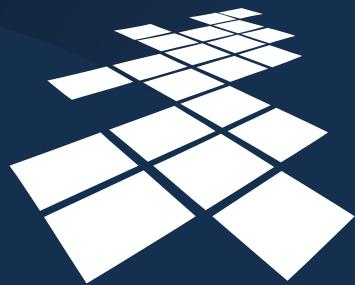


Wprowadzenie do problematyki baz danych

Wykład przygotował:
Robert Wrembel



**UCZELNIA
ONLINE**

BD – wykład 1 (1)

Niniejszy cykl 13 wykładów będzie poświęcony bazom danych.



Plan wykładu

- Podstawowa terminologia
- Charakterystyka baz danych
- Modele danych
- Użytkownicy baz danych
- System zarządzania bazą danych (SZBD)
- Klasyfikacja baz danych

BD – wykład 1 (2)

Celem pierwszego wykładu jest ogólne wprowadzenie do problematyki baz danych.
Zostaną tu omówione:

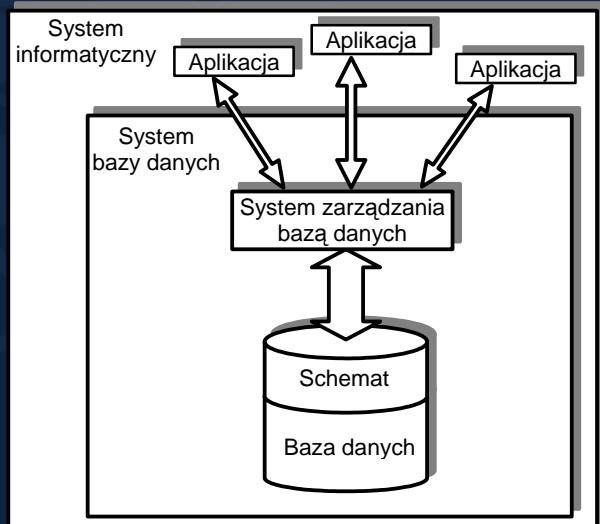
- podstawowa terminologia
- charakterystyka baz danych
- wymagania stawiane bazom danych
- cechy technologii baz danych
- cechy systemu zarządzania bazą danych
- wprowadzenie do modeli danych
- charakterystyka użytkowników baz danych
- charakterystyka sposobów korzystania z bazy danych
- architektury: wewnętrzna i komunikacyjna baz danych
- ogólny podział baz danych



Terminologia



Użytkownicy



BD – wykład 1 (3)

Zbiór danych opisujący pewien wybrany fragment rzeczywistości będziemy nazywać bazą danych. Przykładowo, bazą danych może być zbiór danych banku na temat klientów, ich rachunków, operacji na rachunkach, udzielanych kredytach.

Dane w bazie danych posiadają dwie podstawowe cechy. Po pierwsze, odzwierciedlają rzeczywistość w sposób z nią zgodny (prawidłowy). Po drugie, są zorganizowane w specyficzny sposób, zgodnie z tzw. modelem danych (model danych zostanie omówiony w dalszej części kursu).

Struktura danych i powiązania między nimi są opisane przez tzw. schemat bazy danych.

Baza danych jest zarządzana przez tzw. system zarządzania bazą danych, w skrócie SZBD. Funkcje oferowane przez SZBD zostaną omówione w dalszej części wykładu.

SZBD i bazę danych będziemy dalej nazywać systemem bazy danych.

Z systemem bazy danych współpracują programy użytkowników, zwane aplikacjami. Zadaniem tych programów jest przetwarzanie danych, tj. wstawianie nowych danych, modyfikowanie danych już istniejących, usuwanie danych nieaktualnych, wyszukiwanie danych.

Wszystkie omówione wyżej komponenty (tj. baza danych, SZBD i aplikacje) wchodzą w skład tzw. systemu informatycznego.



Charakterystyka baz danych (1)

1. Trwałość danych

- Długi czas życia – kilka, kilkadziesiąt, kilkaset lat
- Niezależność od działania aplikacji

2. Rozmiar wolumenu danych

- Dane nie mieszczą się w pamięci operacyjnej – wymagana pamięć zewnętrzna
- Danych jest zbyt dużo dla ich liniowego przeglądania przez użytkowników

BD – wykład 1 (4)

Do głównych cech charakteryzujących bazę danych zalicza się: trwałość danych, rozmiar wolumenu danych i złożoność danych.

Trwałość danych oznacza, że dane przechowywane w bazie danych nie są ulotne. W konsekwencji, okres przechowywania danych jest ograniczony wyłącznie okresem żywotności nośnika danych. "Czas życia" danych, po ich zapisaniu do bazy danych jest niezależny od istnienia i działania bądź niedziałania aplikacji. Trwałość danych jest również niezależna od platformy sprzętowo-programowej.

W ogromnej większości zastosowań, dane zgromadzone w bazie danych nie mieszczą się w pamięci operacyjnej, więc do ich składowania jest wymagana pamięć zewnętrzna (dyskowa, optyczna, taśmowa). Tak duże ilości danych nie mogą być przeglądane liniowo ze względu na niewielką efektywność tej techniki. W konsekwencji konieczne jest wykorzystanie innych zaawansowanych mechanizmów efektywnego dostępu do danych.



Charakterystyka baz danych (2)

3. Złożoność danych

- Złożoność strukturalna i złożoność zależności pomiędzy danymi
- Złożoność semantyczna
- Ograniczenia integralnościowe

BD – wykład 1 (5)

Dane gromadzone w bazie danych często są złożone ze względu na:

- złożoność ich struktur i zależności pomiędzy danymi (np. projekt samochodu, złożony z tysięcy elementów),
- złożoność semantyczną danych (np. fakt przyznania kredytu mieszkaniowego jest uzależniony od spełnienia lub niespełnienia wielu wymagań przez petenta).

Ponadto, na dane są często nakładane tzw. ograniczenia integralnościowe gwarantujące, że w bazie danych pojawią się wyłącznie dane spełniające te ograniczenia. Ograniczenia takie mogą być również bardzo złożone.



Wymagania (1)

- Spójność bazy danych
- Efektywne przetwarzanie danych
- Poprawne modelowanie świata rzeczywistego
- Autoryzacja dostępu do danych
- Współbieżność dostępu do danych
- Metadane

BD – wykład 1 (6)

Bazie danych stawia się 6 podstawowych wymagań.

Po pierwsze, musi ona gwarantować spójność danych.

Po drugie, musi zapewniać efektywne przetwarzanie danych.

Po trzecie, musi poprawnie odzwierciedlać zależności w świecie rzeczywistym, który baza danych reprezentuje.

Po czwarte, musi chronić przed nieautoryzowanym dostępem.

Po piąte, musi zapewniać współbieżny dostęp do danych wielu użytkownikom.

Po szóste, musi udostępniać tzw. metadane.



Wymagania (2)

1. Spójność bazy danych

- Poprawność danych z punktu widzenia przyjętych kryteriów
 - wierne odzwierciedlenie danych rzeczywistych
 - spełnienie ograniczeń nałożonych przez użytkowników

BD – wykład 1 (7)

Spójność bazy danych jest definiowana jako poprawność danych z punktu widzenia pewnych przyjętych kryteriów. Definiując spójność wymienia się trzy takie kryteria, tj.:

- wierne odzwierciedlenie danych rzeczywistych,
- spełnienie ograniczeń nałożonych przez użytkowników,
- brak anomalii wynikających ze współbieżnego dostępu do danych.

Kryterium pierwsze należy interpretować następująco: dane przechowywane w bazie danych są takie jak w świecie rzeczywistym, który ta baza reprezentuje. Przykładowo, w bazie danych dziekanatu Wydziału Informatyki i Zarządzania PP są przechowywane dane tylko tych studentów, którzy kiedykolwiek ukończyli studia i dane tylko studentów aktualnie studiujących. Innymi słowy, baza ta nie zawiera danych studentów nieistniejących. Ponadto, zależności między danymi wiernie odzwierciedlają zależności pomiędzy obiektami świata rzeczywistego. Przykładowo, grupa studencka G1 w bazie danych dziekanatu składa się ze studentów należących do tej grupy w świecie rzeczywistym.

Kryterium drugie oznacza, że wszystkie dane w bazie, na które nałożono pewne ograniczenia integralnościowe muszą te ograniczenia spełniać. Przykładowo, w bazie danych dziekanatu zdefiniowano ograniczenie na możliwe wartości oceny i określono, zbiór dozwolonych ocen jako $\{2, 3, 4, 5\}$. Oznacza to, że w bazie danych nie pojawi się żadna inna ocena niż dozwolona.



Wymagania (3)

- Spójność bazy danych cd.
 - Odporność na anomalie będące wynikiem współbieżności dostępu do baz danych
 - Odporność na błędy, awarie i inne anormalne sytuacje wynikające z zawodności środowiska sprzętowo-programowego
 - Odporność na błędy użytkowników

BD – wykład 1 (8)

W przypadku baz danych, z których korzysta przynajmniej dwóch użytkowników mogą powstać dane niepoprawne na skutek równoczesnego modyfikowania tego samego zbioru danych. Baza danych musi być odporna na takie sytuacje niepoprawne.

Spójność to również poprawność danych w przypadku awarii sprzętowo-programowych. W sytuacji wystąpienia awarii, w bazie danych nie mogą powstać dane niepoprawne. Ponadto, żadne dane nie mogą zostać utracone.

Baza danych powinna być również odporna na przypadkowe błędy użytkowników, np. usunięcie danych. W takiej sytuacji musi istnieć mechanizm naprawienia błędu, wycofania akcji użytkownika.



Wymagania (4)

2. Efektywne przetwarzanie danych

- Efektywne metody dostępu do danych
- Optymalizacja metod dostępu do danych
- Niezależność aplikacji od fizycznych metod dostępu

BD – wykład 1 (9)

Drugim wymaganiem stawianym bazie danych jest zapewnienie efektywnego przetwarzania danych, tj. wstawiania nowych danych, modyfikowania istniejących, usuwania i wyszukiwania danych. W tym celu koniecznej jest wykorzystywanie efektywnych metod dostępu do danych z wykorzystaniem specjalizowanych struktur i optymalizacja metod dostępu. Ponadto, program, czy użytkownik korzystający z bazy danych nie zna fizycznej organizacji danych na nośniku. W związku z tym, optymalizacja dostępu powinna być realizowana przez wyspecjalizowany moduł programowy i powinna być niewidoczna dla użytkownika.



Wymagania (5)

3. Poprawne modelowanie świata rzeczywistego

- Wspomaganie procesu projektowania i utrzymania bazy danych
- Różne poziomy modelowania danych
- Transformacje między modelami danych

4. Autoryzacja dostępu do danych

- użytkownicy z hasłami dostępu
- użytkownicy i ich uprawnienia

BD – wykład 1 (10)

Trzecim wymaganiem stawianym bazie danych jest poprawne modelowanie świata rzeczywistego. Oznacza to, że struktura bazy danych musi odzwierciedlać właściwy/poprawny sposób obiektów świata rzeczywistego i powiązania pomiędzy tymi obiektami. Przykładowo, jeżeli dealer samochodowy sprzedaje samochody osobowe i dostawcze w różnych konfiguracjach, to baza danych dla tego dealera musi umożliwiać przechowywanie danych na temat samochodów i osobowych i dostawczych, oraz konfiguracji poszczególnych modeli.

Producenci systemów zarządzania bazami danych oferują narzędzia wspomagające procesy modelowania danych, projektowania bazy danych i transformacje pomiędzy różnymi modelami.

Czwartym wymaganiem jest autoryzacja dostępu do danych. Oznacza to, że dostęp do bazy danych mają tylko jej użytkownicy identyfikowani unikalną nazwą i hasłem. Ponadto, każdy użytkownik posiada określone uprawnienia w bazie danych.



Wymagania (6)

5. Współbieżność dostępu do danych

- równoczesny dostęp do tych samych danych przez wielu użytkowników
- konflikt odczyt-zapis, zapis-zapis

6. Metadane

- dane o danych, strukturach dostępu, użytkownikach i ich prawach

BD – wykład 1 (11)

Piątym wymaganiem jest zagwarantowanie możliwości równoczesnej pracy wielu użytkownikom tej samej bazy danych. Co więcej, użytkownicy ci mogą jednocześnie pracować z tym samym zbiorem danych. W takim przypadku mogą powstać konflikty w dostępie do danych, gdy jeden użytkownik modyfikuje zbiór danych, a drugi próbuje ten sam zbiór odczytać lub zmodyfikować. Baza danych musi zapewnić poprawne rozwiązanie tego typu konfliktów.

Szóstym wymaganiem jest wsparcie dla tzw. metadanych. Metadane to najprościej mówiąc dane o bazie danych. Dane te opisują m.in.: dane przechowywane w bazie, struktury danych, użytkowników i ich uprawnienia.



Technologia baz danych (1)

1. Fizyczne struktury danych i metody dostępu
 - Pliki uporządkowane, haszowe, zgrupowane, indeksy drzewiaste i bitmapowe
 - Metoda połowienia binarnego, haszowanie statyczne i dynamiczne, metody połączenia, sortowanie, grupowanie
 - Składniowe i kosztowe metody optymalizacji dostępu
 - Fizyczna niezależność danych

BD – wykład 1 (12)

Omówione wymagania odnośnie baz danych są zapewniane w ramach tzw. technologii baz danych. Oferuje ona m.in.

fizyczne struktury i metody dostępu. Do fizycznych struktur wykorzystywanych w bazach danych zalicza się pliki uporządkowane, pliki haszowe, pliki zgrupowane, indeksy drzewiaste i indeksy bitmapowe. Do metod dostępu zalicza się: połowienie binarne, haszowanie statyczne i dynamiczne, algorytmy łączenia, sortowania i grupowania.

Dostęp do danych z wykorzystaniem struktur fizycznych i metod dostępu jest optymalizowany za pomocą zaawansowanych technik optymalizacji składniowej i kosztowej.

Ponadto, fizyczna organizacja danych na dysku nie ma wpływu na działanie aplikacji/programów użytkowników korzystających z bazy danych. Oznacza to, że zmiana fizycznej organizacji danych np. o klientach banku, po pierwsze, jest niewidoczna dla użytkownika i po drugie, nie wymaga zmiany kodu aplikacji. Innymi słowy aplikacja działa tak samo dobrze jak poprzednio.



Technologia baz danych (2)

2. Przetwarzanie transakcyjne (spójność baz danych)

- Dostęp do bazy danych za pomocą transakcji o własnościach ACID
- Metody synchronizacji transakcji (2PL, znaczniki czasowe, wielowersyjność danych)
- Metody odtwarzania spójności bazy danych (plik logu, odtwarzanie i wycofywanie operacji, Write Ahead Log, punkty kontrolne)
- Archiwizacja bazy danych i odtwarzanie po awarii

BD – wykład 1 (13)

Technologia baz danych oferuje wsparcie dla tzw. przetwarzania transakcyjnego, zapewniającego spójność całej bazy danych.

