANT) swoich uprawnień (SELECT, UPDATE) innym użytkownikom

Przykład 4

```
REVOKE,UPDATE
TO Pracownicy
FROM Nowak
CASCADE;
```

Odbieranie (REVOKE) uprawnień innym użytkownikom jeśli były nadane (CASCADE)

Administrator bazy danych, zamiast nadawać każdemu użytkownikowi osobno

różne uprawnienia, może określić grupy użytkowników wykonujących podobne operacje i wymagających podobnych uprawnień. Grupy uprawnień, tak zwane role, mogą zawierać przywileje zarówno systemowe, jak i obiektowe. Zaletą używania ról jest uproszczenie zarządzania uprawnieniami w systemach z wieloma użytkownikami. Uproszczenie to polega na:

- nadawaniu i odbieraniu przywilejów jednym poleceniem,
- zmianie uprawnień wielu użytkownikom przez zmianę jednej roli.

Ciekawie rozwiązano problem ról w serwerze MS SQL 2000 [17]. W każdej bazie danych SQL Serwer 2000 zawiera jedną rolę wbudowaną o nazwie PUBLIC. Należą do niej wszyscy użytkownicy bazy danych, role i grupy. Roli PUBLIC nie można usunąć. Jest bardzo przydatna w przypadku, gdy dane uprawnienie ma być przyznane wszystkim. Uprawnienia SELECT, jak również EXECUTE, w wielu systemowych procedurach składowanych są przyznane roli PUBLIC do wszystkich tabel systemowych w każdej z baz danych. Uprawnienia te zapewniają prawidłowe działanie SQL Servera i nie powinny być modyfikowane.

Inny typ to role serwera:

- `Sysadmin` – członkowie roli serwera `sysadmin` mogą wykonać wszelkie operacje na platformie SQL Server. Dysponują oni bardzo efektywnym zbiorem uprawnień i dlatego należy rozważnie przydzielać użytkowników do tej roli. Konto zawsze należy do tej roli

i nie może być usunięte z roli `sysadmin`. Członkowie ustalonej roli systemowej sysadmin są zawsze uważani za właścicieli każdej bazy danych, której używają. Nie można zabronić użytkownikom tej roli dostępu do żadnej z baz danych na platformie SQL Server.

- `Serveradmin` – członkami tej roli powinni być administratorzy serwera, którzy raczej nie będą administrować bazami danych lub innymi obiektami. Użytkownicy korzystający z tej roli, mogą wykonać zdefiniowany zbiór operacji.
- `Setupadmin` – członkowie tej roli są typowymi administratorami, którzy konfigurują zdalne serwery. Mogą wykonywać następujące operacje: dodawać konta logowania do ustalonej roli serwera, dodawać, usuwać i konfigurować serwery przyłączone, określić procedury przechowywane, które mają być uruchamiane podczas uruchomienia systemu.
- `Securityadmin` – członkowie tej roli mogą wykonywać na platformie SQL Server wszelkie operacje związane z bezpieczeństwem całego serwera. Dobrymi kandydatami do tej roli są pracownicy pomocy technicznej tworzący nowe konta.
- `Processadmin` – członkowie tej roli mogą kontrolować procesy działające na serwerze baz danych. Rola ta na ogół dotyczy kasowania działających zapytań; może być przydatna dla pracowników pomocy technicznej.
- `Dbcreator` – członkowie ustalonej roli serwera `dbcreator`, do której należą na ogół doświadczeni administratorzy baz danych, mogą wykonywać operacje związane

z tworzeniem i modyfikacją bazy danych.

• `Diskadmin` – członkowie tej roli serwera mogą zarządzać plikami. Rola ta istnieje dla zachowania zgodności z wcześniejszą wersją platformy SQL Server.

Każda baza danych również zawiera role. Członkowie ustalonych ról bazy danych mają specjalne uprawnienia w każdej z baz danych, różniące się od ról serwera. Inne niż role serwera, ale specyficzne dla każdej bazy danych. Przynależność do ustalonej roli bazy danych w jednej bazie nie ma wpływu na uprawnienia w żadnej innej bazie danych. Można wyróżnić następujące role:

• `db_owner` – członkowie ustalonej roli bazy danych są „właścicielami" bazy danych. Mają w bazie danych wiele uprawnień i mogą wykonywać prawie wszystkie operacje, jakie może wykonać właściciel bazy.

• `db_accessadmin` – członkowie tej roli bazy danych ustalają, które konta logowania mają prawa dostępu do bazy danych. Podobnie jak w przypadku roli `securityadmin`, pracownicy działu pomocy technicznej są najlepszymi kandydatami do przyznania im tej roli.

• `db_securityadmin` – członkowie tej roli bazy danych mogą administrować zabezpieczeniami w bazie oraz mogą uruchamiać polecenia GRANT i REVOKE oraz uruchamiać niektóre systemowe procedury składowe.

• `db_ddladmin` – członkowie tej roli bazy danych mogą uruchamiać polecenia DDL z wyjątkiem GRANT i REVOKE (przyznawania i odbierania uprawnień), przydzielać uprawnienia REFERENCES dla dowolnej tabeli, rekompilować dowolne obiekty i zmieniać nazwy dowolnych obiektów za pomocą systemowej procedury składowej, modyfikować niektóre opcje specyficzne dla tabeli, zmieniać właściciela dowolnego obiektu itp.

• `db_backupoperator` – członkowie tej stałej roli bazy danych mogą wykonywać wszystkie operacje związane z tworzeniem kopii bezpieczeństwa bazy danych.

• `db_datareader` – członkowie tej roli mają uprawnienie SELECT na wszelkich tabelach lub widokach w bazie danych. Nie mogą oni jednak przyznawać (GRANT) lub zabierać (REVOKE) tego polecenia innym.

• `db_datawriter` – członkowie tej roli mają uprawnienia INSERT, UPDATE oraz DELETE (wprowadzania danych, aktualizacji i kasowania) do wszystkich tabel i widoków bazy danych. Nie mogą jednak przyznawać (GRANT) lub zabierać (REVOKE) tych poleceń innym.

• `db_denydatareader` – członkowie tej roli nie mogą uruchamiać polecenia SELECT na żadnej z tabel i widoków bazy danych. Opcja ta jest przydatna, gdy administrator bazy danych (DBA) ma za zadanie skonfigurować obiekty, korzystając z roli `db_ddladmin`, ale nie powinien mieć dostępu do istotnych danych w bazie.

• `db_denydatawriter` – członkowie tej roli bazy danych nie mogą uruchamiać poleceń INSERT, UPDATE lub DELETE na żadnej z tabel lub widoków w bazie danych.

Właściciel bazy danych ma wszystkie prawa, jakie posiadają członkowie roli

db_owner. Każda baza danych może mieć tylko jednego właściciela (dbo).

Jeżeli użytkownik jest właścicielem bazy (dbo), to gdy tworzy on obiekt, nazwą właściciela obiektu staje się nazwa dbo i użytkownik dbo staje się właścicielem tego obiektu bazy danych. Nie odnosi się to do członków roli bazy danych db_owner, ani innych użytkowników bazy danych. Dopóki użytkownik nie skojarzy nazwy swojego obiektu z nazwą dbo, dopóty jako właściciel będzie występować nazwa użytkownika.

Użytkownik jest rozpoznawany w bazie danych jako dbo, jeśli występują następujące sytuacje:

- użytkownik jest twórcą bazy danych; konto logowania, z którego utworzono bazę jest przyjęte jako dbo.
- użytkownik jest przypisany jako właściciel bazy danych,
- użytkownik pracuje na platformie SQL Server jako dowolny użytkownik, korzystający z roli serwera sysadmin,
- użytkownik może połączyć się z bazą danych za pomocą konta, które posiada alias dbo. W danej chwili tylko pojedynczy użytkownik może być zalogowany jako dbo.

Użytkownik, który tworzy obiekt bazy danych jest dla tego obiektu użytkownikiem dboo, czyli właścicielem. Większość osób korzystających z bazy danych to zwykli użytkownicy, a nie właściciele.

Bardzo efektywnym mechanizmem zabezpieczającym, stosowanym w celu ukrycia ważnych danych przez zapewnienie selektywnego dostępu do nich, są perspektywy zwane widokami (ang. view). Perspektywy to wirtualne tablice, udostępniające użytkownikowi fragmenty rzeczywistych tabel.

Przykład 5

```
CREATE VIEW Prac_zesp_5 AS;
SELECT *;
FROM Pracownicy;
WHERE nr_zesp. = 5;
AND pensja = 4000;
GRANT SELECT;
ON Prac_zesp_5;
TO PUBLIC;
```

Mając tak określoną perspektywę, można przypisać uprawnienia do niej określonym użytkownikom. Wiąże się to z tym, że nie mają oni prawa do przeglądania całej tabeli, tylko wybranych kolumn tej tabeli lub wielu tabel. Mechanizm perspektyw pozwala zwiększyć stopień bezpieczeństwa.

Podobnie jak w przypadku uprawnień dotyczących perspektyw, uprawnienia do procedur składowanych pozwalają na blokowanie użytkownikom dostępu do tabel. Inaczej niż w przypadku perspektywy procedury mogą zawierać wiele poleceń i

operacji. Są to miniprogramy, które przeglądają i modyfikują dane w wielu tabelach i perspektywach oraz mogą zbierać informację z innych baz danych. W SQL Serwer 2000 procedury składowane mogą pobierać dane z innych źródeł. Aby uruchomić procedurę, użytkownik potrzebuje jedynie pojedynczego uprawnienia EXECUTE.

W bazach obiektowych, ze względu na dziedziczenie przez podklasy atrybutów i metod z nadklas, pojawiają się poważne problemy z autoryzacją.

5.3. Kod PIN

Kod PIN (ang. Personal Identification Numer) jest kodem numerycznym służącym do uwierzytelniania. Szerokie zastosowanie znalazł w bankowości i telekomunikacji. Zazwyczaj składa się z czterech cyfr, co daje 10000 kombinacji. Są jednak wyjątki, gdzie ze względu na bezpieczeństwo stosuje się dłuższe kody, na przykład 6-bitowe, dające już 1000000 kombinacji. Obecnie stosowane algorytmy generowania PIN-u dają wysoki poziom losowości. Kody PIN nigdy nie są przekazywane w postaci otwartej, zawsze pozostają zakodowane. Do szyfrowania kodu używa się specjalnego algorytmu symetrycznego 3DES. W celu zwiększenia bezpieczeństwa PIN jest szyfrowany bezpośrednio przy wprowadzaniu, a nie przekazywany do innego modułu, który dopiero zajmuje się jego zaszyfrowaniem i wysłaniem. Tak powstały EPP (ang. encrypting PIN pad), popularnie nazywane PIN-pady. Wyeliminowano w ten sposób przekazywanie PIN-u w formie niezaszyfrowanej. Dodatkowo PIN-pady zostały wyposażone w specjalne zabezpieczenia niszczące klucze szyfrujące, w sytuacji gdy ktoś próbuje dostać się do wnętrza, albo próbował wydobyć znajdujące się w nich klucze. Niestety istnieje wiele luk w systemie. Najbardziej popularnym sposobem jest przechwycenie danych z karty i kodu PIN w celu stworzenia fałszywej karty (ang. skimming). Polega to na zdobyciu paska magnetycznego i przyporządkowanego do karty PIN-u kodu przez zamontowanie urządzenia z minikamerą, odczytującego te dane.

5.4. Tokeny

Wzrost zagrożeń w dziedzinie bezpieczeństwa i poufności danych spowodował, że stosowane zabezpieczenia w postaci haseł i kodów PIN przestały stanowić barierę dla potencjalnych włamywaczy. Pojawiły się wtedy elektroniczne urządzenia uwierzytelniające, wielkości kalkulatora czy breloczka, zwane tokenami. Tokeny to urządzenia powiązane bezpośrednio z użytkownikami i ich kontami. Mogą być użyte tylko dla konta, dla którego zostały wydane. Wszystkie tokeny mają zaszyte w

sobie algorytmy kryptograficzne (algorytm symetryczny, funkcja hash, generator pseudolosowy) i klucze kryptograficzne. Bardzo często tokeny zawierają zegar czasu rzeczywistego – zsynchronizowany z serwerem i obsługą kodu uwierzytelniającego wiadomość ((MAC) ang. message authentication code). W oparciu o prywatny klucz, czas, ewentualnie inny ciąg cyfr wcześniej wprowadzony do tokena generuje on ciąg cyfr. Ten wygenerowany ciąg cyfr użytkownik wprowadza do systemu. Jeżeli token działa z uwzględnieniem czasu, to taki ciąg jest ważny tylko przez krótki okres. System komputerowy, znając klucz zawarty w tokenie oraz algorytm stosowany przez token, potrafi stwierdzić, czy ciąg cyfr podany przez użytkownika jest prawidłowy. Dodatkową korzyścią płynącą z tokenów jest to, że są one programowalne, a duża liczba programowanych parametrów, takich jak: długość kodu PIN, liczba błędnych wprowadzeń kodu, typ zastosowanego algorytmu sprawia, że elastyczność tych urządzeń jest bardzo duża. Przykłady tokenów:

- Token DigiPass 300

Urządzenie charakteryzuje się niską ceną, prostotą obsługi oraz dużą wytrzymałością mechaniczną. Implementuje szereg zaawansowanych mechanizmów kryptograficznych typu OTP (ang. One Type Password) i Ch/Rp (ang. Challenge/Response). Token ten jest zaopatrzony w interfejs optyczny, pozwalający na odczytywanie danych bezpośrednio z ekranu komputera. Główne cechy to: zgodność ze standardami DES i 3DES, trzy aplikacje kryptograficzne bazujące na różnych kluczach i parametrach, możliwość wykonywania przez aplikacje różnych funkcji zależnych i niezależnych od czasu i zdarzeń. Ponadto token ten ma wbudowany zegar czasu rzeczywistego i automatyczną blokadę przy kilkakrotnym podaniu złego PIN-u.

- Token RSA SecureID

Jest to generator haseł jednorazowych, który na zewnątrz ma wbudowany jedynie wyświetlacz ciekłokrystaliczny z sześcioma cyframi oraz ze skalą czasu odmierzającą upływające sekundy. Token cały czas wyświetla pseudolosowy ciąg cyfr (ang. card-code), który jest ważny przez 60 sekund. Ciąg cyfr jest funkcją tajnego 64-bitowego klucza, zapisanego w specjalnej karcie (ang. seed value), oraz aktualnego czasu. Serwer autoryzacji potrafi ustalić poprawność ciągu cyfr wygenerowanego przez token. Zegary tokena i serwera są zsynchronizowane.

- Token ActiveCard PLUS

Jest jednym z najciekawszych pod względem zastosowania i bezpieczeństwa rozwiązaniem. Hasła są generowane w trybie synchronizacji z serwerem uwierzytelniania ActivPack lub w trybie asynchronicznym wyzwanie/odpowiedź. Bezpieczeństwo systemu opiera się na takich algorytmach jak DES oraz na tajności klucza zapisanego w specjalnej karcie (ang. seek value). Dostęp do tokena realizowany jest za pomocą indywidualnego kodu PIN użytkownika. W przypadku wprowadzania błędnych kodów PIN token jest automatycznie blokowany. Algorytm generowania haseł dynamicznych posiada skuteczne, oparte na kryptografii, mechanizmy zabezpieczające przed odtworzeniem

tajnego klucza. Wprowadzenie błędnego kodu powoduje skasowanie pamięci. Jakakolwiek próba rozmontowania tokena kończy się samozniszczeniem urządzenia. Przez swojego producenta token może być programowany wielokrotnie.

- TokenDigiPass GO2

Jest to przenośny czytnik kart chipowych. GO2 jest przeznaczony przede wszystkim do masowych wdrożeń, w których wymagany jest niski koszt rozwiązania. Token zbudowano zgodnie z normami ISO:7816-x dla kart chipowych i ze standardami algorytmów AES, DES, 3DES. Posiada wbudowany zegar czasu rzeczywistego, wyświetlacz LCD z matrycą 9×60 oraz inteligentny system zarządzania energią.

Tokeny są produkowane przez wiele firm. Szczegóły ich budowy i zasad działania są na ogół tajemnicą firmy, co zwiększa bezpieczeństwo użycia tokenów.

5.5. Karty inteligentne

Patent na plastikową kartę zawierającą jeden lub więcej układów scalonych do generowania określonych sygnałów powstał w 1970 roku. Pomysł karty inteligentnej (ang. Smart Card), potocznie zwanej kartą mikroprocesorową lub chipową, zrodził się w 1974 roku. Karta inteligentna była następcą karty z paskiem magnetycznym. Dwa układy scalone szybko zastąpiono jednym w celu zwiększenia niezawodności. Karta z paskiem ze względu na małą pojemność pamięci i pasywny sposób działania była łatwa do odczytania, skopiowania i powielenia. W latach 1982–1984 French Bank Card Association zlecił przeprowadzenie badań grupie roboczej IPSO. W testach wzięli udział trzej producenci: Bull, Schlumberger i Philips. W roku 1985 powstała tak zwana elektroniczna portmonetka. W 1986 roku opracowano pierwszy standard dla kart elektronicznych ISO/IEC 7816-1. Następnie w 1991 roku Philips opracował pierwszą kartę z modułem szyfrującym dane i kluczem publicznym, a w 1996 roku pierwszą kartę bezstykową, zgodną ze standardem ISO/IEC 14443-A. Karty inteligentne wykonywane są w dwóch podstawowych technologiach:

- karty kontaktowe (ang. contact smart cards), które mogą być odczytywane po wprowadzeniu ich do specjalnego czytnika, a transmisje danych zapewnia styk odpowiednich złączek elektrycznych wewnątrz czytnika z wyprowadzeniami mikroprocesora na karcie,
- karty bezstykowe (ang. contactless smart cards), które nie mają zewnętrznych wyprowadzeń mikroprocesora, ale wbudowaną wewnątrz antenę służącą do bezkontaktowego transferu danych.

Karty bezstykowe charakteryzują się:

- dużą pewnością czytania,
- wysoką trwałością karty i koniecznością wykonywania skomplikowanych operacji w razie podrobienia karty,

- niższym kosztem eksploatacji systemu kartowego,
- wygodniejszym sposobem posługiwania się kartą,
- prędkością transmisji przekraczającą 100 kb/s,
- krótkim czasem realizacji dowolnej operacji <100 ms,
- większą bezawaryjnością systemu.

Istnieje możliwość połączenia technologii kontaktowej z bezkontaktową i mamy wtedy do czynienia z kartami dualnymi.

Elektroniczne karty inteligentne mają wbudowany procesor 8-bitowy oraz pamięci: RAM, ROM, EPROM lub EEPROM. Pamięć EEPROM (1-64 kB) podzielona jest zwykle na trzy obszary:

- obszar swobodnego odczytu, zapisywany raz podczas personalizacji karty,
- obszar poufny, zapisywany raz i niezmieniany podczas użytkowania karty dostępu wymaga klucza,
- obszar roboczy, gdzie dane są tymczasowe lub zmieniane podczas użytkowania, a dostęp opisywany jest specjalnymi regułami.

Ponieważ karty te bywają często celem ataków, dla większego bezpieczeństwa wprowadzono mechanizmy powodujące, że jakakolwiek ingerencja w budowę karty powoduje jej uszkodzenie lub zablokowanie. Jedną z istotnych wad, związaną z użyciem tego typu kart jest konieczność posiadania specjalnych czytników, co poważnie zwiększa koszty całego zestawu. Czytniki te mają za zadanie:

- wykrywanie karty,
- dostarczanie zasilania,
- dostarczenie sygnału zegarowego,
- umożliwienie komunikacji przy wykorzystaniu przynajmniej jednego protokołu.

Karty nie są również pozbawione wad, które wynikają z technologii ich wytwarzania. Bardziej zaawansowaną odmianą kart inteligentnych są karty kryptograficzne, wyposażone w dodatkowy układ specjalizowany, wykonujące operacje charakterystyczne dla systemów kryptograficznych. Można dokonywać bardzo skomplikowanych obliczeń, potrzebnych do szyfrowania asymetrycznego i symetrycznego. Karty te mają mikroprocesor z wbudowanym akceleratorem szyfrowania, co skutkuje zmniejszeniem pamięci EEPROM, oraz koprocesory arytmetyczne (CORSAIR, FAME, FAMEX), co przyspiesza czas realizacji operacji kryptograficznych nawet dla bardzo długich kluczy. Ponadto wszystkie dane przechowywane na karcie są automatycznie szyfrowane przez zmieszanie sygnałów (ang. computational scrambling encryption). Karty kryptograficzne są doskonałym środkiem do przechowywania takich tajnych informacji jak klucze prywatne i certyfikaty. Dla większego zabezpieczenia karty te wyposażono w mechanizmy powodujące, że jakakolwiek ingerencja w ich budowę powoduje ich uszkodzenie. Błędy typu źle wprowadzony PIN lub hasło powodują zablokowanie karty. Kod PIN wprowadza użytkownik i w każdej chwili może być zmieniony, co wpływa na komfort posługiwania się

kartą. Przykładem tego typu karty może być zestaw proponowany przez firmę ActivCard. Składa się on z karty chipowej ActivCard Gold i bezpiecznego czytnika ActiveReader. ActiveReader jest to wielozadaniowy czytnik kart mikroprocesorowych, wyposażony w specjalną klawiaturę i wyświetlacz. PIN do karty Smart Card wpisywany jest bezpośrednio do czytnika. Czytnik jest bardzo łatwy do zamontowania praktycznie w każdym komputerze.

Obecnie trwają badania nad zwiększeniem bezpieczeństwa kart mikroprocesorowych, polegające na ich integracji z technologią biometryczną wbudowaną w plastik. Operacja taka może służyć do weryfikacji i identyfikacji użytkownika oraz stwierdzenia, czy w danej chwili karta jest używana przez jej właściciela.

5.6. Rozpoznawanie cech biometrycznych

Biometria to nauka zajmująca się ustalaniem i potwierdzeniem tożsamości na podstawie mierzalnych cech organizmu. Inaczej mówiąc, biometria jest zbiorem metod i technik służących do weryfikacji i ustalania tożsamości osób na podstawie ich cech biofizycznych i behawioralnych. Biometria [1] stosowana była już od starożytności. Babilończycy używali odcisków palca w wosku jako pieczęci. Ze względu na dużą dokładność i niezawodność urządzeń odczytujących dane biometryczne oraz dużą moc obliczeniową komputerów, potrafiących te dane analizować, biometria stała się popularnym sposobem ochrony dostępu do danych przed osobami nieupoważnionymi. Biometria przewyższa pod względem bezpieczeństwa inne rodzaje zabezpieczeń, takie jak kody PIN czy tokeny.

Rozróżniamy dwa główne aspekty biometrii:

- statyczną – fizyczno-biologiczną,
- dynamiczną – behawioralną.

Biometria statyczna polega na rozpoznawaniu cech budowy ciała człowieka, takich jak: geometria twarzy, siatkówka i tęczówka oka, głos, geometria dłoni i palców czy linie papilarne.

Biometria dynamiczna rozpoznaje takie cechy zachowania człowieka jak: sposób chodzenia, sposób naciskania klawiszy, cechy podpisu lub sposób podpisywania się.

Nie każda dająca się zmierzyć cecha człowieka w równym stopniu nadaje się do użycia przy weryfikacji tożsamości. Podstawowe wymagania stawiane takim cechom to: powszechność, indywidualność, niezmienność, mierzalność, akceptowalność oraz brak możliwości fałszowania. Na pomiar cech biometrycznych wpływa wiele czynników, takich jak: działanie sensorów, wpływ środowiska, deformacja i zniekształcenia. Nie ma możliwości, aby dwie próbki, pobrane w różnym czasie, były dokładnie takie same. Dlatego do porównania wzorca z pobraną próbką wykorzystywane są algorytmy, które obliczają poziom zgodności. Ten poziom jest później porównywany z założonym wskaźnikiem akceptowalności. Uzyskanie wyniku większego niż wskaźnik

oznacza zidentyfikowanie osoby. Niestety systemy te mogą generować czasami nieprawidłowe odpowiedzi. Główne błędy to:

- FRR (ang. false rejection rate) jest to liczba porównań, w których użytkownik powinien zostać zweryfikowany pozytywnie, a został zweryfikowany negatywnie. FRR nie oznacza, że system działa nieprawidłowo. W przypadku np. systemów opartych na rozpoznawaniu linii papilarnych może to oznaczać nieprawidłowe położenie palca na czytniku.
- FAR (ang. false acceptance rate) jest to liczba porównań, w których użytkownik powinien zostać zweryfikowany negatywnie, a został zweryfikowany pozytywnie.

Ogólnie FRR i FRA zależą od poziomu dokładności czy raczej akceptowalności urządzenia, które jest używane do określenia poziomu bezpieczeństwa systemu. Poziom ten można ustawić. Zwiększenie współczynnika poziomu dokładności ogranicza możliwość zweryfikowania pozytywnie osoby nieuprawnionej do dostępu, ale też zwiększa prawdopodobieństwo, że osoba uprawniona zostanie błędnie odrzucona.

Przykłady systemów biometrycznych:

- BAC Secure Touch 99 – czytnik linii papilarnych, dołączany do portu równoległego, szeregowego lub USB komputera. Pracuje z popularnymi systemami, takimi jak Windows, dostarczany jest ze ściśle zdefiniowanym interfejsem programistycznym, współpracuje z aplikacjami Javy, ActiveX. Rejestracja użytkownika, a więc pobranie linii papilarnych, wymaga od 2 do 10 skanowań. Pobieranie wzorca trwa około pół minuty. Odcisk palca jest rejestrowany z rozdzielczością 320 × 320 pixeli. Wzorzec zawiera 64 charakterystyczne cechy linii papilarnych, które są używane w późniejszej weryfikacji. Autoryzacja użytkownika trwa mniej więcej kilka sekund.
- BioTouch PCMCIA – czytnik odcisków palców, wykonany w formie karty PC-Card Type II do zastosowania w niemal dowolnym notebooku.

Systemy biometryczne w odróżnieniu od tradycyjnych gwarantują obecność użytkownika w momencie identyfikacji. To spowodowało, że do tej pory miały one ograniczone zastosowanie. Inne przeszkody przy wdrażaniu systemów biometrycznych to niska wydajność i wysoka cena. W ostatnich latach wydajność uległa znacznej poprawie, a ceny diametralnie spadły. Dzięki temu systemy biometryczne znajdują coraz większe zastosowanie.

6. Kryptograficzna ochrona danych

Kryptografia to nauka o przekazywaniu informacji w sposób zabezpieczony przed niepowołanym dostępem [9, 10]. Inaczej mówiąc, jest to nauka zajmująca się układaniem szyfrów, czyli procedur takiego przekształcania wiadomości, aby były one niemożliwe lub bardzo trudne do odczytania przez każdego, kto nie posiada odpowiedniego klucza. Można wyróżnić dwa główne nurty kryptografii:

- symetryczna – to rodzaj szyfrowania, w którym jawny tekst ulega przekształceniu na tekst zaszyfrowany za pomocą pewnego klucza, który równocześnie służy do odszyfrowania,
- asymetryczna – to rodzaj kryptografii, w której używa się zestawów dwóch lub więcej powiązanych ze sobą kluczy, umożliwiających wykonywanie różnych operacji kryptograficznych.

Kryptografia ma za zadanie zapewnić podstawowe warunki bezpieczeństwa, takie jak:

- szyfrowanie danych, aby osoby trzecie nie mogły ich odczytać nawet jeśli je przechwycą,
- możliwość sprawdzenia, czy podczas przesyłania dane nie zostały zmodyfikowane,
- zapewnienie wiarygodnej autoryzacji danych przez nadawcę.

Szyfrowanie nie zabezpiecza przed:

- dowolną zmianą całości lub części informacji,
- zamazaniem całości lub części informacji,
- wszczepieniem do informacji powtórzeń.

Szyfrowanie dotyczy danych, kluczy, haseł, funkcji składowych**,** procedur, kopii archiwalnych**.** Szyfrowanie zmniejsza efektywność systemu, w związku z czym metody powinny być w miarę proste. Szyfrowane powinny być przede wszystkim informacje przesyłane przez sieć. Proces szyfrowania i deszyfrowania odbywa się za pomocą algorytmu kryptograficznego i specjalnych kluczy [9, 14, 15].

6.1. Algorytmy symetryczne

Algorytmy symetryczne są to takie algorytmy, w których do szyfrowania i deszyfrowania używa się takiego samego, zwykle generowanego losowo, klucza tajnego.

Najistotniejszą rolę w algorytmach symetrycznych odgrywa długość klucza tajnego. Im dłuższy klucz, tym bezpieczniejszy szyfr. Wymagane bezpieczne minimum to 128 bitów. Algorytmy te można podzielić na dwie grupy:

- algorytmy blokowe (ang. block algorithm) – przetwarzaną jednostką informacji jest grupa bitów, czyli blok o ściśle zdefiniowanej długości; algorytmy te są łatwiejsze w implementacji.
- algorytmy strumieniowe (ang. stream algorithm) – przetwarzaną jednostką informacji jest jeden bit, dane są szyfrowane w sposób ciągły bez podziału na bloki.

Najpopularniejszym algorytmem symetrycznym jest algorytm blokowy DES (ang. Data Encryption Standard). Algorytm ten oparto na metodzie podstawień i permutacji. W miejsce znaku tekstu jawnego podstawia się znak zaszyfrowany oraz przestawia się znaki tekstu jawnego w pewien określony sposób. Szyfr ten przetwarza 64-bitowe bloki danych przy użyciu klucza o długości 56 bitów, w tym osiem bitów parzystości. Zbyt krótki klucz może być przyczyną braku poufności. Algorytm 3DES to kontynuacja DES, który dwukrotnie wydłuża klucz oraz zabezpiecza dane w procesie odpowiedniego szyfrowania, deszyfrowania i ponownego szyfrowania.

Algorytmy symetryczne zostały specjalnie zaprojektowane pod kątem szybkości działania oraz dużej liczby możliwych kluczy. Najlepsze algorytmy bazujące na kluczach symetrycznych gwarantują niemalże doskonałą ochronę danych. Gdy dane zostają zaszyfrowane za pomocą konkretnego klucza, nie ma praktycznie żadnej możliwości zdeszyfrowania wiadomości bez dostępu do identycznego klucza.

Przykładowo [4, 17] SQL Server umożliwia użycie kluczy symetrycznych do kodowania danych. Wszystkie informacje o kluczach symetrycznych są zawarte w procedurze sys.symmetric_keys.

Tworzenie klucza symetrycznego

```
CREATE SYMMETRIC KEY nazwa_klucza_sym
   WITH ALGORITHM = AES_256
   ENCRYPTION BY CERTIFICATE nazwa_cert_kodujacego;
GO
```

Otwieranie klucza

```
OPEN SYMMETRIC KEY nazwa_klucza_sym USING
CERTIFICATE nazwa_cert_kodujacego
```

Kodowanie

```
  DECLARE @wiad_zaszyfrowana NVARCHAR(100)
SET @wiad_zaszyfrowana = EncryptByKey(
Key_GUID(`nazwa_klucza'),     N'Wiadomosc do zaszyfrowania')
SELECT @wiad_zaszyfrowana AS 'Wiadomosc zaszyfrowana`
```

```
SELECT CAST(DecryptByKey(@wiad_zaszyfrowana) AS NVARCHAR)
```

Zamykanie klucza

```
CLOSE SYMMETRIC KEY nazwa_klucza_sym
```

Podstawową wadą algorytmów symetrycznych jest to, że obie strony przesyłające między sobą informacje muszą przed rozpoczęciem transmisji znać klucz. Inne, mniej znane algorytmy symetryczne to: IDEA, RC2, RC4, RC5, Blowfish.