W ramach tego przetwarzania każdy dostęp do bazy danych jest realizowany w ramach pewnej jednostki interakcji, zwanej transakcją. Posiada ona cechy atomowości, spójności, izolacji i trwałości (problematyka transakcji zostanie omówiona w osobnym wykładzie). Transakcje działające równocześnie w systemie muszą być synchronizowane za pomocą specjalizowanych algorytmów (2PL, znaczników czasowych) i stosowania wersji danych.

Zapewnienie spójności danych, np. w przypadku konfliktu transakcji lub awarii sprzętowo-programowych, często wymaga wycofania zmian w bazie danych. Do tego celu konieczne są dodatkowe struktury danych, algorytmy i mechanizmy systemowe.

Awaria sprzętowo-programowa nie może spowodować utraty żadnych danych. W celu zapewnienia tego wymagania stosuje się techniki i systemowe mechanizmy archiwizowania bazy danych i jej odtwarzania po awarii.



Technologia baz danych (3)

3. Modele danych

- Modele pojęciowe (model związków-encji, UML)
- Modele logiczne (relacyjny, obiektowy, obiektowo-relacyjny, semistrukturalny, hierarchiczny, sieciowy)

4. Narzędzia programistyczne

- Języki budowy aplikacji
- Narzędzia modelowania i projektowania
- Metodyki projektowania

BD – wykład 1 (14)

Technologia baz danych oferuje wsparcie dla wielu modeli danych, czyli wielu sposobów reprezentowania danych. Wyróżnia się tu:

- modele pojęciowe (np. związków-encji, UML),
- modele logiczne (np. relacyjny, obiektowy, obiektowo-relacyjny, semistrukturalny, hierarchiczny, sieciowy).

Oprócz technik związanych z zarządzaniem danymi, technologia baz danych oferuje narzędzia programistyczne do budowania aplikacji, modelowania i projektowania bazy danych. Narzędzie te wspierają uznane metodyki projektowania.



System Zarządzania Bazą Danych (SZBD)

- Oprogramowanie zarządzające całą bazą danych
- Funkcjonalność
 - Język bazy danych - tworzenie, definiowanie, wyszukiwanie i pielęgnacja danych w bazie danych
 - Struktury danych - efektywne składowanie i przetwarzanie dużych wolumenów danych
 - Optymalizacja dostępu do danych
 - Współbieżny dostęp do danych
 - Zapewnienie bezpieczeństwa danych zagrożonego awaryjnością środowiska sprzętowo-programowego
 - Autoryzacja dostępu do danych
 - Wielość interfejsów dostępu do bazy danych

BD – wykład 1 (15)

Jak wspomniano przy okazji omawiania slajdu nr 3, jednym z komponentów systemu bazy danych jest tzw. System Zarządzania Bazą Danych (SZBD). Z technologicznego punktu widzenia jest to moduł programowy, którego zadaniem jest zarządzanie całą bazą danych oraz realizowanie żądań aplikacji użytkowników. Podstawowa funkcjonalność SZBD obejmuje:

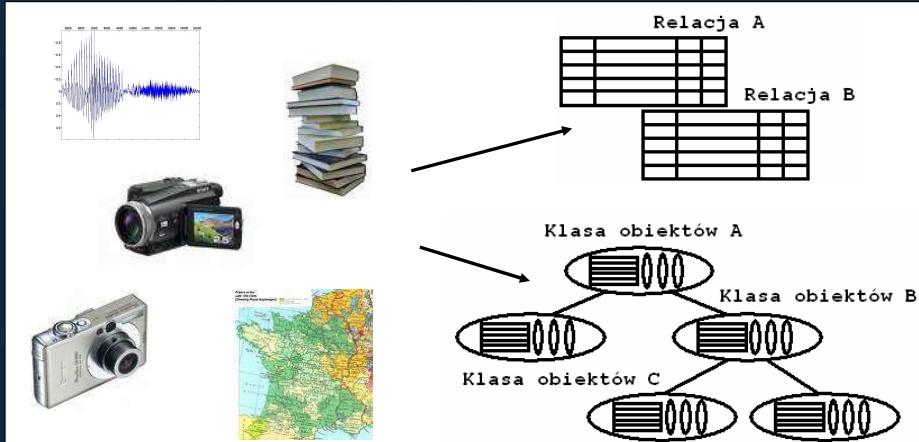
- po pierwsze, wsparcie dla języka bazy danych, który umożliwia m.in. wstawianie, modyfikowanie, usuwanie i wyszukiwanie danych oraz tworzenie, modyfikowanie i usuwanie struktur danych;
- po drugie, wsparcie dla struktur danych zapewniających efektywne składowanie i przetwarzanie dużych wolumenów danych;
- po trzecie, optymalizację dostępu do danych;
- po czwarte, synchronizację współbieżnego dostępu do danych;
- po piąte, zapewnienie bezpieczeństwa danych w przypadku awarii sprzętowo-programowej;
- po szóste, autoryzację dostępu do danych;
- po siódme, wielość interfejsów dostępu do bazy danych.



Model danych (1)

Obiekty świata rzeczywistego

Obiekty modelu danych



BD – wykład 1 (16)

Obiekty ze świata rzeczywistego są reprezentowane w bazie danych za pomocą tzw. modelu danych.

Wyróżnia się następujące modele danych: hierarchiczny, sieciowy, relacyjny, obiektowy, obiektowo-relacyjny, semistrukturalny.

Model hierarchiczny i sieciowy nie są już stosowane w nowo-budowanych systemach. Obecnie w bazach danych najczęściej stosuje się model relacyjny, obiektowo-relacyjny lub semistrukturalny.



Model danych (2)

1. Struktury danych



JAN	NOWAK	47
TADEUSZ	KOWALSKI	34
MACIEJ	NOWAK	26
JANINA	RZEPA	19
KUBA	TARZAN	31
JÓZEF	MALINIAK	29
JAN	NOWAK	56

BD – wykład 1 (17)

Każdy model danych definiuje trzy podstawowe elementy, tj. struktury danych, operacje na danych i ograniczenia intergralnościowe nakładane na dane.

Struktura danych służy do reprezentowania w bazie danych obiektów ze świata rzeczywistego. Przykładowo, grupa pracowników firmy może być reprezentowana w modelu obiektowym jako klasa, lub w modelu relacyjnym jako relacja. Poszczególni pracownicy są reprezentowani odpowiednio jako wystąpienia klasy (w modelu obiektowym) lub krótki relacji (w modelu relacyjnym).



Model danych (3)

2. Operacje (operatory modelu danych)

3. Ograniczenia integralnościowe

Nazwa projektu	Budżet	Data rozpoczęcia	Data zakończenia
Indeksy w BD	500 000,-	1.07.2002	30.06.2005
Magazyny danych	700 000,-	1.09.2002	31.08.2001

BD – wykład 1 (18)

Każdy model danych posiada zbiór predefiniowanych operacji na danych. Przykładowo, w modelu relacyjnym operacje na danych oferowane przez model to: selekcja, projekcja, połączenie i operacje na zbiorach.

Ponadto, model danych umożliwia nałożenie ograniczeń integralnościowych na dane reprezentowane w nim dane. Przykładowo, dla relacji ze slajdu można zdefiniować ograniczenie integralnościowe zapewniające, że data rozpoczęcia projektu będzie zawsze mniejsza niż data jego zakończenia.



Przykładowa baza danych

Pracownicy

IdPrac	Nazwisko	Etat	Szef	DataZat	Płaca	Premia	IdZesp
7340	Kowalski	referent	7548	17.12.90	1800		20
7341	Nowak	asystent	7340	20.02.91	1600	300	30
7342	Tarzan	asystent	7340	22.02.91	1500	500	20
7544	Colargol	kierownik	7600	2.04.91	2500	100	20
7548	Król	księgowy	7600	28.09.91	3500		10
7600	Dziuba	dyrektor		17.10.89	5000		10
7880	Buba	referent	7544	23.05.95	2100		30
7900	Misiek	kierownik	7600	29.10.01	2700		30

Etaty

Nazwa	PłacaMin	PłacaMax
dyrektor	4000	9999
kierownik	2000	5000
referent	1500	2500
asystent	1100	1800
księgowy	2500	4500

Zespoły

IdZesp	Nazwa	Adres
10	Administracyjny	Poznań
20	Sprzedaży	Poznań
30	Reklamy	Gniezno
40	Badań	Oborniki

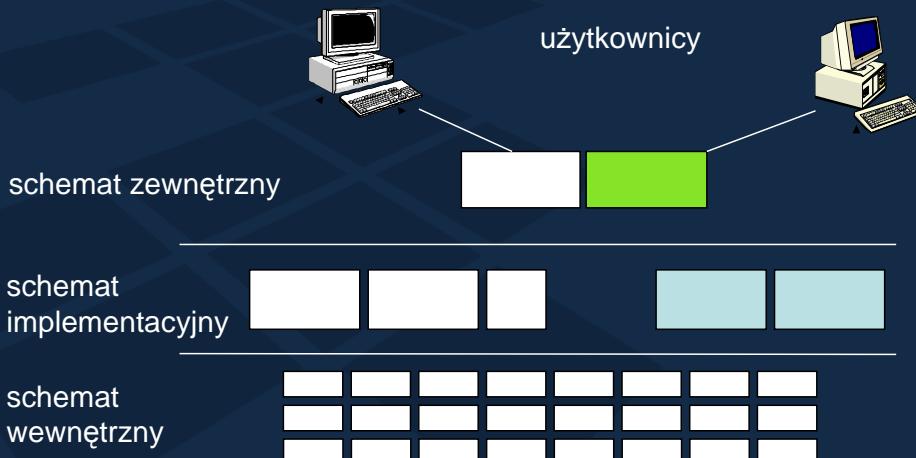
BD – wykład 1 (19)

Przykład prostej bazy danych zaimplementowanej w modelu relacyjnym przedstawiono na slajdzie. Strukturami danych modelu są w tym przypadku trzy relacje: *Pracownicy*, *Zespoły*, *Etaty*. Pierwsza z nich przechowuje dane o pracownikach, druga - o zespołach, w których ci pracownicy są zatrudnieni, a trzecia - zawiera katalog widełek płacowych.



Architektura systemu bazy danych

- 3-warstwowa architektura wg standardu ANSI/SPARC



BD – wykład 1 (20)

Podstawowa architektura systemu bazy danych została zdefiniowana w standardzie ANSI/SPARC. Wyróżnia się w niej 3 następujące tzw. schematy: wewnętrzny, implementacyjny, zewnętrzny.

Schemat wewnętrzny opisuje fizyczny sposób składowania danych na nośnikach. Schemat implementacyjny odwzorowuje schemat wewnętrzny w struktury modelu danych wykorzystywanego w bazie danych. W modelu relacyjnym schemat wewnętrzny jest odwzorowywany w schemat relacyjny. Wreszcie, schemat zewnętrzny stanowi interfejs użytkownika do bazy danych. Schemat ten odwzorowuje schemat implementacyjny w schemat poprzez, który użytkownik widzi bazę danych i pracuje z nią. Należy podkreślić, że schemat zewnętrzny nie zawsze jest stosowany.



Użytkownicy SBD (1)

- Aktorzy na scenie
 - Użytkownicy końcowi
 - Programiści aplikacji
 - Projektanci baz danych
 - Analitycy systemowi
 - Administratorzy systemów baz danych

BD – wykład 1 (21)

Z użytkowaniem bazy danych na różnych etapach są związane różne grupy użytkowników. Wyróżnia się tu: użytkowników końcowych, programistów aplikacji, projektantów baz danych, analityków systemowych i administratorów.

Użytkownicy końcowy charakteryzują się tym, że korzystają z bazy danych głównie poprzez gotowe aplikacje/programy. Ich wiedza zwykle obejmuje sposób obsługi aplikacji i znajomość zagadnień z zakresu obowiązków służbowych.

Zadaniem programistów jest implementowanie aplikacji dla użytkowników końcowych. Funkcjonalność tych aplikacji wynika z wymagań użytkowników.

Projektanci baz danych zajmują się projektowaniem struktury logicznej bazy danych, czyli struktur modelu danych i projektowaniem struktury fizycznej bazy danych, czyli doborem parametrów fizycznego składowania danych na nośnikach. Ponadto, ich zadaniem jest przygotowanie działającej bazy danych.

Analitycy systemowi zajmują się analizą wymagań systemu bazy danych i aplikacji. Wynik ich pracy jest podstawą opracowania struktury logicznej (a często również fizycznej) bazy danych i podstawą dla programistów aplikacji.

Administratorzy systemu bazy danych są odpowiedzialni m.in. za: przygotowanie systemu do pracy produkcyjnej, zagwarantowanie ciągłości pracy systemu, zarządzanie użytkownikami i instalowanie nowych wersji systemu.



Użytkownicy SBD (2)

- Aktorzy poza sceną
 - Administratorzy serwerów, sieci komputerowych
 - Projektanci i programiści SZBD
 - Projektanci narzędzi deweloperskich

BD – wykład 1 (22)

Ponadto, istnieją jeszcze trzy inne grupy użytkowników, których praca nie dotyczy bezpośrednio samej bazy danych, ale bez których system bazy danych nie będzie działał. Do grup tych zaliczamy: administratorów serwerów i sieci komputerowych, projektantów i programistów SZBD, projektantów narzędzi deweloperskich baz danych i aplikacji.



Interakcja z bazą danych (1)

- Język SQL
 - jedyny sposób interakcji z bazą danych
 - język deklaratywny
 - ustandaryzowany
- producenci systemów komercyjnych i niekomercyjnych starają się implementować ten standard

```
select nazwisko, etat, płaca  
from pracownicy  
where idzesp=30  
and etat='kierownik'
```

BD – wykład 1 (23)

Jakakolwiek interakcja programu użytkowego (aplikacji) z bazą danych odbywa się za pomocą języka SQL. Jest to jedyny sposób komunikowania się aplikacji z bazą danych. SQL jest językiem deklaratywnym. Oznacza to, że posługując się nim specyfikujemy tylko co chcemy otrzymać. Nie specyfikujemy sposobu (algorytmu) w jaki ma być zrealizowane zadanie. Przykładem polecenia SQL może być zapytanie do bazy danych poszukujące informacje o klientach banku z Poznania, którzy w ciągu ostatniego miesiąca wypłaciли z bankomatu łącznie powyżej 8000 PLN. W tym zapytaniu specyfikujemy tylko jakie dane nas interesują. Sposób ich wyszukania jest automatycznie dobierany przez SZBD.

SQL jest językiem ustandaryzowanym. Jego standardyzacją zajmuje się specjalny międzynarodowy komitet, w skład którego wchodzą przedstawiciele największych producentów SZBD (IBM, Microsoft, Oracle). Dotychczas opracowano trzy standardy języka SQL, kolejno rozszerzające jego funkcjonalność. Standardy te to: SQL-92, SQL-99, SQL-2003.

Producenci systemów komercyjnych i niekomercyjnych starają się implementować przynajmniej standard SQL-92. Należy jednak pamiętać, że nie ma 100% zgodności implementacji.

Przykład prostego polecenia SQL będącego zapytaniem do bazy danych przedstawiono na slajdzie. Zapytanie to wyszukuje pracowników (nazwisko, etat, płaca) zatrudnionych w zespole o numerze 30 na etacie kierownika.



Interakcja z bazą danych (2)

- Aplikacje
 - formularze
 - elektroniczne formularze z polami, listami, elementami wyboru
 - umożliwiają wstawianie, modyfikowanie, usuwanie, wyszukiwanie danych
 - raporty
 - umożliwiają prezentowanie zawartości bazy danych
 - teksty
 - wykresy
 - grafika

BD – wykład 1 (24)

Język SQL jest narzędziem dostępu do bazy danych stosowanym głównie przez projektantów aplikacji, projektantów baz danych i administratorów baz danych. Standardowym sposobem korzystania z bazy danych przez użytkowników końcowych są aplikacje. Należy jednak pamiętać, że na poziomie programistycznym aplikacje również komunikują się z bazą danych za pomocą polecen SQL.

Ze względu na funkcjonalność, wyróżnia się dwa rodzaje aplikacji, tj. formularze i raporty. Aplikację pierwszego rodzaju należy postrzegać jako elektroniczny formularz (z polami, listami, elementami wyboru) wypełniany przez użytkownika. Formularze umożliwiają pełną obsługę danych, tj. wstawianie, modyfikowanie, usuwanie i wyszukiwanie.

Raporty umożliwiają wyłącznie odczytywanie danych z bazy i prezentowanie ich w różnej postaci, głównie tekstu lub wykresu.

Bazy danych - BD

Formularz - przykład

The screenshot shows a window titled "Oracle Forms Runtime - [Zespoły i pracownicy]". The main area contains two sections: "Zespoły" and "Pracownicy".

Zespoły:

- Nazwa: SYSTEMY EKSPERCKIE
- Adres: STRZELECKA 14
- Pracowników w zespole: 4

Pracownicy:

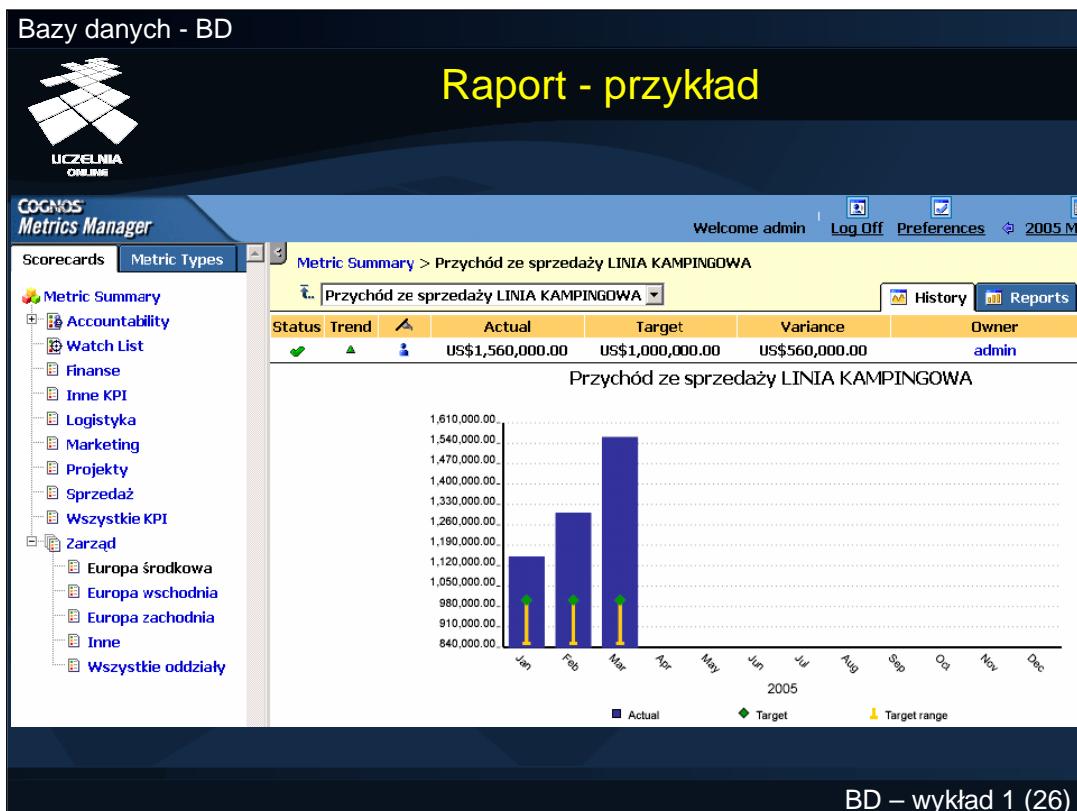
Nazwisko	Etat	Szef	Data zatr.	Płaca pod.	Płaca dod.	Zarobki razem
BINOS	STAZYSTA	SKRABEK	2093.10.15	250.00	170.60	420.60
FARMON	ASYSTENT	MIDAS	2092.09.01	480.00	90.00	570.00
MIDAS	PROFESOR	DRABEK	2077.09.01	1,070.00		1,070.00
TELMAR	STAZYSTA	MIZEK	2094.07.15	208.00		208.00

Sumaryczne zarobki w zespole: 2,268.60

Nazwisko pracownika:
Record: 4/4

BD – wykład 1 (25)

Przykład prostego formularza przedstawiono na slajdzie.



Przykład raportu przestawiono na slajdzie.



Technologie implementacyjne aplikacji

- Języki 3GL
 - np. C, C++, Visual Basic, Visual C++
 - biblioteki umożliwiające zagnieżdżanie poleceń SQL w kodzie
- Języki 4GL
 - np. SAS 4GL, Oracle Forms
 - umożliwiają bezpośrednie umieszczanie poleceń SQL w kodzie aplikacji i bezpośrednią obsługę wyników poleceń SQL
- Java, PHP, Perl
 - stosowane w aplikacjach web'owych pracujących w architekturze 3-warstwowej

BD – wykład 1 (27)

Aplikacje baz danych można implementować w językach trzeciej generacji (3GL) takich jak np. C, C++, Visual Basic, Visual C++. Komunikacja z bazą danych i wykonywanie poleceń SQL i odbiór ich wyników z programów napisanych w tych językach wymaga stosowania specjalizowanych bibliotek, a kod który powstaje jest kodem niskiego poziomu. Z tego względu, w praktyce najczęściej stosuje się albo języki czwartej generacji (4GL) albo języki programowania aplikacji web'owych.

Języki czwartej generacji takie jak np. SAS 4GL lub Oracle Forms, umożliwiają bezpośrednie umieszczanie poleceń SQL w kodzie aplikacji i bezpośrednią obsługę wyników poleceń SQL.

Języki programowania aplikacji web'owych takie jak np. Java, PHP, Perl są stosowane w aplikacjach baz danych pracujących w architekturze 3-warstwowej (omówionej dalej).



BD – wykład 1 (28)

Obecnie, w praktyce stosuje się dwie podstawowe architektury komunikacyjne z systemem bazy danych, tj. architekturę klient-serwer i architekturę 3-warstwową.
W pierwszej z nich, aplikacje użytkowe są zainstalowane na stacjach roboczych i komunikują się z SZBD z wykorzystaniem oprogramowania sieciowego dedykowanego do komunikacji z SZBD. Baza danych znajduje się na dedykowanym serwerze.



BD – wykład 1 (29)

W architekturze 3-warstwowej, pomiędzy stacjami użytkowników, a serwerem bazy danych znajduje się tzw. serwer aplikacji. Jego zadaniem jest udostępnianie umieszczonych na nim aplikacji. Jest to typowa architektura dla aplikacji web'owych. Użytkownik na swojej stacji roboczej posiada tylko przeglądarkę stron www. Aplikacje są udostępniane przez serwer aplikacji w postaci czystych stron html lub w postaci applet'ów Java. W odpowiedzi na polecenia użytkowników realizowane w aplikacjach, serwer aplikacji wysyła odpowiednie żądania do SZBD. SZBD wykonuje polecenia i ich wyniki przesyła do serwera aplikacji, który z kolei przesyła je do aplikacji użytkowników.



Podział systemów baz danych (1)

- Kryteria podziału
 - wykorzystywany model danych
 - liczba węzłów / baz danych
 - cel stosowania
- Model danych
 - relacyjny
 - obiektowy
 - obiektowo-relacyjny
 - semistrukturalny (XML)
 - hierarchiczny
 - sieciowy
- Liczba węzłów / baz danych
 - bazy scentralizowane
 - bazy rozproszone

BD – wykład 1 (30)

Podziału systemów BD można dokonać w oparciu o kilka kryteriów. Najważniejszymi są: wykorzystywany model danych, liczba węzłów, czyli liczba baz danych wchodzących w skład systemu i cel stosowania systemu bd.

Ze względu na model danych SBD dzieli się na: relacyjne, obiektowe, obiektowo-relacyjne, semistrukturalne, hierarchiczne, sieciowe.

Ze względu na liczbę wykorzystywanych BD, wyróżnia się systemy scenralizowane z jedną bazą danych i systemy rozproszone z więcej niż jedną bazą wchodzącą w skład systemu.



Podział systemów baz danych (2)

- Cel stosowania

- przetwarzanie transakcyjne (On-Line Transaction Processing - OLTP)
 - wszelkiego rodzaju systemy ewidencyjne
- przetwarzanie analityczne (On-Line Analytical Processing - OLAP)
 - hurtownie danych
- wspomaganie projektowania (Computer Aided Design - CAD)
 - konstrukcje, budynki, urządzenia

BD – wykład 1 (31)

Ze względu na cel stosowania wyróżnia się bazy danych przetwarzania transakcyjnego, BD przetwarzania analitycznego, wspomagania projektowania, informacji geograficznej, wytwarzania oprogramowania.

Bazy danych przetwarzania transakcyjnego (OLTP) stosuje się w typowych zastosowaniach ewidencyjnych, np. w rezerwacji i sprzedaży biletów, w bibliotekach i wypożyczalniach, w systemach ewidencji ludności, pojazdów, mienia nieruchomości, w bankowości w obsłudze bieżącej, w systemach handlu internetowego i bankowości elektronicznej. Zastosowania tego typu charakteryzują się ogromną liczbą jednocześnie działających użytkowników (tysiące, dziesiątki tysięcy). Interakcja pojedynczego użytkownika z bazą danych jest krótka - kilka kilkanaście sekund.

Bazy danych przetwarzania analitycznego (OLAP) stosuje się w systemach wspomagania zarządzania. Zastosowania tego typu charakteryzują się niewielką liczbą użytkowników (kilku, kilkunastu) ale czas interakcji użytkownika z bazą danych jest długi (godziny, dziesiątki godzin).

Bazy danych dla wspomagania projektowania umożliwiają przechowywanie projektów złożonych obiektów, np. konstrukcji mostów, budynków, schematy urządzeń.



Podział systemów baz danych (3)

- Cel stosowania cd.
 - systemy informacji geograficznej (Geographical Information Systems - GIS)
 - wytwarzanie oprogramowania (Computer Aided Software Engineering - CASE)

BD – wykład 1 (32)

W bazach danych informacji geograficznej przechowuje się zarówno dane tekstowe (np. dane triangulacyjne, opisy terenu) jak i dane przestrzenne (mapy). Tego typu systemy wymagają zaawansowanych technik przeszukiwania map i operacji na nich.

Bazy danych służące do wspomagania wytwarzania oprogramowania przechowują wyniki poszczególnych faz realizacji projektów. Wyniki te są najczęściej reprezentowane w postaci specjalizowanych modeli (diagramów), obiektów i ich własności, projektów i kodów oprogramowania. Tego typu systemy, oprócz standardowej funkcjonalności, wspierają wyszukiwanie zależności pomiędzy obiekty i oraz wywodzenie wersji obiektów (np. oprogramowania) i zarządzanie tymi wersjami.



Dostępne SZBD (1)

- Komercyjne
 - Oracle
 - wersja 9i, 10g
 - IBM
 - DB2 UDB
 - Informix(R) Dynamic Server
 - Microsoft
 - SQL Server2000, SQL Server2005
 - Sybase
 - Adaptive Server Enterprise, Adaptive Server Anywhere

BD – wykład 1 (33)

Na rynku istnieje wiele komercyjnych systemów BD. Do najpopularniejszych producentów zalicza się Oracle, IBM, Microsoft i Sybase. Oracle oferuje SZBD o nazwie Oracle9i, Oracle10g. IBM oferuje systemy DB2 i Informix(R) Dynamic Server. Microsoft oferuje popularny SQL Server w wersjach 2000 i 2005. Sybase jest producentem systemu Adaptive Server Enterprise i Adaptive Server Anywhere.



Dostępne SZBD (2)

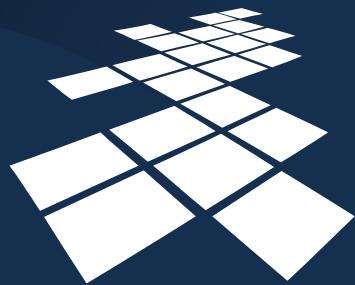
- Niekomercyjne
 - MySQL
 - PostgreSQL
 - FireBird

BD – wykład 1 (34)

Ponadto, dostępne są rozwiązania niekomercyjne, spośród których najpopularniejszymi są MySQL, PostgreSQL i FireBird.

Relacyjny model danych

Wykład przygotował:
Robert Wrembel



UCZELNIA
ONLINE

BD – wykład 2 (1)



Plan wykładu

- Relacyjny model danych
- Struktury danych
- Operacje
- Ograniczenia integralnościowe

BD – wykład 2 (2)

W ramach drugiego wykładu z baz danych zostanie przedstawiony relacyjny model danych, który w praktyce jest najczęściej stosowany. W szczególności wykład omówi: struktury danych tego modelu, operacje modelu i ograniczenia integralnościowe.



Model danych

- Definiuje
 - struktury danych
 - operacje
 - ograniczenia integralnościowe
- Relacyjny model danych
 - relacje
 - selekcja, projekcja, połączenie, operacje na zbiorach
 - klucz podstawowy, klucz obcy, zawężenie dziedziny, unikalność, wartość pusta/niepusta

BD – wykład 2 (3)

W ogólności model danych definiuje:

- struktury wykorzystywane do reprezentowania danych,
- operacje na danych,
- ograniczenia integralnościowe, czyli reguły poprawności danych.

Jednym z fundamentalnych modeli jest model relacyjny. Jest on wykorzystywany w większości komercyjnych i niekomercyjnych systemów baz danych. W modelu tym, strukturą danych jest relacja; operacje na danych obejmują selekcję, projekcję, połączenie i operacje na zbiorach. Ograniczenia integralnościowe w tym modelu to: klucz podstawowy, klucz obcy, zawężenie dziedziny, unikalność wartości, możliwość nadawania wartości pustych/niepustych.