6.2. Algorytmy asymetryczne

W algorytmach asymetrycznych (kryptografii z kluczem publicznym) do szyfrowania i deszyfrowania używa się zestawu dwóch lub więcej powiązanych kluczy. Szyfrowanie i podpisy cyfrowe zakładają istnienie dwóch kluczy: klucza prywatnego i klucza publicznego. Jeżeli zależy nam, aby informacja pozostała poufna, to nadawca szyfruje ją kluczem publicznym odbiorcy, a odbiorca deszyfruje ją swoim prywatnym kluczem. Jeżeli zależy nam, aby informacja pozostała autentyczna, to nadawca szyfruje ją swoim kluczem prywatnym, a odbiorca deszyfruje ją kluczem publicznym nadawcy. Jeżeli zależy nam, aby informacja pozostała poufna i autentyczna, to nadawca szyfruje ją swoim prywatnym kluczem, a uzyskany rezultat kluczem publicznym odbiorcy. Odbiorca deszyfruje ją swoim prywatnym kluczem, a uzyskany rezultat kluczem publicznym nadawcy.

Najpopularniejszy algorytm asymetryczny to RSA, który zachowuje autentyczność i poufność. Inne znane to: DSS, ECC, NTRU oraz Diffiego–Hellmana. Obecnie kryptografia asymetryczna jest szeroko stosowana do wymiany informacji poprzez kanały o niskiej poufności, takie jak Internet. Stosowana jest także w systemach elektronicznego uwierzytelniania, obsługi podpisów cyfrowych czy szyfrowania poczty.

Przykładowo istnieje możliwość tworzenia pary kluczy przy wykorzystaniu platformy SQL Server lub też można użyć zewnętrznej pary kluczy ładowanych z pliku. Klucze są tworzone z wykorzystaniem algorytmu RSA, a długość klucza wynosi 512, 1024 lub 2048 bitów. Podobnie jak w certyfikatach klucz prywatny może być zabezpieczony hasłem. Aby zakodować (zdekodować) informacje, korzystamy z funkcji EncryptByAsym() i DecryptByAsym(). Niżej znajdują się przykłady tworzenia oraz wykorzystania kluczy.

Tworzenie kluczy

```
CREATE ASYMMETRIC KEY nazwa_klucza
WITH ALGORITHM = RSA_2048
ENCRYPTION BY PASSWORD = 'hasło';
```

```
GO
```

Tworzenie (importowanie) klucza z pliku i uprawnienia użytkownika do jego używania

```
CREATE ASYMMETRIC KEY nazwa_klucza
AUTHORIZATION nazwa_uzyt
 FROM FILE = ' c:\key.tmp'
 ENCRYPTION BY PASSWORD = 'hasło';
GO
```

Szyfrowanie przy użyciu kluczy asymetrycznych

```
   DECLARE @wiad_zaszyfrowana NVARCHAR(100)
SET @wiad_zaszyfrowana =
EncryptByAsymKey(AsymKey_ID('nazwa_klucza'),
                N'Wiadomosc do zaszyfrowania')
SELECT @wiad_zaszyfrowana AS 'Wiadomosc
zaszyfrowana'
SELECT
CAST(DecryptByAsymKey(AsymKey_ID('nazwa_klucza'
),                    @wiad_zaszyfrowana) AS
NVARCHAR)
```

Niestety algorytmy asymetryczne mają wiele wad. Algorytmy z kluczem publicznym są bardzo wolne w działaniu, gdy tymczasem algorytmy symetryczne działają około 1000 razy szybciej. Rosną także wymagania dotyczące przepustowości. Zawsze trzeba będzie szyfrować dane szybciej niż jest to w stanie wykonać kryptografia z kluczem publicznym. Algorytmy asymetryczne są również bardzo podatne na ataki. Najczęściej stosuje się kombinację dwóch technik: symetrycznej i asymetrycznej jako algorytmy hybrydowe.

6.3. Algorytmy hybrydowe

Wady algorytmów asymetrycznych skłoniły do opracowania trzeciego rodzaju algorytmów, zwanych algorytmami hybrydowymi. Algorytmy te wykorzystują wolniejsze metody kryptograficzne oparte na kluczach publicznych do zabezpieczenia i dystrybucji losowo wygenerowanego jednorazowego klucza sesyjnego. Klucz ten następnie stosowany jest w algorytmach symetrycznych do zabezpieczenia transmisji wiadomości. Dane są szyfrowane za pomocą algorytmów symetrycznych, a klucze szyfrowane są algorytmem asymetrycznym. Odbiorca otrzymuje dwa zaszyfrowane ciągi bitów: dane i klucz sesji. Najpierw odszyfrowuje klucz sesji swoim kluczem prywatnym, a następnie dane, korzystając z rozszyfrowanego klucza. Technika ta zwana jest techniką hybrydo-

wą. W technice hybrydowej długości kluczy powinny być tak dobrane, aby system był jednakowo trudny do zaatakowania [7]. W tabeli 1 pokazano zestawienie kluczy symetrycznych i publicznych o podobnej odporności na ataki. Długość kluczy symetrycznych i publicznych podano w bitach.

Tabela 1

Długość klucza symetrycznego	Długość klucza publicznego
56	384
64	512
80	768
112	1792
128	2304

Ogólna zasada jest taka, że należy wybrać długość klucza algorytmu asymetrycznego bardziej bezpieczną niż ta porównywalna dla algorytmu symetrycznego. Klucze publiczne są na ogół dłużej używane i stosowane do ochrony większej liczby informacji.

6.4. Jednokierunkowa funkcja skrótu

Jednokierunkowa funkcja skrótu (ang.one-way hash function, fingerprint, message digest, cryptographic checksum) jest stosowana wtedy, gdy chcemy być pewni, że otrzymane dane podczas transmisji nie zostały zmodyfikowane. Algorytm ma za zadanie stworzenie
z dużego zbioru danych niewielkiego, ok. 160 bitów, skrótu tych danych. Zaletą metody jest to, że bardzo trudno jest odtworzyć pełną wiadomość ze skrótu. W kryptografii funkcja skrótu jest to funkcja przekształcająca ciąg binarny wejściowy o nieokreślonej z góry długości, nazwany ciągiem wstępnym (ang. pre-image), w ciąg wyjściowy o stałej długości, zwany wartością skrótu (ang. hash value). Jak sama nazwa wskazuje, funkcja jest jednokierunkowa, co oznacza, że bardzo łatwo można obliczyć skrót na podstawie ciągu wejściowego, ale trudno wygenerować ciąg, który daje w rezultacie określoną wartość skrótu. Funkcja jest również deterministyczna. W identycznym ciągu danych wejściowych powoduje wygenerowanie identycznego ciągu danych wyjściowych. Jedna zmiana w dowolnym bicie wejściowym powoduje zmianę przeciętnie połowy bitów wyjściowych. Cechy funkcji to: jednokierunkowość, łatwość wyliczenia, odporność na kolizje. Przykładowe funkcje jednokierunkowe to: iloczyn liczb pierwszych, funkcja plecakowa, funkcja Rabina, funkcja RSA czy logarytm dyskretny. Najczęściej stosowane algorytmy funkcji skrótu to: MD-2, MD-4, MD-5 i SHA. Każdy z algorytmów dzieli tekst na fragmenty o ustalonej wielkości
i przeprowadza serie operacji matematycznych na każdym fragmencie. MD-2 i MD-4 są dziś uważane za niewystarczająco bezpieczne. Uważany za standard MD-5 jest modyfikacją metody MD-4 i generuje skrót o długości 128 bitów. SHA oraz jej poprawiona wer-

sja SHA-1 na potrzeby standardu DSS generują skrót o długości 160 bitów. Jest to najwydajniejszy spośród obecnie stosowanych algorytmów.

Jednokierunkowa funkcja skrótu może być wykorzystana to szyfrowania haseł, dzięki czemu nawet system nie będzie mógł ich odszyfrować.

6.5. Podpis elektroniczny

Podpis elektroniczny to ciąg znaków w postaci elektronicznej, które wraz z danymi, do których są dołączone lub z którymi są logicznie powiązane, służą do identyfikacji osoby go składającej. Obecnie podpis elektroniczny najczęściej traktuje się na równi z podpisem cyfrowym. Należy mieć jednak świadomość, że podpis elektroniczny jest pojęciem znacznie szerszym niż podpis cyfrowy.

Według polskiej normy PN-I-02000 podpis elektroniczny to przekształcenie kryptograficzne danych, umożliwiające odbiorcy informacji sprawdzenie autentyczności i integralności danych oraz zapewniające nadawcy ochronę przed sfałszowaniem danych przez odbiorcę.

Cechy podpisu elektronicznego to:

- autentykacja – uniemożliwienie podszywania się pod daną osobę i wysłania w jej imieniu danych,
- integralność – zapewnienie wykrywalności wszelkich zmian w trakcie przesyłania i przechowywania informacji,
- autoryzacja – zapewnienie niemożności wyparcia się podpisu i treści informacji przez jej autora,
- umożliwienie weryfikacji podpisu przez osobę niezależną.

Podpis elektroniczny może być wykonany z wykorzystaniem kryptografii asymetrycznej za pomocą klucza prywatnego nadawcy. Wówczas jest on weryfikowany za pomocą klucza publicznego nadawcy. Metoda gwarantuje, że informacja pochodzi od określonego nadawcy, jednak może być rozszyfrowana przez każdego, kto ma klucz publiczny nadawcy. Nie jest to bezpieczny sposób przesyłania danych. W praktyce podpis elektroniczny jest wykonywany przez zaszyfrowanie skrótu wiadomości przy użyciu klucza prywatnego nadawcy z wykorzystaniem na przykład algorytmu RSA. Do wykonania podpisu elektronicznego może służyć również algorytm DSA. Jego bezpieczeństwo opiera się na trudności obliczenia dyskretnych logarytmów. DSA podpisuje nie dokument, ale jego wartość – po zastosowaniu funkcji haszującej SHA. Tym samym autor nie zdradza od razu, co podpisał.

Podpis elektroniczny umożliwia niepodważalne oznaczenie czasu złożenia podpisu. Wyklucza to wielokrotne wprowadzenie do obiegu tego samego dokumentu. Algorytmy podpisu elektronicznego są na ogół wyposażone w datowniki (ang. time stamps). Data i czas podpisu są dołączane do dokumentu i podpisywane razem

z jego treścią.

Samo zastosowanie technik kryptograficznych do utworzenia podpisu elektronicznego nie daje gwarancji, że osoba która użyła klucza prywatnego jest tą, za którą się podaje. Gwarancję taką ma dać system certyfikacji kluczy. Certyfikat cyfrowy to elektroniczne zaświadczenie o tym, że dane służące do weryfikacji podpisu elektronicznego są przyporządkowane do właściwej osoby i potwierdzają jej tożsamość. Certyfikat cyfrowy posiada:

- unikalny numer seryjny,
- tożsamość urzędu wydającego certyfikat,
- okres ważności certyfikatu,
- identyfikator właściciela certyfikatu,
- klucz publiczny właściciela certyfikatu,
- podpis cyfrowy urzędu certyfikacji, potwierdzający jego autentyczność.

Ogólnie można powiedzieć, że bezpieczny podpis elektroniczny to taki, który jest powiązany z danymi, do których został dołączony i jest przyporządkowany wyłącznie do osoby składającej ten podpis.

7. Protokoły przesyłania danych

Cały wysiłek włożony w zabezpieczenie danych nie ma sensu bez zabezpieczenia transmisji danych między klientami a serwerem podczas normalnej pracy użytkowników. Ze względu na specyfikę transmisji sieciowej praktycznie niemożliwe jest zabezpieczenie się przed podsłuchem w sieci. Standardowe protokoły miały przede wszystkim za zadanie zapewnić udaną transmisję, natomiast przy ich tworzeniu nie było mowy o zabezpieczeniu poufności danych. Nowoczesne systemy zarządzania bazą danych coraz częściej pozwalają na zastosowanie mechanizmów kryptograficznych, zapewniających poufność danych w czasie transmisji [6, 13, 14, 16]. Intensywny rozwój Internetu sprawia, że, istotnym zagadnieniem staje się bezpieczeństwo przesyłanych tą drogą informacji. Używany protokół HTTP (ang. HyperText Transfer Protocol) w warstwie komunikacyjnej opiera się na protokole TCP/IP, który jest pozbawiony możliwości zabezpieczenia informacji przed jej przechwyceniem. Nie ma również mechanizmów potwierdzających tożsamość odbiorców i nadawców komunikatu. HTTP umożliwia wprawdzie weryfikację tożsamości użytkowników na podstawie haseł, jednak hasła są przesyłane przez sieć w postaci niezaszyfrowanej i mogą zostać przechwycone w trakcie transmisji. Konieczne jest więc wprowadzenie dodatkowych środków ochrony. Konieczność zachowania modelu stosu TCP/IP i enkapsulacji pakietów spowodowała, że zaczęto szukać warstw, które należałoby zabezpieczyć tak, aby wprowadzenie nowych rozwiązań było jak najmniej uciążliwe dla końcowego użytkownika. Powstały dwa rozwiązania tego problemu.

Jedno z nich to stworzenie protokołu w warstwie trzeciej obok protokołu IP (ang. Internal Protocol). Rozwiązanie to wiąże się bezpośrednio ze sprzętem. Zabezpieczenie pakietów staje się niezależne od użytych protokołów i może być całkowicie przeźroczyste dla użytkownika. Przykład to protokół IPSec.

Drugie podejście zakłada zabezpieczenie danych w warstwie aplikacji. W tym wypadku do zapewnienia bezpiecznej transmisji potrzebny jest odpowiedni program. Powstało wiele protokołów spełniających to założenie, takich jak: TLS(SSL), SSH, S/MIME i pochodne HTTPs, SFTP, SCP itp. Wszystkie protokoły wykorzystują szyfry kryptograficzne z algorytmami symetrycznymi i asymetrycznymi. Również bardzo ważnym problemem stała się wymiana klucza. Powstały odpowiednie protokoły i mechanizmy, które pozwalają ustalić nienaruszalność i autentyczność danych.

7.1. Protokół IPSec

Protokół IP istniejący od około 20 lat jest protokołem niegwarantującym bezpieczeństwa przesyłanych danych. Dane są przesyłane otwartym tekstem, osoby nieupoważnione mogą podglądać całe pakiety poufnych informacji. Niemniej jednak jest to jeden z najlepiej zaprojektowanych protokołów, o czym świadczy czas jego wykorzystania. Na bazie tego protokołu powstał protokół IPSec (ang. Internal Protocol Security), który umożliwia:

- uwierzytelnianie – pakiety są podpisywane, co pozwala na jednoznaczne określenie i zweryfikowanie nadawcy,
- poufność przesyłanych informacji – dane są szyfrowane, na przykład algorytmami DES lub 3DES,
- integralność danych – sprawdzane są sumy kontrolne, pozwalające wykryć modyfikacje danych.

Protokół IPSec jest podrzędny w stosunku do IP. Oznacza to, że pakiety poruszające się w sieci mają zawsze nagłówek IP, a zaraz po nim enkapsulowany odpowiedni protokół bezpieczeństwa AH (ang. Authentication Header) lub ESP (ang. Encapsulated Security Payload).

Protokół AH jest odpowiedzialny za sprawdzenie integralności zarówno enkapsulowanego protokołu wyższej warstwy, jak i części nagłówka IP znajdującego się pod spodem. Ochroną obejmowane są te pola nagłówka, które nie ulegają zmianie podczas przesyłania przez sieć. Dla zapewnienia integralności wykorzystywane są funkcje skrótu, takie jak MD-5 czy SHA. AH zapewnia integralność, autentyczność oraz zabezpiecza przed autopowtarzaniem pakietów przez odpowiedni kod uwierzytelniający wiadomość (MAC) oraz włączeniem numerów sekwencji do pakietu.

Protokół ESP, którego podstawowym zadaniem jest zapewnienie poufności danych, jest bardziej złożony w porównaniu do AH. Zapewnia szyfrowanie i ochronę integralności danych, która obejmuje wyłącznie dane wyższej warstwy, bez nagłówka IP. Szyfrowanie i uwierzytelnianie jest opcjonalne. Algorytmy funkcji skrótu są takie same jak w AH, poufność zapewniają natomiast szyfry blokowe, takie jak DES czy 3DES.

Ciekawym rozwiązaniem jest protokół automatycznej negocjacji parametrów bezpieczeństwa IKE (ang. Internet Key Exchange), opracowany przez grupę roboczą IPSec. Głównym zadaniem tego protokołu jest wzajemne uwierzytelnianie hostów nawiązujących połączenie z wykorzystaniem IPSec, a następnie uzgadnianie krótkoterminowych kluczy kryptograficznych na potrzeby poszczególnych kanałów, przez które jest przesyłana informacja. Obydwie te funkcje są realizowane na podstawie skonfigurowanych na stałe danych uwierzytelniających, takich jak: hasła między parami hostów, certyfikaty czy klucze PGP.

IPSec działa w dwóch trybach: transportowym i tunelowym. Najbardziej intuicyjnym zastosowaniem protokołu IPSec jest tak zwany tryb transportowy. Pomiędzy

nagłówkiem IP a danymi dodawany jest dodatkowy nagłówek IPSec i dodatkowe pola, chroniące enkapsulowane wewnątrz dane. Tak utworzony pakiet, pomimo że może się on poruszać po sieci globalnej, stosuje się – ze względu na wymóg docierania pakietów w odpowiedniej kolejności – prawie wyłącznie w sieciach lokalnych. W trybie tunelowym ochrona kryptograficzna jest zapewniona dla całego pakietu IP. Zabezpieczony pakiet oraz dodatkowe pola (nagłówek IPSec i inne pola) są traktowane jako dane i dodawany jest do nich nowy nagłówek IP. Następuje enkapsulacja wewnętrznego pakietu IP w zewnętrzny. W ESP lub AH enkapsulowany jest pakiet wyższej warstwy wraz z nagłówkiem IP. Adresy źródłowe i docelowe umieszczone w nagłówku IP, znajdującym się pod IPSec, są z reguły adresami odpowiednich ruterów szyfrujących, tworzących pomiędzy sieciami lokalnymi tunele VPN (ang. Virtual Private Network). IPSec oraz VPN to technologie pozwalające w sposób bezpieczny przesyłać dowolne pakiety, pochodzące z różnych aplikacji i wykorzystujące różne protokoły.

Obecnie pracuje się nad utworzeniem w sieci publicznej, takiej jak Internet, osobnego kanału, przez który byłyby przesyłane zaszyfrowane i podpisane dane.

7.2. Protokół SSL

SSL (ang. Secure Socket Laser) jest protokołem ogólnego przeznaczenia do przesyłania zaszyfrowanych informacji w aplikacjach sieciowych. SSL został przyjęty jako standard szyfrowania na stronach www. SSL jest to zestaw reguł i standardów umożliwiających uwierzytelnianie, negocjowanie algorytmów, wymianę kluczy i szyfrowanie danych pomiędzy przeglądarką a serwerem z wykorzystaniem certyfikatów. SSL jest jedną z metod, zapewniających bezpieczeństwo w transakcjach finansowych dzięki wykorzystaniu funkcji skrótu. Podstawowym zadaniem protokołu SSL jest zapewnienie bezpiecznego kanału dla protokołów wyższych warstw. Ponadto jednym z podstawowych założeń SSL jest otwartość i rozszerzalność protokołu, czyli brak przywiązania do jednego, konkretnego algorytmu szyfrującego. Można wybrać dowolny algorytm, na przykład DES, 3DES lub RC4.

SSL gwarantuje następujące funkcje:

- autentyczność stron,
- poufność transmisji,
- integralność danych.

Autentyczność stron jest realizowana przez zbiór standardów, określanych jako infrastruktura klucza publicznego. Jest to system pozwalający na hierarchiczne potwierdzenie tożsamości klientów i serwerów. SSL wykorzystuje dwa podstawowe rodzaje szyfrów: symetryczne i asymetryczne. Szyfry symetryczne, jako szybsze, nadają się do wydajnego kodowania danych, przesyłanych w dużych strumieniach. Szyfry asymetryczne stosowane są do uwierzytelnienia stron oraz bezpośredniej wymiany klucza

symetrycznego. Klucz jest generowany losowo dla każdego połączenia. Wersja protokołu SSL 3.0 wykorzystuje algorytmy DES,3DES, RSA, IDEA, a także jednokierunkową funkcję skrótu MD-5.

W 1999 roku na bazie SSL 3.0 powstał ulepszony protokół, nazwany SSL wersja 3.1 lub TLS (ang. Transport Layer Security). TLS łączy się generalnie z protokołem HTTP, jednak można go wykorzystywać również z innymi protokołami warstwy aplikacyjnej, takimi jak: Telnet, SMTP, POP, IMAP, FTP.

Sesja TLS przebiega według stałego schematu:

- Ustalenie połączenia TCP klienta z serwerem.
- Klient wysyła do serwera komunikat, który zawiera: wersję protokołu, obsługiwane algorytmy kryptograficzne, metody kompresji i liczbę losową potrzebną przy generowaniu kluczy (ClientHello).
- Serwer odpowiada klientowi, podając wersję protokołu, rodzaj szyfrowania i kompresji oraz liczbę losową (ServerHello).
- Serwer wysyła swój certyfikat, pozwalający klientowi na sprawdzenie swojej tożsamości (Certificate).
- Serwer wysyła informacje o swoim kluczu publicznym; klucz jest określony przez algorytm przesyłany w poprzednim komunikacie (ServerKeyExchange).
- Serwer informuje, że klient może przejść do następnej fazy zestawienia połączenia (ServerHelloDone).
- Klient na podstawie ustalonych w poprzednich komunikatach dwóch liczb losowych, swojej i serwera, generuje klucz sesji używany do faktycznej wymiany danych i wysyła go serwerowi, używając jego klucza publicznego (ClientKeyExchange).
- Klient zawiadamia serwer, że ten może się przełączyć na komunikację szyfrowaną (ChangeCipherSpec).
- Klient jest gotowy do odbierania zakodowanych danych (Finished).
- Serwer zawiadamia, że odtąd będzie wysyłał zakodowane dane (Finished).

Możliwa jest również autentykacja klienta. Dodatkowo serwer żąda od klienta certyfikatu, klient wysyła certyfikat po kroku ServerHelloDone, potwierdza certyfikat komendą Certificate Verify poprzez zaszyfrowanie kluczem prywatnym wszystkich ustalonych już danych i wysyła je do serwera.

8. Audyt

Zgodnie z definicją audyt (ang. audit) to szczegółowa analiza działalności danej organizacji, prowadzona przez zewnętrznych, niezależnych specjalistów w celu ujawnienia ewentualnych problemów czy nieprawidłowości w jej funkcjonowaniu. Obiektem audytu mogą być serwery bazujące na popularnych systemach operacyjnych, routery i firewalle, badane pod kątem reguł kontroli dostępu i zabezpieczeń systemu operacyjnego oraz ogólna topologia sieci i usługi sieciowe.

8.1. Rodzaje audytu

Ogólnie można wyróżnić:

- audyt informatyczny, czyli proces zbierania i oceniania dowodów, który ma na celu zweryfikowanie działania systemu informatycznego, określenie czy system działa poprawnie, czy chroni przed niepożądanymi zdarzeniami lub czy umożliwia wczesne ich wykrywanie i zapobieganie skutkom,
- audyt bezpieczeństwa, czyli przegląd i ocenę działania systemu komputerowego pod kątem adekwatności istniejących zabezpieczeń do ustalonej polityki bezpieczeństwa oraz w celu wykrycia potencjalnych zagrożeń,
- audyt baz danych, czyli proces mający na celu określenie statusu bezpieczeństwa danych w firmie w odniesieniu do uregulowań prawnych obowiązujących w danym kraju.

Podstawą audytu jest wszechstronna analiza zarówno struktury organizacyjnej i technicznej baz danych, jak i norm oraz standardów jej zabezpieczenia. Audyt daje gwarancje zastosowania sprawdzonych metod i narzędzi.

Istnieje wiele sytuacji, w których powinno się przeprowadzić audyt, na przykład w celu:

- sprawdzenia poziomu bezpieczeństwa systemu,
- stwierdzenia incydentów włamań do sieci w systemie,
- wprowadzenia zmian w systemie,
- udostępnienia nowych usług sieciowych użytkownikom wewnętrznym lub publicznym.

Audytowi mogą podlegać różne elementy systemu, takie jak:
- mechanizmy przechowywania danych,
- mechanizmy kontroli dostępu,
- bezpieczeństwo kodu,
- obsługa błędów kodu,
- obciążenie serwera.

8.2. Przykłady audytu w systemach

Jako przykład usług komercyjnego systemu bazy danych może posłużyć oferta Oracle Export Services. Oracle oferuje audyt bazy danych między innymi w postaci takich działań, jak:
- monitorowanie obciążeń procesorów serwera,
- monitorowanie zajętości pamięci,
- przegląd plików konfiguracyjnych instancji,
- przegląd plików logów bazy danych,
- przegląd definicji struktur logicznych i fizycznych,
- badanie podstawowych współczynników wydajności,
- badanie podstawowych najmniej wydajnych zapytań.

Oracle od wersji 9 umożliwia nawet audyt wzdłuż kolumn w zapytaniach.
Inne podejście prezentuje firma EXE GROUP, której audyt obejmuje:
- audyt funkcjonalny,
- identyfikację wersji systemu, modułów i komponentów,
- przegląd i analizę architektury systemu, możliwość rozbudowy,
- przegląd struktur logicznych i fizycznych systemu, plików konfiguracyjnych,
- testy konfiguracji i logów bazy SQLNet/Net8,
- szczegółowy audyt bazy danych (słowników, nadawanych przywilejów w systemie, ról, analizę aktywności użytkowników),
- prawa dostępu do modułów systemu,
- analizę stopnia wykorzystania zasobów informatycznych systemu,
- bezpieczeństwo systemu.

Pierwszym etapem przeprowadzenia audytu jest uzyskanie informacji na temat działania systemu, usług sieciowych, topologii oraz haseł dostępu do urządzeń. Następnie przeprowadzana jest w sieci klienta sesja zbierania danych bezpośrednio z systemu – automatyczna i półautomatyczna – za pomocą wyspecjalizowanych urządzeń analitycznych. Na podstawie uzyskanych informacji sporządzany jest raport w formie wstępnej w celu konsultacji z klientem. Bardzo istotną rzeczą jest jednolity i ściśle określony format przekazywanych dokumentów. Umożliwia to szybkie wyciąganie wniosków, porównywanie okresowo przeprowadzanych raportów, łatwość

implementacji wskazanych zaleceń. Wyniki przeważnie są przekazywane w postaci graficznej, ułatwiającej ich analizę.

Efektem przeprowadzonego audytu jest uzyskanie zarówno aktualnej szczegółowej listy nieprawidłowości w zabezpieczeniach serwerów sieciowych, jak i wielu wskazówek dotyczących poprawy poziomu bezpieczeństwa.

Podsumowanie

Bezpieczeństwo danych jest procesem, a nie stanem, który można osiągnąć. Wymaga ono stałych nakładów czasowych i finansowych. Jednak, jak się okazuje, najsłabszym ogniwem są ludzie. Jak powiedział sławny kraker Kelvin Mitnick:

Inwestujecie miliony w firewalle, systemy biometryczne i najprzeróżniejsze zabezpieczenia techniczne. A potem okazuje się, że wasz pracownik zapisał hasło na karteczce i przylepił do monitora [11].

Bibliografia

[1] Anderson R., *Inżynieria zabezpieczeń*, WNT, Warszawa 2005.
[2] Chałon M., *Ochrona i bezpieczeństwo systemów baz danych*, [w:] *Inżynieria komputerowa*, praca zbiorowa pod redakcją W. Zamojskiego, WKŁ, Warszawa 2005.
[3] Cole E., Krutz R., Conley J., *Bezpieczeństwo sieci. Biblia*, Wydawnictwo Helion, Gliwice 2005.
[4] Gallagher S., *SQL Server 7. Księga eksperta*, Wydawnictwo Helion, Gliwice 2001.
[5] Greene J., *Oracle8 Server. Księga eksperta*, Wydawnictwo Helion, Gliwice 2000.
[6] Horton M., Mugge C., *Bezpieczeństwo sieci. Notes antyhakera*, WNT, Warszawa 2004.
[7] Kahn D., *Łamacze kodów. Historia kryptologii*, WNT, Warszawa 2004.
[8] Kifner T., *Polityka bezpieczeństwa i ochrony informacji*, Wydawnictwo Helion, Gliwice 1999.
[9] Koblitz N., *Algebraiczne aspekty kryptografii*, WNT, Warszawa 2000.
[10] Menezes A., Oorschot P., Vanstone S., *Kryptografia stosowana*, WNT, Warszawa 2005.
[11] Mitnick K., Simon W., *Sztuka podstępu. Łamałem ludzi, nie hasła*, Wydawnictwo Helion, Gliwice 2005.
[12] Potter B., Fleck B., *802.11.Bezpieczeństwo*, Wydawnictwo Helion, Gliwice 2004.
[13] Schetina E., Green K., Carlson J., *Bezpieczeństwo w sieci*, Wydawnictwo Helion, Gliwice 2002.
[14] Schneier B., *Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C*, WNT, Warszawa 2002.
[15] Stinson D., *Kryptografia. W teorii i praktyce*, WNT, Warszawa 2005.
[16] Strebe M., *Bezpieczeństwo sieci*, Wydawnictwo Mikom, Seria: Podstawy, Warszawa 2005.
[17] Waymire R., Sawtell R., *MS SQL Server 2000 dla każdego*, Wydawnictwo Helion, Gliwice 2002.

CZĘŚĆ III

Integralność baz danych

Integralność (ang. data integrity) to formalna poprawność bazy danych, jej fizycznej organizacji, zgodność ze schematem bazy danych i regułami dostępu.

Encyklopedia of Internet and New Technologies

9. Zapewnienie integralności w bazach danych

Bardzo ważnym zadaniem jest zapewnienie integralności danych [1, 2, 3], czyli odpowiednich mechanizmów zabezpieczających przed skutkami przypadkowych błędów logicznych, konfliktów we współbieżnym dostępie do danych oraz skutkami awarii oprogramowania i sprzętu komputerowego. System integralny to taki, który dostarcza wiarygodnych danych i jest zabezpieczony przed nieautoryzowaną modyfikacją informacji. Systemy baz danych powinny zapewniać możliwość sprawdzania i ewentualnej korekty wprowadzanych danych oraz zawierać odpowiednie mechanizmy, zapewniające prawidłowe przetwarzanie danych. Integralność to zapewnienie kompletności, poprawności i wiarygodności danych zgromadzonych w bazie. Proces ochrony integralności obejmuje:

- kontrolę danych wejściowych oraz synchronizację dostępu do danych,
- poprawianie, czyli korektę danych, cofanie i odtwarzanie stanu bazy,
- archiwizację przez tworzenie kopii bazy oraz zapisów działania systemu,
- testowanie, czyli sprawdzanie poprawności zawartości bazy.

Pojęcie integralność obejmuje integralność statyczną i transakcyjną. Źródłem naruszenia integralności statycznej są błędy logiczne w danych oraz brak poprawnie skonstruowanego schematu bazy danych. Zagrożeniem integralności transakcyjnej są awarie oprogramowania i sprzętu oraz współbieżny dostęp do danych.