Struktury danych (1)

- Baza danych jest zbiorem relacji
- Schemat relacji R , oznaczony przez $R(A_1, A_2, \dots, A_n)$, składa się z nazwy relacji R oraz listy atrybutów A_1, A_2, \dots, A_n
- Liczbę atrybutów składających się na schemat relacji R nazywamy stopniem relacji
- Każdy atrybut A_i schematu relacji R posiada domenę, oznaczoną jako $\text{dom}(A_i)$
- Domena definiuje zbiór wartości atrybut relacji poprzez podanie typu danych

BD – wykład 2 (4)

W modelu relacyjnym, baza danych jest zbiorem relacji. Każda relacja posiada swój tzw. schemat, który składa się z listy atrybutów. Schemat relacji R jest często oznaczany jako $R(A_1, A_2, \dots, A_n)$, gdzie A_1, A_2, \dots, A_n oznaczają atrybuty. Liczbę atrybutów składających się na schemat relacji R nazywamy stopniem relacji.

Każdy atrybut posiada swoją domenę, zwaną także dziedziną. Definiuje ona zbiór wartości jakie może przyjmować atrybut poprzez określenie tzw. typu danych, np. liczba całkowita, data, ciąg znaków o długości 30.



Struktury danych (2)

- Relacją r o schemacie $R(A_1, A_2, \dots, A_n)$, oznaczoną $r(R)$, nazywamy zbiór n-tek (krotek) postaci $r=\{t_1, t_2, \dots, t_m\}$.
- Pojedyncza krotka t jest uporządkowaną listą n wartości $t=<v_1, v_2, \dots, v_n>$, gdzie $v_i, 1 \leq i \leq n$, jest elementem $\text{dom}(A_i)$ lub specjalną wartością pustą (NULL)
- i-ta wartość krotki t, odpowiadająca wartości atrybutu A_i , będzie oznaczana przez $t[A_i]$
- Relacja $r(R)$ jest relacją matematyczną stopnia n zdefiniowaną na zbiorze domen $\text{dom}(A_1), \text{dom}(A_2), \dots, \text{dom}(A_n)$ będącą podzbiorem iloczynu kartezjańskiego domen definiujących R:

$$r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$$

BD – wykład 2 (5)

Formalna definicja relacji jest następująca:

Relacją r o schemacie $R(A_1, A_2, \dots, A_n)$, oznaczoną $r(R)$, nazywamy zbiór n-tek (krotek) postaci $r=\{t_1, t_2, \dots, t_m\}$.

Pojedyncza krotka t jest uporządkowaną listą n wartości $t=<v_1, v_2, \dots, v_n>$, gdzie $v_i, 1 < i < n$, jest elementem $\text{dom}(A_i)$ lub specjalną wartością pustą (NULL).

i-ta wartość krotki t, odpowiadająca wartości atrybutu A_i , będzie oznaczana przez $t[A_i]$.

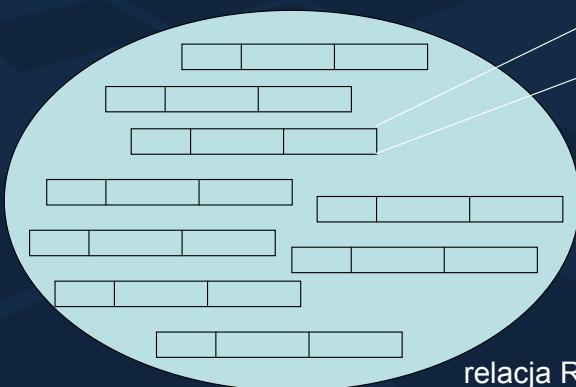
Matematyczna definicja relacji jest następująca:

Relacja $r(R)$ jest relacją matematyczną stopnia n zdefiniowaną na zbiorze domen $\text{dom}(A_1), \text{dom}(A_2), \dots, \text{dom}(A_n)$ będącą podzbiorem iloczynu kartezjańskiego domen definiujących R.



Struktury danych (3)

Relacja jest zbiorem krotek (k-wartości), które są listami wartości



ROR | 9345 PLN | 07.08.2006

BD – wykład 2 (6)

Innymi słowy, relacja jest zbiorem krotek (k-wartości), które są listami wartości. Przykładowo, relacja Rachunki jest złożona ze zbioru krotek. Każda z nich przechowuje trzy wartości, tj. rodzaj rachunku, saldo i datę jego ważności.



Alternatywna definicja relacji

- Wyświetlana relacja ma postać tabeli
 - krotki są wierszami tej tabeli
 - nagłówki kolumn są atrybutami

IdPrac	Nazwisko	Estat	Szef	Zatrudniony	Placa	IdZesp
7369	SMITH	SPRZEDAWCA	7902	1980-12-17	2048	20
7499	ALLEN	SPRZEDAWCA	7698	1981-02-20	4096	10
7566	JONES	KIEROWNIK	7839	1981-04-02	7616	20
7654	MARTIN	SPRZEDAWCA	7698	1981-09-28	3200	30
7782	CLARK	KIEROWNIK	7839	1981-06-09	6272	10
7788	SCOTT	KIEROWCA	7566	1982-12-09	7680	20
7839	KING	DYREKTOR		1981-11-17	12800	10
7844	TURNER	SPRZEDAWCA	7698	1981-09-08	3840	30

BD – wykład 2 (7)

Intuicyjnie, relacja ma postać klasycznej tabeli z kolumnami i wierszami. Kolumny odpowiadają atrybutom relacji, a wiersze (zwane również rekordami) odpowiadają krotkom.



Baza danych

- Baza danych = zbiór relacji
- Schemat bazy danych = zbiór schematów relacji
- Schemat relacji = zbiór {atrybut, dziedzina, [ograniczenia integralnościowe]}
- Relacja = zbiór krotek
- Krotka = lista wartości atomowych

BD – wykład 2 (8)

Slajd ten podsumowuje omówione wcześniej definicje.

Baza danych jest zbiorem relacji.

Schemat relacji jest zbiorem {atrybut, dziedzina, [ograniczenia integralnościowe]}.

Schemat bazy danych jest zbiorem schematów relacji.

Relacja jest zbiorem krotek.

Krotka jest listą wartości atomowych.



Charakterystyka relacji

- Każdy atrybut relacji ma unikalną nazwę
- Porządek atrybutów w relacji nie jest istotny
- Porządek krotek w relacji nie jest istotny i nie jest elementem definicji relacji
- Wartości atrybutów są atomowe (elementarne)
- Relacja nie zawiera rekordów powtarzających się

BD – wykład 2 (9)

Relacja posiada następujące cechy:

- każdy atrybut relacji ma unikalną nazwę,
- porządek atrybutów w relacji nie jest istotny,
- porządek krotek w relacji nie jest istotny i nie jest elementem definicji relacji,
- wartości atrybutów są atomowe (elementarne),
- relacja nie zawiera rekordów powtarzających się. Ponieważ relacja jest zbiorem krotek, więc, z definicji zbioru, wszystkie krotki relacji muszą być unikalne.



Unikalność krotek relacji - klucze (1)

- Ograniczenie na unikalność krotek relacji
 - Każdy podzbiór S atrybutów relacji R, taki że dla każdych dwóch krotek ze zbioru $r(R)$ zachodzi $t1[S] \neq t2[S] \Leftrightarrow$ superkluczem (super key) R
 - Superklucz
 - cały schemat relacji

BD – wykład 2 (10)

Każdy podzbiór S atrybutów relacji R, jest nazywany superkluczem (ang. super key) relacji R jeżeli dla każdych dwóch krotek ze zbioru $r(R)$ zachodzi $t1[S] \neq t2[S]$. W ogólności, cały schemat relacji jest superkluczem.



Unikalność krotek relacji - klucze (2)

- Superklucz może posiadać nadmiarowe atrybuty
- **Kluczem K** schematu relacji R nazywamy superklucz schematu R o takiej własności, że usunięcie dowolnego atrybutu A z K powoduje, że $K' = K - A$ nie jest już superkluczem
- Klucz jest minimalnym superkluczem zachowującym własność unikalność krotek relacji
- Schemat relacji może posiadać więcej niż jeden klucz

BD – wykład 2 (11)

Superklucz może posiadać nadmiarowe atrybuty. **Kluczem K** schematu relacji R nazywamy superklucz schematu R o takiej własności, że usunięcie dowolnego atrybutu A z K powoduje, że $K' = K - A$ nie jest już superkluczem.

Klucz jest minimalnym superkluczem zachowującym własność unikalność krotek relacji.

Schemat relacji może posiadać więcej niż jeden klucz.



Unikalność krotek relacji - klucze (3)

- Wyróżniony klucz \Leftrightarrow klucz podstawowy
- Pozostałe klucze \Leftrightarrow klucze wtórne lub kandydujące

BD – wykład 2 (12)

Jeden z kluczy relacji może być wyróżniony jako tzw. klucz podstawowy, który jednoznacznie identyfikuje krotki relacji. W związku z tym, klucz podstawowy nie może przyjmować wartości pustych. Pozostałe klucze schematu relacji nazywamy kluczami wtórnymi lub kandydującymi.



Ograniczenie integralnościowe

- Mechanizm (reguła), który gwarantuje że dane wpisane do relacji spełniają nałożone na nie warunki
 - czuwa nad tym SZBD
- Definiuje się na poziomie
 - pojedynczego atrybutu
 - całej relacji
- Rodzaje
 - klucz podstawowy (primary key)
 - klucz obcy (foreign key)
 - unikalność (unique)
 - zawężenie domeny/dziedziny (check)
 - wartość pusta/niepusta (NULL/NOT NULL)

BD – wykład 2 (13)

Każda relacja może posiadać jawnie zdefiniowane ograniczenia integralnościowe. Ograniczenie integralnościowe jest pewną regułą gwarantującą, że dane znajdujące się w relacji spełniają tę regułę. W praktyce nad zapewnieniem integralności danych czuwa SZBD. Ograniczenie integralnościowe definiuje się albo dla pojedynczego atrybutu albo dla całej relacji.

Wyróżnia się następujące ograniczenia integralnościowe:

- klucz podstawowy (primary key),
- klucz obcy (foreign key),
- unikalność (unique),
- zawężenie domeny/dziedziny (check),
- wartość pusta/niepusta (NULL/NOT NULL).



Klucz podstawowy

- Klucz podstawowy relacji (primary key)
 - atrybut (lub zbiór atrybutów), którego wartość jednoznacznie identyfikuje krotkę
 - wartość ta jest unikalna w obrębie całej relacji i jest niepusta
- Przykłady:
 - adres e-mail, NIP, PESEL, nr dowodu, nr paszportu

BD – wykład 2 (14)

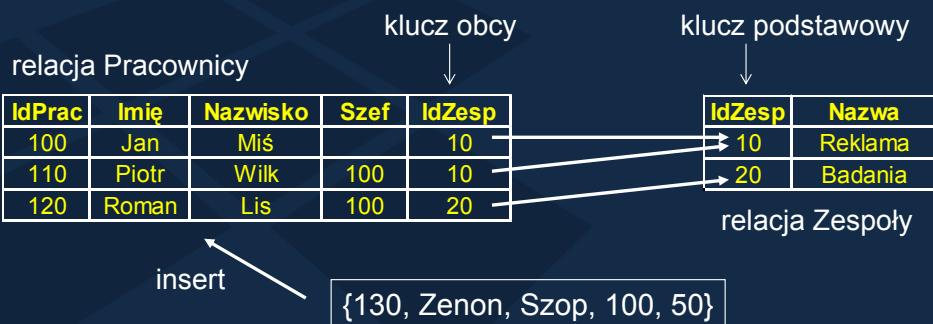
Klucz podstawowy relacji (ang. primary key) jest to atrybut lub zbiór atrybutów, którego wartość jednoznacznie identyfikuje krotkę relacji. Z definicji, wartość atrybutu, który zdefiniowano jako klucz podstawowy jest unikalna w obrębie całej relacji i jest niepusta.

Przykładami atrybutów, które mogły by być kluczami podstawowymi są np. adres e-mail, NIP, PESEL, nr dowodu, nr paszportu.



Klucz obcy (1)

- Klucz obcy relacji (foreign key)
 - atrybut (lub zbiór atrybutów), który wskazuje na klucz podstawowy
 - służy do reprezentowania powiązań między danymi (łączenia relacji)



BD – wykład 2 (15)

Klucz obcy relacji (ang. foreign key) jest atrybutem lub zbiorem atrybutów, który wskazuje na klucz podstawowy w innej relacji. Klucz obcy służy do reprezentowania powiązań między danymi (łączenia relacji). Dziedziną wartości klucza obcego jest dziedzina wartości klucza podstawowego, na który ten klucz obcy wskazuje.

W przykładzie ze slajdu, w relacji Zespoły kluczem podstawowym jest atrybut IdZesp. W relacji Pracownicy kluczem obcym jest IdZesp i wskazuje on na IdZesp w relacji Zespoły. Wartościami atrybutu IdZesp w relacji Pracownicy mogą być tylko te wartości, które przyjmuje IdZesp w relacji Zespoły.

Przykładowy rekord {130, Zenon, Szop, 100, 50} nie zostanie wstawiony do relacji Pracownicy, ponieważ wartość atrybutu IdZesp (50) nie znajduje się w relacji Zespoły. Naruszono w tym przypadku ograniczenie integralnościowe klucza obcego.



Klucz obcy (2)

- Dane są relacje R1 i R2. Podzbiór FK atrybutów relacji R1 nazywany jest kluczem obcym R1 jeżeli:
 - atrybuty w FK mają taką samą domenę jak atrybuty klucza podstawowego PK relacji R2
 - dla każdej krotki t1 relacji R1 istnieje dokładnie jedna krotka t2 relacji R2, taka że $t1[FK] = t2[PK]$, lub $t1[FK] = \text{null}$
- Klucz obcy (ograniczenie referencyjne) gwarantuje, że rekordy z tabeli R1 występują w kontekście związanego z nim rekordu z tabeli R2

BD – wykład 2 (16)

Bardziej formalna definicja klucza obcego jest następująca.

Dane są relacje R1 i R2. Podzbiór FK atrybutów relacji R1 nazywany jest kluczem obcym R1 jeżeli:

- atrybuty w FK mają taką samą domenę jak atrybuty klucza podstawowego PK relacji R2,
- dla każdej krotki t1 relacji R1 istnieje dokładnie jedna krotka t2 relacji R2, taka że $t1[FK] = t2[PK]$, lub $t1[FK] = \text{null}$.

Klucz obcy, zwany również ograniczeniem referencyjnym, gwarantuje, że rekordy z tabeli R1 występują w kontekście związanego z nim rekordu z tabeli R2.



Zawężenie dziedziny

- Zawężenie dziedziny (ograniczenie domeny) atrybutu (check)
 - ograniczenie dozwolonych wartości do pewnego podzbioru przez wyrażenie logiczne określające przedział lub za pomocą wyliczeniowej listy wartości
- Przykłady:
 - płeć: K, M, nieznana, N/A
 - pensja: wartości dodatnie
 - kolor oczu: niebieskie, szare, piwne

BD – wykład 2 (17)

Zbiór wartości domeny atrybutu może być zawężony przez wyrażenie logiczne do pewnego podzbioru: przedziału lub wyliczeniowej listy wartości. Jest to tzw. ograniczenie integralnościowe zawężenia dziedziny (domeny). Przykładami tego typu ograniczenia są np.

- ograniczenie dopuszczalnych wartości atrybutu płeć do: K, M, nieznana, N/A (zgodnie ze standardem ISO),
- zagwarantowanie dodatnich wartości atrybutu pensja,
- ograniczenie dopuszczalnych wartości atrybutu kolor_oczu do trzech wartości: niebieskie, szare, piwne.



Zawężenie dziedziny - przykład

- Etat - dziedzina: {'Analyst', 'Dyrektor', 'Referent', 'Kierownik', 'Sekretarka'}
- Płaca - dziedzina: płaca>500
- IdPrac - klucz podstawowy

relacja Pracownicy

IdPrac	Nazwisko	Etat	Płaca	Szef	IdZesp
120	Kowalski	Analyst	850	100	10
100	Tarzan	Dyrektor	1700		10
130	Nowak	Referent	600	100	10
110	Józick	Kierownik	1200	100	20
140	Nowacki	Analyst	800	110	20
150	Bunio	Sekretarka	700	100	10

insert

{200, 'Szop', ' Księgowy', 900, 10}
 {130, 'Borsuk', ' Kierownik', 1000, 20}
 {210, 'Rosomak', ' Kierownik', 400, 20}

BD – wykład 2 (18)

Jako przykład rozważmy relację Pracownicy ze slajdu. Przyjmijmy, że dla atrybutu Etat zdefiniowano ograniczenie zawężające zbiór jego wartości do analityka, dyrektora, referenta, kierownika i sekretarki. Dla atrybutu płaca określono dziedzinę wartości większych niż 500. Atrybut IdPrac zdefiniowano jako klucz podstawowy relacji Pracownicy. Do tak zdefiniowanej relacji nie da się wstawić żadnej z trzech krotek. Pierwsza z nich narusza integralność etatu, druga narusza integralność klucza podstawowego, a trzecia - integralność płacy.



Podstawowe operacje algebry relacji

- Selekcja (SELECT)
- Projekcja (PROJECT)
- Połączenie (JOIN)
 - Iloczyn kartezjański
- Operacje na zbiorach
 - suma (UNION)
 - część wspólna (INTERSECTION)
 - różnica (MINUS, DIFFERENCE)

BD – wykład 2 (19)

W modelu relacyjnym wykorzystuje się tzw. algebrę relacji, definiującą zbiór operacji na danych i semantykę tych operacji. Operacjami tymi są: selekcja, projekcja, połączenie, iloczyn kartezjański jako specjalny przypadek połączenia, operacje na zbiorach (suma, część wspólna i różnica).



Operacja selekcji

- Przeznaczenie:
 - wyodrębnienie podzbioru krotek relacji, które spełniają warunek selekcji
- Notacja: $\sigma_{<\text{warunek selekcji}>} (<\text{Nazwa relacji}>)$
 - **warunek selekcji** jest zbiorem predykatów postaci
 - $<\text{atrybut}><\text{operator relacyjny}><\text{literals}>$
 - lub
 - $<\text{atrybut}><\text{operator relacyjny}><\text{atrybut}>$
 - predykaty są łączone operatorami logicznymi: AND lub OR
- Własności: operacja selekcji jest komutatywna:
 $\sigma_{<\text{war1}>} (\sigma_{<\text{war2}>} (R)) = \sigma_{<\text{war2}>} (\sigma_{<\text{war1}>} (R))$

BD – wykład 2 (20)

Operacja selekcji umożliwia wyodrębnienie podzbioru krotek relacji, które spełniają warunek selekcji.

Operacja ta jest oznaczana symbolem sigma z pewnym warunkiem selekcji. Operacja ta działa na relacji o pewnej nazwie. Warunek selekcji jest zbiorem predykatów postaci $<\text{atrybut}><\text{operator relacyjny}><\text{literals}>$ lub $<\text{atrybut}><\text{operator relacyjny}><\text{atrybut}>$.

Predykaty są łączone operatorami logicznymi: AND lub OR.

Rozważmy dwie operacje selekcji. Operacja S1 jest realizowana jako pierwsza. S1 posiada warunek W1 i jest realizowana na relacji R. Operacja S2 jest realizowana jako druga. S2 posiada warunek W2 i jest realizowana na wyniku operacji S1. Przyjmijmy, że wynik operacji S1 i S2 wykonanych w takiej kolejności jest zbiorem krotek $\{k1, k2, k3\}$. Jeżeli zamienimy kolejność wykonywania operacji selekcji, tzn. najpierw zostanie wykonana operacja S2 z warunkiem W2 na relacji R, a następnie S1 z warunkiem W1 na wyniku działania operacji S2, to w wyniku końcowym otrzymamy identyczny zbiór krotek jak poprzednio. Taką własność operacji selekcji nazywamy komutatywnością.



Operacja selekcji - przykłady (1)

- $\sigma_{IdZesp = 10}$ (Pracownicy)

```
select IdPrac, Nazwisko, Etat, Szef,
       Zatrudniony, Płaca, IdZesp
  from Pracownicy
 where IdZesp=10
```

- $\sigma_{Płaca > 7000}$ (Pracownicy)

```
select IdPrac, Nazwisko, Etat, Szef,
       Zatrudniony, Płaca, IdZesp
  from Pracownicy
 where Płaca>10
```

BD – wykład 2 (21)

Na slajdzie przedstawiono dwa przykłady operacji selekcji. Pierwszy z nich wybiera z relacji Pracownicy te rekordy, dla których wartość atrybutu IdZesp jest równa 10. Drugi przykład wybiera z relacji Pracownicy tylko tych pracowników których wartość atrybutu Płaca jest większa niż 7000. Obie operacje wyrażono w notacji ogólnej i w języku SQL.



Operacja selekcji - przykłady (2)

- $\sigma_{(IdZesp=10 \text{ AND } Płaca>7000) \text{ OR } (IdZesp=20 \text{ AND } Płaca>8000)}$ (Pracownicy)

```
select IdPrac, Nazwisko, Etat, Szef,
       Zatrudniony, Płaca, IdZesp
  from Pracownicy
 where (IdZesp=10 and Płaca>7000)
   or      (IdZesp=20 and Płaca>8000)
```

- $\sigma_{Estat='Księgowy' \text{ AND } (Płaca}>6000 \text{ AND } Płaca<9000)} (Pracownicy)$

```
select IdPrac, Nazwisko, Etat, Szef,
       Zatrudniony, Płaca, IdZesp
  from Pracownicy
 where Etat='KSIĘGOWY'
   and (Płaca}>6000 and Płaca<9000)
```

BD – wykład 2 (22)

W pierwszym przykładzie z tego slajdu operacja selekcji wybiera z relacji Pracownicy krotki dla których wartość atrybutu IdZesp=10 i Płaca>7000 lub IdZesp=20 i Płaca>8000. Należy zwrócić tu uwagę na priorytety operatorów. AND ma wyższy priorytet niż OR, co dodatkowo zostało zaznaczone za pomocą nawiasów.

Drugi przykład ilustruje selekcję z relacji Pracownicy wszystkich księgowych zarabiających w przedziale między 6000 i 9000.



Operacja projekcji

- Przeznaczenie:
 - wyodrębnienie wybranych atrybutów relacji
- Notacja: $\pi_{<\text{atrybuty}>} (<\text{Nazwa relacji}>)$
 - atrybuty jest podzbiorem atrybutów ze schematu relacji
- Własności: operacja projekcji nie jest komutatywna
- Składanie operacji projekcji jest możliwe jeżeli lista2 zawiera wszystkie atrybuty lista1

$$\pi_{<\text{lista1}>} (\pi_{<\text{lista2}>} (R)) = \pi_{<\text{lista1}>} (R)$$

BD – wykład 2 (23)

Drugą operacją modelu relacyjnego jest projekcja. Umożliwia ona wyodrębnienie (wybór) tylko określonych atrybutów relacji.

Operacja ta jest oznaczana symbolem pi z podzbiorem wybieranych atrybutów z całego zbioru atrybutów relacji. Operacja ta działa na relacji o pewnej nazwie.

Operacja projekcji nie jest komutatywna, a składanie operacji projekcji jest możliwe jeżeli lista2 zawiera wszystkie atrybuty lista1. Notację operacji składania projekcji przedstawiono na slajdzie.



Operacja projekcji - przykłady

- $\pi_{\text{Nazwisko}} (\text{Pracownicy})$

```
select Nazwisko  
from Pracownicy
```

- $\pi_{\text{Nazwisko}, \text{Etat}, \text{Płaca}} (\text{Pracownicy})$

```
select Nazwisko, Etat, Płaca  
from Pracownicy
```

BD – wykład 2 (24)

Na slajdzie przedstawiono dwa przykłady projekcji. W pierwszym, ze zbioru atrybutów relacji Pracownicy jest wybierany tylko atrybut Nazwisko. Wynikiem tej operacji projekcji jest zbiór nazwisk wszystkich pracowników. W drugim przykładzie, ze zbioru atrybutów relacji Pracownicy są wybierane atrybuty Nazwisko, Etat i Płaca. W tym przypadku, wynikiem jest zbiór krotek wszystkich pracowników, ale każda z krotek posiada tylko 3 wartości: nazwiska, etatu i pensji.

Obie przykładowe operacje projekcji wyrażono w notacji ogólnej i w języku SQL.



Składanie operacji

- Wynik danej operacji może być zbiorem wejściowym dla innej operacji

$$\sigma_{\text{IdZesp} = 10} (\text{Pracownicy}) \Leftrightarrow \text{PracZesp10}$$
$$\pi_{\text{IdPrac, Nazwisko}} (\text{PracownicyZesp10}) \Leftrightarrow \text{PracZesp10Wynik}$$

$$\text{PracZesp10Wynik} = \pi_{\text{IdPrac, Nazwisko}} (\sigma_{\text{IdZesp} = 10} (\text{Pracownicy}))$$

BD – wykład 2 (25)

Sekwencja wielu operacji, w której kolejne operacje są wykonywane na pośrednich wynikach operacji poprzednich, może być zastąpiona pojedynczą operacją złożoną, powstałą przez zagnieźdżenie operacji elementarnych.

Jako przykład rozważmy operację selekcji z warunkiem IdZesp=10. Przyjmijmy, że jej wynikiem jest relacja tymczasowa o nazwie PracZesp10. Następnie na tej relacji wykonujemy operację projekcji atrybutów IdPrac i Nazwisko.

Przyjmijmy, że jej wynikiem jest relacja tymczasowa o nazwie PracZesp10Wynik.

Obie operacje można złożyć w jedną, której wynik będzie identyczny z zawartością relacji PracZesp10Wynik, jak pokazano na slajdzie.



Operacje na zbiorach (1)

- Kompatybilność relacji
 - Dwie relacje: $R(A_1, \dots, A_n)$ i $S(B_1, \dots, B_n)$ są kompatybilne, jeżeli mają ten sam stopień i jeżeli $\text{dom}(A_i) = \text{dom}(B_i)$ dla $1 \leq i \leq n$
- Operacje na zbiorach
 - dla dwóch kompatybilnych relacji: $R(A_1, \dots, A_n)$ i $S(B_1, \dots, B_n)$

BD – wykład 2 (26)

W modelu relacyjnym są dostępne operacje na zbiorach o takiej samej semantyce, jak standardowe operacje na zbiorach znane z kursu matematyki. W modelu relacyjnym operacje te są wykonywane na relacjach, które jak wiemy są zbiorami krotek. Relacje te muszą być kompatybilne.

Dwie relacje są kompatybilne jeśli mają ten sam stopień i dziedziny odpowiadających sobie atrybutów są takie same.

Operacje sumy, iloczynu i różnicy dwóch kompatybilnych relacji R i S są zdefiniowane następująco.



Operacje na zbiorach (2)

- **Suma:**

- Wynikiem tej operacji, oznaczanej przez $R \cup S$, jest relacja zawierająca wszystkie krotki, które występują w R i wszystkie krotki, które występują w S , z wyłączeniem duplikatów krotek
- Operacja sumy jest operacją komutatywną: $R \cup S = S \cup R$

- **Iloczyn:**

- Wynikiem tej operacji, oznaczonej przez $R \cap S$, jest relacja zawierająca krotki występujące zarówno w R i S
- Operacja iloczynu jest operacją komutatywną: $R \cap S = S \cap R$

BD – wykład 2 (27)

Suma: wynikiem tej operacji, oznaczanej przez R SUMA S , jest relacja zawierająca wszystkie krotki, które występują w R i wszystkie krotki, które występują w S , z wyłączeniem duplikatów krotek. Operacja sumy jest operacją komutatywną: R SUMA $S = S$ SUMA R .

Iloczyn: wynikiem tej operacji, oznaczonej przez R ILOCZYN S , jest relacja zawierająca krotki występujące zarówno w R i S . Operacja iloczynu jest operacją komutatywną: R ILOCZYN $S = S$ ILOCZYN R .



Operacje na zbiorach (3)

- **Różnica:**

- Wynikiem tej operacji, oznaczonej przez $R-S$, jest relacja zawierająca wszystkie krotki, które występują w R i nie występują w S
- Operacja różnicy nie jest operacją komutatywną:
 $R - S \neq S - R$

BD – wykład 2 (28)

Różnica: wynikiem tej operacji, oznaczonej przez $R-S$, jest relacja zawierająca wszystkie krotki, które występują w R i nie występują w S . Operacja różnicy nie jest operacją komutatywną: $R - S \neq S - R$.



Operacje na zbiorach - przykłady

Uczniowie

Imię	Nazwisko
Ala	Kusiak
Edek	Musiał
Adam	Zając
Olek	Struś
Ola	Buba

Instruktorzy

Imię	Nazwisko
Jan	Kuc
Edek	Musiał
Wacek	Misiek

Uczniowie \cup Instruktorzy

Imię	Nazwisko
Ala	Kusiak
Edek	Musiał
Adam	Zając
Olek	Struś
Ola	Buba
Jan	Kuc
Wacek	Misiek

Uczniowie - Instruktorzy

Imię	Nazwisko
Ala	Kusiak
Adam	Zając
Olek	Struś
Ola	Buba
Jan	Kuc
Wacek	Misiek

Uczniowie \cap Instruktorzy

Imię	Nazwisko
Edek	Musiał

Instruktorzy - Uczniowie

Imię	Nazwisko
Jan	Kuc
Wacek	Misiek

BD – wykład 2 (29)

Na slajdzie przedstawiono dwie kompatybilne relacje Uczniowie i Instruktorzy oraz wyniki operacji sumy, iloczynu i różnicę tych relacji.



Operacje na zbiorach - SQL

```
select Imię, Nazwisko  
from Uczniowie  
UNION  
select Imię, Nazwisko  
from Instruktorzy;
```

```
select Imię, Nazwisko  
from Uczniowie  
MINUS  
select Imię, Nazwisko  
from Instruktorzy;
```

```
select Imię, Nazwisko  
from Uczniowie  
INTERSECT  
select Imię, Nazwisko  
from Instruktorzy;
```

```
select Imię, Nazwisko  
from Instruktorzy  
MINUS  
select Imię, Nazwisko  
from Uczniowie;
```

BD – wykład 2 (30)

Na slajdzie przedstawiono polecenia zapisane w języku SQL realizujące operacje sumy, iloczynu i różnicę relacji z poprzedniego slajdu.