10. Integralność statyczna

Integralność statyczna dotyczy poprawnie zaprojektowanego schematu bazy danych oraz spełnienia ograniczeń nałożonych na wartości atrybutów opisujących obiekty w bazie. Ważne jest, aby nienaruszona była spójność danych. Kontrola zgodności danych prowadzona jest przez zdefiniowanie, a następnie sprawdzenie ograniczeń integralnościowych, zwanych regułami integralności. Ograniczenia integralnościowe są zabezpieczeniem przed nadawaniem danym niewłaściwych wartości oraz przed niewłaściwymi związkami między nimi. Kontrola ograniczeń może odbywać się na poziomie serwera lub aplikacji.

10.1. Integralność semantyczna

Jeżeli wartości danych spełniają wcześniej zdefiniowane i nałożone ograniczenia, to mówimy, że zachowana jest integralność semantyczna. Niżej podano opisany w języku SQL przykład zapewnienia integralności semantycznej w bazie relacyjnej. Integralność encji i dziedziny zapewniono przez zdefiniowanie klucza głównego, określenie jego właściwości (NOT NULL UNIQUE) i sprawdzenie poprawności wartości atrybutu NrStud (CHECK).

Przykład 6

```
CREATE TABLE Studenci;
(NrIndeksu Number(5) NOT NULL UNIQUE;
NazwiskoStud Varchar (15);
NazwaWydziału Varchar (20);
Stypendium Decimal (7,2);
PRIMARY KEY (NrIndeksu));
CHECK (NrStud BETWEEN 10 AND 500);
```

Utrata integralności semantycznej może wynikać na przykład z następujących przyczyn:

- niewłaściwej wartości pola danych (np. wiek 200 lat),
- niezgodności między wartościami pól tej samej jednostki danych,

- niespójności między polem jednostki, a polem jednostki połączonej,
- obecności pustych pól, nieważnych wartości.

Zapewnienie integralności semantycznej ma na celu zabezpieczenie danych przed celową lub przypadkową błędną modyfikacją danych, a więc odrzucenie wszelkich akcji powodujących niespójność bazy lub uruchomienie akcji, które przywracają poprawność i spójność bazy. Integralność można wymusić w sposób deklaratywny poprzez więzy integralności oraz w sposób proceduralny poprzez tzw. wyzwalacze (ang. triggers).

Więzy integralności są to pewne warunki, które muszą być spełnione przez określony podzbiór danych w bazie. Warunki te muszą pozostać prawdziwe w przypadku każdej operacji modyfikacji w bazie danych. Każda operacja naruszająca te więzy musi zostać anulowana. Typowy przykład narzuconych ograniczeń: wartości atrybutów w danej tabeli muszą pochodzić z określonej dziedziny. Więzy integralności dzielimy na statyczne i dynamiczne. Więzy statyczne muszą być spełnione w bieżącym i następnym stanie bazy, więzy dynamiczne, temporalne określają poprawność danych w odniesieniu do historii stanów przechowywanych w bazie.

Wyzwalacze są to procedury uruchamiane automatycznie przez system przy zajściu jakiegoś zdarzenia dotyczącego danego obiektu. Ich zadaniem jest kontrola poprawności wprowadzanych lub przetwarzanych danych do postaci akceptowalnej przez użytkownika. Wyzwalacze nie dopuszczają do zmian lub cofają zmiany, które naruszają zasady integralności. Utworzenie wyzwalacza polega na określeniu zdarzenia uruchamiającego wyzwalacz tabeli będącej jego celem i zaprojektowaniu akcji, jaką ma ta procedura wykonać.

W języku SQL instrukcje INSERT, UPDATE, DELETE służą do uruchamiania procedury wyzwalacza. Akcja może być uruchamiana przed (wyzwalacz typu BEFORE) lub po (wyzwalacz typu AFTER) określonym zdarzeniu. Może też chronić przed zajściem zdarzenia lub może zmienić wynik zdarzenia. Wyzwalacz może zadziałać raz dla operacji, która go wywołuje lub wielokrotnie dla każdego wiersza (z klauzulą FOR EACH ROW). Wykorzystanie tej klauzuli umożliwia dostęp zarówno do wartości sprzed zajścia zdarzenia (OLD. atrybut relacji), jak i do nowych wartości atrybutu (NEW. atrybut relacji), powstałych w wyniku wstawienia, modyfikacji czy skreślenia wiersza w trakcie zdarzenia. Wyzwalacze mogą zagnieżdżać w sobie inne wyzwalacze. Każdy wyzwalacz może uruchamiać inny. Liczba poziomów zagnieżdżenia zależy od systemu. Typowym zastosowaniem gniazda wyzwalaczy jest zapisywanie kopii wierszy modyfikowanych przez inny wyzwalacz. Wyzwalacze mogą wykonywać proste analizy oraz porównania przed i po wykonaniu modyfikacji danych. Mogą wykonywać akcje zależne od wyniku porównania. Wyzwalacze są dobrym sposobem na egzekwowanie bardziej złożonych reguł integralności. Na ogół jednak powinno się używać przede wszystkim więzów integralności, które mówią nam, jakie warunki powinny być spełnione, a nie jak je sprawdzać. Szczegółowy opis procedur wyzwalających z przykładami można znaleźć w cytowanej pracy autorki [5].

Bardzo efektywnym mechanizmem zabezpieczającym integralność danych są procedury składowane [4], poprzez które realizowany jest dostęp do danych. Są one wywoływane bezpośrednio przez aplikacje klienta. Klient podając nazwę i parametry, wywołuje procedurę, która sprawdza, czy wprowadzone zmiany nie naruszają integralności semantycznej.

10.2. Integralność encji i integralność referencyjna

Integralność encji [4, 9, 12] zapewnia się na etapie definiowania schematu bazy danych. Każda tabela musi posiadać klucz główny (ang. PRIMARY KEY), a wartości kolumny wybranej jako klucz główny muszą być w ramach tabeli unikatowe (ang. UNIQUE) i różne od wartości NULL [8]. Klucz główny to wyróżniony atrybut lub minimalny zbiór wyróżnionych atrybutów, które w sposób jednoznaczny identyfikują dane w tabeli [5]. Przykład zapewnienia integralności encji w języku SQL przedstawiono poniżej.

Przykład 7

```
CREATE TABLE Studenci;
(NrIndeksu Number(5) NOT NULL UNIQUE;
NazwiskoStud Varchar (15);
NazwaWydziału Varchar (20);
Stypendium Decimal (7,2);
PRIMARY KEY (NrIndeksu));
```

W schemacie bazy danych tabele powiązane są między sobą kluczami Powiązania te realizowane są przez klucze obce (ang. FOREIGN KEY).

Powiązania między kluczami encji pociągają za sobą konieczność określenia reguł postępowania w wypadku wykonywania operacji na tabelach nadrzędnych w stosunku do innych tabel. To właśnie integralność referencyjna [4, 5, 12, 15] określa stany, w jakich może znajdować się wartość klucza obcego w danej tabeli. Wartość klucza obcego w danej tabeli musi być albo równa wartości klucza głównego w tabeli z nią powiązanej, albo wartości NULL. Niżej podano przykłady zapewnienia integralności referencyjnej w języku SQL.

Przykład 8

```
CREATE TABLE Wydziały;
(NazwaWydziału Char (15);
RokStudiów Smallint;
```

```
KodKursu Char (3);
NrStud Number (5);
PRIMARY KEY (NazwaWydziału);
FOREIGN KEY(NrIndeksuIDENTIFIES Studenci);
ON DELETE SET NULL;
ON UPDATE CASCADE);
```

Przykład 9

```
CREATE TABLE Wydziały;
(NazwaWydziału Char (15);
RokStudiów Smallint;
KodKursu Char (3);
NrStud Number (5);
PRIMARY KEY (NazwaWydziału);
FOREIGN KEY(NrIndeksu IDENTIFIES Studenci);
DELETE RESTRICTED;
ON UPDATE CASCADE);
```

Definicję dwóch tabel z uwzględnieniem więzów integralności oraz określenie asercji, czyli więzów niezależnych od tabeli i dziedziny, podano w przykładzie 10.

Przykład 10

```
CREATE TABLE Grupy;
(NrGrupy Integer (2), NOT NULL UNIQUE;
NazwaGrupy Varchar (30);
NrStarGrupy Integer (2));
PRIMARY KEY (NrGrupy));
CREATE TABLE Studenci;
(NrIndeksu Integer(2), NOT NULL;
Nazwisko Varchar (20);
Adres Varchar (50),
DataRozp.Stud DATE DEFAULT Sysdate;
Stypendium Decimal (8,2);
NrGrupy Integer (2));
PRIMARY KEY (NrIndeksu);
FOREIGN KEY (NrGrupy);
REFERENCES Grupy;
ON UPDATE CASCADE;
ON DELETE SET NULL;
CHECK (NrStud BETWEEN 10 AND 500);
```

```
CREATE ASSERTION MinStypendium;
AFTER;
INSERT ON Studenci;
UPDATE OF Stypendium ON Studenci;
CHECK;
( NOT EXIST;
(SELECT *;
FROM Studenci;
WHERE Stypendium > 700));
```

Dzięki więzom asercji uniemożliwia się przyznanie studentom zbyt niskich stypendiów.

Innym sposobem zachowania integralności referencyjnej są wspomniane wcześniej procedury, zwane wyzwalaczami.

10.3. Integralność w różnych modelach baz danych

Jeżeli mamy do czynienia z obiektową bazą danych, to problem integralności jest bardziej skomplikowany niż w bazach relacyjnych. W bazie obiektowej każdy obiekt musi mieć unikatowy identyfikator w bazie. Ponieważ dostęp do obiektów odbywa się za pomocą metod tych obiektów, ograniczenia integralności muszą być definiowane poprzez pewne warunki umieszczone w treściach metod. Integralność danych wynika ze związków między obiektami i klasami. Model obiektowy obejmuje cztery rodzaje integralności:

- integralność klasa–klasa – nadklasa nie może być usunięta, dopóki nie zostaną usunięte powiązane z nią podklasy,
- integralność klasa–obiekt – klasa nie może być usunięta, dopóki nie zostaną usunięte powiązane z nią obiekty,
- integralność dziedziny – atrybuty są definiowane na wcześniej utworzonych klasach lub na zbiorach identyfikatorów obiektów,
- integralność referencyjna – ponieważ klasy mogą być powiązane z innymi klasami poprzez związki, w modelu obiektowym mamy więc do czynienia z integralnością referencyjną, podobną do integralności referencyjnej w modelu relacyjnym.

Ciekawostkę stanowią bazy temporalne, czyli modelujące historię zmian rzeczywistości. Model temporalny to model uwzględniający czas rejestracji danych oraz rzeczywisty czas zajścia zdarzeń. Temporalne więzy integralności określają poprawność danych zarówno dla bieżącego stanu bazy danych, jak i stanów poprzednich i przyszłych. Ogólnie są one przydatne wszędzie tam, gdzie są istotne zależności czasowe między danymi. Można na przykład zadać taki warunek: pensja pracownika nie

może wzrosnąć o więcej niż 12% w ciągu czterech kolejnych posiedzeń zarządu. W bazach statycznych możliwe jest tylko sprawdzenie warunku, czy np. pensja pracownika nie może wzrosnąć jednorazowo o więcej niż 12%. Do specyfikacji temporalnych więzów integralności stosowana jest logika temporalna. Logika ta umożliwia rozważanie zależności czasowych bez wprowadzenia czasu explicite. Zazwyczaj operuje na czasie składającym się z dyskretnych wydarzeń. W logice temporalnej wykorzystuje się operatory odnoszące się do przeszłości. Jeżeli chce się dołączyć nowy stan, należy zweryfikować poprzednie; jeśli są spełnione, to nowa historia stanów jest zapamiętana.

11. Integralność transakcyjna

Transakcja jest to ciąg operacji wykonywanych na danych w bazie, inaczej mówiąc ciąg instrukcji języka SQL tworzących pewną całość [3,7,10,11]. Transakcja, od momentu jej rozpoczęcia aż do chwili jej zakończenia, może znajdować się w jednym z pięciu stanów: aktywna, częściowo zatwierdzona, zatwierdzona, nieudana, odrzucona.

Transakcja może zakończyć się:

- powodzeniem, wówczas jest zatwierdzana (ang. COMMIT),
- niepowodzeniem, wówczas jest odrzucana (ang. ABORT) lub wycofywana (ang. ROLLBACK).

Transakcja powinna posiadać następujące cechy:

- niepodzielność (ang. atomicity) – gwarantuje, że transakcja musi być wykonana w całości lub całkowicie anulowana,
- spójność (ang. consistency) – zapewnia nienaruszalność zasad integralności, czyli baza po operacji musi być spójna,
- izolację (ang. isolation) – dane, które są modyfikowane przez jedną transakcję przed jej zakończeniem muszą być niedostępne dla innych transakcji,
- trwałość (ang. durability) – gwarantuje, że po pomyślnym zakończeniu transakcji zmiany w bazie będą na stałe zapisane i nie zostaną utracone w wyniku jakiejkolwiek późniejszej awarii.

Źródłem zagrożeń dla integralności transakcyjnej są:

- awarie programowe i sprzętowe zaistniałe w trakcie wykonywania transakcji; na ogół wynikają one z utraty zawartości pamięci operacyjnej komputera,
- błędy transakcji spowodowane wykonywaniem zabronionych operacji (dzielenie przez zero), czy niemożnością wykonania danej operacji (niewystarczający stan konta),
- nieprawidłowa realizacja równoczesnego dostępu do tych samych danych przez różnych użytkowników,
- błędy zapisu lub odczytu danych z dysku podczas wykonywania transakcji,
- zanik napięcia, zamazanie danych przez operatora, pożar, kradzież, sabotaż.

Utrzymanie integralności wymaga:

- właściwego zarządzania transakcjami, aby po awarii przywrócić spójny stan bazy danych,

• odpowiednich mechanizmów sterowania współbieżnym dostępem do danych.

Można wyróżnić dwie strategie działania transakcji:

Transakcje działają na własnych kopiach, które są zamieniane z oryginalnymi obiektami w fazie zatwierdzenia. Rozwiązanie to, wymagające większego obszaru pamięci, jest rzadko stosowane ze względu na większą możliwość awarii.

Transakcje działają bezpośrednio na bazie danych w specjalnych plikach, zwanych dziennikami, potocznie nazywanych logami (ang. log). W logach zapisują wszelkie operacje aktualizacyjne, które wykonały, wraz z obiektami przed i ewentualnie po aktualizacji. W razie awarii baza jest odtwarzana przez analizę informacji zapisanej w logu.

11.1. Dzienniki transakcji

Jeżeli w wyniku awarii systemu ciąg operacji tworzących transakcje zostanie przerwany, to stosuje się zasadę wszystko albo nic, to znaczy należy wycofać z bazy efekty częściowego wykonania transakcji. Jest to możliwe dzięki prowadzeniu dziennika transakcji, zwanego w skrócie logiem. Log jest to plik, w którym rejestruje się przebieg wykonywania operacji. W logu rejestrowany jest identyfikator transakcji, adresy wszystkich obiektów aktualizowanych przez daną transakcję, wartości obiektów przed i po modyfikacji oraz informacje dotyczące przebiegu transakcji. Dopiero po zapisaniu polecenia zatwierdzającego transakcje (ang. COMMIT), a dokładnie punktu zatwierdzenia transakcji (ang. COMMIT POINT), wszelkie modyfikacje wykonywane przez transakcje są przepisywane do fizycznej bazy danych. Przed osiągnięciem punktu zatwierdzenia wszystkie takie aktualizacje trzeba uważać za tymczasowe, ponieważ mogą być odwołane.

Technika ta, zwana wyprzedzającym zapisem do logu, pozwala łatwo wycofać transakcję, która nie może zostać dokończona. Zostawia się więc stan bazy bez zmiany i pozostaje tylko poinformować użytkownika, że transakcja nie doszła do skutku.

Jeśli transakcja została zatwierdzona, to zmiany przez nią wprowadzone muszą być na trwałe zapamiętane w bazie, nawet jeśli wystąpiła awaria i nie zostały przepisane do fizycznej bazy. Należy wówczas ustalić, które transakcje zostały zatwierdzone, a nie przepisane do fizycznej bazy, które zaś nie były zatwierdzone i powinny być anulowane. W tym celu jest określany tzw. punkt kontrolny logu (CP), czyli punkt zapisu do logu wszystkich transakcji i uaktualnianie ich.

Tworzenie punktu kontrolnego polega na:

• wstrzymaniu uruchomień nowych transakcji,

• zaczekaniu, aż wszystkie rozpoczęte transakcje wprowadzą do logu zapisy potwierdzenia <COMMIT> lub odrzucenia <ABORT>,

• przesłaniu logu na dysk,

- wprowadzeniu do logu zapisu <CP> i ponownym przesłaniu logu na dysk,
- wznowieniu wstrzymanych transakcji.

Jeśli wystąpi awaria, to bada się przebieg transakcji od ostatniego punktu kontrolnego. Wszystkie transakcje, które po punkcie kontrolnym są zatwierdzone instrukcją COMMIT uaktualnia się i zapisuje na dysku. Transakcje te mogły nie być fizycznie przepisane. Inne transakcje anuluje się, stosując instrukcję ROLLBACK.

Można wyróżnić kilka rodzajów logów. Na szczególną uwagę zasługują [7, 8]:

- logi z unieważnieniem,
- logi z powtarzaniem,
- logi hybrydowe, powstałe z połączenia tych dwóch typów.

W logach z unieważnieniem w celu odtwarzania stanu bazy danych likwiduje się zmiany wprowadzone przez niezatwierdzone transakcje. Odtwarzanie polega na przywróceniu uprzednich wartości wszystkich niezatwierdzonych transakcji. Zmiany w bazie danych muszą być zapamiętane na dysku, zanim na nim zostaje zapamiętany zapis COMMIT. Dane należy zapisywać na dysk natychmiast po zakończeniu transakcji, co może prowadzić do wielokrotnego zwiększenia liczby potrzebnych dostępów do dysku.

W logach z trybem powtarzania ignoruje się transakcje niezatwierdzone. Zmiany wykonywane przez transakcje zatwierdzone powtarza się. Odtwarzanie polega na przypisaniu nowych wartości ze wszystkich transakcji zatwierdzonych. Zapis COMMIT zostaje zapamiętany na dysku, zanim zapisze się na nim wartości zmienione. Prowadzi to do konieczności przechowywania zmodyfikowanych bloków w buforach. Bloki te trzyma się aż do zatwierdzenia transakcji i zakończenia tworzenia zapisów w logu. Może to zwiększyć liczbę buforów, potrzebnych przy wykonywaniu transakcji.

Oba typy logów mogą spowodować powstanie sprzecznych wymagań. Dotyczy to obsługi buforów w trakcie wstawiania punktów kontrolnych, jeśli elementy danych nie są całymi blokami lub zbiorami bloków. Dlatego też stosuje się logi stanowiące połączenie dwóch technik: odtwarzania i powtarzania. Logi te są bardziej elastyczne w zakresie kolejności wykonywania czynności, ale wymagają zapisywania większej liczby danych do logu. Odtwarzanie polega na powtórzeniu transakcji zatwierdzonych oraz unieważnieniu transakcji niezatwierdzonych.

Aby w sposób jasny i klarowny opisać efekty wykonania transakcji w warunkach awaryjnych, wprowadzono pseudojęzyk oparty na komendach SQL.

11.2. Pseudojęzyk

Do prześledzenia algorytmów zarządzania transakcjami wprowadzono notację, opisującą operacje związane z przesyłaniem danych.

INPUT < X > pobieranie do bufora pamięci elementów bazy danych X z dysku

OUTPUT < X > kopiowanie bufora pamięci na dysk
READ < X, y > kopiowanie elementu bazy danych X do lokalnej zmiennej transakcji y
WRITE < X, y > kopiowanie wartości zmiennej y do elementu bazy danych X w buforze pamięci
SET < X > tworzenie przez transakcje nowej wartości w swojej przestrzeni adresowej
<SEND LOG> przesyłanie logu, czyli skopiowanie pliku na dysk
< BEGIN T > rozpoczęcie transakcji
<COMMIT T > zatwierdzenie transakcji
<ABORT T > unieważnienie transakcji
<T, y, old w > zapis aktualizujący w logu z unieważnieniem
<T, y, new w > zapis aktualizujący w logu z powtórzeniem
<T, y, old w, new w > zapis aktualizujący w logu hybrydowym (z unieważnieniem/powtórzeniem)
<CP> punkt kontrolny
<BEGIN CP (T_1, T_2, ... T_K) > początek bezkolizyjnego punktu kontrolnego (transakcje aktywne)
<END CP> koniec punktu kontrolnego
R_i (X) transakcja T_i czyta element z bazy danych
W_i (X) transakcja T_i zapisuje element do bazy danych
$P(T_i,T_j)$ plan sekwencyjny gdy *i*-ta transakcja poprzedza transakcję *j*-tą
$P(T_j,T_i)$ plan sekwencyjny gdy *j*-ta transakcja poprzedza transakcję *i*-tą
$O_k(T_i)$ *k*-ta operacja *i*-tej transakcji,
$STOP_i$ (X) – blokada do odczytu zakładana przez transakcję T_i
GO_i (X)- zwolnienie blokady przez transakcję T_i
Zazwyczaj przebieg transakcji odbywa się w dwóch krokach:

Krok 1. Operacja odczytania elementu bazy danych:

INPUT < X > pobieranie do bufora pamięci elementów bazy danych X z dysku
READ < X, y > wczytanie zawartości buforów do przestrzeni adresowej transakcji

Krok 2. Operacja zapisania nowej wartości elementu do bazy danych:

SET < X > tworzenie przez transakcję nowej wartości w swojej przestrzeni adresowej
WRITE < X, y > kopiowanie wartości zmiennej y do elementu bazy danych X w buforze pamięci
OUTPUT < X > zapisanie z bufora pamięci na dysk
Wszystkie trzy typy logów mają podobną postać, różnią się tylko zapisem aktuali-

zacyjnym. Niżej przedstawiono zapis w logach, przyjmując dla uproszczenia, że mamy do czynienia z jedną transakcją, składającą się z dwóch elementów A i B.

Postać logu typu unieważnienie:
< BEGIN T >
< T, A, old w >
< T, B, old w >
<COMMIT T >

Postać logu typu powtarzanie:
< BEGIN T >
< T, A, new w >
< T, B, new w >
<COMMIT T >

Postać logu typu unieważnienie/powtarzanie:
< BEGIN T >
< T, A, old w, new w >
< T, B, old w, new w >
<COMMIT T>

Cały problem polega na tym, w jaki sposób i kiedy zapisuje się powstałe w wyniku transakcji zmiany na dysku. W logu z unieważnieniem nie można skopiować A i B na dysk, zanim nie znajdzie się tam log z zapisem zmian. Najpierw wykonujemy więc operację SEND LOG, potem dopiero następuje zapisanie wartości A i B na dysk, zatwierdzenie transakcji T, zapisanie <COMMIT T> w logu. Kolejny krok to ponowne umieszczenie logu na dysku, aby zapewnić, że zapis <COMMIT T> jest trwały. W przypadku logu z powtarzaniem w zapisie postaci logu są zawarte nowe wartości (new w) dla A i B. Po zakończeniu transakcji WRITE < X, y> następuje jej zatwierdzenie <COMMIT T>. Następnie log jest przesyłany na dysk <SEND LOG>, czyli wszystkie zapisy dotyczące zmian wykonywanych przez transakcję T zostają zapamiętane na dysku. Dopiero wtedy można na dysku zapamiętać nowe wartości A i B. W przypadku logu typu unieważnienie/powtarzanie istotne jest, aby – zanim na dysku zapisze się zmianę wartości elementu X spowodowaną działaniem transakcji T – najpierw na dysk wprowadzić zapis < T, X, old w, new w >.

12. Modele odtworzenia bazy danych

12.1. Logi z unieważnieniem

Jeżeli w wyniku awarii transakcja nie została wykonana w całości, należy wówczas odtworzyć spójny stan bazy danych wykorzystując zawartość logu. Służy do tego podany niżej algorytm. Dla uproszczenia założymy, że mamy do czynienia z całymi logami, bez względu na ich wielkość. Wprowadzone zmiany do bazy są zgodne z danymi zawartymi w logach.

Krok 1. Badamy, czy dla danej transakcji T wykonała się operacja SEND LOG.

Krok 2. Jeśli TAK, to przechodzimy do kroku 8.

Krok 3. Jeśli nie, to badamy, czy dla danej transakcji T <COMMIT T> jest zapisana na dysku.

Krok 4. Jeśli TAK, to przechodzimy do kroku 8.

Krok 5. Jeśli nie, to kolejno od ostatniej komendy sprawdzamy wszystkie operacje i przypisujemy elementom stare wartości < T, y, old w >.

Krok 6. Do logu wprowadzamy zapis < ABORT T >.

Krok 7. Wykonujemy SEND LOG.

Krok 8. STOP.

Jeżeli w trakcie odtwarzania wystąpiła awaria, to powtarzamy algorytm.

Bardzo często wiele transakcji wykonuje się równocześnie. Aby określić, jakie zmiany nastąpiły i w jaki sposób przywrócić spójny stan bazy danych, wprowadza się punkty kontrolne.

Na rysunku 3 przedstawiono przykład realizacji w pewnym przedziale czasowym czterech transakcji: T_1, T_2, T_3, T_4.

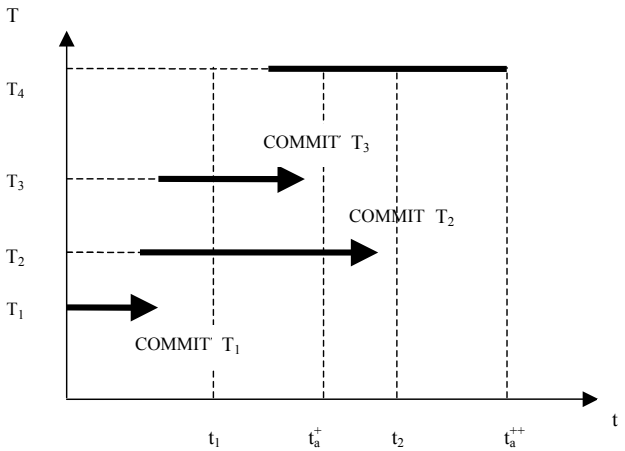
Bezkolizyjne punkty kontrolne oznaczono t_1 i t_2, moment wystąpienia awarii t_a gdzie:

t_1 : < BEGIN CP (T2, T3) >

t_2 : < END CP >

Jeśli $t_1 < t_a < t_2$, czyli $t_a = t_a^+$, to awaria nastąpiła między punktami kontrolnymi,

jeśli $t_a > t_2$, czyli $t_a = t_a^{++}$, to awaria nastąpiła po zakończeniu punktu kontrolnego.



Rys. 3. Przykład transakcji zapisywanych w logach z unieważnieniem

Ograniczymy się tylko do transakcji aktywnych w trakcie wystawiania punktu kontrolnego. Transakcja T_1 nie jest więc brana pod uwagę.

Rozważymy dwa przypadki wystąpienia awarii, które w pełni oddają jej skutki:

1. $ta = t_a^+$

W chwili t_a^+ transakcje T_2, T_4 są aktywne, nie zostały zakończone, dlatego należy unieważnić zmiany wprowadzone przez te transakcje; czyli wprowadzamy zapisy aktualizacyjne <T2, X, old w>, <T4, X, old w> i komendy <ABORT T2>, <ABORT T4>.

Badając operacje od punktu t_a^+ wstecz (w logu z unieważnieniem zaczynamy sprawdzanie wszystkich transakcji od końca logu), natrafiamy na punkt t_1: <BEGIN CP (T_2, T_3)>. Oczywiste jest, że awaria nastąpiła w tym przedziale kontrolnym. Oprócz transakcji T_2, T_4 tylko transakcja T_3 może być niezatwierdzona. Napotykamy jednak na zapis <COMMIT T_3>, a więc transakcja została zatwierdzona i nie bierze się jej pod uwagę. Dlatego nie ma sensu sprawdzania logu wstecz dalej, niż do początku najwcześniejszej z tych transakcji, w tym wypadku T_2.

2. $t_a = t_a^{++}$

Awaria nastąpiła w chwili t_a^{++}. Badając operacje od punktu t_a^{++} wstecz, napotkamy na koniec punktu kontrolnego w chwili t_2 : <END CP>. Wiadomo, że wszystkie niezakończone transakcje zaczęły się po poprzednim zapisie: <BEGIN CP (T_2,T_3)>. Sprawdzając wstecz, należy uaktualnić niezatwierdzoną transakcję T_4 wprowadzając <T_4, X, old w>, następnie usuwamy transakcję T_4 komendą <ABORT T_4>. Wszystkie inne transakcje (T_2,T_3) są zatwierdzone przez <COMMIT T_2 > i <COMMIT T_3>, czyli nic się nie zmienia.

12.2. Logi z powtarzaniem

Jeżeli w wyniku awarii transakcja nie została wykonana w całości, należy wówczas odtworzyć spójny stan bazy danych za pomocą logu. Korzystając z logu z powtarzaniem, skupiamy się na transakcjach zatwierdzonych komendą COMMIT. Transakcje te należy powtórzyć. Jeśli transakcja T nie jest zatwierdzona, oznacza to, że zmiany nie zostały zapisane na dysku i traktujemy T jakby jej w ogóle nie było. Dwie różne transakcje zatwierdzone mogą zmieniać w różnym czasie wartość tego samego elementu bazy danych. Zapisy w logach sprawdzamy od najwcześniejszego do najpóźniejszego. W wypadku transakcji zatwierdzonej pojawia się problem, które zmiany zostały na dysku zapamiętane.

Aby odzyskać dane po awarii, wykonujemy następujące kroki:

Krok 1. Badamy, czy dla danej transakcji T_i wykonała się operacja <COMMIT T_i>.

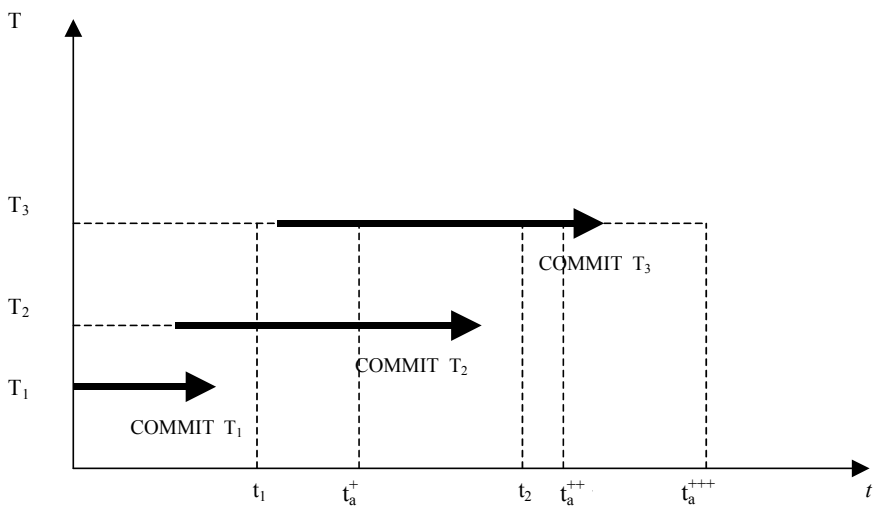
Krok 2. Jeśli NIE, to przechodzimy do kroku 6.

Krok 3. Jeśli TAK, badamy, czy dla danej transakcji T_i wykonano SEND LOG.

Krok 4. Jeśli TAK, to transakcja T_i jest zakończona, sprawdzamy log, aktualizujemy zapisy < T_i, X, new w >.