Iloczyn kartezjański

- Dane są dwie relacje: $R(A_1, \dots, A_n)$ i $S(B_1, \dots, B_m)$
 - Wynikiem iloczynu kartezjańskiego relacji R i S , oznaczonym przez $R \times S$, jest relacja Q stopnia $n+m$ i schemacie: $Q(A_1, \dots, A_n, B_1, \dots, B_m)$
- Krotkom w relacji Q odpowiadają wszystkie kombinacje krotek z relacji R i S
- Jeżeli relacja R ma N krotek, a relacja S ma M krotek, to relacja Q będzie miała $N*M$ krotek

BD – wykład 2 (31)

Kolejną operacją modelu relacyjnego jest połączenie. Szczególnym przypadkiem połączenia jest tzw. iloczyn kartezjański, zdefiniowany następująco.

Dane są dwie relacje: $R(A_1, \dots, A_n)$ i $S(B_1, \dots, B_m)$. Wynikiem iloczynu kartezjańskiego relacji R i S , oznaczonym przez $R \times S$, jest relacja Q stopnia $n+m$ i schemacie: $Q(A_1, \dots, A_n, B_1, \dots, B_m)$. Krotkom w relacji Q odpowiadają wszystkie kombinacje krotek z relacji R i S . Jeżeli relacja R ma N krotek, a relacja S ma M krotek, to relacja Q będzie miała $M*N$ krotek. Innymi słowy, iloczyn kartezjański polega na połączeniu każdej krotki z relacji R z każdą krotką relacji S .



Iloczyn kartezjański - przykład

Pracownicy

Imię	Nazwisko
Ala	Kusiak
Edek	Musiał
Adam	Zając

Zespoły

Nazwa	Lokalizacja
Reklama	Krucza 10
Badania	Piotrowo 3A

Pracownicy x Zespoły

Imię	Nazwisko	Nazwa	Lokalizacja
Ala	Kusiak	Reklama	Krucza 10
Edek	Musiał	Reklama	Krucza 10
Adam	Zając	Reklama	Krucza 10
Ala	Kusiak	Badania	Piotrowo 3A
Edek	Musiał	Badania	Piotrowo 3A
Adam	Zając	Badania	Piotrowo 3A

BD – wykład 2 (32)

Na slajdzie przedstawiono dwie relacje, tj. Pracownicy i Zespoły oraz wynik iloczynu kartezjańskiego tych relacji.



Operacja połączenia (1)

- Przeznaczenie:

– łączenie na podstawie warunku połączniowego wybranych krotek z dwóch relacji w pojedynczą krotkę

- Notacja: operacja połączenia relacji $R(A_1, \dots, A_n)$ i $S(B_1, \dots, B_m)$, jest oznaczona jako:

$$R \bowtie_{\text{warunek połączniowy}} S$$

– warunek połączniowy jest zbiorem predykatów połączonych operatorami logicznymi AND
– predykaty są postaci: $A_i \theta B_j$

- A_i i B_j są atrybutami połączniowymi
- A_i jest atrybutem R , B_j jest atrybutem S
- $\text{dom}(A_i) = \text{dom}(B_j)$,
- θ jest operatorem relacyjnym ze zbioru $\{ =, \neq, <, \leq, >, \geq \}$

BD – wykład 2 (33)

Operacja połączenia umożliwia łączenie wybranych krotek z dwóch relacji w pojedynczą krotkę. Krotki są łączone na podstawie podanego warunku połączniowego.

Notację operacji łączenia relacji R i S przedstawiono na slajdzie. Warunek połączniowy jest zbiorem predykatów połączonych operatorami logicznymi AND. Predykaty te są postaci: $A_i \text{ THETA } B_j$, gdzie

- A_i i B_j są atrybutami połączniowymi,
- A_i jest atrybutem R , B_j jest atrybutem S ,
- $\text{dom}(A_i) = \text{dom}(B_j)$,
- THETA jest operatorem relacyjnym ze zbioru $\{ =, \neq, <, \leq, >, \geq \}$.



Operacja połączenia (2)

- Ogólna postać operacji połączenia (theta join)
 - $R \bowtie_{\theta} S$
- Połączenie równościowe (equi join)
 - θ jest operatorem $=$
- Połączenie nierównościowe (non-equi join)
 - θ jest operatorem różnym od $=$

BD – wykład 2 (34)

Ogólna postać operacji połączenia, gdzie THETA jest dowolnym operatorem relacyjnym jest nazywana połączeniem typu THETA (ang. theta join).

Operacja połączenia, dla której THETA jest operatorem $=$, nazywana jest połączeniem równościowym (ang. equi join).

Operacja połączenia, dla której THETA jest operatorem różnym od $=$, nazywana jest połączeniem nierównościowym (ang. non-equi join).



Operacja połączenia (3)

- Połączenie naturalne (natural join)
 - połączenie równościowe
 - jeden z atrybutów połączeniowych jest usunięty ze schematu relacji wynikowej
 - oznaczane jako: $R * S$
 - atrybuty połączeniowe w obu relacjach muszą mieć taką samą nazwę

BD – wykład 2 (35)

Operacja połączenia równościowego, w której jeden z atrybutów połączeniowych jest usunięty ze schematu relacji wynikowej, jest nazywana połączeniem naturalnym (ang. natural join). Połączenie naturalne jest oznaczane jako: $R * S$, przy czym wymagane jest, by atrybuty połączeniowe w obu relacjach miały taką samą nazwę.



Pracownicy

IdPrac	Imię	Nazwisko	Szef	IdZesp
100	Jan	Miś		10
110	Piotr	Wilk	100	10
120	Roman	Lis	100	20

Zespoły

IdZesp	Nazwa
10	Reklama
20	Badania

Pracownicy \bowtie Pracownicy

Szef=IdPrac

IdPrac	Imię	Nazwisko	Szef	IdZesp	IdPrac	Imię	Nazwisko	Szef	IdZesp
110	Piotr	Wilk	100	10	100	Jan	Miś		10
120	Roman	Lis	100	20	100	Jan	Miś		10

Pracownicy * Zespoły

IdPrac	Imię	Nazwisko	Szef	IdZesp	Nazwa
100	Jan	Miś		10	Reklama
110	Piotr	Wilk	100	10	Reklama
120	Roman	Lis	100	20	Badania

BD – wykład 2 (36)

Na slajdzie przedstawiono dwie relacje, tj. Pracownicy i Zespoły oraz wynik połączenia równościowego i naturalnego tych relacji.



Operacja połączenia - SQL

```
select *  
from pracownicy p join zespoły z  
on p.id_zesp=z.id_zesp
```

połączenie równościowe
(niestandardowe)

```
select nazwisko, nazwa  
from pracownicy p join zespoły z  
on p.id_zesp=z.id_zesp
```

połączenie równościowe
(standardowe)

połączenie naturalne
(standardowe)

```
select *  
from pracownicy p natural join zespoły z
```

BD – wykład 2 (37)

Na slajdzie przedstawiono polecenia zapisane w języku SQL realizujące operacje połączenia równościowego i naturalnego relacji z poprzedniego slajdu.

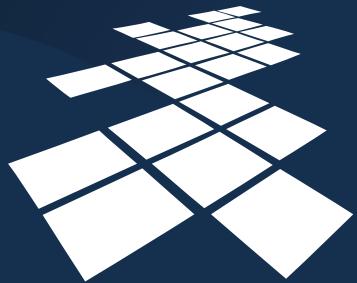
Połączenie równościowe zapisano w dwóch postaciach, pierwsza nie jest zgodna ze standardem SQL, ale jest wspierana przez wiele SZBD. Druga notacja jest zgodna ze standardem języka. Połączenie naturalne wyspecyfikowano zgodnie ze standardem.

Modelowanie danych

Model związków-encji

Wykład przygotował:

Robert Wrembel



**UCZELNIA
ONLINE**

BD – wykład 3 (1)



Plan wykładu

- Wprowadzenie do modelowania i projektowania systemów informatycznych
- Model związków-encji
 - encje
 - atrybuty encji
 - związki pomiędzy encjami
 - hierarchia generalizacji

BD – wykład 3 (2)

Celem wykładu jest omówienie metodyki modelowania danych. Wynik tej metodyki, czyli tzw. model konceptualny jest podstawą schematu bazy danych. W ramach wykładu zostaną omówione:

- wprowadzenie do modelowania i projektowania systemów informatycznych,
- model związków-encji, jako jeden z fundamentalnych modeli wykorzystywanych przy projektowaniu relacyjnych baz danych.

W szczególności, zostaną omówione obiekty modelu związków-encji, czyli encje i ich atrybuty i różnego typu związki pomiędzy encjami oraz hierarchie encji.



Modelowanie - modele

- Modelowanie - odwzorowanie rzeczywistych obiektów świata rzeczywistego w systemie informatycznym (bazie danych)
- Modele
 - konceptualne
 - reprezentacja obiektów w uniwersalnym modelu niezależnym od modelu implementacyjnego
 - model związków-encji
 - model UML
 - implementacyjne
 - modele wykorzystywane do implementacji modeli konceptualnych
 - modele danych (relacyjne, obiektowe, itp.)

BD – wykład 3 (3)

Każdy system informatyczny jest komputerową reprezentacją jakiegoś fragmentu świata rzeczywistego. Należy zatem zbudować taki model świata rzeczywistego, który łatwo i sensownie da się odzwierciedlić w systemie informatycznym. Budowaniem takich modeli zajmuje się dziedzina wiedzy zwana modelowaniem.

Wyróżnia się dwa podstawowe rodzaje modeli, tj. konceptualne i implementacyjne. Model konceptualny umożliwia reprezentowanie obiektów w uniwersalnym modelu niezależnym od modelu implementacyjnego. Wśród modeli konceptualnych najpopularniejszymi są model związków-encji i model UML.

Model implementacyjny (zwany również modelem danych) jest wykorzystywany na etapie implementacji systemu. Odzwierciedla on model konceptualny w konkretne struktury danych. Wśród modeli danych najpopularniejszymi obecnie są: relacyjny, obiektowy, obiektowo-relacyjny i semistrukturalny.



Cykl projektowy systemu informatycznego



BD – wykład 3 (4)

Każdy system informatyczny projektuje się etapami. Do najważniejszych z nich należą: analiza, projektowanie, implementowanie, wdrożenie.

Etap analizy polega na zebraniu wymagań użytkowników odnośnie struktury i funkcjonalności SI. Wynikiem tej fazy są abstrakcyjne modele konceptualne. Modele te opisują najczęściej rodzaje i struktury informacji przetwarzanych w informatyzowanej instytucji oraz funkcjonalność oprogramowania przetwarzającego te informacje.

Etap projektowania polega na przejściu od abstrakcyjnych modeli konceptualnych do modeli implementacyjnych bazy danych i aplikacji.

Etap implementacji polega na zbudowaniu bazy danych w wybranej technologii implementacyjnej i zimplementowaniu aplikacji.

Po przetestowaniu systemu jest on wdrażany u użytkownika. Po wdrożeniu do użytkowania, system należy utrzymywać w ciągłej, efektywnej i niezawodnej pracy.



Obiekty świata rzeczywistego

- Obiekty materialne
 - samochody, budynki, sprzęt komputerowy
 - zasoby ludzkie (grupa pracowników)
- Obiekty niematerialne
 - wiedza (znajomość technologii)
 - zdarzenia (otrzymanie nagrody, urlopu)
 - stany rzeczywistości (stan rachunku bankowego, polisa ubezpieczeniowa)



BD – wykład 3 (5)

W informatyzowanym świecie rzeczywistym występują obiekty różnego typu. Wyróżnia się obiekty materialne i niematerialne. Przykładami tych pierwszych mogą być, np. samochody, budynki, sprzęt komputerowy, zasoby ludzkie (grupa pracowników). Przykładami tych drugich mogą być, np. wiedza (znajomość technologii), zdarzenia (otrzymanie nagrody, urlopu), stany rzeczywistości (stan rachunku bankowego, stan polisy ubezpieczeniowej).



Model związków-encji

- Model związków-encji (entity-relationship model - ER)
 - obiekty świata rzeczywistego reprezentowane za pomocą encji (entities)
 - powiązania między obiektami świata rzeczywistego reprezentowane za pomocą związków (relationships) pomiędzy encjami
- Notacje modelu ER
 - Chen
 - Barker (Oracle) - stosowana na wykładzie

BD – wykład 3 (6)

Jak wspomniano, zadaniem modelu konceptualnego jest odzwierciedlenie obiektów świata rzeczywistego w inne abstrakcyjne obiekty, które w pewnym dalszym momencie da się reprezentować w systemie informatycznym. Jednym z fundamentalnych modeli konceptualnych wykorzystywanym w projektowaniu relacyjnych baz danych jest model związków-encji (ang. entity-relationship model - ER). W tym modelu, obiekty świata rzeczywistego są reprezentowane za pomocą encji (ang. entities), a powiązania między obiektami - za pomocą związków pomiędzy encjami (ang. relationships).

Model ER można przedstawić w wielu różnych notacjach graficznych. Najpopularniejszymi są tu notacja Chena i notacja Barkera (wykorzystywana w produktach Oracle). Notacja Barkera będzie wykorzystywana na wykładzie.



Encja

- Reprezentuje zbiór obiektów opisany tymi samymi cechami (atributami, własściami)
- Informacje o tych obiektach będą przechowywane w bazie danych
- Konkretny obiekt świata rzeczywistego jest reprezentowany jako wystąpienie encji (instancję encji)

BD – wykład 3 (7)

Encja reprezentuje zbiór obiektów opisany tymi samymi cechami (atributami, własściami). Informacje o tych obiektach będą przechowywane w bazie danych.

Konkretny obiekt świata rzeczywistego jest reprezentowany jako wystąpienie encji (instancja encji).



Modelowanie encji (1)

Obiekty świata rzeczywistego

Firma zatrudnia pracowników. Chcemy przechowywać informacje nt. danych personalnych pracowników (imię, nazwisko, adres i numer telefonu).

wspólne cechy pracowników



Zenon Sikora
ul. Polska 3
333333

Herman Klos
ul. Rycerska 45
444444

Zdzisław Pirat
ul. Helska 44
555555

Encja

<u>Pracownik</u>
imię
nazwisko
adres
nr_telefonu

Wystąpienia encji

<u>Pracownik</u>
imię = Zenon
nazwisko = Sikora
adres = ul. Polska 3
nr_telefonu = 333333

BD – wykład 3 (8)

Jako przykład encji rozważmy następujący mikro-opis fragmentu pewnego świata rzeczywistego.

"Firma zatrudnia pracowników. Chcemy przechowywać informacje nt. danych personalnych pracowników (imię, nazwisko, adres i numer telefonu)."

Na podstawie tego opisu można zbudować prosty model encji. Ponieważ chcemy przechowywać dane na temat pracowników, obiektem modelu będzie encja o nazwie Pracownik. Ponieważ wszyscy pracownicy firmy mają takie same cechy, więc encja będzie posiadała 4 atrybuty: imię, nazwisko, adres, nr_telefonu. Encja ta będzie reprezentowała wszystkich pracowników firmy i aktualnie zatrudnionych i zatrudnianych w przyszłości. Wystąpieniami tej encji są konkretni pracownicy firmy, jak pokazano na slajdzie.