Krok 5. Jeśli NIE, to (mimo że zapis mógł znaleźć się w logu, ale może nie być na dysku) T_i traktujemy jako niezakończoną i wprowadzamy zapis < ABORT T_i>.

Krok 6. STOP.



Rys. 4. Przykład transakcji zapisywanych w logach z powtarzaniem

Na rysunku 4 przedstawiono przykład realizacji w pewnym przedziale czasowym trzech transakcji: T_1, T_2, T_3,

Punkty kontrolne oznaczono jako t_1 i t_2, moment wystąpienia awarii t_a gdzie:

t_1 : < BEGIN CP (T_2, T_3) >

t_2 : < END CP >

Jeśli $t_1 < t_a < t_2$, czyli $t_a = t_a^+$, to awaria nastąpiła między punktami kontrolnymi oraz przed zatwierdzeniem transakcji T_2, T_3.

Jeśli $t_a > t_2$, to są dwa przypadki wystąpienia awarii:

a) Jeśli $t_a = t_a^{++}$, to awaria nastąpiła po zatwierdzeniu transakcji T_2 i przed zatwierdzeniem transakcji T_3, i po komendzie < END CP >

b) Jeśli $t_a = t_a^{+++}$, to awaria nastąpiła po zatwierdzeniu transakcji T_2, T_3 i po komendzie < END CP >.

Rozważymy trzy przypadki wystąpienia awarii:

1. $t_a = t_a^+$

Awaria nastąpiła między punktami kontrolnymi przed <END CP>. Wówczas wyszukujemy wstecz najbliższy zapis początku bezkonfliktowego punktu kontrolnego <BEGIN CP (T_2, T_3) >. Zbiór wszystkich transakcji to {T_2, T_3}. Ponieważ <BEGIN CP (T_2, T_3) > jest to jedyny punkt kontrolny, sprawdzamy więc cały log. Jedyną zatwierdzoną transakcją jest T_1. Powtarzamy jej działanie <T_1, X, new w> i po odtworzeniu do logu wprowadzamy <ABORT T_2> i <ABORT T_3>

2. $t_a = t_a^{++}$

Awaria wystąpiła pomiędzy komendami <COMMIT T_2> i <COMMIT T_3> i po <END CP>. Szukamy wstecz pierwszego wystąpienia komendy <END CP>. Jest to punkt t_2. Wiadomo więc, że wystarczy powtórzyć tylko te transakcje, które albo zaczęły się po zapisie <BEGIN CP>, albo są ze zbioru {T_2, T_3}. Znajdujemy zapis <COMMIT T_2>, czyli powtarzamy transakcję T_2. Aktualizujemy zapisy tylko dla transakcji T_2, gdzie <T_2, X, new w>, nic nie zmieniając dla T_3. Po odtworzeniu do logu wprowadza się zapis usunięcia transakcji T_3 <ABORT T_3>.

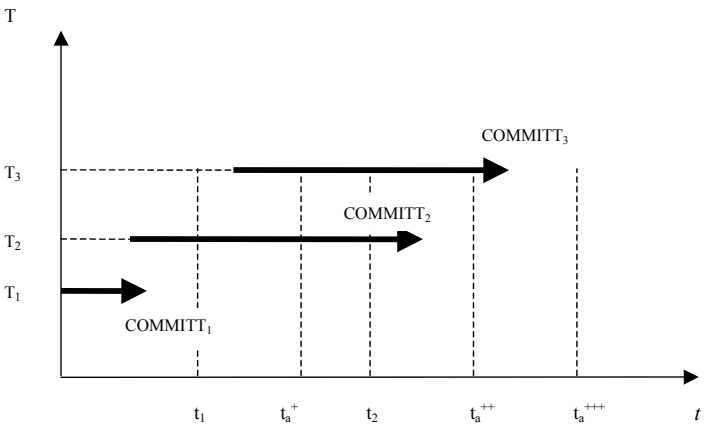
3. $t_a = t_a^{+++}$

Awaria wystąpiła poza przedziałem kontrolnym. Szukamy wstecz pierwszego wystąpienia <END CP>. Jest to punkt t_2. Wiadomo więc, że wystarczy powtórzyć tylko te transakcje, które albo zaczęły się po zapisie <BEGIN CP (T_2, T_3)>, albo są ze zbioru {T_2, T_3}. Znajdujemy zapis <COMMIT T_2> i <COMMIT T_3>. Wiadomo, że trzeba je powtórzyć. Przeszukujemy log wstecz do napotkania rekordów: <BEGIN T_2>, <BEGIN T_3>, aktualizujemy wszystkie zapisy <T_2, X, new w> i <T_3, X, new w> dla transakcji T_2 i T_3.

Aby uniknąć przechowywania zbyt dużej liczby informacji, można w zapisie: <BEGIN CP (T_1 T_2 ... T_m) oprócz nazwy transakcji dodać wskaźnik do tych adresów w logu, w których te transakcje się zaczynają.

12.3. Logi hybrydowe (unieważnienie/powtarzanie)

Algorytm odtwarzania po awarii za pomocą logów hybrydowych polega na powtórzeniu wszystkich transakcji zatwierdzonych poleceniem <COMMIT T_i>, zaczynając od najwcześniejszych. Następnie unieważniamy wszystkie niezakończone transakcje, zaczynając od najpóźniejszych.



Rys. 5. Przykład transakcji zapisywanych w logach hybrydowych

Rozważymy trzy przypadki wystąpienia awarii:

1. $t_a = t_a^+$

Awaria nastąpiła, gdy obydwie transakcje T_2 i T_3 nie zostały zatwierdzone. Unieważniamy transakcje T_2 i T_3, aktualizujemy wszystkie zapisy <T_2, X, old w> i <T_3, X, old w> dla transakcji T_2 i T_3, cofamy transakcje T_2 i T_3 poprzez <ABORT T_2> i <ABORT T_3 >.

2. $t_a = t_a^{++}$

Awaria nastąpiła, gdy transakcja T_2 została zatwierdzona, a transakcja T_3 nie jest kompletna. Powtarzamy transakcję T_2, aktualizujemy wszystkie zapisy <T_2, X, new w> dla transakcji T_2 zapamiętując na dysku <new w>. Aktualizujemy wszystkie zapisy <T3, X, old w> dla transakcji T_3 i cofamy transakcje T_3 <ABORT T_3>.

3. $t_a = t_a^{+++}$

Awaria nastąpiła, gdy obydwie transakcje T_2 i T_3 zostały zatwierdzone. Ponieważ transakcja T_1 była zatwierdzona przed <BEGIN CP>, zakładamy, że jest zapisana na dysk. Powtarzamy transakcje T_2 i T_3, aktualizujemy wszystkie zapisy <T_2, X, new w> i <T_3,X, new w> dla transakcji T_2 i T_3, zapamiętując na dysku <new w>.

13. Współbieżność

W przypadku sekwencyjnego wykonywania poszczególnych transakcji baza danych zawsze będzie w stanie spójnym, jeśli oczywiście jest zachowana semantyczna integralność. W systemach wielodostępnych, a szczególnie w rozproszonych bazach danych, zachowanie spójności przy współbieżnym wykonywaniu transakcji jest dużym problemem. Niemniej jednak ze względu na efektywność systemu ważne jest, aby wiele transakcji było wykonywanych równocześnie.

Celem sterowania współbieżnością jest ochrona spójności bazy danych w sytuacji równoczesnego dostępu do tych samych danych przez wielu użytkowników. Wiąże się to z koniecznością zapewnienia bezkolizyjności przebiegu wielu transakcji. Jedną z metod sterowania współbieżnością jest planowanie.

13.1. Plany

Każda transakcja T_i składa się z ciągu operacji o_1 (T_i), o_2 (T_i)….o_n (T_i). Plan P = {T_1, T_2, … T_i} jest to zbiór transakcji lub zbiór operacji uporządkowanych w czasie. Zakładamy, że kilka transakcji może mieć dostęp do tego samego elementu bazy danych. Mówimy wtedy o współbieżności dostępu do danych.

Przeanalizujemy następujące przypadki:

- Każda transakcja T_i wykonywana jest w całości

Zachowana zostaje zasada izolacji. Po wykonaniu transakcji T_i baza przechodzi z jednego stanu spójnego w drugi stan spójny. Wprowadzimy pojęcie planu sekwencyjnego. Plan sekwencyjny jest to taki plan, w którym wszystkie operacje jednej transakcji poprzedzają wszystkie operacje innej transakcji. Jeśli jakaś operacja o_n (T_i) poprzedza operację o_k (T_j), to wszystkie operacje transakcji T_i muszą poprzedzać wszystkie operacje transakcji T_j.

Przykład 11

Mamy dane dwie transakcje T_i i T_j. Ograniczamy się tylko do operacji na buforach READ < X, y > i WRITE < X, y > bez przepisywania informacji na dysk. Transakcja T_i składa się z następujących operacji:

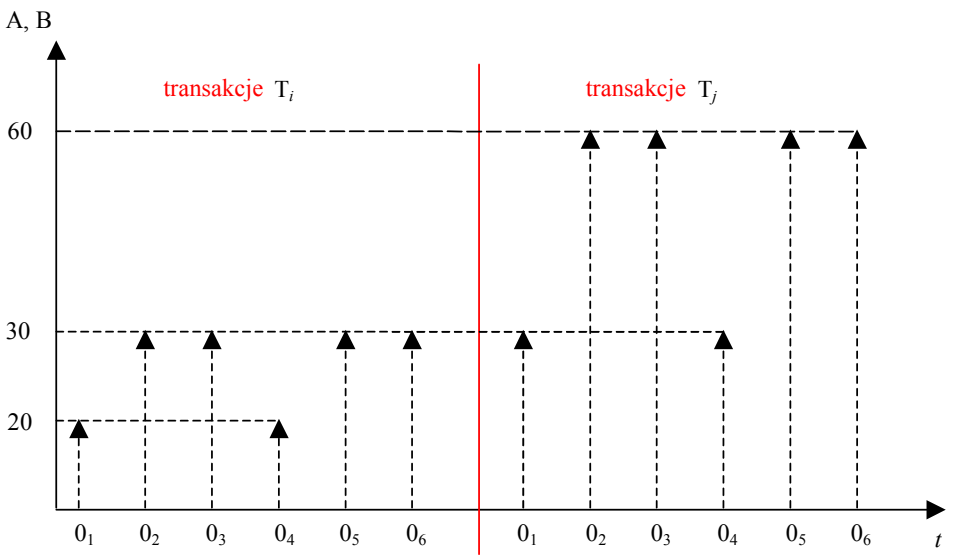
READ < A, x > operacja o_1 (T_i)
x:=x+10 operacja o_2 (T_i)
WRITE < A, x > operacja o_3 (T_i)
READ < B, x > operacja o_4 (T_i)
x:=x+10 operacja o_5 (T_i)
WRITE < B, x > operacja o_6 (T_i)

Transakcja T_j składa się z następujących operacji:

READ < A, y > operacja o_1 (T_j)
y:=y*2 operacja o_2 (T_j).
WRITE < A, y > operacja o_3 (T_j)
READ < B, y > operacja o_4 (T_j)
y:=y*2 operacja o_5 (T_j)
WRITE < B, y > operacja o_6 (T_j)

Wartości początkowe A = B = 20.

Na rysunku 6 przedstawiono plan sekwencyjny P = {T_i, T_j). Najpierw są wykonywane wszystkie operacje transakcji T_i, a potem wszystkie operacje transakcji T_j.



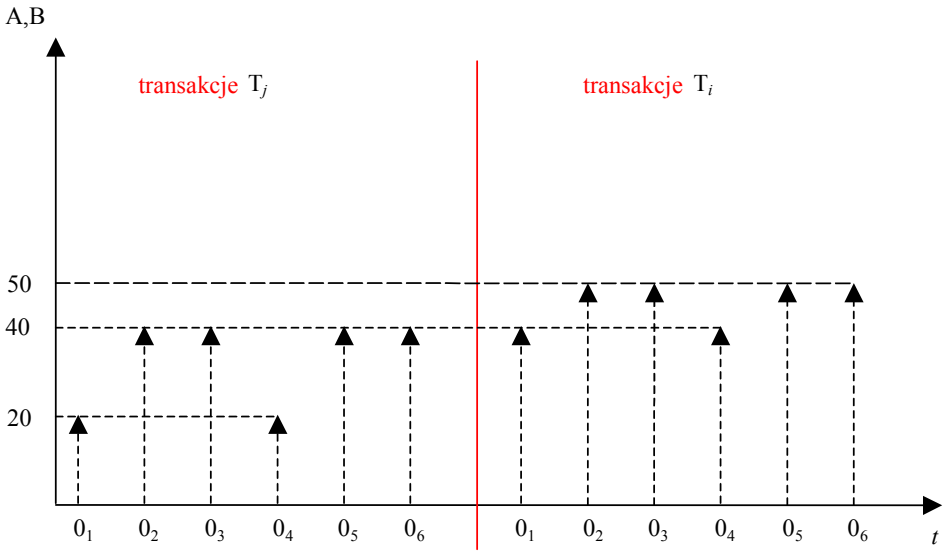
Rys. 6. Plan sekwencyjny P = {T_i, T_j)

Wartości początkowe A = B = 20.
Po wykonaniu transakcji T_i wartość A = 30, wartość B = 30.
Po wykonaniu transakcji T_j wartość A = 60, wartość B = 60.
Stan bazy po wykonaniu operacji jest spójny.

Ciąg operacji planu jest następujący:

$$P\{T_i, T_j) = R_i(A); W_i(A); R_i(B); W_i(B); R_j(A); W_j(A); R_j(B); W_j(B);$$

Na rysunku 7 przedstawiono plan sekwencyjny $P = \{T_j, T_i)$. Najpierw są wykonywane wszystkie operacje transakcji T_j, a potem wszystkie operacje transakcji T_i.



Rys. 7. Plan sekwencyjny $P = \{T_j,T_i)$

Wartości początkowe A = B = 20.
Po wykonaniu transakcji T_j wartość A = 40, wartość B = 40
Po wykonaniu transakcji T_i wartość A = 50, wartość B = 50
Stan bazy po wykonaniu operacji jest spójny A = B
Ciąg operacji planu jest następujący:

$$P\{T_j,T_i) = R_j(A);W_j(A);R_j(B);W_j(B);R_i(A);W_i(A); R_i(B);W_i(B);$$

Końcowe wartości A i B w obu planach są różne. W wyniku wykonania planu $P = \{T_i,T_j)$ najpierw wykonuje się transakcję T_i, a A i B uzyskują wartość 60, natomiast gdy pierwsza jest transakcja T_j w planie $P = \{T_j,T_i)$, wtedy wartość A i B wynosi 50. Końcowa wartość nie jest istotna. Ważne jest, że przy planach sekwencyjnych kolejność wykonywania operacji zależy tylko od kolejności wykonywania całych transakcji, a więc spójność jest zachowana.

- Transakcje nie są wykonywane w całości

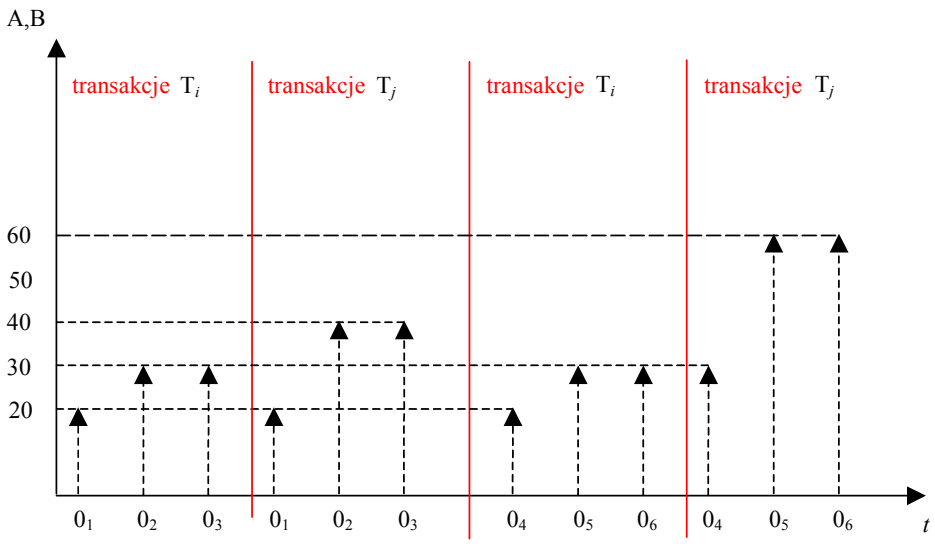
Wprowadzimy pojęcie planu szeregowanego. Plan szeregowany to taki plan, którego wpływ na bazy danych jest taki sam jak planu sekwencyjnego, niezależnie od

stanu początkowego bazy danych. Korzystając z przykładu zawierającego transakcje T_i i T_j, na rysunku 8 przedstawiono plan szeregowany.

Wynik przetwarzania tego planu jest taki sam jak planu P = {T_i,T_j). Można udowodnić, że w przypadku planu szeregowanego dowolny stan spójny bazy danych zostanie przeprowadzony w inny stan spójny.

Ciąg operacji planu:

$$P\{T_i,T_j) = R_i(A);W_i(A);R_j(A);W_j(A);R_i(B);W_i(B);\ R_j(B);W_j(B);$$



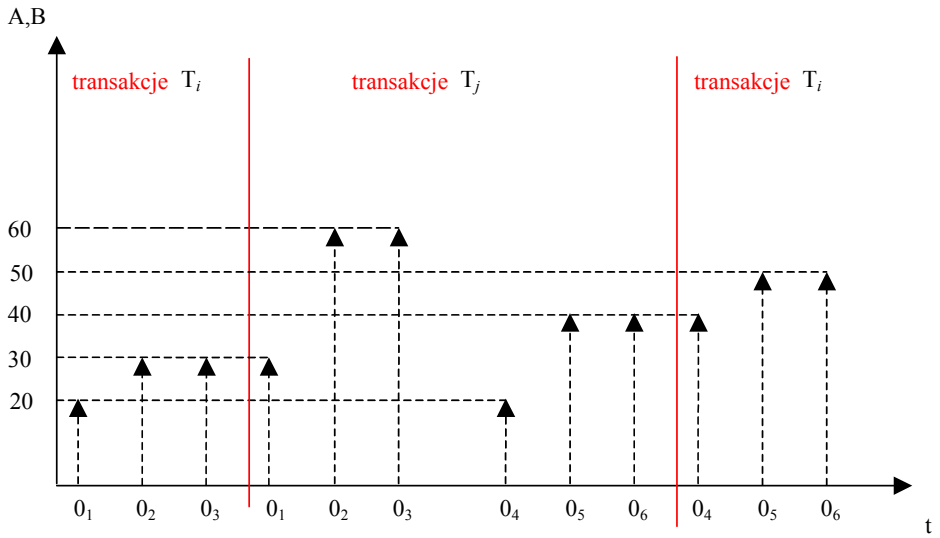
Rys. 8. Plan szeregowany P_{sz} = {T_i,T_j)

• Transakcje nie są wykonywane w całości. Plan nie jest ani sekwencyjny, ani szeregowy. Nazwiemy go planem przypadkowym

Korzystając z przykładu zawierającego transakcje T_i i T_j, na rysunku 9 przedstawiono przypadek, gdy plan jest przypadkowy.

Wartości początkowe A = B = 20, po wykonaniu operacji A = 60, B = 50, czyli zaczynając od stanu spójnego przechodzimy w stan niespójny. Jeśli jedna z operacji transakcji T_i działa jako pierwsza na A, to również powinna działać jako pierwsza na B. Inaczej mamy do czynienia z pojawieniem się stanu niespójnego.

W naszym przypadku po wykonaniu obu transakcji A = 2(A+10), a B = 2B+10.



Rys. 9. Plan przypadkowy $P_p = \{T_i, T_j\}$

Ciąg operacji tego planu przedstawia się następująco:

$$P = R_i(A);W_i(A);R_j(A);W_j(A);R_j(B);W_j(B);\ R_i(B);W_i(B);$$

13.2. Konflikty

Zasada poprawności bazy danych brzmi, że transakcje wykonywane w izolacji przekształcają spójny stan bazy w inny spójny stan. Stanem spójnym nazywamy stan bazy danych, w którym są spełnione wszystkie deklarowane lub ustalone przez projektanta więzy. Jeżeli nie zapewnimy izolacji transakcji, mogą pojawić się następujące problemy:

- Brudny odczyt (ang. dirty reads) występuje wtedy, gdy wiele transakcji próbuje uzyskać dostęp do tej samej tabeli w tym samym czasie. Może się zdarzyć, że jedna transakcja modyfikuje dane, inna czyta te dane przed ich potwierdzeniem, następnie pierwsza zostaje cofnięta i stan bazy powraca do stanu sprzed zmian. Druga transakcja może później próbować modyfikować tabelę w oparciu o dane uzyskane podczas odczytu, które jednak nie są już poprawne.
- Niepowtarzalny odczyt (ang. nonrepetable reads) występuje wtedy, gdy jedna transakcja czyta z tabeli, a potem inna transakcja modyfikuje dane w tabeli. Jeśli pierwsza transakcja próbuje potem jeszcze raz odczytać dane z tej tabeli, odczyt powoduje uzyskanie innych danych.

• Odczyt fantomów (ang. phantom reads) występuje wtedy, gdy jedna transakcja odczytuje dane w oparciu o pewne warunki wyszukiwania, inna transakcja modyfikuje dane w tabeli, a następnie pierwsza transakcja odczytuje dane z tabeli ponownie, opierając się na tych samych warunkach wyszukiwania. W wyniku zmiany danych w tabeli jako rezultat wyszukiwania otrzymuje się zupełnie inne dane.

Aby zapobiec tego typu problemom, w standardzie SQL wprowadzono cztery poziomy izolacji. Poziom izolacji to stopień, w jakim dana transakcja może mieć dostęp do danych, modyfikowanych przez inne transakcje przed ich zatwierdzeniem. Inaczej mówiąc, jest to wymuszanie szeregowalności transakcji. Wyróżniamy poziomy:

READ UNCOMMITED – odczyt niezatwierdzony. Jest to najmniej restrykcyjny z poziomów izolacji. Dopuszcza czytanie przez transakcje danych jeszcze niezatwierdzonych. Jest używany do transakcji zawierających ogólne informacje, takie jak dane statystyczne, które nie powinny być modyfikowane. Poziom ten dopuszcza brudny i niepowtarzalny odczyt oraz odczyt fantomów.

READ COMMITED – odczyt zatwierdzony. Poziom ten zabrania odczytu danych niezatwierdzonych, umożliwia jednak zapisywanie danych w transakcjach niezatwierdzonych. Dopuszcza występowanie niepowtarzalnych odczytów i odczytu fantomów, zabrania występowania brudnych odczytów.

REPEATABLE READ – odczyt powtarzalny. Poziom ten zabrania zapisywania w transakcjach niezatwierdzonych. Jeśli więc transakcja niezatwierdzona przeczytała daną, to dana ta może być tylko czytana przez inną transakcję. Jeśli transakcja niezatwierdzona zapisała jakąś daną, to nie można jej ani odczytać, ani zapisać dopóki ta transakcja nie zostanie zatwierdzona. Poziom ten dopuszcza występowanie odczytu fantomów, jednak blokuje występowanie brudnych odczytów i niepowtarzalnych odczytów.

SERIALIZABLE – szeregowalność, czyli jedyny bezpieczny poziom izolacji. Poziom ten nie dopuszcza występowania brudnych i niepowtarzalnych odczytów oraz odczytu fantomów.

Zapewnienie zachowania wysokiego poziomu poprawności danych przy jednoczesnym realizowaniu transakcji nazywamy szeregowalnością, a nawet szeregowalnością ze względu na konflikt. Sytuacje konfliktowe mogą wystąpić wtedy, gdy dwie transakcje T_i i T_j, gdzie $i \neq j$, są zainteresowane dostępem do tego samego obiektu lub kolejność występowania operacji w transakcji zostaje zmieniona. Mamy dane dwie transakcje T_i i T_j, gdzie $i \neq j$. Rozważymy następujące przypadki:

• Odczyt ($R_i(X)$) – Odczyt ($R_j(Y)$) dla $X = Y$ i $X \neq Y$

Para ta nie stanowi konfliktu, ponieważ żadna z operacji odczytu $R_i(X)$) i ($R_j(Y)$) nie zmienia ani X, ani Y. Każda z tych operacji może być wykonywana w dowolnej kolejności.

• Odczyt ($R_i(X)$) – Zapis ($W_j(Y)$) dla $X \neq Y$

Para ta nie stanowi konfliktu, ponieważ X nie jest zmieniane i transakcja T_j może zapisać wartość Y przed i po transakcji T_i.

• Zapis ($W_i(X)$) – Odczyt ($R_j(Y)$) dla $X \neq Y$

Para ta nie stanowi konfliktu, ponieważ Y nie jest zmieniane i transakcja T_i może zapisać wartość X przed i po transakcji T_j.

• Zapis ($W_i(X)$) – Zapis ($W_j(Y)$) dla $X \neq Y$

Para ta nie stanowi konfliktu.

• Zapis ($W_i(X)$) – Zapis ($W_j(Y)$) dla $X=Y$

Dwa zapisy tego samego elementu przez dwie różne transakcje T_i i T_j są konfliktem. Zmiana kolejności wykonywania operacji powoduje, że wartości wyliczane przez T_i i T_j mogą być różne.

• Odczyt ($R_i(X)$) – Zapis ($W_i(Y)$)

Odczyt i zapis tej samej transakcji stanowi konflikt. Kolejność działań danej transakcji jest ustalona i nie może być zmieniona.

• Odczyt ($R_i(X)$) – Zapis ($W_j(X)$) i Zapis ($W_i(X)$) – Odczyt ($R_j(X)$)

Odczyt i zapis oraz zapis i odczyt tego samego elementu bazy danych wykonywany przez różne transakcje stanowi konflikt. Zmiana kolejności operacji $R_i(X)$ i $W_j(X)$) ma wpływ na wartość X.

Z przytoczonych przypadków wynika, że nie można zmieniać kolejności wykonywania operacji, jeśli dotyczą one tego samego elementu bazy danych lub co najmniej jedną czynnością jest operacja zapisu WRITE.

W przykładzie 12 pokazano przekształcenie planu szeregowanego (z rys. 8) na plan sekwencyjny (z rys. 6) bez konfliktów.

Przykład 12

$R_i(A);W_i(A); R_j(A);\underline{W_j(A); R_i(B)}; W_i(B); R_j(B); W_j(B);$ zapis–odczyt $x \neq y$
$R_i(A);W_i(A); \underline{R_j(A); R_i(B)}; W_j(A); W_i(B); R_j(B); W_j(B);$ odczyt–odczyt
$R_i(A);W_i(A); R_i(B); R_j(A); \underline{W_i(A); W_i(B)}; R_j(B); W_j(B);$ zapis–zapis $x \neq y$
$R_i(A);W_i(A); R_i(B); \underline{R_j(A); W_i(B)}; W_j(A); R_j(B); W_j(B);$ zapis–odczyt $x \neq y$
$R_i(A);W_i(A); R_i(B); W_i(B); R_j(A); W_j(A); R_j(B); W_j(B);$

Rozróżnia się trzy metody synchronizacji współbieżnego dostępu do danych: blokady, optymistyczne sterowanie dostępem oraz znaczniki czasowe.

13.3. Blokady

Mechanizm blokad polega na tym, że każda transakcja przed uzyskaniem dostępu do danego obiektu bazy danych musi założyć odpowiednią blokadę na tym obiekcie. Zapewnia to użytkownikowi wyłączność dostępu do modyfikowanych przez siebie danych [6, 7, 8]. Blokada uniemożliwia dostęp do obiektu innym transakcjom i jest usuwana w momencie zakończenia transakcji. Żadne dwie transakcje nie mogą zablokować tego samego obiektu, jeśli blokada nie została wcześniej

zwolniona. Rozróżnia się: blokadę do odczytu $STOP_i(X)$ oraz blokadę do zapisu $GO_i(X)$.

Mechanizm blokad w pewnym stopniu wymusza szeregowalność transakcji.

Ogólnie mechanizm blokad jest bardzo złożony, gdyż blokady mogą być zakładane na pojedyncze krotki, wartości atrybutów, tabele i całe bazy danych. Nazywa się to ziarnistością blokowania.

Wyróżnia się:

- grube ziarna, to znaczy blokady zakłada się na całą bazę danych, poszczególne krotki; efektem tego jest mały stopień współbieżności, a tym samym mała wydajność,
- miałkie ziarna, gdzie blokady zakłada się na przykład na elementy krotki. Zapewnia to większą współbieżność, lepszą wydajność, ale wiąże się ze znacznym nakładem czasu na zakładanie blokad oraz zapotrzebowaniem na dodatkową pamięć na blokady.

Ziarnistość blokowania odbywa się pod kontrolą tzw. protokołu blokowania zapobiegającego. Im drobniejsza ziarnistość blokowania, tym większy jest poziom współbieżności w dostępie do danych. Często stosuje się tzw. eskalacje blokad. Wraz ze wzrostem aktywności systemu serwery zaczynają blokować większe sekcje informacji, aby ograniczyć zużycie pamięci. Eskalacja powoduje spadek współbieżności w dostępie do danych.

Plan z rysunku 6 można przedstawić następująco:

$P_{blok}\{T_i,T_j) = STOP_i(A);R_i(A);W_i(A);GO_i(A);STOP_i(B);R_i(B);W_i(B);GO_i(B);$
$STOP_j(A);R_j(A);W_j(A);\ GO_j(A);STOP_j(B);\ R_j(B);W_j(B);\ GO_j(B);$

Plan z rysunku 7 można przedstawić następująco:

$P_{blok}\{T_j,T_i) = STOP_j(A);R_j(A);W_j(A);GO_j(A);STOP_j(B);R_j(B);W_j(B);GO_j(B);$
$STOP_i(A);R_i(A);W_i(A);\ GO_i(A);STOP_i(B);\ R_i(B);W_j(B);\ GO_i(B);$

Plan z rysunku 8 można przedstawić następująco:

$P_{blok} = STOP_i(A);R_i(A);W_i(A);GO_i(A);STOP_j(A);R_j(A);W_j(A);GO_j(A);$
$STOP_i(B);R_i(B);W_i(A);\ GO_i(B);STOP_j(B);\ R_j(B);W_j(B);\ GO_j(B);$

Samo blokowanie nie zapewnia szeregowalności. Znany jest algorytm blokowania dwufazowego 2PL, który wymaga w ramach transakcji najpierw założenia wszystkich blokad, a dopiero potem zwolnienia pierwszej blokady. Aby uniknąć niepotrzebnego blokowania, system zazwyczaj stosuje kilka trybów blokowania i różne zasady przyznawania blokad dla poszczególnych trybów. Znane są blokowania aktualizujące, przyrostowe, elementów hierarchii ziarnistości i inne.

W trakcie dostępu współbieżnego istnieje niebezpieczeństwo zakleszczenia, czyli wza-
jemnej blokady wykonywanych równocześnie transakcji. Zakleszczenie to sytuacja, w któ-

rej dwie lub więcej transakcji znajduje się w stanie oczekiwania na uwolnienie z blokady.

Przykład 13

Mamy dwie transakcje:

T_i: $STOP_i(A);R_i(A);A:=A+10;W_i(A);STOP_i(B);$
$GO_i(A);R_i(B);B:=B+10\ W_i(B);\ GO_i(B);$

T_j: $STOP_j(B);R_j(B);B=B*2;\ W_j(B);\ STOP_j(A);$
$GO_j(B);R_j(A);A=A*2;W_j(A);\ GO_j(A);$

Jeżeli operacje w transakcjach będą się przeplatać następująco:

$STOP_i(A);\ R_i(A);\ STOP_j(B);\ R_j(B);\ A:=A+10;\ B=B*2,\ W_i(A);\ W_j(B);\ STOP_i(B)$ **zabronione**; $STOP_j(A)$ **zabronione**,

to żadna z transakcji nie może się wykonać i czas oczekiwania będzie nieograniczony. Element B został wcześniej zablokowany przez transakcję T_j i nie została zwolniona blokada, w związku z czym nie jest możliwe założenie blokady $STOP_i(B)$ na tym elemencie przez transakcję T_i. To samo dotyczy elementu A.

Rozwiązanie tego typu problemów to:

- spełnienie warunku, aby każda transakcja zakładała wszystkie potrzebne jej blokady na samym początku w postaci jednej akcji lub nie zakładała wcale (znaczne ograniczenie współbieżności),
- wykrywanie wzajemnej blokady w chwili, gdy ona wystąpi i wycofanie działania jednej akcji (rola ofiary), co pozwala drugiej na kontynuowanie działania,
- ustalenie limitu czasu oczekiwania na założenie blokady, a po upływie tego czasu automatyczne wycofanie oczekującej transakcji.

Standard SQL pozwala na kontrolowanie blokady ustawionej przez transakcje. Odbywa się to z wykorzystaniem opisanych poprzednio poziomów izolacji w SQL. I tak:

READ UNCOMMITED – brak blokady dla operacji odczytu, dający możliwość równoczesnego korzystania z danych różnym transakcjom, czyli dana transakcja będzie mogła odczytywać dane, które właśnie są modyfikowane przez inną transakcję, co może spowodować niespójność danych.

READ COMMITED – zwalnianie blokad zakładanych na rekordy natychmiast po ich odczycie.

REPEATABLE READ – zwolnienie blokad rekordów pobranych do odczytu, ale nigdy nie odczytanych. Umożliwia to równoczesne wprowadzanie danych, realizowane przez inną transakcję.

SERIALIZABLE – utrzymanie blokad założonych przy odczycie do zakończenia transakcji. Niemożliwe jest wtedy dokonywanie jakichkolwiek zmian na blokowanych danych, łącznie z dodawaniem nowych rekordów.

W przypadku obiektowych baz danych należy wziąć pod uwagę, że klasa dziedziczy atrybuty i metody od swoich nadklas. W związku z tym, gdy jedna transakcja przetwarza

instancje klasy, inna nie powinna modyfikować definicji tej klasy, ani żadnej nadklasy tej klasy. Dla zapytania dotyczącego danej klasy blokada jest więc zakładana nie tylko na danej klasie, ale dla wszystkich następników w hierarchii danej klasy.

13.4. Inne metody sterowania współbieżnym dostępem

Oprócz blokad jednym ze sposobów sterowania współbieżnym dostępem jest podejście optymistyczne. Zakłada się, że konflikty między transakcjami występują bardzo rzadko, dzięki czemu transakcje mają nieograniczony dostęp do danych. Dopiero przed zatwierdzeniem transakcji sprawdza się, czy nie wystąpił konflikt. Jeśli tak, to jedną transakcję anuluje się i wykonuje od początku. Istnieje też metoda znaczników czasowych (ang. timestamps). Szeregowalność transakcji uzyskuje się przez nadanie każdej transakcji unikalnych znaczników, które ustawiają je w pewnym ciągu czasowym. Metodę tę stosuje się przede wszystkim w systemach rozproszonych, gdzie blokowanie nie zdaje egzaminu. Znaczniki czasowe przypisuje się w kolejności rosnącej w momencie, gdy zaczyna się transakcja. Istnieją różne sposoby określania znaczników. Można korzystać z zegara systemowego, co nie zawsze jest bezpieczne, lub z licznika, który zostaje zwiększony o jeden w chwili rozpoczęcia transakcji.

Kolejnym typem optymistycznego sterowania współbieżnością jest walidacja. W procesie walidacji utrzymuje się zapis czynności aktywnych transakcji, a nie czas zapisu i odczytu wszystkich elementów bazy danych. Zanim zostaje zapisana wartość elementu bazy danych, zbiory elementów przeczytanych przez transakcje i tych, które mają być przez nią zapisane porównuje się ze zbiorami zapisów z innych transakcji aktywnych. Jeśli występują jakieś nieprawidłowości, transakcja zostaje cofnięta.

Wszystkie metody szeregowalności transakcji, czyli: blokowanie, znaczniki czasowe i walidacja mają swoje zalety i wady. Dotyczy to przede wszystkim przestrzeni pamięci potrzebnej dla tych operacji oraz możliwości zakończenia transakcji bez opóźnień. W przypadku blokad potrzebna przestrzeń pamięci jest proporcjonalna do liczby blokowanych elementów bazy danych. W pozostałych dwóch przypadkach pamięć jest potrzebna do zapamiętania czasów zapisu i odczytu, związanych z każdym elementem bazy danych, czy zapisu aktywnych transakcji; bez względu na to, czy dostęp do nich następuje w danej chwili. Znaczniki czasowe i walidacja wymagają więcej przestrzeni pamięci, ponieważ muszą utrzymywać dane dotyczące transakcji zakończonych, które nie są potrzebne przy blokowaniu. Wydajność tych trzech metod zależy również od tego, czy transakcja wymaga dostępu, który jest potrzebny innej transakcji współbieżnie. Gdy ta współbieżność jest mała, znaczniki czasu i walidacja nie powodują wielu cofnięć i mogą być znacznie efektywniejsze od blokad. Gdy wy-

maganych jest wiele cofnięć, wtedy wygodniej stosować blokadę.

13.5. Zarządzanie transakcjami w systemach rozproszonych baz danych

W rozproszonych bazach danych dane są podzielone poziomo – czyli krotki tej samej relacji znajdują się w różnych miejscach lub pionowo – wówczas na schemacie relacji wykonane są operacje projekcji, dzięki czemu uzyskamy szereg schematów niższego stopnia. Można też powielić dane, czyli replikować, co zapewnia identyczne kopie relacji w różnych miejscach. Dzięki temu uzyskuje się wysoki stopień niezawodności systemów rozproszonych.

Ze względu na replikację danych w kilku węzłach problem zarządzania transakcjami i ochrona integralności danych są znacznie trudniejsze. Należy zadbać o dodatkowe zachowanie zgodności danych w całym rozproszonym systemie. Transakcja rozproszona składa się z wielu składowych, działających w różnych miejscach i obejmuje wiele lokalnych baz danych. W związku z tym jej zatwierdzenie może wystąpić tylko w przypadku pomyślnego zakończenia wszystkich transakcji lokalnych, będących całością transakcji rozproszonej. Realizacja przetwarzania rozproszonych transakcji wymaga implementacji protokołów zarządzających kontrolą dostępu i niezawodnością transakcji rozproszonego systemu.

Typowy algorytm zarządzania kontrolą dostępu to dwufazowy protokół zatwierdzania 2PC (ang. two-phase commit). W pierwszej fazie lokalne bazy informują wyróżniony węzeł, zwany koordynatorem transakcji o tym, czy ich lokalna część transakcji może zostać zatwierdzona, czy musi być wycofana. W drugiej, jeżeli wszystkie lokalne bazy danych biorące udział w transakcji mogą ją zatwierdzić, koordynator nakazuje zatwierdzenie (notuje to w swoim dzienniku transakcji) i w każdej bazie lokalnej zatwierdzone są transakcje. W przeciwnym wypadku, jeśli choć jedna z lokalnych transakcji zostaje wycofana, należy wycofać wszystkie.

Istnieje również protokół 3PC. Uwzględnia on sytuacje, gdy koordynator ulegnie awarii. Umożliwia wówczas wybranie nowego koordynatora. Na ogół komercyjne systemy zarządzania bazą danych nie realizują tego protokołu.

Replikacja danych zwiększa poziom niezawodności, jednak zawsze pozostaje ryzyko, że w systemie istnieje kopia danych, która nie mogła być uaktualniona ze względu na jej chwilowe uszkodzenie lub niedostępność, spowodowaną uszkodzeniem łącza komunikacyjnego. Utrzymywanie zgodności replik wymaga implementacji protokołów kontroli replik.

Najprostszym protokołem kontroli replik jest protokół ROWA (ang. Read One Write All). Zakłada on, że jedna operacja logicznego odczytania zreplikowanych danych jest zrealizowana przez pojedynczą operację fizycznego odczytania z dowolnej

repliki, zaś operacja logicznego zapisu wymaga wykonania operacji fizycznego zapisu na wszystkich kopiach.

13.6. Zarządzanie transakcjami w systemie Oracle

System Oracle w przeciwieństwie do MYSQL inicjalizuje automatycznie transakcje. Transakcja rozpoczyna się w chwili, gdy nawiązane zostanie połączenie z bazą danych. Transakcja kończy się w momencie przerwania połączenia z bazą danych lub gdy wystąpi polecenie zatwierdzenia transakcji – COMMIT, lub wycofania transakcji – ROLLBACK. Oracle pozwala również dzięki komendzie SAVEPOINT, umieszczonej wewnątrz bloku transakcji, na częściowe cofnięcie transakcji zamiast całej. Oczywiście występuje to przy połączeniu SAVEPOINT z ROLLBACK.

Aby zachować integralność danych w bazie w Oracle, istnieje mechanizm nazwany logicznym znacznikiem czasowym SCN (ang. System Change Number), służący do śledzenia kolejności, w jakiej zachodzą transakcje w bazie. Informacje te są przydatne, gdy zachodzi potrzeba poprawnego i zgodnego z kolejnością wystąpień odtworzenia transakcji po awarii.

Bardzo wygodną strukturą są tak zwane segmenty powrotu (ang. rollback segments), które składują informacje umożliwiające odtworzenie bazy danych do stanu sprzed wystąpienia transakcji. Segmenty różnią się od logów tym, że nie zapamiętują zmian spowodowanych przebiegiem transakcji, dają tylko możliwość cofania transakcji i współbieżnego dostępu. Segmenty udostępniają spójny obraz bazy danych w określonym momencie w przeszłości. Segmenty (w Oracle wersja 9) pozwalają również na cofnięcie się do momentu w przeszłości przed zapytanie, które spowodowało utratę spójności bazy danych. Zapewnia to mechanizm zapytań wstecz (ang. Flashback Query). Mechanizm ten został mocno rozbudowany w Oracle 10.

Oracle wykorzystuje takie poziomy izolacji jak: READ COMMITED i SERIALIZABLE. Trzeci dostępny to READ ONLY, który jednocześnie zabrania operacji zapisu i udostępnia dokładny obraz danych w momencie zapytania.

Do rozwiązania współbieżnego dostępu do danych w bazie Oracle służy system MVRC (ang. Multiversion Read Consistency). System ten gwarantuje spójny obraz danych. MVRC ignoruje zmiany, wprowadzane przez niezatwierdzone w chwili zapytania transakcje.

Oracle jest jedną z najszybszych baz danych. Na szybkość jej działania wpływa wiele czynników, przede wszystkim system zarządzania transakcjami, umożliwiający bezkolizyjny dostęp współużytkowników w tym samym czasie. Uzyskano to

w wyniku zmniejszenia do niezbędnego minimum liczby operacji we/wy oraz niezakładania blokad na operacje odczytu.

Podsumowanie

Integralność polega na zapewnieniu, że baza danych pozostaje dokładnym odbiciem reprezentowanej rzeczywistości. Wyróżniamy schemat bazy danych i stan. Schemat jest stały, niezmienny. Zmiana schematu to nowa baza danych. Stan bazy jest określony w danym momencie czasowym. Pod wpływem czynników zewnętrznych i wewnętrznych, takich jak na przykład transakcje, baza danych przechodzi przez ciąg zmian stanów. W zbiorze możliwych stanów tylko niektóre z nich są poprawne. Stany bazy danych, w których są spełnione wszystkie deklarowane lub zamierzone więzy są nazwane stanami spójnymi. Istotne jest, aby transakcja przekształcała spójny stan bazy danych w inny, również spójny. Transakcje, działając w izolacji, zapewniają spójność bazy. Dąży się do tego, aby spójność była utrzymana również w trakcie współbieżnego dostępu. Uzyskuje się to dzięki opracowaniu planów sekwencyjnych, szeregowanych i szeregowanych ze względu na konflikt. W celu zachowania spójności bazy danych stosuje się blokady i inne formy zabezpieczania danych przy współbieżnym dostępie.

Bibliografia

[1] Allen S., *Modelowanie danych*, Wydawnictwo Helion, Gliwice 2006.

[2] Berenstein P.A., Goodman N., Hadzilacos V., *Recovery algorithms for Database Systems*, Proc. 1983 IFIP Congress, North Holland, Amsterdam, s. 799–807.

[3] Berenstein P.A., Goodman N., Hadzilacos V., *Concurrency Control and Recovery in Database Systems*, Addison-Wesley 1987.

[4] Beynon-Davies P., *Systemy baz danych*, WNT, Warszawa 1998.

[5] Chałon M., *Ochrona i bezpieczeństwo systemów baz danych*, [w:] *Inżynieria komputerowa*, praca zbiorowa pod redakcją W. Zamojskiego, WKŁ, Warszawa 2005.

[6] Elmasri R., Navathe S., *Wprowadzenie do systemów baz danych*, Wydawnictwo Helion, Gliwice 2003.

[7] Garcia-Molina H., Ullman J., Widom J., *Implementacja systemów baz danych*, WNT, Warszawa 2003.

[8] Garcia-Molina H., Ullman J., Widom J., *Systemy baz danych. Pełny wykład*, WNT, Warszawa 2006.

[9] Graves M., *Projektowanie baz XML. Vademecum profesjonalisty*, Wydawnictwo Helion, Gliwice 2002.

[10] Gray J.N., Reuter A., *Transaction Processing: Concepts and Techniques*, The Morgan-Kaufman, Series in Data Management Systems, San Francisco 1993.

[11] Kumar V., Hsu M., *Recovery Mechanisms in Database Systems*, Prentice-Hall, Englewood Cliffs NJ, 1998.

[12] Peterson J., *Wprowadzenie do baz danych*, Wydawnictwo Helion, Gliwice 2003.

[13] Thomasian A., *Concurrency Control: Methods, Performance and Analysis*, ACM Computing Surveys 1998, s. 70–119.

[14] Thuraisingham B. Ko H., *Concurrency Control in Trusted Database Management Systems*: *a survey*, ACM Press NY USA 1993.

[15] Whitehorn M., Marklyn B., *Relacyjne bazy danych*, Wydawnictwo Helion, Gliwice 2003.

CZĘŚĆ IV

Zabezpieczenie przed uszkodzeniami i utratą danych

Dzienniki transakcji chronią tylko przed skutkami awarii systemu, obejmującymi utratę danych w pamięci operacyjnej, natomiast archiwizację stosuje się na wypadek utraty zawartości dysku. Archiwa są to kopie baz danych przechowywane w bezpiecznych miejscach.

H. Garcia-Molina, J. Ullman, J. Widom

14. Zabezpieczanie danych

Priorytetowym zadaniem jest dążenie, aby system charakteryzował się:

- wysoką niezawodnością, czyli bezawaryjnym działaniem w ciągu ustalonego czasu,
- dostępnością, czyli prawdopodobieństwem, że w ustalonej chwili system będzie działał i będzie zdolny do realizacji określonych usług.

O tym, jak ważna jest niezawodność i poprawność danych przekonujemy się wtedy, gdy tracimy dane, bądź gdy odkrywamy w nich błędy. Konsekwencją takiego stanu rzeczy jest zmniejszenie zysków firmy, utrata dobrego wizerunku i zaufania klientów. Jednym z podstawowych zadań projektanta systemu i administratora jest zabezpieczenie przed częściową lub całkowitą utratą danych oraz opracowanie skutecznej strategii tworzenia kopii zapasowych. Zabezpieczenia te mają charakter zarówno sprzętowy, jak i programowy i powinny umożliwić rekonstrukcję bazy danych. Bezpieczeństwo bazy danych zależy przede wszystkim od poprawności i częstości sporządzania kopii. Kopia to stan bazy w chwili archiwizacji. Innym sposobem odtworzenia stanu sprzed awarii systemu jest wykorzystanie dziennika transakcji. Istotną rolę odgrywa zwiększenie niezawodności systemu. Niezawodność systemu komputerowego zwiększa się przez duplikowanie komputerów, podzespołów lub całych systemów informatycznych. Im ważniejsza informacja, im większa musi być dostępność do systemu, tym bardziej rosną wymagania wobec nadmiarowości sprzętu i oprogramowania. System bazodanowy powinien być tak zorganizowany, aby zapewniać stałą gotowość operacyjną i minimalizować przestoje serwera związane z awariami sprzętu.

15. Archiwizacja i odtwarzanie bazy po awarii

Archiwizacja i odtwarzanie danych to zbiór strategii i operacji, mających na celu zabezpieczenie bazy danych przed skutkami awarii i umożliwiających rekonstrukcję tej bazy. Archiwizacja i sporządzenie kopii (ang. backup) zabezpieczają nie tylko przed utratą danych w wyniku awarii nośnika, lecz również przed nieautoryzowanymi zmianami, dokonanymi przez włamywacza. Określenia archiwizacja i backup oznaczają procesy składowania danych, zachodzą one jednak z użyciem innych mechanizmów i nośników i są realizowane w innym celu. Backup służy do jak najszybszego odtworzenia środowiska w sytuacjach awaryjnych. Podstawowym kryterium realizacji backupu jest czas odtwarzania. Archiwizacja natomiast są to pewne procedury postępowania, określające zasady gromadzenia, przechowywania i odtwarzania danych, składowanych do późniejszego wykorzystania.

Archiwizacja charakteryzuje się:

- znacznie mniejszą częstotliwością wykonywania kopii,
- zapisem na nośnikach wyższej jakości,
- wykorzystywaniem nośników jednorazowego zapisu.

Backup charakteryzuje się:

- dużą częstotliwością wykonywania kopii okresowych,
- relatywnie dużą liczbą przechowywanych kopii,
- zapisem na nośnikach wielokrotnego zapisu/odczytu,
- złożonym cyklem sporządzania i przechowywania kopii.

Można wyróżnić:
backup pełny (ang. full backup),
backup różnicowy (ang. differential backup),
backup przyrostowy (ang. cummulative incremental backup).

Zawsze na początku wykonuje się pełny backup, a następnie – w zależności od potrzeb i przyjętej strategii bezpieczeństwa – backup różnicowy lub backup przyrostowy. Backup pełny to rozwiązanie konieczne wtedy, gdy chcemy zabezpieczyć dane. Zaletą tej metody jest łatwość wyszukiwania dowolnych danych i szybkość odtworzenia danych w czasie awarii systemu. Sporządzanie backupu pełnego to proces pracochłonny, a czasami nawet zbędny. Wady to przede wszystkim nieefektywność wykorzystania nośników oraz długi czas wykonywania operacji. Okres przechowywania

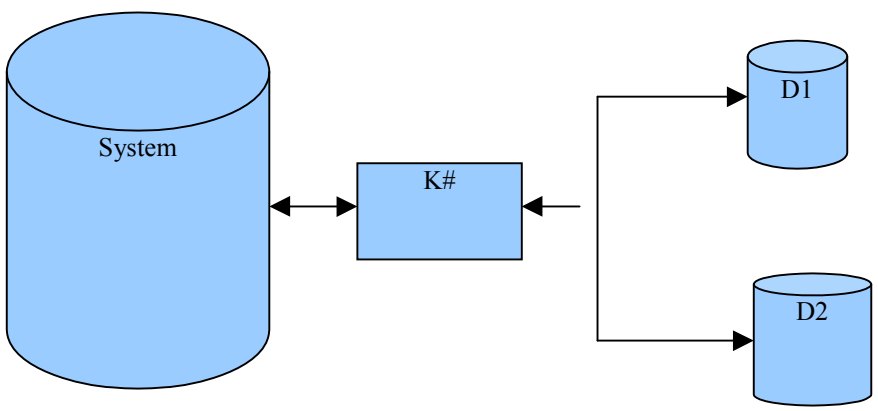
sporządzonych kopii zapasowych i archiwalnych nazywamy czasem retencji. W trakcie jednego okresu retencyjnego stosuje się kilka backupów przyrostowych lub różnicowych. W trakcie backupu różnicowego lub przyrostowego zapisuje się tylko te pliki, które zostały zmodyfikowane od czasu ostatniego backupu pełnego, nie jest przy tym istotne, jakie pliki zostały zmienione. Wybór tych plików jest wykonywany automatycznie przez oprogramowanie archiwizacyjne. Każdy plik czy katalog w systemie ma przydzielone odpowiednie, charakteryzujące go atrybuty, takie jak informacje, że plik jest archiwalny i jaka jest data ostatniej modyfikacji pliku, która została zapisana razem z plikiem. Jeśli od poprzedniej kopii wiele danych uległo zmianie, to operacje porównania trwają niejednokrotnie dłużej niż pełny backup. Okazuje się, że na ogół w krótkim czasie jedynie nieznaczna część plików znajdujących się w komputerze jest modyfikowana przez użytkownika lub system operacyjny. Dlatego stosując metodę przyrostową podczas backupu, zapisuje się tylko te dane, które były w jakikolwiek sposób zmieniane od czasu pełnej bądź przyrostowej archiwizacji. Dodatkową zaletą metody przyrostowej jest znacznie efektywniejsze wykorzystanie nośników informacji, ponieważ zapisujemy tylko zmiany. Czas tego typu backupu jest bardzo krótki. Wadą jest trudność odnalezienia właściwych danych. Aby odnaleźć określony zbiór danych, należy dysponować wszystkimi nośnikami, zawierającymi wszystkie dotychczas zrobione kopie oraz dodatkowo nośnik z ostatnią pełną kopią. Powoduje to znaczne wydłużenie czasu ewentualnego odtworzenia danych.

Sprzętowe zabezpieczenie danych to stworzenie zapasowej bazy. Istnienie zapasowej bazy danych jest konieczne w razie awarii bazy aktywnej. Wówczas zapasowa baza jest przełączana w stan aktywności. W systemie z rezerwową bazą pracują dwa komputery o podobnej konfiguracji. Serwer zapasowy zawiera kopie bazy danych serwera podstawowego. Uaktualnienie bazy rezerwowej następuje co pewien czas na podstawie zarchiwizowanych plików dziennika transakcji, przesyłanych z serwera podstawowego.

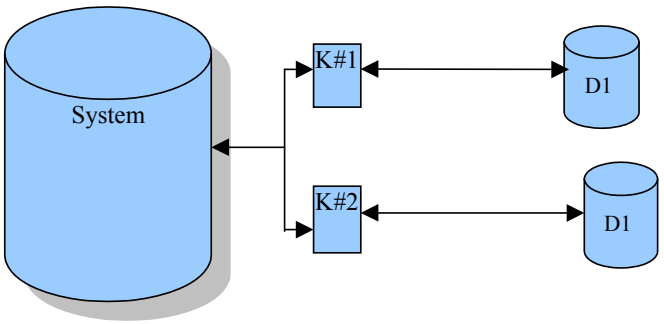
16. Kopie lustrzane

16.1. Kopie lustrzane proste

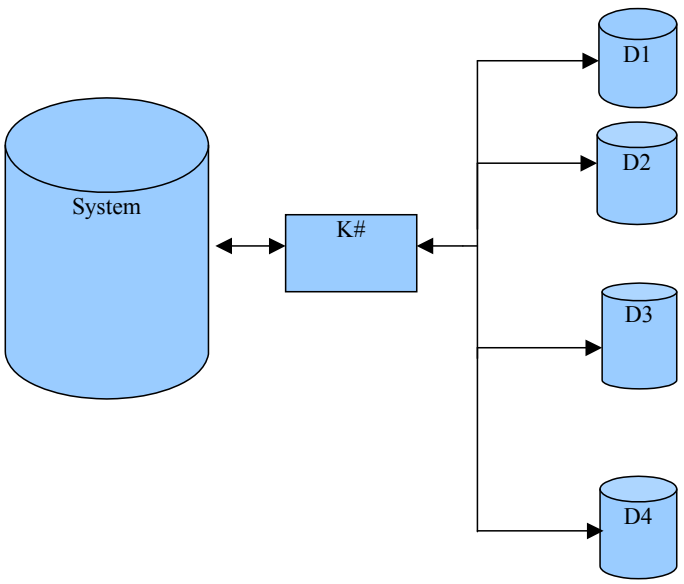
Proces dublowania w czasie rzeczywistym zapisu danych na drugim urządzeniu nosi nazwę tworzenia kopii lustrzanych (ang. mirroring). Dzięki temu oba urządzenia przechowują te same informacje. Mirroring służy do zabezpieczenia danych przed skutkami awarii sprzętu. W trakcie mirroringu (rys. 10) dane przekazywane są do dwóch napędów D1 i D2 (obydwa dyski zawierają te same dane). Jeśli jeden dysk ulegnie awarii, drugi wciąż jest sprawny. Innym sposobem zabezpieczenia jest powielanie (ang. duplexing). W metodzie tej dane są kopiowane przez dwa kanały dyskowe i przechowywane na dwóch dyskach (rys. 11). Dzięki temu odporny na awarię jest nie tylko napęd, ale również kontroler. Można też dublować serwer (ang. server duplexing). W metodzie tej dubluje się cały serwer plików, dzięki czemu użytkownicy w przypadku awarii jednego mają do dyspozycji drugi. Przykładem takiego rozwiązania jest system tolerancji uszkodzeń Fault Tolerance Level III firmy Novell.



Rys. 10. Tworzenie kopii lustrzanych (mirroring)



Rys. 11. Powielanie danych (duplexing)



Rys. 12. Fragmentacja dysków (striping)

Inną metodą zapisu zbiorów na dysku jest fragmentacja. Fragmentacja (ang. striping) polega na podziale pamięci dyskowej na fragmenty, zwykle o rozmiarze 512 kB lub jego wielokrotności, co powoduje rozdzielenie zapisywanej informacji na poszczególne dyski. Podział danych przyspiesza szybkość zapisu i odczytu, gdyż może być wykonywany równolegle i równocześnie maksymalnie wykorzystuje dostępny obszar pamięci. Zasadniczą wadą tego podziału jest brak właściwego bezpieczeństwa danych, a awaria jednego z dysków może powodować zniszczenie wszystkich danych. Załóżmy, że mamy do dyspozycji cztery dyski (rys. 12). Fragmentacja polega na tym, że pierwszy blok zbioru 1 przesyłany jest na dysk #1, blok 2 na dysk #2, blok 3 na dysk #3, a blok 4 zbioru 1 na dysk #4. Następnie blok 5 zbioru 1 ponownie na dysk #1, blok 6 na dysk # 2. Blok 1

zbioru 2 umieszczono na dysku #3, blok 2 zbioru 2 na dysku #4, blok 3 zbioru 2 na dysku #1 i tak aż do wypełnienia. Jeśli zbiór znajduje się cały na dysku (na przykład zbiór 3 na dysku #3) oznacza to, że rozmiar zbioru jest mniejszy niż fragment dysku). Fragmentację stosuje się między innymi w macierzach RAID.

16.2. Replikacja

W rozproszonych bazach danych, w każdym z węzłów można zainstalować lokalną bazę danych. Informacje poufne mogą się znajdować w centralnej bazie danych, a użytkownicy baz lokalnych nie mają do nich dostępu. W systemach rozproszonych baz danych najczęściej wykorzystuje się replikację. Replikacja (ang. replication) jest to mechanizm, który w szybki i pewny sposób pozwala na rozsyłanie informacji do wielu stanowisk. Replikacji może podlegać cała baza danych lub poszczególne relacje. Replikacja zwiększa dostępność do danych, niezawodność systemu bazodanowego oraz zmniejsza natężenie ruchu w sieci. Uniezależnia od przepustowości i aktualnego obciążenia sieci oraz od wydajności zdalnych węzłów i ich aktualnego obciążenia. Dzięki replikacji unika się uniezależnienia od czasowej niedostępności węzłów, czy awarii sieci.

Podstawowe typy replikacji to:

- replikacja migawkowa,
- replikacja transakcyjna,
- replikacja łączona.

Typy pochodne to:

- subskrypcje możliwe do uaktualniania,
- replikacja migawkowa z uaktualnieniem subskrybentów, czyli serwerów przechowujących replikowane dane,
- replikacja transakcyjna z uaktualnieniem subskrybentów,
- replikacja multimaster, czyli taka, w której wszystkie węzły są równorzędne.

Replikacja migawkowa jest stosunkowo prosta do konfiguracji i zarządzania. Podobnie jak replikacja transakcyjna ma cechę jednokierunkowości. Zmiany danych mogą być wykonywane jedynie na serwerze publikatora.

Replikacja oparta na transakcjach kopiuje do bazy danych subskrybenta jedynie te transakcje, które zostały zatwierdzone. Monitorowanie zmian odbywa się na poziomie transakcji. Propagacja zmian następuje zazwyczaj w czasie bliskim rzeczywistemu.

Replikacja łączona pozwala w każdej lokalizacji na wykonywanie zmian w kopii lokalnej replikowanych danych. Przy tego typu replikacji zaniedbywana jest spójność transakcyjna. Replikacja wymaga mechanizmu zapewniającego przeniesienie lokalnych zmian dokonywanych na danych na wszystkie istniejące kopie, aby użytkownicy – niezależnie od miejsca pobytu – uzyskiwali na to samo pytanie taką samą odpo-

wiedź. Proces ten nazywa się synchronizacją (ang. synchronization) lub odświeżaniem (ang. refreshing) repliki. Związane są z nim dwie istotne sprawy: moment synchronizacji i sposób odświeżenia. Utrzymanie kilku kopii tej samej tabeli niesie za sobą wiele niebezpieczeństw, które prowadzą do powstawania konfliktów. Konflikt jest to sytuacja, w której istnieje niemożność uzgodnienia stanu replikowanych obiektów pomiędzy węzłami, co może doprowadzić do utraty zgodności schematów w węzłach sieci.

Według [1] wyróżniamy:

- konflikt aktualizacji, gdy transakcje pochodzące z różnych węzłów dokonują aktualizacji tego samego rekordu w tym samym czasie,
- konflikt unikalności, gdy replikacja rekordu powoduje naruszenie ograniczeń typu klucz podstawowy czy klucz unikalny,
- konflikt usunięcia, gdy jedna transakcja usuwa rekordy z replikowanej tabeli, które w tym samym momencie są modyfikowane lub usuwane przez transakcję z innego węzła,
- konflikt kolejnościowy, powstający w wyniku różnej kolejności propagacji transakcji pomiędzy węzłami.

Po wykryciu konfliktu system realizuje procedury rozwiązywania konfliktów i zapewnienia zgodności replikowanych schematów w węzłach. Jako przykład niech posłuży system zarządzania bazą danych Oracle. Posiada on wbudowane procedury do usuwania konfliktów aktualizacji i konfliktów unikalności, takie jak:

- metoda najpóźniejszego znacznika (ang.latest timestamp),
- metoda nadpisania (ang. overwrite),
- metoda addytywna (ang. additive),
- metoda dodania nazwy węzła (ang. append site name),
- metoda dodania sekwencji (ang. append sequence),
- metoda zignorowania (ang. discard).