Modelowanie encji (2)

Obiekty świata rzeczywistego

Parking firmy jest przeznaczony do parkowania wielu różnych samochodów. Chcemy przechowywać informacje o samochodach (marka, model, numer rejestracyjny), które mogą parkować na parkingu firmy.

wspólne cechy samochodów



Subaru
Forester
PO0233A

Peugeot
206
PO1236U

Opel
Astra
PZI932Y

Encja

Samochód
marka
model
nr_rejestracyjny

Wystąpienia encji

Samochód
marka = Subaru
model = Forester
nr_rejestracyjny = PO0233A

BD – wykład 3 (9)

Jako kolejny przykład rozważmy następujący opis mikro-świata rzeczywistego.

"Parking firmy jest przeznaczony do parkowania wielu różnych samochodów. Chcemy przechowywać informacje o samochodach (marka, model, numer rejestracyjny), które mogą parkować na parkingu firmy."

Z tego opisu wynika, że najważniejszym obiektem modelu tego mikro-świata jest samochód opisany marką, modelem i numerem rejestracyjnym. Każdy samochód będzie więc reprezentowany za pomocą encji o nazwie Samochód z 3 atrybutami: marka, model, nr_rejestracyjny. Konkretne samochody na parkingu firmy będą wystąpieniami tej encji.



Modelowanie encji (2)

- Każda encja posiada
 - unikalną nazwę
 - zbiór cech (atrybutów)
- Encje wchodzą w związki z innymi encjami
 - wyjątkiem są encje reprezentujące dane słownikowe i konfiguracyjne
- Dowolna rzecz lub obiekt może być reprezentowana tylko przez jedną encję
- Nazwa encji powinna być rzeczownikiem w liczbie pojedynczej

BD – wykład 3 (10)

Modelując encje należy przestrzegać następujących reguł.

1. Każda encja posiada unikalną nazwę.
2. Każda encja posiada zbiór cech (atrybutów).
3. Encje wchodzą w związki z innymi encjami (wyjątkiem są encje reprezentujące dane słownikowe i konfiguracyjne).
4. Dowolna rzecz lub obiekt może być reprezentowana tylko przez jedną encję.
5. Nazwa encji powinna być rzeczownikiem w liczbie pojedynczej.



Atrybuty encji (1)

- Identyfikator
 - atrybut lub zbiór atrybutów jednoznacznie identyfikujący wystąpienie encji
 - zbiór atrybutów + związki
 - związki
- Identyfikatory naturalne
 - PESEL, NIP, nr dowodu, nr paszportu, nr rejestracyjny, ISBN
- Identyfikatory sztuczne
 - numer pozycji katalogowej, identyfikator pracownika

BD – wykład 3 (11)

Jak już wiemy, encja posiada atrybuty. Wyróżnia się dwa rodzaje atrybutów, tj. identyfikatory i deskryptory.

Identyfikator encji jest to atrybut lub zbiór atrybutów jednoznacznie identyfikujący wystąpienie encji. W skład identyfikatora mogą wchodzić również atrybuty i związki lub same związki.

Wyróżnia się identyfikatory naturalne np. PESEL, NIP, nr dowodu, nr paszportu, nr rejestracyjny, ISBN i identyfikatory sztuczne, np. numer pozycji katalogowej, identyfikator pracownika.



Atrybuty encji (2)

- Deskryptory (atrybuty deskrypcyjne)
 - wszystkie inne atrybuty poza identyfikatorami
 - reprezentują podstawowe cechy/własności encji
 - cechy te będą przechowywane w bazie danych
 - atrybuty z wartościami opcjalnymi
 - atrybuty z wartościami obowiązkowymi

BD – wykład 3 (12)

Deksryptory (atrybuty deskrypcyjne) są to wszystkie inne atrybuty poza identyfikatorami. Reprezentują one podstawowe cechy/własności encji (cechy te będą przechowywane w bazie danych). Wartości deskryptorów mogą być albo opcjonalne albo obowiązkowe. Wynika to z analizy potrzeb informacyjnych użytkowników.



Definicja atrybutu encji

- Nazwa
- Dziedzina
 - typ danych i maksymalny rozmiar
 - zbiór dozwolonych wartości
 - zakres dozwolonych wartości
- Dozwolone / niedozwolone wartości puste
- Opcjonalnie unikalność wartości

} ograniczenia integralnościowe

BD – wykład 3 (13)

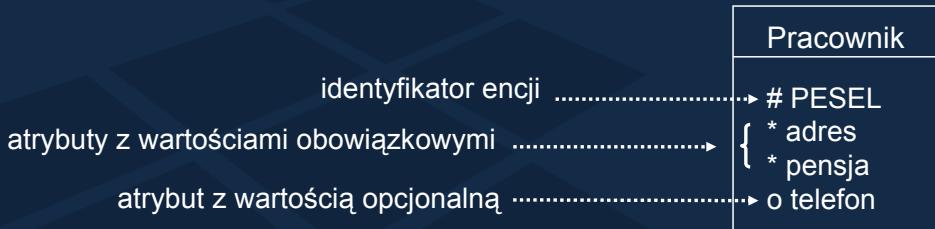
Definicja pojedynczego atrybutu encji obejmuje:

- nazwę,
 - dziedzinę definiującą: typ danych i maksymalny rozmiar, zbiór dozwolonych wartości lub zakres dozwolonych wartości,
 - informację czy są dozwolone / niedozwolone wartości puste,
- Opcjonalnie, dla atrybutu można zdefiniować unikalność wartości.



Atrybuty encji - przykład

- Pracownicy firmy są opisani numerem PESEL, adresem zamieszkania, pensją i opcjonalnie numerem telefonu



BD – wykład 3 (14)

Przykłady atrybutów opisujących każdego pracownika (czyli encji Pracownik) przedstawiono na slajdzie. Identyfikatorem encji jest atrybut PESEL. Identyfikator oznacza się poprzedzając nazwę atrybutu znakiem #. Adres i pensja zdefiniowano jako atrybuty z wartościami obowiązkowymi (gwiazdka przed nazwą atrybutu). Natomiast telefon zdefiniowano jako atrybut mogący przyjmować wartości puste (kółko przed nazwą atrybutu).



Związek

- Związek (asocjacja) reprezentuje powiązania pomiędzy obiektymi świata rzeczywistego
 - klienci posiadają rachunki bankowe
 - studenci otrzymują oceny z egzaminów
- W modelu ER związek łączy encje
- Związek z każdego końca posiada krótki opis ułatwiający interpretację związku

BD – wykład 3 (15)

Kolejnym obiektem modelu ER jest związek, zwany również asocjacją. Reprezentuje on powiązania pomiędzy obiektymi świata rzeczywistego. W modelu ER związek łączy encje. Przykładowo, jeśli klienci posiadają rachunki bankowe, to w modelu ER powiązanie encji Klient z encją Rachunek jest reprezentowane obiektem typu związek. Związek z każdego końca posiada krótki opis ułatwiający interpretację tego związku.

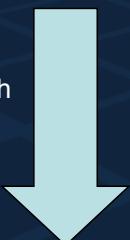


Modelowanie związków (1)

Związki

Pracownicy firmy posiadają różne samochody. Chcemy przechować informację na temat faktu posiadania samochodu przez pracownika.

Identyfikacja
związków posiadających
wspólne własności



Zenon Sikora	posiada	Subaru Forester
Herman Klos	jest własnością	Peugeot 206
Zdzisław Pirat		Opel Astra

Związki pomiędzy encjami

związek

Pracownik

posiada

Samochód

jest własnością

opis związku

BD – wykład 3 (16)

Jako przykład związku rozważmy następujący mikro-opis fragmentu pewnego świata rzeczywistego.

"Pracownicy firmy posiadają różne samochody. Chcemy przechować informację na temat faktu posiadania samochodu przez pracownika."

Na podstawie tego opisu można zbudować prosty model związków-encji. Ponieważ chcemy przechowywać dane na temat pracowników i samochodów, obiektami modelu będą encje Pracownik i Samochód. Encje te będą połączone związkiem. Na slajdzie jest on reprezentowany jako linia przerywana łącząca obie encje. Dodatkowo, oba końce związku są opisane tekstami. Związek ten należy interpretować następująco: pracownik posiada samochód; samochód jest własnością pracownika. Opisy związków powinny być krótkie, zwykle powinny to być czasowniki lub rzeczowniki odczasownikowe.



Modelowanie związków (2)

- Wiemy, że istnieje związek pomiędzy pracownikami a samochodami
- Chcielibyśmy wiedzieć:
 - Ile samochodów może posiadać pracownik?
 - Ile pracowników może posiadać ten sam samochód?
 - Czy każdy samochód musi do kogoś należeć?
 - Czy każdy pracownik musi posiadać samochód?

BD – wykład 3 (17)

Z poprzedniego opisu wiemy, że istnieje związek pomiędzy pracownikami a samochodami.

Chcielibyśmy dodatkowo wiedzieć:

Ile samochodów może posiadać pracownik?

Ile pracowników może posiadać ten sam samochód?

Czy każdy samochód musi do kogoś należeć?

Czy każdy pracownik musi posiadać samochód?



Cechy związku

- Stopień związku
 - unarny (binarny rekursywny)
 - binarny
 - ternarny
 - n-arny
- Typ asocjacji (kardynalność)
 - jeden-do-jeden (1:1)
 - jeden-do-wiele (1:M)
 - wiele-do-wiele (M:N)
- Istnienie (klasa przynależności)
 - opcjonalny
 - obowiązkowy

BD – wykład 3 (18)

Każdy związek posiada trzy cechy, tj. stopień związku, typ asocjacji i istnienie. Stopień związku określa liczbę encji połączonych związkiem. Wyróżnia się związki unarne (łączące encję samą z sobą), binarne (łączące dwie encje), ternarne (łączące trzy encje) i n-arne (łączące n encji). Typ asocjacji, zwany kardynalnością związku, określa ile wystąpień jednej encji może być powiązanych z iloma wystąpieniami innej encji. Wyróżnia się związki 1:1, 1:M, M:N. Istnienie, zwane również klasą przynależności związku określa, czy związek jest opcjonalny, czy obowiązkowy.



Cechy związku - przykład (1)

- Pracownicy firmy posiadają samochody
- W celu udostępnienia miejsca parkingowego należy zarejestrować pracownika i jego samochód
- Każdy pracownik ma prawo parkować tylko jeden konkretny samochód
- Nie każdy pracownik ma samochód
- Zarejestrowany w rejestrze parkingowym samochód na pewno jest własnością jednego pracownika

związek Pracownik-Samochód
stopień związku: binarny

typ asocjacji
Pracownik (1) : Samochód (1)

istnienie
Pracownik może posiadać

istnienie
Samochód musi być własnością

typ asocjacji
Pracownik (1) : Samochód (1)

BD – wykład 3 (19)

Jako przykład rozważmy następujący opis mikro-świata rzeczywistego.

"Pracownicy firmy posiadają samochody. W celu udostępnienia miejsca parkingowego należy zarejestrować pracownika i jego samochód. Każdy pracownik ma prawo parkować tylko jeden konkretny samochód. Nie każdy pracownik ma samochód. Zarejestrowany w rejestrze parkingowym samochód na pewno jest własnością jednego pracownika."



Cechy związku - przykład (2)

- Związek binarny (łączy dwie encje)
- Związek opcjonalny od strony pracownika (linia przerywana)
- Związek obowiązkowy od strony samochodu (linia ciągła)
- Związek 1:1 (1 pracownik posiada 1 samochód)



BD – wykład 3 (20)

Z opisu tego wynika konieczność zbudowania modelu, w którym wystąpią encje: Pracownik i Samochód. Związek obu encji jest binarnym (łączy encję Pracownik i Samochód), opcjonalnym od strony encji Pracownik (pracownik może mieć samochód), obowiązkowy od strony encji Samochód (samochód musi być własnością pracownika), o typie asocjacji 1:1 (jeden pracownik posiada tylko jeden samochód i jeden samochód jest własnością tylko jednego pracownika).

Na diagramie opcjonalność związku jest oznaczana linią przerywaną, a obowiązkowość - linią ciągłą.



Typ asocjacji 1:1 - przykład (1)

Związek binarny jeden-do-jeden (1:1)

Każdy dział musi mieć kierownika, natomiast pracownik może być kierownikiem co najwyżej jednego działu.



BD – wykład 3 (21)

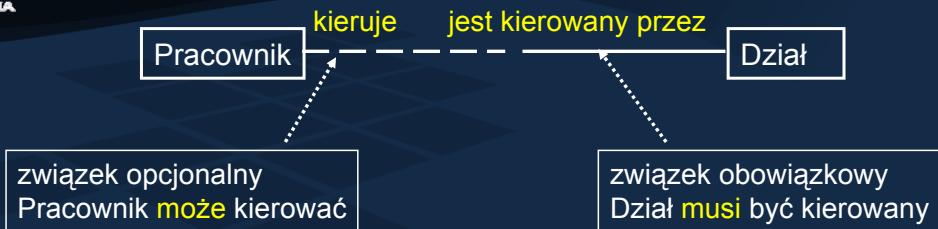
Jako przykład związku binarnego 1:1 rozważmy następujący opis mikroświatu.

"Każdy dział musi mieć kierownika, natomiast pracownik może być kierownikiem co najwyżej jednego działu."

Związek ten łączy encję Pracownik z encją Dział. Jest to związek 1:1 opcjonalny od strony pracownika (linia przerywana) i obowiązkowy - od strony działu (linia ciągła). Oznacza to, że spośród wszystkich pracowników tylko jeden jest kierownikiem konkretnego działu. Istnieją pracownicy, którzy nie są kierownikami żadnych działów. Z drugiej strony, każdy dział musi mieć dokładnie jednego kierownika, którym jest pracownik.



Typ asocjacji 1:1 - przykład (2)



- Interpretacja
 - pracownik może być kierownikiem tylko jednego działu
 - istnieją pracownicy, którzy nie kierują żadnym działem
 - każdy dział musi być kierowany przez dokładnie jednego pracownika

BD – wykład 3 (22)

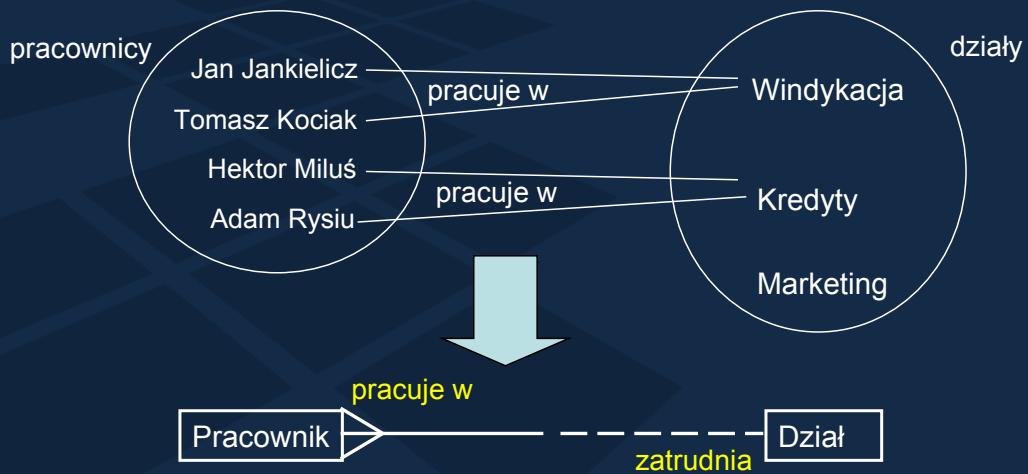
Podsumowanie omawianego przykładu przedstawiono na slajdzie.