Niestety nie ma żadnych wbudowanych metod rozwiązywania konfliktów usunięcia i konfliktów kolejnościowych. W razie ich wystąpienia administrator może zaimplementować własne metody. Jeżeli nie udało się żadną z metod usunąć konfliktu, informacja o tym trafia do kolejki błędów węzła, w którym konflikt wystąpił. Kolejka jest implementowana przez migawkę, inaczej nazwaną perspektywą zmaterializowaną DEFERROR (Oracle). Perspektywa ta zawiera informacje o konfliktach, które zakończyły się niepowodzeniem.

W systemie MS SQL Server są dostępne, z pewnymi modyfikacjami, wszystkie trzy podstawowe typy replikacji. Zaimplementowane zostały dwa typy subskrypcji: push i pull. O konfiguracji subskrypcji push mówi się, gdy subskrypcja została ustawiona w tym samym czasie, w którym tworzone były publikacje. Przez pojęcie publikacji rozumie się zbiór artykułów, grupujących powielane dane w postaci całych tabel, kolumn, wierszy lub nawet procedur składowanych. Konfiguracja push pozwala scentralizować zarządzanie subskrypcjami, wymaga jednak dodatkowych nakładów na

zarządzanie synchronizacją. Subskrypcja typu pull jest ustawiana z poziomu każdego indywidualnego subskrybenta. Jest ona użyteczna dla aplikacji pozwalających na utrzymanie zabezpieczeń na niskim poziomie oraz w przypadku dużej liczby subskrybentów. Proces wykonywania poszczególnych zadań w ramach replikacji jest zautomatyzowany przez moduł SQL Server Agent, konfigurowany przez administratora.

Zastosowanie replikacji przynosi wiele korzyści, a przede wszystkim umożliwia szybszy dostęp do danych ze względu na to, że dane są dostępne lokalnie. Również możliwe jest częściowe uniezależnienie od awarii sieciowych. Niestety z replikacją wiąże się wiele problemów implementacyjnych.

16.3. Kopie lustrzane zdalne

Zdalne kopie lustrzane (ang. remote mirror), tworzące tak zwane kopie wtórne (replikacje), mogą być realizowane sprzętowo lub programowo. Mają różną lokalizację, dzięki czemu w przypadku awarii centrum podstawowego, centrum z kopiami zapasowymi może w każdej chwili przejąć jego zadania. Możemy wyróżnić:

- kopie synchroniczne,
- kopie semisynchroniczne,
- kopie asynchroniczne.

Replikacja, czyli powielanie synchroniczne, to najbardziej zaawansowana technologicznie i wymagająca operacja tworzenia kopii w czasie rzeczywistym. W tym przypadku zapis nowej lub zmienionej informacji następuje zarówno w centrum podstawowym, jak i zapasowym. Dopiero po potwierdzeniu obu zapisów serwer dostaje potwierdzenie wykonania całej operacji. Takie wspólne wykonywanie operacji pozwala w razie awarii natychmiast przejąć przez kopię rolę aktywną, nie tracąc czasu ani danych. Obydwa zapisy – podstawowy i jego replika – są przecież aktualne. Powielanie synchroniczne wymaga dużej przepustowości łącza, co bezpośrednio związane jest z wysokimi kosztami. Dodatkowo opóźnienia wnoszone przez łącza oraz czas oczekiwania na potwierdzenie zrealizowania operacji powielania mają duży wpływ na wydajność systemu. Również wszystkie błędy operatora lub aplikacji są powielane w kopiach. Wymaga to wprowadzenia dodatkowych mechanizmów zabezpieczających od strony serwera. Kolejną wadą są sytuacje, w których awaria głównego systemu dyskowego może spowodować utratę spójności danych (ang. rolling disaster) i dostępności do danych. Reasumując, powielanie synchroniczne może być realizowane zarówno sprzętowo, jak i programowo. Zapis na dysk jest uznany dopiero po zakończonym zapisie kopii. Ze względu na opóźnienia w łączach odległość między wersją podstawową a kopiami jest ograniczona. W przypadku dużej liczby zapisów na dysk utrzymywanie kopii synchronicznej negatywnie wpływa na wydajność systemu. Niemniej jednak tego typu powielanie zapewnia najkrótszy czas przywrócenia funkcjo-

nalności systemu po awarii i minimalizację utraty danych. Podczas powielania synchronicznego nie występują konflikty. Każda modyfikacja w węźle lokalnym powoduje założenie lokalnej blokady na modyfikowanych rekordach, a następnie na odpowiednich rekordach w zdalnej replice tabeli. Żadna inna transakcja nie może wykonać operacji modyfikacji zablokowanych rekordów. Nie dochodzi więc do konfliktów. Największą barierę stanowią wysokie koszty realizacji.

Przykłady replikacji synchronicznej:

IBM ESS: Peer-to-peer Copy,

Hitachi Data Systems: True Copy,

EMC: SRDF Synchronous.

W przypadku kopii semisynchronicznych EMC: SRDF Semi-synchronous różnica w stosunku do kopii synchronicznych polega na tym, że zapis jest uznany za zakończony bez oczekiwania na potwierdzenie z centrum zapasowego, ale kolejne zapisy muszą czekać na potwierdzenie.

Powielanie asynchroniczne stanowi rozwiązanie zapewniające wysoki stopień bezpieczeństwa przy znacznie niższych wymaganiach i kosztach. Centrum podstawowe buforuje operacje zapisu, grupując je w paczki (ang. consistency group) i przesyła je do centrum zapasowego w określonych momentach czasowych. Nie jest wymagane tworzenie kopii w czasie rzeczywistym. Aplikacja dostaje natychmiastowe potwierdzenie dokonania zapisu. W ten sposób minimalizuje się wpływ procesu powielania na wydajność systemu, znacznie obniżają sie również koszty bieżące. Opóźnienie w transmisji pozwala na zastosowanie skutecznych zabezpieczeń przeciwko utracie spójności danych, propagacji błędów operatora i aplikacji. Jedyną wadą tego typu rozwiązania jest tworzenie się luk pomiędzy centrami obejmującymi te dane, które nie zostały jeszcze przeniesione do centrum zapasowego. Przykłady systemów replikacji asynchronicznej:

IBM ESS: Extended Remote Copy (XRC),

Hitachi Data Systems: HXRC.

Rozwinięciem konfiguracji kopii zdalnej jest kopia lustrzana z rozszczepieniem (ang. split mirror backup). Zamiast tworzyć pełną kopię zdalną w określonym czasie, decydujemy się na utrzymywanie zdalnej kopii lustrzanej dysków w centrum zapasowym, czyli mamy prawie cały czas aktualną kopię po stronie centrum zapasowego. Co pewien czas należy przerywać połączenie, aby zamrozić spójny obraz danych. Po zamrożeniu następuje tak zwana resynchronizacja, czyli uzupełnienie danych o dane zapisywane w centrum podstawowym w trakcie zamrożenia kopii zapasowej. W centrum zapasowym może istnieć wiele kopii zapasowych. Mówimy wtedy o wielokrotnych kopiach lustrzanych z rozszczepieniem.

17. Kopie migawkowe

17.1. Migawka

Mechanizmem wykorzystywanym do replikacji jest również zbiór danych, zwanych migawką (ang. snapshot). Migawka może być fizyczną kopią danych jak w rozwiązaniu firmy EMC TimeFinder lub jest to odpowiedni mechanizm, pozwalający na dostęp do danych zamrożonych w momencie wykonywania kopii, które nie zostały jeszcze zapisane w postaci fizycznej kopii. Kopia może być pełna lub różnicowa. Pełne kopie dają większe bezpieczeństwo danych, wprowadzają jednak ogromne wymagania dotyczące pojemności dysków. Na przykład firma Network Appliance szacuje, że przy 31 kopiach narzut potrzebny do przechowania ich waha się między 10–35% pojemności systemu. Inny problem stwarza wpływ czynności kopiowania na wydajność systemu. Zależy on przede wszystkim od systemu, który powinien sam optymalizować proces zapisu kopii migawkowej i obsługę bieżących operacji odczytu i zapisu danych. Stosuje się tryb zapisu danych zwany WAFL (ang. Write Anywhere File Layout). W tym trybie zapis nowej lub zmienionej informacji następuje zawsze w obszarach dysku, oznaczonych jako wolne. Dzięki temu stara informacja nie jest zastępowana przez nową. Kopię migawkową tworzy się przez skopiowanie tabeli wskaźników, przypisujących poszczególne bloki w systemie dysków RAID (ang. Redundant Array of Inexpensive or Independent Disks) poszczególnym plikom. Proces ten trwa zaledwie kilka sekund i w jego wyniku informacja zapisana na dyskach zostaje uporządkowana w postaci niezależnych zbiorów (ang. filesystem), z których jeden jest zmienny, a pozostałe – zwane migawkami – zamrożone. Bloki zajęte przez informację zostają przeniesione do puli wolnych dopiero po wykasowaniu wszystkich migawek. Odpada więc problem kopiowania ochranianych danych, dzięki czemu zwiększa się wydajność systemu.

Migawka opisana jest za pomocą takich parametrów, jak:

- typ migawki,
- nazwa migawki,
- moment wypełnienia migawki danymi,
- specyfikacja sposobu odświeżenia,

- specyfikacja momentu rozpoczęcia automatycznego odświeżenia,
- specyfikacja częstotliwości odświeżenia,
- informacja określająca zakres danych dostępnych w migawce.

Ze względu na moment odświeżania dzielimy migawki na:

- synchroniczne, odświeżane natychmiast po modyfikacji danych źródłowych,
- asynchroniczne, odświeżane na żądanie użytkownika, po zatwierdzeniu transakcji po upływie pewnego interwału czasowego.

Ze względu na sposób odświeżania wyróżniamy migawki:

- pełne, polegające na kopiowaniu całej zawartości tabeli źródłowej,
- przyrostowe, polegające na kopiowaniu zmian.

W języku SQL migawkę tworzy się poleceniem CREATE SNAPSHOT lub CREATE MATERIALIZED VIEW.

Mechanizm tworzenia migawki pokazuje poniższy przykład.

Przykład 14

```
CREATE SNAPSHOT studenci;
BUILD IMMEDIATE;
REFRESH COMPLETE;
START WITH sysdate+(1/24*60);
NEXT sysdate+(1/24*60*6);
AS SELECT * FROM emp@wroclaw;
UNION;
SELECT* FROM emp@opole;
```

Definiujemy tu migawkę o nazwie studenci, wypełnioną danymi w momencie tworzenia (BUILD IMMEDIATE), o sposobie odświeżania pełnym, momencie rozpoczęcia odświeżania automatycznego START WITH sysdate+(1/24*60), częstotliwości odświeżania NEXT sysdate+(1/24*60*6).

Specyfikacja sposobu odświeżania migawki umożliwia określenie, czy migawka ma być odświeżana w sposób pełny, czy przyrostowy. Odświeżanie przyrostowe jest możliwe, jeśli został utworzony dziennik migawki na tabeli źródłowej i migawka jest typu prostego. Migawka typu prostego bazuje na jednej tabeli, niepołączonej samej z sobą, nie posiada operatorów zbiorowych (np. UNION) funkcji grupowych ani klauzul typu START WITH itp. Możliwe jest również zdefiniowanie migawki nigdy nieodświeżanej (NEVER REFRESH). Wszelkiego rodzaju zmiany dotyczące migawki zapisywane są w dzienniku migawki.

Mechanizm tworzenia dziennika migawki pokazuje poniższy przykład.

Przykład 15

```
CREATE SNAPSHOT LOG ON emp;
PCTFREE 5;
```

```
TABLESPACE users;
STORAGE;
```

Korzystając z migawki, można wykonywać różnego rodzaju operacje. W poniższym przykładzie pokazano modyfikacje danych.

Przykład 16

```
CREATE SNAPSHOT kopia ewf;
REFRESH NEXT sysdate+1;
FOR UPDATE;
SELECT* FROM emp@warszawa;
```

Podstawowe zastosowanie migawek to tworzenie natychmiastowych backupów, pozwalające na odzyskiwanie w ciągu kilkunastu sekund skasowanych, zniszczonych lub zmienionych plików. Kopie migawkowe znakomicie sprawdzają się jako źródło dla backupu taśmowego. Wykonanie kopii migawkowej trwa znacznie krócej. Migawki pozwalają również na ekstrakcje danych operacyjnych do hurtowni. Dzięki możliwości natychmiastowego przywracania systemu plików do zadanej konfiguracji migawki idealnie nadają się dla środowisk testowych i do tworzenia aplikacji.

17.2. Zdalna replikacja danych

W ostatnich latach obserwuje się, że w rozwiązaniach klasy Network Attache Storage (NAS) są dostępne technologie, pozwalające na zdalną replikację danych. Przykładem jest tu produkt tej firmy SnapMirror, oparty na zastosowaniu kopii migawkowych. Migawki są wykorzystane do efektywnej i prostej replikacji danych między serwerami plików, przy minimalnych wymaganiach co do przepustowości łączy. Migawki lustrzane (ang. SnapMirror) to rozwiązanie asynchroniczne, oparte na kolejno wykonywanych kopiach migawkowych i transmisji danych różnicowych na poziomie bloków dyskowych między poszczególnymi kopiami. Transmisja jest tu ograniczona do minimum.

Przykład 17

Mamy dane dwa systemy: system źródłowy (ang. source) i docelowy (ang. target). Cały proces można przedstawić w postaci następujących kroków:

Krok 1. W systemie źródłowym Source wykonana jest kopia migawkowa.

Krok 2. Dane przesyłamy z Source do systemu docelowego Target jako transfer zerowy.

Krok 3. W trakcie transferu Source jest aktywny i zapisuje nowe informacje.

Krok 4. Target jest w pełni zgodny z pierwszą kopią Source.

Krok 5. W Source wykonana jest kolejna kopia migawkowa.
Krok 6. Różnica między kopiami Source i Target jest transmitowana do Target.
Krok 7. Powtarzamy operacje i transmisje.

Wszystkie te operacje są wykonywane zgodnie z wymaganiami bezpieczeństwa i przepustowością sieci.

18. Macierze dyskowe

18.1. Kontrolery macierzy RAID

Jedną z metod zapewnienia bezpieczeństwa przechowywanych danych oraz zwiększenia niezawodności systemu jest stosowanie macierzy dyskowych zamiast pojedynczych dysków, dysków lustrzanych czy dysków zdwojonych. Zwiększenie niezawodności i uodpornienie na awarię osiąga się przez zastosowanie macierzy dyskowych RAID (ang. Redundant Array of Inexpensive or Independent Disks). Pomysł zbudowania tego typu macierzy powstał około 30 lat temu, ale – stale ulepszany – dotrwał do chwili obecnej. Macierz RAID jest to zbiór dysków zapewniający współdzielenie i podział informacji, sterowany kontrolerem nadzorującym ich pracę. Liczba dysków, sposób zapisu na nich danych, połączenia między dyskami pozwalają na zwiększenie integralności danych, odporności na uszkodzenia oraz na zwiększenie przepustowości systemu.

Można wyróżnić trzy kategorie typów RAID:

- podstawowe,
- zagnieżdżone,
- dowolne kombinacje podstawowych i zagnieżdżonych.

Kontrolery macierzy RAID możemy podzielić na:

- programowe,
- sprzętowe,
- hybrydowe.

Kontrolery programowe charakteryzują się niską ceną i prostotą, ponieważ nie trzeba instalować dodatkowego sprzętu. Są jednak mało wydajne i często występują problemy z kompatybilnością oprogramowania.

Kontrolery sprzętowe charakteryzują się dużą szybkością, są kosztowne, mają możliwość wymiany dysku bez konieczności wyłączania macierzy z pracy. Są odporne na braki w zasilaniu. Biorąc pod uwagę wady i zalety obu rozwiązań zbudowano kontrolery hybrydowe, łączące w sobie cechy obu rozwiązań.

Jako przykład kontrolera może posłużyć JetStor SATA 416S, którego cechy wyspecjalizowano poniżej:

- 400 MHz RISC CPU z 266 MHZ magistralą pamięci z układem ASIC do generowania informacji nadmiarowej,

- przepustowość magistrali wewnętrznej 1600 MB/s,
- interfejs Hosta: 2×320 MB/s LVD SCSI,
- iterfejsy dysków: 16×3 Gb PHY SATAII, adresowanie 48-bitowe,
- 1GB pamięci cache (DDR ECC SDRAM),
- wielkość: 3U (montaż w szafach typu rack),
- waga 18 kg bez dysków,
- możliwość konfiguracji w trybie RAID 0, 1, 0+1, 3, 5, 6,
- przełączanie dysków bez wyłączania całego urządzenia (ang. hotswapping).

Dodatkową zaletą tego kontrolera jest 48-bitowe adresowanie SATAII, które pozwala zaadresować 144 petabajtów obszaru dyskowego, jednakże fizycznie przy wykorzystaniu największych obecnie dysków 500 GB maksymalna pojemność macierzy wynosi 8 TB. Możliwa jest również obsługa przez sieć (ang. Simple Network Management Protocol). Złącze ethernetowe pozwala zarządzać macierzą przez interfejs przeglądarki web.

18.2. Podstawowe typy RAID

Do kategorii podstawowych typów możemy zaliczyć RAID poziomu 0, 1, 2, 3, 4, 5, 6.

Podstawowe typy RAID mają następujące zalety:

Z1 – zwiększoną szybkość zapisu lub odczytu dzięki temu, że każdy z dysków ma do zapisania lub odczytania tylko część pliku, tak zwany zapis z przeplotem (ang. stripping),

Z2 – prosta implementacja kontrolera, w związku z czym niski koszt,

Z3 – system może przetrwać uszkodzenia jednego z dysków,

Z4 – dobre transfery odczytu i zapisu,

Z5 – system może obsłużyć współbieżnie wiele żądań dostępu do plików z uwagi na podział danych na poziomie bloków,

Z6 – zerowy czas odbudowy macierzy.

Zalety podstawowych typów macierzy RAID podano w tabeli 2.

Tabela 2

	RAID 0	RAID 1	RAID 2	RAID 3	RAID 4	RAID 5	RAID 6
Z1	X	X					
Z2	X	X					
Z3	X			X	X	X	X
Z4				X	X	X	X
Z5					X	X	X
Z6		X					

Wśród wad wyróżniamy:

W1 – brak tolerancji uszkodzeń, a nawet pogorszoną odporność na uszkodzenia z uwagi na to, że przy uszkodzeniu jednego z dysków dane w macierzy nie nadają się do użytku,

W2 – marnowanie przestrzeni dyskowej na dużą nadmiarowość (dyski lustrzane),

W3 – konieczność zapisu informacji na każdym z dysków,

W4 – nie można obsłużyć współbieżnie wielu żądań z uwagi na aktywność każdego dysku przy odczycie bloku danych,

W5 – trudny w w implementacji sprzętowej i programowej,

W6 – długi czas odbudowy macierzy,

W7 – spowolnienie działania podczas odbudowy,

W8 – nieefektywny, w przypadku małej liczby dysków w macierzy, ponieważ duży procent pojemności jest wykorzystywany do zapisania informacji nadmiarowej,

W9 – zmniejszona prędkość zapisu wynikającą z konieczności generacji aż dwóch nadmiarowych informacji.

Wady podstawowych typów macierzy RAID podano w tabeli 3.

Tabela 3

	RAID 0	RAID 1	RAID 2	RAID 3	RAID 4	RAID 5	RAID 6
W1	X						
W2		X	X				
W3		X					
W4				X			
W5					X	X	X
W6						X	X
W7						X	X
W8							X
W9							X

Systemy podstawowe RAID znajdują zastosowanie:

AP1 – podczas przetwarzania i edycji dużych plików audio-video,

AP2 – wszędzie tam, gdzie potrzebna jest duża wydajność odczytu/zapisu, a gdzie nie zależy nam na tolerowaniu uszkodzeń,

AP3 – w bankowości i finansach,

AP4 – wszędzie tam, gdzie potrzebna jest wysoka niezawodność systemu,

AP5 – w systemach, w których potrzebne są dobre transfery,

AP6 – we wszelkiego rodzaju serwerach: email, www, bazy danych, gdzie potrzebny jest kompromis między niezawodnością a szybkością odczytu,

AP7 – tam, gdzie potrzebna jest duża tolerancja uszkodzeń.

Zastosowania podstawowych typów macierzy RAID wymieniono w tabeli 4.

Tabela 4

	RAID 0	RAID 1	RAID 2	RAID 3	RAID 4	RAID 5	RAID 6
AP1	X			X	X		
AP2	X						
AP3		X					
AP4	X						
AP5				X	X		
AP6						X	X
AP7							X

Z podanych tabel wynika, że macierz RAID2 nie znajduje obecnie zastosowania.

18.3. Zagnieżdżone typy RAID

Podstawowe typy RAID są ze sobą łączone i tworzą typy zagnieżdżone.

Do zagnieżdżonych typów RAID należą przede wszystkim RAID poziomu 0+1, 1+0, 5+0, 10+0 i dowolne kombinacje podstawowych typów RAID. Przykładowo macierz RAID 0+1 jest realizowana jako RAID 1, której elementami są macierze RAID 0. Macierz taka posiada zarówno zalety macierzy RAID 0, jak i macierzy RAID 1. Pojedyncza awaria dysku powoduje, że całość staje się praktycznie RAID 0. Macierz RAID 1+0 realizowana jest jako RAID 0, której elementami są macierze RAID 1. W porównaniu do macierzy RAID 0+1 realizuje tę samą koncepcję połączenia zalet RAID 0 i RAID 1, ale w odmienny sposób. Dzięki technice zapisu informacji na dyskach podczas wymiany uszkodzonego dysku odbudowywany jest tylko fragment całej macierzy. Macierz RAID 5+1 znana jest jako macierz RAID 6. Zawiera dwie niezależne sumy kontrolne. Jest bardzo kosztowna w implementacji, ale daje bardzo duże bezpieczeństwo.

Zagnieżdżone typy RAID mają następujące zalety:

Z1 – wydajność porównywalna z RAID 0,

Z2 – odporność na uszkodzenia porównywalna z RAID 5,

Z3 – wytrzymałość uszkodzenia kilku dysków do czasu, aż w każdym poziomie działa jeden dysk,

Z4 – szybkość porównywalna z RAID 0,

Z5 – bardzo dobre transfery,

Z6 – tolerancja uszkodzenia jednego z dysków,

Z7 – duża tolerancja uszkodzeń.

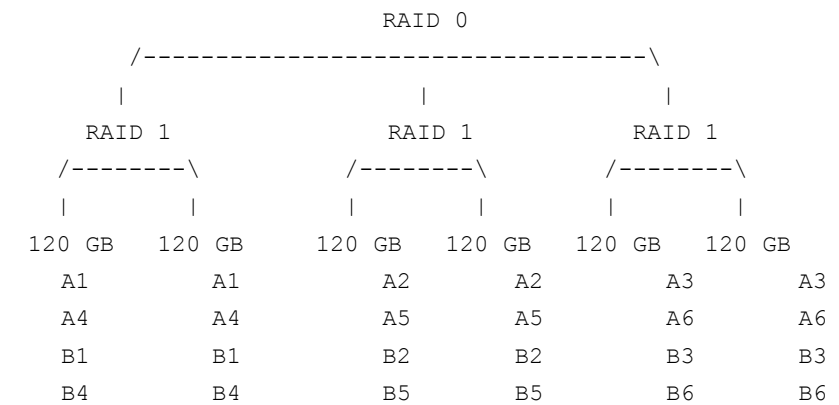
Z8 – migracja tak zwanego punktu krytycznego macierzy (ang. hotspot).

Zalety zagnieżdżonych typów macierzy RAID podano w tabeli 5.

Na rysunku 13 pokazano zagnieżdżoną macierz typu RAID 1+0.

Tabela 5

	RAID 0+1	RAID 1+0	RAID 5+0	RAID 10+0
Z1	X			
Z2	X			
Z3		X		
Z4		X		
Z5			X	
Z6			X	
Z7				X
Z8				X



gdzie: A1, B1, itd. przedstawia jeden blok danych, każda kolumna przedstawia jeden dysk.

Rys. 13. Zagnieżdżona macierz RAID 1+0 (www.acnc.com)

Zagnieżdżone typy macierzy RAID mają następujące wady:

W1 – marnowanie przestrzeni dyskowej,

W2 – zła skalowalność; jeśli chcemy zwiększyć macierz, musimy dodać po jednym dysku do każdego poziomu,

W3 – uszkodzenie jednego z dysków sprowadza macierz do postaci RAID 0,

W4 – bardzo drogie rozwiązanie sprzętowe,

W5 – trudne do zaimplementowania,

W6 – dyski muszą być zsynchronizowane, co limituje wybór dysków,

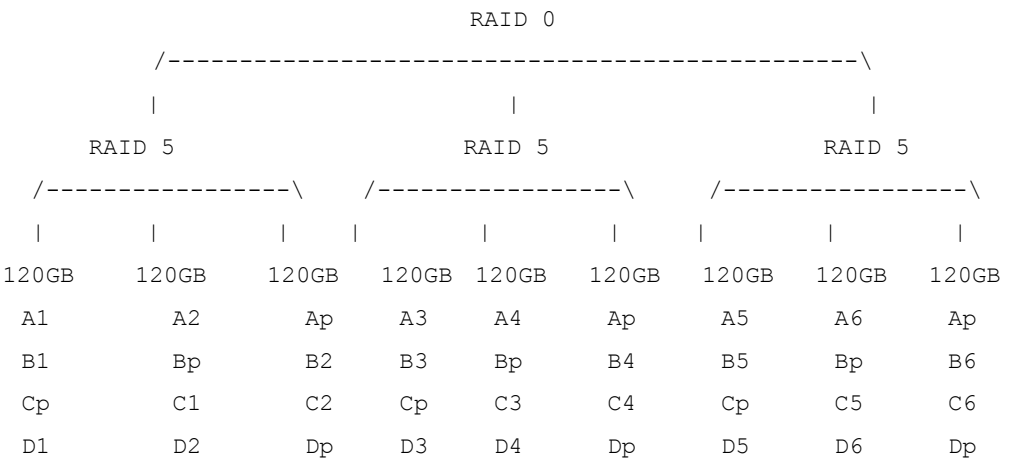
W7 – uszkodzenia dwóch dysków z jednego poziomu dla RAID 5+0 i uszkodzenie najniższego poziomu w RAID 10+0 niszczy macierz.

Wady, jakie mają zagnieżdżone typy macierzy RAID podano w tabeli 6.

Tabela 6

	RAID 0+1	RAID 1+0	RAID 5+0	RAID 10+0
W1	X			
W2	X	X		
W3	X			
W4	X		X	
W5		X		
W6			X	
W7			X	X

Na rysunku 14 pokazano zagnieżdżoną macierz typu RAID 5+0.

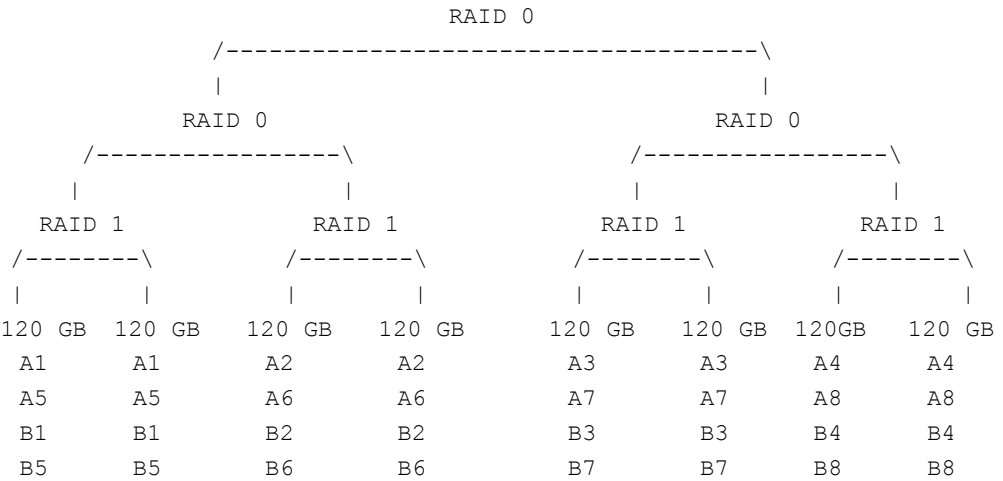


gdzie: A1, B1, itd. przedstawia jeden blok danych, każda kolumna przedstawia jeden dysk.

Rys. 14. Zagnieżdżona macierz RAID 5+0 (www.acnc.com)

Systemy zagnieżdżone RAID 1+0 i RAID 5+0 znajdują zastosowanie jako serwery bazodanowe, wymagające dużej niezawodności i przepustowości. Systemy RAID 0+1 służą głównie jako serwery plików do przetwarzania audio-video i wszędzie tam, gdzie nie musimy mieć wysokiej niezawodności.

Na rysunku 15 pokazano zagnieżdżoną macierz typu RAID 10+0.



Rys. 15. Zagnieżdżona macierz RAID 10+0 (www.acnc.com)

18.4. Kombinacje systemów RAID

Popularność systemu RAID sprawiła, że coraz większa liczba firm zainteresowała się rozwiązaniami tego typu. Dla własnych potrzeb firmy tworzyły różne wariacje istniejących rozwiązań. I tak powstały:

- JBOD (ang. Just a Bunch of Disks) – konkatenacja dysków w jedną logiczną całość. Nie jest to faktyczny poziom RAID, gdyż nie posiada nadmiarowości. Konkatenacja stanowi odwrotność partycjonowania dysku. Zaletą rozwiązania jest to, że przy uszkodzeniu tracone są tylko i wyłącznie dane na zniszczonym dysku.
- RAID 7 – produkt firmy Storage Computer Corporation. Praktycznie jest to RAID 3 lub RAID 4 z tą różnicą, że posiada dodatkowy bufor, funkcjonujący podobnie do buforu cache w procesorach, poprawiający odczyty i zapisy z tego samego obszaru wynikające z zasady lokalności, czyli korzystania z niewielkiego obszaru danych. Zalety takiego rozwiązania to praktycznie zerowy czas dostępu z punktu widzenia prędkości dysków. Zasadniczą wadą jest bardzo wysoki koszt pamięci i kontrolerów. Ponadto istnieje konieczność podtrzymywania zasilania pamięci przez zasilacze UPS.
- Matrix RAID – specjalna opcja Biosu ICH6R RAID BIOS firmy INTEL. Działa na dwóch dyskach, tworząc dwa oddzielne poziomy RAID 0 i RAID 1. RAID 1 stanowi tak zwaną bezpieczną część systemu. Jest przeznaczony na ważne dokumenty i pliki. RAID 0 przeznaczony jest na aplikacje, oprogramowanie systemu i inne komponenty, które wymagają dużej szybkości działania.

19. Klastry

Inny sposób zabezpieczenia baz danych to grupowanie (ang. clustering). Chodzi tu o grupę serwerów wykorzystujących wspólnie te same zasoby i zapewniających użytkownikom równorzędne usługi. Jeśli jeden z serwerów ulegnie awarii i zostanie wyłączony, pozostałe przejmą jego rolę. Serwery znajdujące się w tym samym klastrze mogą mieć dostęp do tych samych systemów plików, a te z kolei mogą być zabezpieczone przez kopie lustrzane lub przez wykorzystanie konfiguracji typu RAID. Klastry (ang. cluster), zwane także gronami, to zaawansowana technologia, która pozwala na utrzymanie niemal stałej dostępności systemu informatycznego oraz dużej wydajności. Klaster to połączona grupa jednostek komputerowych, zwanych węzłami, tworzących jednolity system, który współpracuje przy udostępnianiu i dzieleniu różnego rodzaju usług i zasobów. W klastrach dla podniesienia bezpieczeństwa stosuje się zamiast pojedynczego dysku macierze. Czasami stosuje się ponadto dodatkowe węzły, które w czasie normalnej pracy nie są używane, lecz tylko w czasie awarii, co pozwala na utrzymanie sprawności systemu. Niektóre systemy klastrowe pozwalają na tolerowanie uszkodzeń. Są to jednak rozwiązania bardzo kosztowne. W praktyce częściej stosuje się klastry zapewniające dużą dostępność do danych. Ich głównym celem jest minimalizacja czasu spowodowanego awarią. Systemy te, w odróżnieniu od tolerujących uszkodzenia, nie gwarantują ciągłej pracy, lecz umożliwiają prawie natychmiastowe jej wznowienie.

Klastry możemy podzielić na:

- wydajnościowe, które pracują tak jak równoległe komputery, a ich zadaniem jest zwiększenie mocy obliczeniowej systemu,
- niezawodnościowe, które dublują swoje funkcje na wypadek awarii.

W praktyce dąży się do tego, aby obydwie funkcje były realizowane w jednym rozwiązaniu. Dotyczy to przede wszystkim serwerów www. Inna stosowana klasyfikacja klastrów to podział na klastry: aplikacyjne i systemowe. Jedną z najbardziej popularnych implementacji klastrów jest klaster typu Beowulf, gdzie rolę węzłów pełnią wydajne komputery klasy PC, pracujące pod kontrolą GNU/Linuxa oraz z zainstalowanym oprogramowaniem, pozwalającym uzyskać przetwarzanie równoległe. Jeden z największych obecnie klastrów jest zbudowany z 1100 podwójnych procesorów Apple G5 (2.0 GHz) i jest trzecim co do wielkości mocy obliczeniowej superkomputerem na świecie.

Od systemów klastrowych oczekuje się znacznego zredukowania lub wyeliminowania przestojów w pracy serwera obsługującego systemy krytyczne. Podstawowe korzyści, jakie przynosi ta technologia to odporność na awarie i zrównoważone obciążenia. Główne jej wady to złożoność i koszty.

20. Technika przeciwdziałania awariom

W poprzednich rozdziałach opisano różnego typu zabezpieczenia przed skutkami awarii. Obecnie postaramy się określić, jakiego typu techniki trzeba użyć, aby była skutecznym zabezpieczeniem przed awarią.

20.1. Awarie sprzętowe

Dla poszczególnych awarii można wyróżnić następujące techniki zabezpieczenia:

- Przy awarii przestrzeni dyskowej należy stosować:
  – macierze RAID ,
  – kopie lustrzane zdalne,
  – backup pełny, gdy awarii ulega cały RAID,
  – klastry,
  – replikację danych.

Niewskazane jest sporządzanie kopii migawkowych, gdy awarii ulega cały RAID.

- Przy uszkodzeniu serwera, które uniemożliwia pracę stosuje się:
  – klastry,
  – replikację,
  – centralne składowanie danych i serwery zapasowe,
  – backup przyrostowy i różnicowy,
  – redundancję sprzętową zarówno zasilaczy, jak i serwerów.
- Przy utracie zasilania należy zastosować:
  – redundancję zasilaczy,
  – urządzenia UPS,
  – odpowiednią konfigurację obwodów zasilania,
  – zapasowe centrum danych,
  – backup w celu odtworzenia uszkodzonych danych.
- Przy całkowitym zniszczeniu serwerowni (pożar, powódź itp.) stosuje się:
  – zapasowe, oddalone centrum danych,
  – przechowuje się backup w oddalonych archiwach.

20.2. Awarie programowe

Można wyróżnić następujące techniki zabezpieczenia w poszczególnych awariach:

- Przy błędach programowych wymagających odtworzenia pojedynczych plików:
  – backup na dyski lub taśmy,
  – migawki.

Niewskazane są kopie lustrzane, replikacja, klastry.

- Przy błędach programowych, wymagających odtworzenia całego systemu plików:
  – backup na dyski lub taśmy,
  – backup plikowy baz danych,
  – migawki plikowe
  – backupy obrazu dysków.

Niewskazane są kopie lustrzane sprzętowe lub programowe, replikacja, klastry i migawki.

Podsumowanie

Niezależnie od poziomu bezpieczeństwa i dostępności danych, które udało nam się zapewnić stosując różne konfiguracje programowe i sprzętowe, zawsze musimy się liczyć z niebezpieczeństwem utraty danych, wynikającym z błędów człowieka czy też ukrytych błędów oprogramowania. Nawet wielokrotna replikacja danych nie zagwarantuje zabezpieczenia przed przypadkowym skasowaniem danych przez użytkownika czy włamaniem się intruza do systemu. Bardzo ważne jest posiadanie dostatecznie wydajnego systemu archiwizacji i odtwarzania, który pozwoli zautomatyzować operacje tworzenia i odzyskiwania kopii.

Bibliografia

[1] Bębel B., Wrembel R., *Oracle*. *Projektowanie rozproszonych baz danych*, Wydawnictwo Helion, Gliwice 2003.

[2] Chałon M., *Ochrona i bezpieczeństwo systemów baz danych*, [w:] *Inżynieria komputerowa*, praca zbiorowa pod redakcją W. Zamojskiego, WKŁ, Warszawa 2005.

[3] Chang B., Scardina M., Kiritzov., *Oracle 9i i XML*, Wydawnictwo Helion, Gliwice 2003.

[4] Gallagher S., *SQL Server 7. Księga eksperta*, Wydawnictwo Helion, Gliwice 2001.

[5] Greene J., *Oracle8 Server. Księga eksperta*, Wydawnictwo Helion, Gliwice 2000.

[6] Loney K., *Oracle Database 10g. Kompendium administratora*, Wydawnictwo Helion, Gliwice 2005.

[7] Loney K., Theriault M., *Oracle 8i . Podręcznik administratora bazy danych*, Wydawnictwo Helion, Gliwice 2003.

[8] Otey M., *SQL Server 2005. Nowe możliwości*, Wydawnictwo Helion, Gliwice 2005.

[9] Theriault M., Carmichael R., Viscusi J., *Oracle 9i.Administrowanie bazami danych od podstaw*, Wydawnictwo Helion, Gliwice 2003.

[10] Urman S., *Oracle 8. Programowanie w języku PL/SQL*, Wydawnictwo Helion, Gliwice 2002.

[11] Whalen E., Schroeder M., *Oracle. Optymalizacja wydajności*, Wydawnictwo Helion, Gliwice 2003.

[12] Wrembel R., Jezierski J., Zakrzewicz M., *System zarządzania bazą danych Oracle 7 i Oracle 8*, Wydawnictwo Nakom, Poznań 1999.

CZĘŚĆ V

Tendencje rozwojowe baz danych

Współczesne bazy danych rozwijają się w różnych kierunkach, ale duża liczba zastosowań (...) *polega na budowaniu jednej dużej bazy danych* (...) *wirtualnej* (...) *tak, aby można było pytać o dane, jakby pochodziły z jednego źródła* (...).

H. Garcia Molina, J. Ullman, J. Widom

21. Integracja danych

We współczesnym świecie pojawiła się potrzeba gromadzenia danych pochodzących z różnych źródeł, pozwalających na przechowywanie danych historycznych i potrafiących w dogodny sposób udostępnić dane do analizy. Dane zarządzane są przez system zarządzania baz danych jako jedna duża całość. Jednym z tego typu rozwiązań jest federacyjny system baz danych, który ma niezależne źródła danych, ale każde źródło może pozyskiwać obce dane. Bardziej nowoczesne rozwiązanie polega na tworzeniu hurtowni baz danych. Hurtownia w jednej bazie przechowuje kopie danych z różnych źródeł, które wcześniej zostały przetworzone, czyli filtrowane, złączane lub agregowane. Alternatywę dla hurtowni stanowią mediacje. Mediator jest składową oprogramowania, nie przechowuje żadnych własnych danych. Przekształca zapytanie użytkownika i syntetyzuje odpowiedź. Tak utworzona baza jest wirtualną bazą danych. Zarówno hurtownie, jak i mediatory potrzebują w każdym źródle danych składowych, które tłumaczą zapytania i ich wyniki. Są to ekstraktory i wrapery.

Okazuje się, że korzystanie z różnych źródeł stanowi bardzo poważny problem. Różnice między źródłami nie dotyczą bowiem tylko schematu, lecz także danych, które w różnych źródłach, mimo tego samego znaczenia, mogą być reprezentowane inaczej. Różnice dotyczą między innymi:

- typów danych,
- wartości danych,
- semantyki danych,
- wartości pominiętych w niektórych rozwiązaniach.

Tego typu niespójności należy rozwiązywać za pomocą programów translacyjnych jeszcze przed implementacją zintegrowanej bazy danych.

Nowe spojrzenie na dane, czyli korzystanie z różnych źródeł informacji spowodowało, że problem ochrony i bezpieczeństwa danych znacznie się skomplikował.

22. Federacyjne systemy baz danych

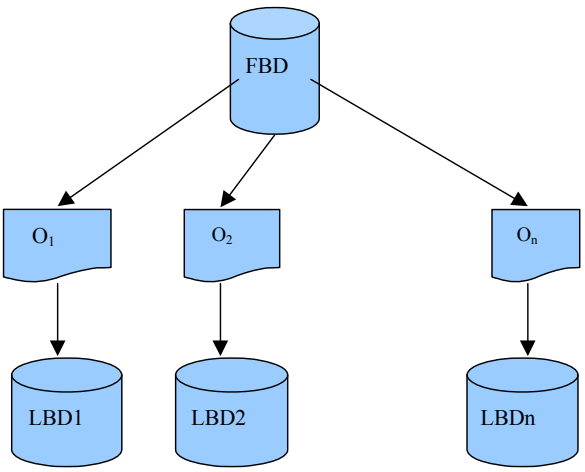
Federacyjne Bazy Danych (FBD) to najprostsza architektura wspomagająca integrowanie danych. FBD to kolekcja heterogenicznych, autonomicznych baz połączonych siecią, zarządzanych przez Federacyjny System Zarządzania Bazami Danych (FSZBD). System taki umożliwia tworzenie aplikacji globalnych, działających na całych bazach danych. Powinien ponadto zapewnić przeźroczystość i niezależność danych. Ogólnie przeźroczystość oznacza, że ukrywa się przed użytkownikami informacje, które opisują rozproszenie systemu. Użytkownik musi mieć wrażenie, że jest jedynym pracującym w systemie. Wyróżnia się kilka rodzajów przeźroczystości:

- przeźroczystość sieci i rozproszenia – uniezależnienie użytkowników od wszystkich szczegółów sieci, takich jak miejsce przechowywania danych czy nazwa użyta w systemie,
- przeźroczystość współbieżności przy zachowaniu spójności danych,
- przeźroczystość skalowania – wprowadzenie nowych elementów do bazy bez wpływu na działanie programów i pracę użytkowników,
- przeźroczystość replikacji, umożliwiającą synchronizację wszystkich replik w różnych węzłach sieci bez wiedzy użytkownika i wpływu na jego pracę,
- przeźroczystość fragmentacji, która pozwala na realizację zapytań cząstkowych bez wiedzy użytkownika,
- przeźroczystość awarii, umożliwiająca pracę w systemie w przypadku awarii niektórych węzłów sieci,
- przeźroczystość migracji danych w sieci,
- przeźroczystość wydajności, czyli możliwość dodawania nowych elementów systemu bez wpływu na pracę użytkowników,
- przeźroczystość dostępu, czyli pozbawienie użytkowników informacji na temat położenia danych.

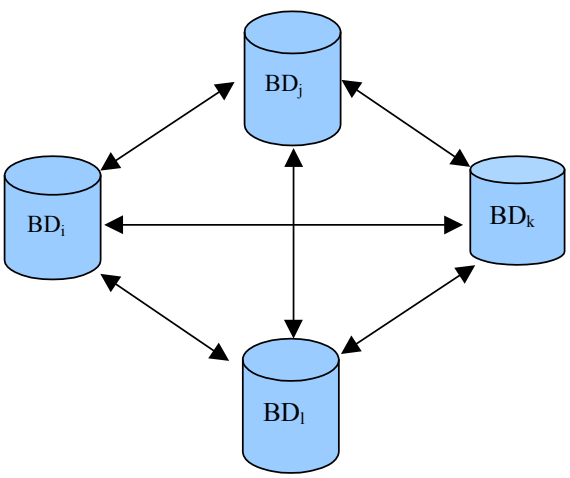
FSZBD powinien zapewniać niezależność danych zarówno logiczną, jak i fizyczną, czyli dostęp do danych na odpowiednich poziomach abstrakcji, gdy niewidoczne są szczegóły implementacji danych. Niezależność danych to inaczej odporność aplikacji na zmiany w strukturze danych i sposobie dostępu do nich. Jest to możliwość przeprowadzenia zmian w jakimś schemacie, znajdującym się na pewnym poziomie sys-

temu, przy jednoczesnym braku konieczności nanoszenia zmian w innym schemacie, który jest umieszczony na następnym poziomie. Wyróżnia się niezależność zarówno fizyczną, jak i logiczną. Dopuszcza się ewolucje schematu bazy danych. Niezależność danych, podobnie jak przeźroczystość, jest cechą bazy idealnej i, podobnie jak przeźroczystość, jest najczęściej osiągalna tylko częściowo przy projektowaniu i implementacji FBD.

Na rysunku 16 przedstawiono system FBD. Udostępnienie danych i usług lokalnych baz danych (LBDi) dla aplikacji globalnych odbywa się przez prosty interfejs, zwany osłoną (O_i) lub wraperem.



Rys. 16. Lokalne bazy dla systemu federacyjnego



Rys. 17. Kolekcja czterech federacyjnych baz danych

Inne istotne cechy FSZBD to: prostota, naturalność, wysoki poziom abstrakcji interfejsów programistycznych, dostęp do informacji poprzez języki zapytań. FBD mogą działać na różnym sprzęcie, w różnych systemach operacyjnych, z różnymi protokołami komunikacyjnymi. Mogą bazować na różnych systemach zarządzania bazami danych, wymieniać pomiędzy sobą niejednorodne dane, podlegające różnym modelom danych, schematom, semantyce czy mechanizmom dostępu.

Na rysunku 17 przedstawiono kolekcję czterech federacyjnych baz danych. Aby można było korzystać z tak powiązanych baz danych, należy utworzyć połączenia 1:1 między wszystkimi parami baz danych.

Połączenia te pozwalają na wysyłanie zapytań z jednego systemu bazy danych BD_i do innego BD_j, wyrażonych za pomocą pojęć zrozumiałych dla BD_j. Przy założeniu, że każda z n baz chce się łączyć z (n–1) bazami, trzeba napisać $n(n–1)$ fragmentów kodów obsługujących zapytania. Oczywiście bazy danych, które są wykorzystywane w FSZBD rządzą się własnymi prawami. Podlegają lokalnym priorytetom, regułom własności, autoryzacji dostępu, bezpieczeństwa i mogą odrzucać zlecenia, które naruszają ich lokalne ograniczenia. Trzeba pamiętać o tym, że zapytania muszą działać poprawnie w tej bazie, do której są skierowane, a niejednokrotnie te same zapytania mają różną postać w zależności od bazy, do której są skierowane.

Przykładowo mamy daną bazę danych producenta komputerów BD_1 o schemacie:

Komputery (*numer*, *procesor*, *pamięć*, *dysk*…);

Monitory (*numer*, *ekran*, *maxrozX*, *maxrozY*);

oraz drugą bazę BD_2, w której komputery i monitory tworzą jeden schemat o nazwie **System**:

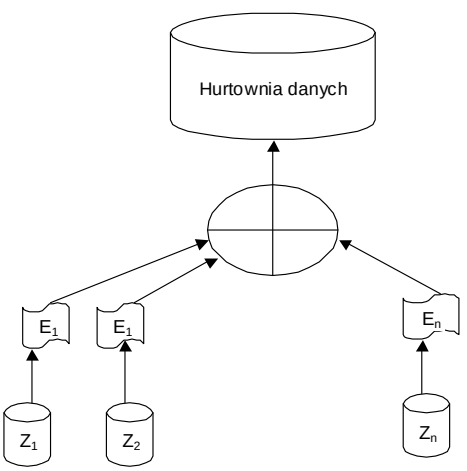
System (*id*, *procesor*, *pamięć*, *dysk*…),

Atrybuty *id*, *pamięć i dysk* z BD_2 odpowiadają atrybutom *numer*, *pamięć*, *dysk* z BD_1, pozostałe nie mają swoich odpowiedników (procesor w jednym przypadku oznacza typ procesora, w drugim częstotliwość zegara). Zapytania SELECT... FROM… WHERE, wysyłane przez producenta z BD_1, muszą być zrozumiałe dla BD_2 i uwzględniać różnice w semantyce atrybutów.

23. Hurtownie baz danych

Hurtownia baz danych (ang. data warehouse) jest rozbudowaną bazą danych, przechowującą olbrzymią ilość danych zbieranych w czasie. Przechowywane dane są tematycznie spójne oraz zintegrowane. Operacje przeprowadzane na danych mają charakter analityczny. Hurtownie najczęściej wykorzystuje się do tworzenia systemów, które wymagają podejmowania decyzji. Twórca pojęcia hurtownie danych W.H. Inmon zdefiniował hurtownie jako *ematyczny, zintegrowany określony w czasie i nie ulotny zbiór danych używanych do wspierania procesu podejmowania decyzji kierowniczych* [4]. Hurtownia powinna mieć następujące własności:

- integrację danych, a więc ujednolicenie danych z różnych źródeł,
- orientację tematyczną, czyli nastawienie na określoną tematykę, a nie obszar zastosowań,
- nieulotność danych – dane umieszczone w hurtowni nie ulegają modyfikacji, służą tylko do odczytu; na ogół nie stosuje się typowych transakcji w bazie,
- orientacje czasową – dane mają przypisany identyfikator czasowy i są poprawne tylko w pewnym określonym przedziale czasowym.



Rys. 18. Hurtownia danych

Hurtownia przedstawiona na rysunku 18 stanowi jeden globalny schemat, zasilany z wielu różnych źródeł danych (Z_1, Z_2, … Z_n). Ich zróżnicowanie powoduje, że konieczne jest przekształcenie danych w jednolite, przejrzyste i wiarygodne źródło informacji, korzystając z pewnych programów zwanych ekstraktorami (E_1, E_2, … E_n), które tłumaczą zapytania i ich wyniki między schematem globalnym i lokalnymi schematami w źródle.

Ekstraktory składają się z:

- zapytania lub wielu zapytań do źródła, czyli do lokalnych baz danych (LBD),
- mechanizmów komunikacji, które umożliwiają przekazywanie danych do hurtowni.

Konstruując zapytania, na ogół korzysta się z języka SQL lub z jakiegokolwiek języka właściwego dla lokalnej bazy danych.

Dla przykładowych baz danych BD_1 i BD_2, opisanych w rozdziale 26, chcemy utworzyć następujący schemat hurtowni danych:

CompMag (*id, procesor, pamięć, dysk, producent)*

Aby utworzyć schemat globalny, należy za pomocą odpowiedniego programu wydobyć dane z dwóch baz. Wydobycie danych z bazy BD_2 **(SYSTEM)** jest bardzo proste, a kod programu jest następujący:

```
INSERT INTO CompMag(id, procesor, pamięć, dysk, producent)
SELECT id, procesor, pamięć, dysk, producent2
FROM SYSTEM;
```

W przypadku bazy pierwszego producenta BD_1 kod programu jest znacznie bardziej złożony, gdyż należy wziąć pod uwagę wiele warunków.

Zestaw operacji wykonywanych na danych pochodzących z różnych źródeł, którymi zasilana jest hurtownia, określa się mianem procedur ETL (ang. Extraction Transformation Loading). Procedura ETL składa się z trzech etapów:

- pobranie danych z ich pierwotnego źródła,
- łączenie danych, ich weryfikowanie, oczyszczanie i znakowanie czasowe,
- wprowadzanie danych do hurtowni.

Istnieje kilka sposobów organizowania hurtowni:

- Systematyczne rekonstruowanie na podstawie aktualnych danych ze źródeł danych. Wadą tej metody jest konieczność wyłączania działania hurtowni na pewien czas oraz przesyłanie pełnej kopii, zawierającej dużą liczbę danych.
- Aktualizacja przyrostowa na podstawie zmian, które nastąpiły w źródłach od chwili ostatniej aktualizacji.
- Natychmiastowa aktualizacja, system reaguje bezpośrednio na zachodzące zmiany. Tego typu organizacja jest bardzo kosztowna i niewygodna.

Pierwszy sposób organizacji jest niezwykle popularny, ma jednak wiele wad. Hurtownia na czas rekonstrukcji nie działa, a długi okres pomiędzy kolejnymi rekonstrukcjami powodują, że dane tracą na wartości. Aktualizacja przyrostowa wymaga kopio-

wania mniejszej liczby danych, co znacznie skraca czas kopiowania, niemniej jednak sam algorytm rekonstrukcji przyrostów informacji jest bardzo złożony. Natychmiastowa aktualizacja wymaga przesyłania dużej liczby komunikatów, sprawdza się więc tylko wtedy, gdy źródła są rzadko aktualizowane.

Ważnym zastosowaniem hurtowni danych jest możliwość obsługi złożonych zapytań, które wymagają na ogół agregowania danych. Te zapytania, zwane inaczej zapytaniami wspomagającymi podejmowanie decyzji lub OLAP – analityczne działania bezpośrednie (ang. On-line Analytic Processing), sprawdzają bardzo dużo danych. Są to bardzo długie transakcje, wymagające dużych przepustowości sieci. Bardzo często blokują całą bazę danych, uniemożliwiając wykonywanie zwykłych operacji – zwanych OLTP (ang. On-line Transaction Processing). Podstawowym zadaniem operacji OLTP jest przechowywanie danych na ogół w bazach relacyjnych. OLTP to stosunkowo duża liczba małych transakcji, które najczęściej modyfikują dane. Zapis i odczyt danych jest zarówno interaktywny, jak i wsadowy. Podstawowe problemy to:

- udostępnianie danych wielu użytkownikom jednocześnie,
- nieprzerwanie utrzymana spójność danych,
- maksymalizacja średniej wydajności.

Analiza danych, które w późniejszym etapie mają być wykorzystane w systemach wspomagających podejmowanie decyzji to zadania OLAP. Odczyt danych jest przeważnie interaktywny, zapis danych tylko wsadowy. Podstawowe problemy przy tego typu operacjach to ogromne ilości danych i analizy wielowymiarowe. Aby uniknąć kolizji między operacjami OLTP i OLAP, najczęściej tworzy się kopię surowych danych w hurtowni, na których uruchamia się zapytania typu OLAP, a w źródłach danych wykonuje się zapytania OLTP i modyfikację danych.

Korzyści wynikające z utworzenia hurtowni to:

- zapewnienie jednolitej, łatwej w zarządzaniu struktury danych, które dotyczą wspomagania decyzji,
- umożliwienie użytkownikom zadawania złożonych zapytań, dotyczących nie jednego, lecz kilku obszarów zastosowań,
- możliwość wykorzystania aplikacji OLAP czy też programów wspomagających eksplorację danych.

Niezwykle ważnym, odnoszącym się do hurtowni zagadnieniem jest kontrola jakości danych oraz spójność danych. Największy problem pojawia się wtedy, gdy mamy do czynienia z danymi pochodzącymi z niezależnych od siebie źródeł o różnym nazewnictwie, z różnie zdefiniowanymi wartościami, czy na przykład z różnie przypisanymi numerami identyfikacyjnymi. Administrowanie hurtownią danych i zapewnienie danym bezpieczeństwa jest dużo bardziej złożone i wymaga odpowiednio większych umiejętności w porównaniu z administrowaniem zwykłą bazą danych. Na ogół zajmuje się tym grupa ekspertów, dysponująca odpowiednią wiedzą i dużym doświadczeniem.

24. Mediatory

Mediator jako baza wirtualna stanowi bardzo ciekawe rozwiązanie problemu korzystania ze wspólnych danych. Mediator, nie posiadając własnych danych, w odpowiedzi na zapytanie musi pozyskać stosowne dane ze źródeł i na ich podstawie sformułować odpowiedź. Mediator stanowi warstwę pośrednią, która oddziela lokalną bazę danych od globalnych aplikacji. Podstawowe zadania mediatorów to:

- określenie wszelkich różnic, jakie zachodzą między schematem lokalnym a schematem globalnym,
- wybór odpowiednich danych metodą selekcji,
- wspomaganie wyszukiwania znaczących cech w danych, które nie zostały sformatowane,
- wspomaganie szeroko pojętego problemu eksploracji danych.

Mediator przekazuje zapytanie do wraperów, które tak jak ekstraktory tłumaczą zapytania i ich wyniki między schematem globalnym i lokalnymi schematami w źródle (rys. 19). Wrapery łączą mediatory ze źródłami. Wrapery, z których korzystają mediatory są na ogół bardziej skomplikowane niż ekstraktory. Wraper musi posiadać umiejętność akceptowania różnego typu zapytań od mediatora i tłumaczenia ich na pojęcia zrozumiałe dla lokalnej bazy danych. Tworzenie wrapera polega na sklasyfikowaniu możliwych zapytań, które przekazywane są poprzez szablony z mediatora. Szablony są to pytania z parametrami reprezentującymi stałe. Mediator dostarcza te stałe, a wraper wykonuje odpowiednie zapytania z tymi stałymi. Innymi słowy, wraper przekształca szablon T na zapytanie S źródła ($T \Rightarrow S$).

Przykładowo mamy dany następujący wraper dla źródła – producent P1, o schemacie:

Komputery (*numer, procesor, pamięć, dysk….)*

Schemat mediatora jest następujący:

CompMed (*numer, procesor, pamięć, dysk, producent1)*

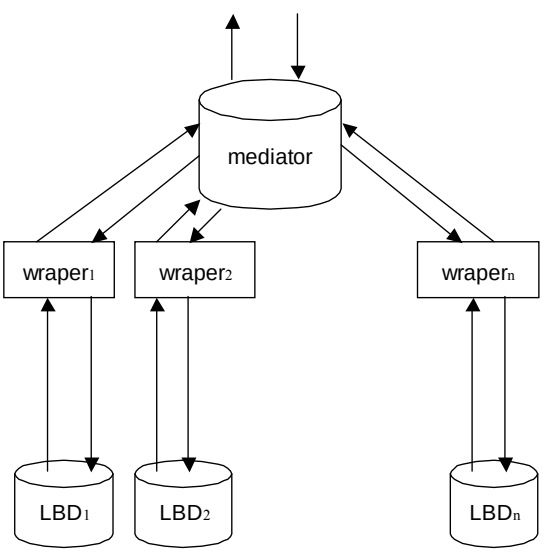
Interesują nas komputery o określonej szybkości. Kod reprezentujący szybkość oznaczymy $f. Szablon będzie wyglądał następująco:

```
SELECT*
FROM CompMed
WHERE szybkość=$f;
```

⇒
SELECT *numer,procesor, pamięć, dysk, producent1*
FROM **Komputery**
WHERE szybkość=”$f”;

Zapytania mogą mieć postać bardziej skomplikowaną:
SELECT*
FROM **CompMed**
WHERE szybkość=”$f” AND dysk=”$d”;
⇒
SELECT *numer,procesor, pamięć, dysk, producent1*
FROM **Komputery**
WHERE szybkość=”$f” AND dysk=”$d”;

Wzorce zapytań zawarte w szablonach oraz źródła zapytań, które są związane z każdym z nich są przechowywane w tabelach, tworzonych przez generatory wraperów. Generatorem wraperów nazywa się oprogramowanie tworzące wrapery. Po pobraniu zapytania od mediatora wyszukiwany jest w tablicy szablon, pasujący do zapytania. Jeżeli znajdzie się taki szablon, to wartości parametru zapytania są używane do utworzenia zapytania do lokalnej bazy danych. Jeśli nie można znaleźć odpowiedniego szablonu, to zapytanie nie jest utworzone. Gdy zachodzi taka potrzeba, wraper przetwarza odpowiedź i dopiero wtedy przekazuje ją do mediatora. Istnieje możliwość stosowania większej liczby zapytań poprzez zastosowanie wrapera filtrującego wyniki zapytań, tak aby do mediatora przekazać tylko informacje potrzebne.



Rys. 19. Mediatory

Wrapery umożliwiają wykonywanie różnego typu operacji, takich jak: projekcja, agregacja czy łączenie. W ten sposób do mediatora przesyłany jest dopiero wynik przeprowadzonej operacji. Na podstawie uzyskanych od wraperów wyników zapytań tworzona jest odpowiedź dla użytkownika. Z mediatorami skojarzone jest również pojęcie perspektywy, czyli tablicy wirtualnej, która zawiera dane z różnych źródeł, fizycznie nie istniejąc. Oczywiście zapytania mogą być przesyłane do perspektywy tak jak do każdej rzeczywistej tablicy. Perspektywa między innymi:

- zwiększa bezpieczeństwo danych,
- zmniejsza złożoność modeli konceptualnych,
- upraszcza zapytania,
- zwiększa ochronę prywatności dzięki zmniejszeniu możliwości dostępu do obiektów,
- umożliwia zapewnienie logicznej niezależności aplikacjom i obiektom,
- wspomaga współpracę między systemami heterogenicznymi poprzez tworzenie wspólnego schematu,
- umożliwia hurtowniom danych przeprowadzanie analiz danych pochodzących z wielu źródeł.

Mediatory w dogodny sposób wspomagają konstruowanie perspektyw wirtualnych. Taka perspektywa wirtualna istnieje tylko jako definicja, a jej wyliczenie następuje w momencie jej użycia. Wynik pracy perspektywy wirtualnej jest kasowany zaraz po jego wykorzystaniu. Dzięki temu unika się dublowania danych, znikają problemy związane z aktualizacjami danych, z transakcjami i ich przetwarzaniem. Do wad natomiast należy zaliczyć czas, potrzebny na utworzenie odpowiednich perspektyw i zapytań, którego poziom, jeśli nie jest optymalizowany, najczęściej jest niedopuszczalnie wysoki. Można by pokusić się o stwierdzenie, że mediator sam jest perspektywą, która jest zdefiniowana na co najmniej jednym źródle danych. Wprawdzie nie posiada on żadnych danych własnych, ale ma możliwość wykonywania określonych operacji w taki sposób, jakby rzeczywiście zawierał dane. Podstawowym zaś zadaniem mediatora jest odwoływanie się do poszczególnych źródeł danych w celu wykonywania określonych operacji, takich jak na przykład wyszukiwanie danych czy też ich modyfikacja.

Można wyróżnić co najmniej dwie metody definiowania mediatorów [11, 12]:

- Podejście LAV (local – as – view) – w metodzie tej wszystkie schematy lokalne definiuje się w postaci widoków nad schematem globalnym. Używa się w tym celu pojęć, które są stosowane właśnie w schemacie globalnym.
- Podejście GAV (global – as – view) – w metodzie tej schemat globalny definiuje się jako widok ponad zbiorem danych schematów lokalnych.

Mediatory odgrywają ogromną rolę w procesie integracji danych. Można powiedzieć, że podstawowym zadaniem mediatora jest zbieranie informacji, które dotyczą pewnego obiektu, z jednego bądź też kilku źródeł danych. Podczas gromadzenia takich informacji usuwa się powtarzające się dane i rozwiązuje się problemy doty-

czące niespójności danych. Wynikiem pracy mediatora jest ujednolicenie semantyki danych znajdujących się w systemie oraz stworzenie translatora operacji, które dotyczą schematu lokalnego i lokalnych schematów źródeł.

25. Analityczne przetwarzanie bezpośrednie

25.1. Cechy przetwarzania analitycznego

Pojęcie OLAP (analityczne działania bezpośrednie) zostało wprowadzone przez Tedda Codda [6] w celu zdefiniowania architektury, która jest w stanie obsłużyć skomplikowane czynności analityczne, takie jak na przykład:

- konsolidację, polegającą na agregowaniu pewnych, wcześniej zagregowanych danych w dane o innych obiektach; na przykład w bazie opisującej wyższe uczelnie agregujemy dane o instytutach i katedrach w inne obiekty o nazwie wydziały, a te z kolei agregujemy w dane o uczelniach,
- drążenie, czyli operację odwrotną do konsolidacji, która polega na przedstawianiu jak największej ilości szczegółowych danych,
- obracanie, czyli czynność pozwalającą na przeanalizowanie tych samych danych w kontekście różnych sytuacji, najczęściej na osi czasu,
- obsługę wielu użytkowników równocześnie w zakresie dostępu do danych,
- zapewnienie wewnętrznej wymiarowości, czyli sprawienie, aby wszystkie wymiary w bazie danych miały swój odpowiednik w strukturze oraz w możliwościach działania.

W technice OLAP posługujemy się narzędziami, które powinny spełniać określone reguły, takie jak:

- przezroczystość, czyli niewidoczność dla użytkownika,
- dostępność, czyli możliwość korzystania z danych zawartych w źródłach w różnych formatach,
- stała efektywność raportowania, czyli użytkownicy nie powinni odczuwać spadku efektywności pracy systemu w razie takich zmian jak chociażby powiększanie się rozmiarów baz danych,
- architektura klient–serwer,
- elastyczność raportowania, wygląd danych wygodny dla użytkownika,
- wielowymiarowa perspektywa koncepcyjna; narzędzia OLAP powinny posługiwać się łatwym w użyciu, wielowymiarowym modelem, który niejako obrazowałby, w jaki sposób firma widziana jest przez użytkowników,

• nieograniczoność wymiarów i poziomów agregacji; żadne narzędzie OLAP nie może mieć wpływu na liczbę wymiarów w modelu analitycznym.

Technika OLAP znalazła zastosowanie przede wszystkim w hurtowniach danych.

Przetwarzanie analityczne powinno spełniać następujące warunki [12]:

• analiza olbrzymiej ilości danych dla dużej liczby użytkowników powinna być procesem efektywnym,

• sposób przechowywania danych nie może mieć wpływu na ich późniejszą prezentację,

• obsługa zapytań czy wszelkiego rodzaju obliczeń musi następować w sposób niezwykle szybki,

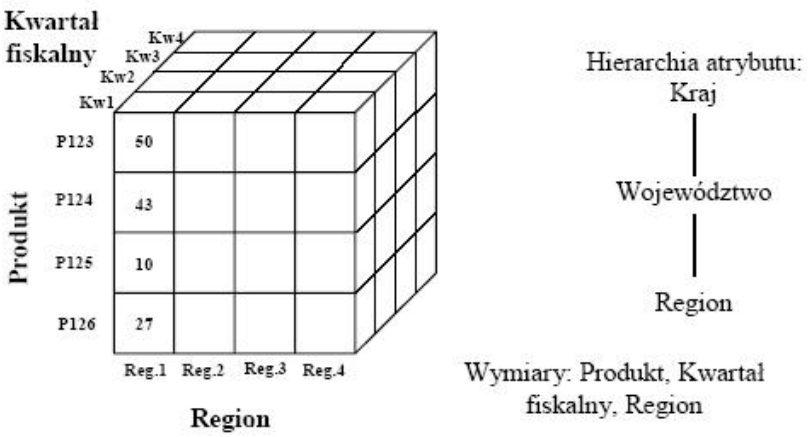
• powinna istnieć możliwość wykonywania takich operacji, jak: agregacja, prognozowanie, wykonywanie obliczeń statystycznych, obsługa operacji wielowymiarowych,

• powinno być możliwe bezproblemowe przedstawianie wyników analiz w różnych formach, takich jak raporty czy też arkusze kalkulacyjne.

Aby można było spełnić te wszystkie warunki, konieczne jest stosowanie odpowiednich, wielowymiarowych struktur danych.

25.2. Struktury wielowymiarowe

Podstawowym elementem struktury wielowymiarowej jest fakt. Fakty z kolei opisuje się tak zwanymi miarami, które zazwyczaj są atrybutami liczbowymi. Zbiór faktów tworzy tabelę faktów. Tabela ta reprezentuje zdarzenia lub obiekty. Obiekty



Rys. 20. Dane zorganizowane jako kostka wielowymiarowa [1]

są osadzone w przestrzeni wielowymiarowej, np. trójwymiarowej, reprezentowanej jako punkty w kostce sześciennej, gdzie jednym z wymiarów może być na przykład czas. Takie podejście niesie za sobą konieczność wprowadzenia i używania drugiego obok faktów rodzaju tabeli: tabeli wymiarów. Tabele wymiarów zbudowane są z krotek atrybutów odpowiednich wymiarów. Tabele faktów natomiast można sobie wyobrażać jako pewną strukturę, która zawiera w sobie po jednej krotce dla każdego z faktów zarejestrowanych w hurtowni danych. Każdy fakt zawiera zmienne, które rozpoznaje za pomocą wskaźników do odpowiednich tabel wymiarów. Tabela faktów zawiera takie informacje, które dają możliwość zidentyfikowania wszystkich krotek w odpowiednich wymiarach.

Dane wielowymiarowe poddaje się kilku operacjom, takim jak [15]:

- obracanie – można zmienić perspektywę, z której widzi się dane,
- selekcja – można wybrać tylko te elementy wymiarów, które są dla nas istotne, a pozostałe pominąć,
- projekcja, czyli zredukowanie liczby wymiarów i zaprezentowanie zagregowanych danych z usuniętych wymiarów w pozostałych wymiarach,
- wycinanie, czyli połączenie operacji selekcji oraz projekcji, co pozwala na przeprowadzenie projekcji tylko dla wybranych elementów odpowiednich wymiarów,
- ranking, czyli szeregowanie elementów wymiaru według odpowiednich wytycznych,
- zwijanie oraz rozwijanie, czyli nawigacja po hierarchii wymiarów, przy czym zwijanie polega na uogólnianiu poziomów w tej hierarchii, rozwijanie natomiast na jej uszczegółnianiu.

Interpretacja danych wielowymiarowych prowadzi do wyodrębnienia dwóch typów pojęć:

ROLAP – relacyjne systemy OLAP,

MOLAP – wielowymiarowe systemy OLAP.

W relacyjnych systemach ROLAP dane są przechowywane w relacjach o specjalnej strukturze, nazwanej schematem typu gwiazda lub płatek śniegu. Jedną z tych relacji jest tabela faktów, zawierająca tak zwane dane surowe, czyli niezagregowane. Inne relacje zawierają informacje o wartościach uporządkowanych wzdłuż każdego wymiaru. Dla ROLAP jest specjalnie dobrany język zapytań, który pozwala na:

- przechowywanie ogromnych ilości danych,
- występowanie złożonych struktur danych, spowodowane tym, że zależności wielowymiarowe muszą w jakiś sposób być odwzorowane relacyjnie,
- stosunkowo niską wydajność, której główną przyczyną jest to, że struktury relacyjne nie są w dogodny sposób dostosowane do analiz wielowymiarowych,
- dość łatwą modyfikację danych.

Największe wady tego typu systemów to mała wydajność języka zapytań i czas, jaki jest potrzebny na uzyskanie odpowiedzi. Systemy te są jednak bardzo często wykorzystywane przy budowie hurtowni danych, głównie ze względu na to, że mogą przechowywać niezwykle duże ilości danych.

W systemach MOLAP dane są przechowywane w specjalizowanej, często zagregowanej strukturze. Struktura ta zwana jest formalną kostką danych, w odróżnieniu od kostki danych surowych. Podstawowe cechy systemów MOLAP to:

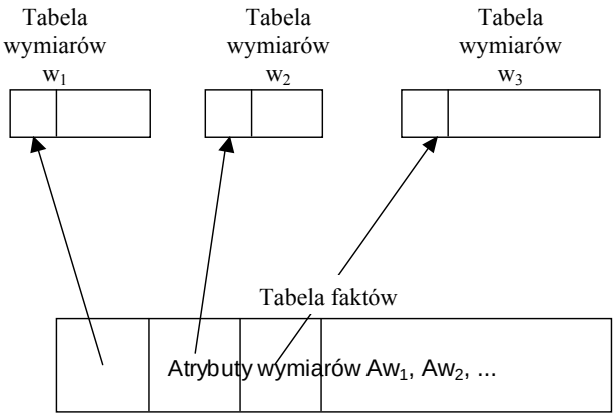
- mniejsze możliwości w zakresie ilości przechowywanych danych,
- reprezentowanie wielowymiarowych struktur w sposób naturalny,
- wysoki poziom wydajności przeprowadzanych analiz wielowymiarowych,
- stosunkowo trudna do realizowania modyfikacja danych, głównie z powodu stosowania struktur wielowymiarowych.

Systemy MOLAP wykorzystuje się najczęściej jako składnice danych lub struktury pomocnicze dla hurtowni danych. Idealnym rozwiązaniem byłoby połączenie architektury ROLAP i MOLAP w taki sposób, aby umożliwiało ono przechowywanie ogromnych ilości danych oraz przeprowadzanie efektywnych analiz wielowymiarowych. Relacyjna baza danych służyłaby do przechowywania zarówno historycznych, jak i elementarnych danych, podczas gdy zadaniem serwerów wielowymiarowych byłoby przechowywanie i przetwarzanie roboczych zbiorów danych [6].

Modele wielowymiarowe danych, zwane kostkami, w rzeczywistości są macierzami wielowymiarowymi. Jeśli macierz taka ma więcej niż trzy wymiary, to wtedy określa się ją terminem hiperkostki. Kostka danych jest skonstruowana w ten sposób, że można ją dzielić wzdłuż każdego wymiaru na różnym poziomie granulacji, czyli każda komórka w kostce danych powstaje w wyniku przecięcia wszystkich jej wymiarów. Taki sposób ułożenia danych w strukturze wielowymiarowej zwiększa wydajność zapytań w porównaniu z relacyjnym modelem danych. Jeżeli jednym z wymiarów jest czas, to – dzieląc go na lata, miesiące czy dni – uzyskamy wszystkie informacje z tego okresu. Wybór sposobu dzielenia dla każdego wymiaru zależy od tego, jak szczegółowe informacje chcemy uzyskać. W efekcie podziału uzyskujemy mniejsze kostki, zawierające informacje podobne do uzyskanych przez klauzulę języka SQL znaną pod nazwą GROUP BY. Przy pomocy klauzuli WHERE zapytanie ma możliwość ograniczenia do wybranych przedziałów wzdłuż jednego lub więcej wymiarów. Zapytanie stanowi podstawowy blok kwalifikacyjny języka SQL, czyli SELECT…FROM…WHERE z możliwością zastosowania dodatkowych klauzul.

Ogólna postać zapytania, nazwanego krojeniem i kratkowaniem (ang. sliping and dicing):

SELECT	*atrybuty grupujące i agregacja*
FROM	*tabela faktów powiązana przez klucze obce z tabelami wymiarów*
WHERE	*ograniczenia do wybranych podziałów wzdłuż jednego lub więcej wymiarów*
GROUP BY	*atrybuty grupujące*



Rys. 21. Atrybuty wymiarów z tabeli faktów wskazują klucze w tabelach wymiaru

W relacyjnych systemach danych występują wspomniane już schematy typu gwiazda oraz schemat płatka śniegu. Centrum gwiazdy stanowi tabela faktów, która zawiera odpowiedni zestaw faktów. Tabela ta jest powiązana z innymi, wspomnianymi wyżej tabelami wymiarów. Tabele wymiarów opisują wartości faktów wzdłuż każdego wymiaru. Nawiązując do baz relacyjnych, każdy atrybut wymiaru Aw_i w tabeli faktów jest kluczem obcym (rys. 21), odwołującym się do odpowiedniej tabeli wymiarów W_1, W_2, ..., W_n.

W schemacie gwiazdy wszystkie tabele wymiarów są połączone z tabelą faktów relacjami typu „jeden do wielu". Oznacza to, że każda krotka danego wymiaru jest połączona nie z jedną, lecz z kilkoma krotkami faktów. Klucz główny tabeli faktów natomiast tworzony jest przez połączenie kluczy głównych wszystkich tabel wymiarów. Często zdarza się tak, że tabele faktów mają duże rozmiary, w przeciwieństwie do tabel wymiarów, ponieważ niemal wszystkie dane w hurtowniach danych reprezentowane są właśnie przez fakty.

Podstawowe cechy charakterystyczne schematu gwiazdy to:

- prostota struktury,
- duża efektywność zapytań dzięki małej liczbie połączeń pomiędzy tabelami,
- dość długi czas potrzebny na załadowanie danych do tabel wymiarów,
- możliwość dogodnej współpracy z hurtowniami danych dzięki istnieniu wielu narzędzi wspomagających.

Schemat płatka śniegu natomiast można uznać za odmianę schematu gwiazdy. Podstawowa różnica polega na tym, że tabele wymiarów są tu tak znormalizowane, aby tworzyły hierarchię [5, 6]. Schemat płatka śniegu charakteryzuje się przede wszystkim:

- spadkiem wydajności zapytań w odniesieniu do schematu gwiazdy (większa liczba połączeń pomiędzy tabelami),
- strukturą bardziej sprzyjającą wprowadzaniu modyfikacji,

• krótkim czasem potrzebnym na załadowanie danych do tabel wymiarów.

Ze względu na małą efektywność zapytań częściej jest stosowany schemat gwiazdy.

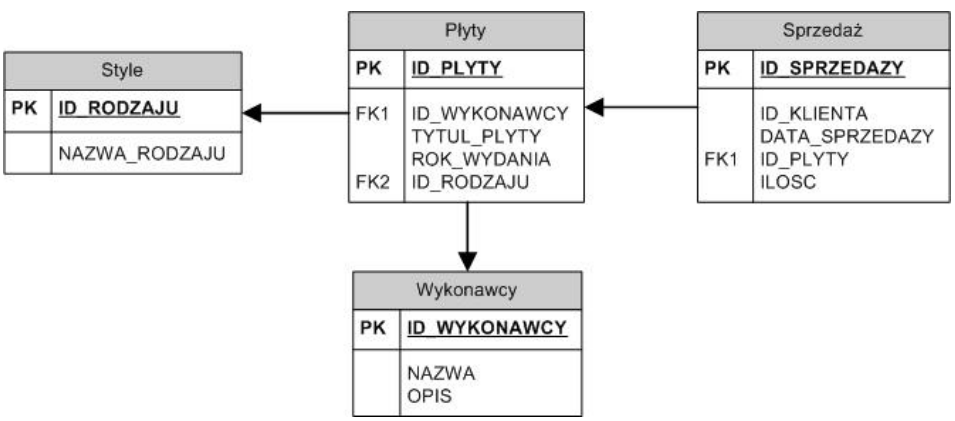
Dzięki modelom wielowymiarowym możliwe jest stosowanie bardzo złożonych zapytań, które dotyczą ogromnych ilości informacji. Modele te pozwalają utrzymywać integrację środowiska i tworzyć wszelkiego rodzaju podsumowania danych z uwzględnieniem wielu wymiarów.

25.3. Implementacja kostek za pomocą perspektyw zmaterializowanych

Komercyjne systemy kostek danych mogą pomagać użytkownikowi w określeniu perspektyw zmaterializowanych kostki danych. Perspektywa zmaterializowana jest to wynik pewnego zapytania, który chcemy na trwale zapamiętać w bazie danych. Zazwyczaj perspektywy te są typowymi agregatami kostki. Aby można zwiększyć użyteczność takiej perspektywy, muszą być one dość duże, aby zgromadzenie wymiarów było dostatecznie szczegółowe.

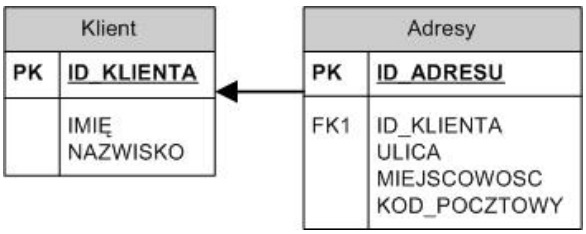
Przykład perspektyw zmaterializowanych

Pierwszą z baz danych wykorzystanych w przykładzie jest baza „Sklep". Ogólnie można powiedzieć, że jest to baza, która zawiera informacje na temat posiadanych przez sklep płyt muzycznych oraz szczegółowe informacje dotyczące samej sprzedaży.



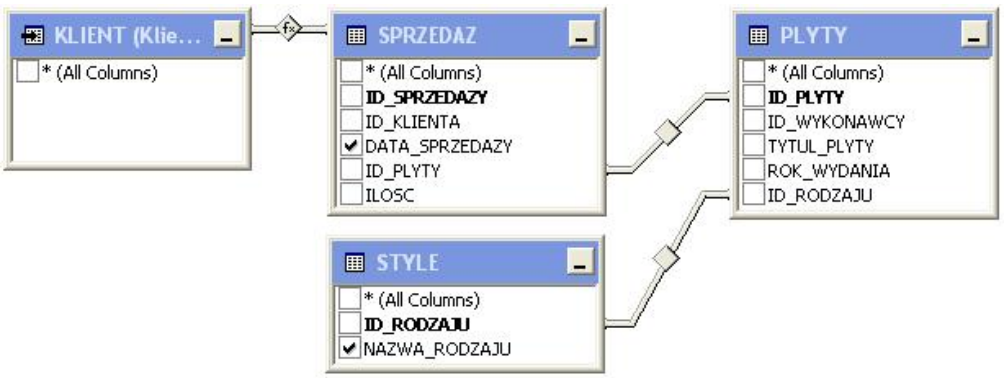
Rys. 22. Diagram związków encji bazy „Sklep"

Drugą z baz wykorzystywanych w przykładzie jest baza „Klienci". Jest to prosta baza, zbudowana jedynie z dwóch tabel, przechowująca podstawowe dane dotyczące klientów sklepu.



Rys. 23. Diagram związków encji bazy „Klienci”

Widok pierwszy został zaimplementowany w taki sposób, aby połączyć obie wykorzystywane w projekcie bazy danych. Jego zadaniem jest takie przefiltrowanie odpowiednich tabel obu baz danych, aby można było uzyskać informacje o imieniu oraz nazwisku klienta, stylu muzycznym, do którego zaliczana jest kupiona przez niego płyta oraz dacie, która powie nam, kiedy dokonano sprzedaży.

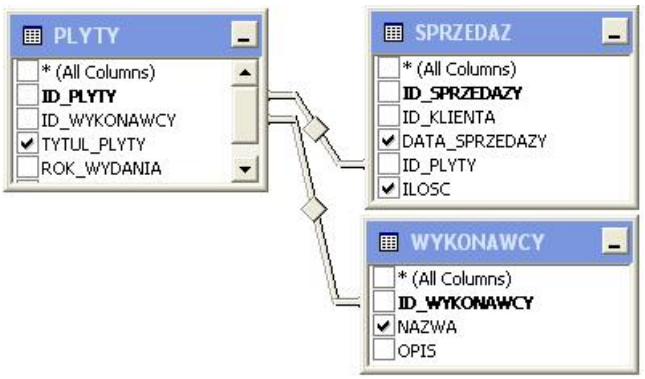


Rys. 24. Diagram widoku pierwszego zmaterializowanego

Żądany widok powstał za pomocą następującego kodu:

```
SELECT TOP (100) PERCENT KLIENT.DBO.KLIENT.IMIE, KLIENT.DBO.KLIENT.NAZWISKO,
DBO.STYLE.NAZWA_RODZAJU, DBO.SPRZEDAZ.DATA_SPRZEDAZY
FROM KLIENT.DBO.KLIENT INNER JOIN
        DBO.SPRZEDAZ ON KLIENT.DBO.KLIENT.ID_KLIENTA =
DBO.SPRZEDAZ.ID_KLIENTA INNER JOIN
        DBO.PLYTY ON DBO.SPRZEDAZ.ID_PLYTY = DBO.PLYTY.ID_PLYTY INNER JOIN
        DBO.STYLE ON DBO.PLYTY.ID_RODZAJU = DBO.STYLE.ID_RODZAJU
ORDER BY KLIENT.DBO.KLIENT.NAZWISKO
```

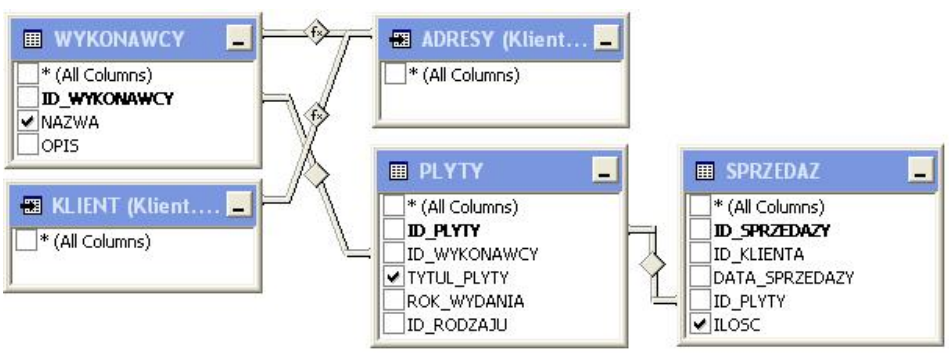
Widok drugi z kolei został zaimplementowany jedynie na bazie „Sklep”. Celem jego utworzenia było wybranie z tej bazy informacji o liczbie poszczególnych płyt każdego wykonawcy, które zostały sprzedane w sklepie oraz o dacie tejże sprzedaży.



Rys. 25. Diagram widoku drugiego zmaterializowanego

Kod, który pozwolił na utworzenie tego widoku wygląda następująco:

```
SELECT DBO.WYKONAWCY.NAZWA, DBO.PLYTY.TYTUL_PLYTY,
                DBO.SPRZEDAZ.DATA_SPRZEDAZY, DBO.SPRZEDAZ.ILOSC
FROM DBO.PLYTY INNER JOIN
    DBO.WYKONAWCY ON DBO.PLYTY.ID_WYKONAWCY =
    DBO.WYKONAWCY.ID_WYKONAWCY
    INNER JOIN
    DBO.SPRZEDAZ ON DBO.PLYTY.ID_PLYTY = DBO.SPRZEDAZ.ID_PLYTY
```



Rys. 26. Diagram widoku trzeciego zmaterializowanego

Widok trzeci, podobnie jak widok pierwszy, został zaimplementowany w taki sposób, aby łączył dwie utworzone na potrzeby przykładu bazy. Widok ten pozwala uzyskać informacje na temat tego, jakie ilości poszczególnych płyt poszczególnych wykonawców sprzedano w różnych miejscowościach.

Kod, który pozwolił na utworzenie tego widoku wygląda natomiast tak:

```
SELECT KLIENT.DBO.ADRESY.MIEJSCOWOSC, DBO.WYKONAWCY.NAZWA,
    DBO.PLYTY.TYTUL_PLYTY, DBO.SPRZEDAZ.ILOSC
FROM DBO.WYKONAWCY INNER JOIN
    DBO.PLYTY ON DBO.WYKONAWCY.ID_WYKONAWCY =
DBO.PLYTY.ID_WYKONAWCY INNER JOIN
    DBO.SPRZEDAZ ON DBO.PLYTY.ID_PLYTY = DBO.SPRZEDAZ.ID_PLYTY
INNER JOIN
    KLIENT.DBO.ADRESY INNER JOIN
    KLIENT.DBO.KLIENT ON KLIENT.DBO.ADRESY.ID_KLIENTA =
    KLIENT.DBO.KLIENT.ID_KLIENTA ON
    DBO.SPRZEDAZ.ID_KLIENTA = KLIENT.DBO.KLIENT.ID_KLIENTA
```

Istnieje możliwość przyspieszenia wielu zapytań przez zastosowanie specjalizowanego operatora kostki CUBE, który służy do wstępnego agregowania tabeli faktów wzdłuż wszystkich podzbiorów wymiarów. Mocniejszym sposobem jest utworzenie siatki ziarnistości do agregowania wzdłuż wszystkich wymiarów. Siatka ta nosi nazwę kraty perspektyw. Umieszczanie zapytań w kracie perspektyw pomaga przy projektowaniu baz danych opartych na kostkach danych. W procesie projektowania najpierw ustala się zbiór zapytań, jakie są przewidywane do określonych zastosowań. Następnie określa się zbiór perspektyw do zmaterializowania.

Podsumowanie

Idealnym rozwiązaniem jest możliwość uzyskania potrzebnych informacji, bez konieczności projektowania baz danych dla konkretnych zastosowań. Prowadzi to do korzystania z wirtualnej bazy danych, która fizycznie nie istnieje, ale stanowi filtr nałożony na różne źródła danych. Problemem jest niestety różnorodność źródeł pod względem semantyki danych, typów, wartości, schematów. Istnieje konieczność tworzenia translatorów, które ujednoliciłyby dane w różnych źródłach i zadbały o ich spójność. Innym sposobem jest tworzenie zagregowanych perspektyw, będących odpowiedziami na zapytania do bazy. Perspektywy te można trwale przechowywać w bazie, nie rekonstruując ich w razie potrzeby.

Bibliografia

[1] Bembenik R., Jędrzejec B., *Technologia hurtowni danych, transformacja i integracja danych w systemach sieciowych*, http: //www.zsi.pwr.wroc.pl./zsi/missi2002/pdf/ s. 2007 pdf.
[2] Benon-Davies P., *Systemy baz danych*, WNT, Warszawa 1998.
[3] Chang B., Scardina M., Kiritzov S., *Oracle 9i i XML*, Wydawnictwo Helion, Gliwice 2003.
[4] Elmasri R., Navathe S., *Wprowadzenie do systemów baz danych*, Wydawnictwo Helion. Gliwice 2003.
[5] Garcia-Molina H., Ullman J., Widom J., *Implementacja systemów baz danych*, WNT, Warszawa 2003.
[6] Garcia-Molina H., Ullman J., Widom J., *Systemy baz danych. Pełny wykład*, WNT, Warszawa 2006.
[7] Greene J., *Oracle8 Server. Księga eksperta*, Wydawnictwo Helion, Gliwice 2000.
[8] Gupta A., Mumick I., *Materialized Views: Techniques, Implementations and Applications*, MIT Press, Cambridge MA, 1999.
[9] Loney K., *Oracle Database 10g. Kompendium administratora*, Wydawnictwo Helion, Gliwice 2005.
[10] Loney K., Theriault M., *Oracle 8i. Podręcznik administratora baz danych*, Wydawnictwo Helion, Gliwice 2003.
[11] Pankowski T., Stokłosa J., Bilski T., *Bezpieczeństwo danych w systemach informatycznych*, PWN, Warszawa 2001.
[12] Pankowski T., *Dane wielowymiarowe i język MDX w systemach OLAP*, VI Konferencja PLOUG, Zakopane 2000.
[13] Poe V., Klauer P., *Brobst. Tworzenie hurtowni danych*, WNT, Warszawa 2000.
[14] Theriault M., Carmichael R., Viscusi J., *Oracle 9i. Administrowanie bazami danych od podstaw*, Wydawnictwo Helion, Gliwice 2003.
[15] Traczyk T., *Hurtownie danych – wprowadzenie*, http: //www.ia.pw.edu.pl/~ttraczyk/pdf/intotest98_art.pdf.
[16] Urman S., *Oracle 8. Programowanie w języku PL/SQL*, Wydawnictwo Helion, Gliwice 2002.
[17] Whalen E., Schroeter M., *Oracle. Optymalizacja wydajności*, Wydawnictwo Helion, Gliwice 2003.
[18] http://www.ploug.org.pl/szkola/szkola_4/materialy/02_MW_logiczne.pdf
[19] http://www.e-informatyka.pl/article/show-bw/430

Uwagi końcowe

Ogromne koszty związane z utratą danych, a co za tym idzie – z przestojami systemu spowodowały, że w każdym rodzaju działalności istotne jest zaplanowanie właściwej strategii i technologii, gwarantującej bezpieczeństwo przechowywanych, przesyłanych i modyfikowanych danych. Różnego rodzaju badania i analizy wykazały, że jednym z najsłabszych ogniw systemu jest czynnik ludzki. Ponad 32% przyczyn przestoju systemu to umyślny i nieumyślny błąd człowieka. Tylko awaria sprzętu lub systemu, około 44% przyczyn przestoju systemu, jest groźniejsza. Pozostałe przyczyny, takie jak błędy oprogramowania, wirusy komputerowe i inne stanowią zaledwie 24%. Powszechny dostęp do Internetu, szybki transfer danych i zaawansowane technologie przetwarzania danych spowodowały, że kontrola przenoszonych informacji stała się praktycznie niemożliwa. Cała nadzieja w bezpiecznych protokołach przesyłania danych, limitowaniu dostępu do określonych danych, autoryzowaniu wglądu do plików w zabezpieczających hasłach o często bardzo skomplikowanej budowie, wykorzystujących szyfrowanie danych.

Nie istnieją niezawodne systemy ani urządzenia. Zawsze może wystąpić nieprzewidziana awaria, która nie tylko wstrzyma działanie systemu, uszkodzi program, ale może także spowodować częściową, a nawet całkowitą utratę danych. Dlatego też, korzystając z gotowych systemów czy projektując nowe, trzeba odpowiedzieć na pytania: co należy chronić? przed czym? i czy warto ponieść koszty ochrony? Trzeba zatem oszacować ryzyko.

Pierwszym elementem analizy ryzyka jest określenie, jakie materialne i niematerialne zasoby systemu powinniśmy chronić. Następnie ustala się hierarchię ważności tych danych i określa dane strategiczne. Drugi etap to rozpoznanie zagrożeń. Opracowuje się listę zjawisk zarówno prawdopodobnych, jak i bardzo mało prawdopodobnych. Kalkulacja ryzyka polega na określeniu prawdopodobieństwa wystąpienia tych zagrożeń. Po oszacowaniu ryzyka wiemy już, co chronić, przed czym i czy warto. Chodzi o to, aby nakłady na koszty ochrony nie były większe od spodziewanych korzyści. Wyliczenie kosztów i zysków jest zadaniem bardzo trudnym, a często wręcz niemożliwym, gdyż niektórych kosztów nie jesteśmy w stanie przewidzieć, a inne są niewymierne. Różnica między kosztami poniesionymi w wyniku strat, a kosztami ochrony i zabezpieczeń określa uzyskany zysk.

Ochrona i bezpieczeństwo danych, aby spełniały swoją rolę, powinny być przeprowadzane na kilku płaszczyznach równocześnie. Powinny dotyczyć:

- archiwizacji danych,
- ochrony antywirusowej,
- zabezpieczeń przed utratą zasilania,
- ochrony poufności przechowywanych i przesyłanych danych,
- zapewnienia ciągłości pracy systemu informatycznego.

Nie ma idealnych systemów, nie ma niezawodnych systemów i nie ma nieomylnych ludzi. Dlatego kluczowym elementem poprawnego funkcjonowania systemów jest opracowanie i zaimplementowanie zbioru zasad, norm i procedur postępowania w celu ochrony przed ujawnieniem, nieuprawnioną modyfikacją oraz zniszczeniem informacji, czyli opracowanie i zaimplementowanie właściwej polityki bezpieczeństwa.

Spis rzeczy