Typ asocjacji 1:M (1)

Związek binarny typu jeden-do-wiele (1:M)

Każdy pracownik pracuje dokładnie w jednym dziale. Dział może zatrudniać (ale nie koniecznie) wielu pracowników.



BD – wykład 3 (23)

Jako przykład związku binarnego 1:M rozważmy następujący opis mikroświatu.

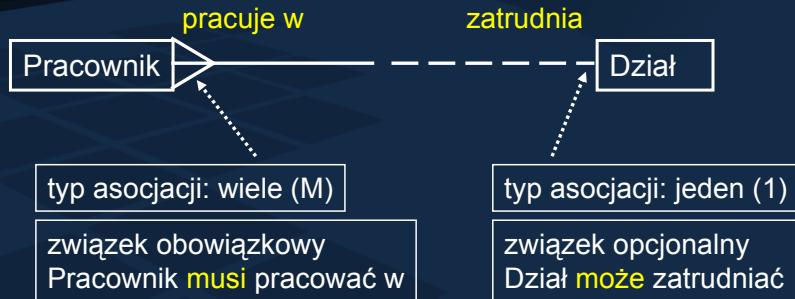
"Każdy pracownik pracuje dokładnie w jednym dziale. Dział może zatrudniać (ale nie koniecznie) wielu pracowników."

Związek ten łączy encję Pracownik z encją Dział. Jest to związek 1:M obowiązkowy od strony pracownika i opcjonalny - od strony działu. Oznacza to, że każdy pracownik musi pracować w jakimś dziale. Wielu pracowników pracuje w tym samym dziale. Z drugiej strony, każdy dział może zatrudniać przynajmniej jednego pracownika. Mogą istnieć działy, które nikogo nie zatrudniają.

Typ asocjacji "wiele" jest oznaczony na diagramie w postaci rozwidlającej się linii dochodzącej do encji, w naszym przykładzie - Pracownik. Jak się domyśliliśmy z poprzednich przykładów, typ asocjacji "jeden" jest reprezentowany jako normalna linia dochodząca do encji, w naszym przykładzie - Dział.



Typ asocjacji 1:M (2)



- Interpretacja
 - każdy pracownik musi pracować w jakimś dziale
 - w jednym dziale pracuje jeden lub wielu pracowników
 - dział może zatrudniać pracowników
 - istnieją działy, które nie zatrudniają pracowników

BD – wykład 3 (24)

Podsumowanie omawianego przykładu przedstawiono na slajdzie.



Typ asocjacji 1:M (3)

- Związek binarny 1:M obustronnie obowiązkowy
 - Drużyna piłkarska musi być złożona z zawodników
 - nie ma drużyny bez zawodników
 - Każdy piłkarz należy do dokładnie jednej drużyny
 - piłkarz, który nie należy do drużyny (nie gra) nie jest piłkarzem



BD – wykład 3 (25)

Jako przykład kolejnego związku binarnego 1:M rozważmy związek encji Piłkarz z encją Drużyna. Jest to związek obustronnie obowiązkowy. Interpretacja tego związku jest następująca: drużyna piłkarska musi być złożona z zawodników. Innymi słowy, nie istnieje drużyna, która nie posiada zawodników. Każdy piłkarz należy do dokładnie jednej drużyny, a piłkarz, który nie należy do drużyny (nie gra) nie jest piłkarzem.



Typ asocjacji 1:M (4)

- Związek binarny 1:M obustronnie obowiązkowy
 - z każdym rachunkiem bankowym musi być związana historia operacji na nim
 - istniejąca operacja została wykonana na konkretnym rachunku
 - nie istnieją operacje nie związanych z rachunkiem



BD – wykład 3 (26)

Jako przykład kolejnego związku binarnego 1:M rozważmy związek encji Rachunek z encją Operacja. Jest to również związek obustronnie obowiązkowy. Interpretacja tego związku jest następująca: z każdym rachunkiem bankowym musi być związana historia (przynajmniej jedna) operacji na nim. Istniejąca operacja musi dotyczyć konkretnego rachunku i nie istnieją operacje nie związanych z rachunkiem.



Typ asocjacji M:N (1)

Związek binarny typu wiele-do-wiele (M:N)

Pracownik może brać udział w jednym lub wielu projektach; może też nie brać udziału w żadnym projekcie. Każdy projekt realizuje przynajmniej jeden pracownika.



BD – wykład 3 (27)

Jako przykład związku binarnego M:N rozważmy następujący opis mikroświatu.

"Pracownik może brać udział w jednym lub wielu projektach; może też nie brać udziału w żadnym projekcie. Każdy projekt realizuje przynajmniej jeden pracownika."

Związek ten łączy encję Pracownik z encją Projekt. Jest to związek M:N opcjonalny od strony pracownika i obowiązkowy - od strony projektu. Oznacza to, że każdy pracownik może realizować jeden lub wiele projektów. Mogą również istnieć pracownicy nie realizujący żadnego projektu. Z drugiej strony, każdy projekt musi być realizowany przez jednego lub wielu pracowników.



Typ asocjacji M:N (2)



- Interpretacja
 - pracownik może brać udział w projekcie
 - istnieją pracownicy nie biorący udziału w żadnym projekcie
 - projekt musi być realizowany przez przynajmniej jednego pracownika
 - w tym samym projekcie może brać udział wielu pracowników

BD – wykład 3 (28)

Podsumowanie omawianego przykładu przedstawiono na slajdzie.



Typ asocjacji M:N (3)

- Związek binarny M:N obustronnie opcjonalny
 - każdy student może należeć do jednej lub wielu organizacji studenckich
 - mogą istnieć studenci nie należący do żadnej organizacji
 - dana organizacja może zrzeszać jednego lub wielu studentów
 - mogą istnieć organizacje, które nie zrzeszają żadnego studenta



BD – wykład 3 (29)

Kolejnym przykładem związku M:N jest związek studenta z organizacją. Jest to związek obustronnie opcjonalny. Interpretacja tego związku jest następująca:

- każdy student może należeć do jednej lub wielu organizacji studenckich,
- mogą istnieć studenci nie należący do żadnej organizacji,
- dana organizacja może zrzeszać jednego lub wielu studentów,
- mogą istnieć organizacje, które nie zrzeszają żadnego studenta.



Atrybuty związku (1)

Związek binarny typu wiele-do-wiele (M:N)

Pracownik może brać udział w jednym lub wielu projektach; może też nie brać udziału w żadnym projekcie. Każdy projekt realizuje przynajmniej jeden pracownika. Dla pracowników, którzy biorą udział w projektach należy zapamiętać ich funkcję, wynagrodzenie oraz daty początku i końca ich udziału w projekcie.



BD – wykład 3 (30)

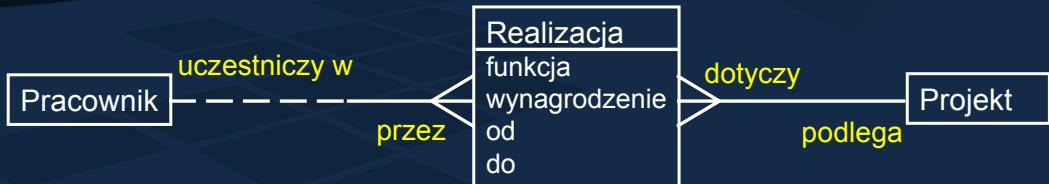
Związek może mieć również swoje atrybuty (cechy). Poniższy opis ilustruje konieczność wprowadzenia związku z atrybutami.

"Pracownik może brać udział w jednym lub wielu projektach; może też nie brać udziału w żadnym projekcie. Każdy projekt realizuje przynajmniej jeden pracownika. Dla pracowników, którzy biorą udział w projektach należy zapamiętać ich funkcję, wynagrodzenie oraz daty początku i końca ich udziału w projekcie."

Z poniższego opisu wynika konieczność wprowadzenia encji Pracownik i Projekt powiązanych tak jak poprzednio związkiem M:N opcjonalnym od strony pracownika i obowiązkowym od strony projektu. Dodatkowo, udział pracownika w projekcie jest opisany funkcją pracownika, wynagrodzeniem oraz datą początku i końca uczestnictwa. Są to cechy związku.



Atrybuty związku (2)



- Jeśli związek posiada dodatkowe cechy \Rightarrow należy wprowadzić dodatkową encję (Realizacja)
- Do encji tej dochodzą **obowiązkowe związki typu wiele**
 - interpretacja obowiązkowości związków
 - jeśli istnieje wystąpienie encji Realizacja, to musi ono dotyczyć jakiegoś projektu i pracownika
 - nie może istnieć realizacja bez pracownika i projektu

BD – wykład 3 (31)

Omawiana notacja Barkera nie umożliwia wiązania atrybutów ze związkami. Konstrukcją modelującą takie przypadki jest dodatkowa encja pośrednia pomiędzy oryginalnym encjami.

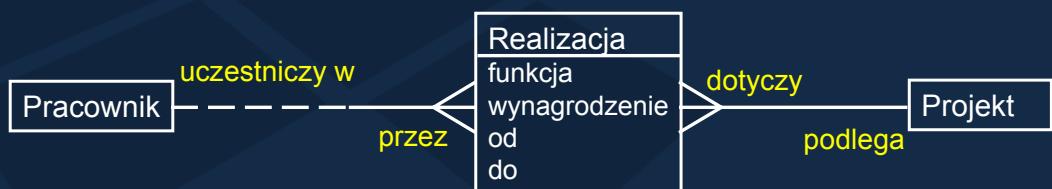
W naszym przykładzie, należy wprowadzić encję Realizacja z atrybutami: funkcja, wynagrodzenie, od, do, reprezentującymi cechy związku. Do encji Realizacja dochodzą związki typu wiele, oba obowiązkowe od strony tej encji. Interpretacja obowiązkowości tych związków jest następująca:

- jeśli istnieje wystąpienie encji Realizacja, to musi ono dotyczyć jakiegoś projektu i pracownika,
- nie może istnieć realizacja bez pracownika i projektu.



Encja słaba

- Encja słaba (weak entity)
 - nie posiada swojego identyfikatora
 - wystąpienia encji mogą istnieć tylko w kontekście wystąpień encji powiązanych z encją słabą
 - konkretne wystąpienie encji Realizacja może wystąpić wyłącznie w kontekście konkretnego pracownika i konkretnego projektu



BD – wykład 3 (32)

W modelu ER wyróżnia się tzw. encje słabe (ang. weak entity). Encja słaba nie posiada swojego identyfikatora, a jej wystąpienia mogą istnieć tylko w kontekście wystąpień encji z nią powiązanych. Przykładowo, konkretne wystąpienie encji Realizacja może wystąpić wyłącznie w kontekście konkretnego pracownika i konkretnego projektu.



Identyfikator encji słabej

- Idenfikatorem encji słabej są wszystkie związki, w które wchodzi ta encja



BD – wykład 3 (33)

Identyfikatorem encji słabej są wszystkie związki, w które wchodzi ta encja z innymi encjami. Związek wchodzący w skład identyfikatora encji jest oznaczany na diagramie jako krótka linia prostopadła do związku umieszczona przy końcu związku. W przykładzie ze slajdu, w skład identyfikatora encji Realizacja wchodzą związki z encją Pracownik i Projekt.



Związek binarny rekursywny (1)

- Określa powiązanie pomiędzy wystąpieniem encji a innym wystąpieniem tej samej encji
- Modelowanie zależności służbowych

Pracownicy posiadają swich kierowników. Istnieją pracownicy, którzy nie są kierownikami.



BD – wykład 3 (34)

Związek binarny rekursywny określa powiązanie pomiędzy określonym wystąpieniem encji a innym wystąpieniem tej samej encji.

Przykładem takiego związku jest model zależności służbowych, w których pracownicy posiadają swoich kierowników. Z jednej strony pracownik może mieć kierownika, tj. nie wszyscy pracownicy mają kierowników. Przykładowo, kierownik nie ma nad sobą kierownika. Z drugiej strony istnieją pracownicy, którzy nie są kierownikami.

Tego typu związek modelujący zależności hierarchiczne musi być opcjonalnym z obu stron. W przeciwnym przypadku, powstałaby hierarchia nieskończona.



Związek binarny rekursywny (2)

- Modelowanie elementów złożonych

Istnieją podzespoły elementarne, niedekomponowalne i podzespoły złożone. Podzespół złożony składa się z kolejnych podzespołów. Każdy z kolejnych podzespołów może być złożony z innych podzespołów. Poziom złożoności podzespołów nie może być dowolny.



BD – wykład 3 (35)

Innym przykładem związku binarnego rekursywnego jest model podzespołów złożonych, dla poniższego opisu mikro-światła.

"Istnieją podzespoły elementarne, niedekomponowalne i podzespoły złożone. Podzespół złożony składa się z kolejnych podzespołów. Każdy z kolejnych podzespołów może być złożony z innych podzespołów. Poziom złożoności podzespołów może być dowolny."

W modelu ze slajdu, encja Podzespół jest powiązana sama z sobą związkiem 1:M opcjonalnym z obu stron. Oznacza to, że każdy podzespół może się składać z wielu innych podzespołów. Z drugiej strony, każdy podzespół może być częścią innego podzespołu złożonego.



Związki ternarne (1)

Związek ternarny

Kierowca może otrzymać mandat za popełnione wykroczenie. Mandat jest wystawiany przez konkretnego policjanta.



BD – wykład 3 (36)

Związek ternarny łączy 3 encje. Jako przykład takiego związku rozważmy model dla poniższego opisu mikro-świata.

"Kierowca może otrzymać mandat za popełnione wykroczenie. Mandat jest wystawiany przez konkretnego policjanta."

W modelu dla tego opisu należy wykorzystać trzy encje: Kierowca, Policjant, Wykroczenie. Związek ternarny łączy wystąpienia tych trzech encji, jak pokazano na slajdzie.



Związki ternarne (2)



- W omawianej notacji Barkera **związek ternarny jest reprezentowany jako encja (Mandat)**
 - do encji Mandat dochodzą związki obowiązkowe
 - jeśli wystawiono mandat to jest on dla konkretnej osoby, został wystawiony przez konkretnego policjanta i dotyczy konkretnego wykroczenia

BD – wykład 3 (37)

W omawianej notacji Barkera związek ternarny jest reprezentowany jako encja (w naszym przykładzie Mandat). Do encji Mandat dochodzą związki obowiązkowe o kardynalności "wiele". Ich interpretacja jest następująca: jeśli wystawiono mandat to jest on dla konkretnej osoby, został wystawiony przez konkretnego policjanta i dotyczy konkretnego wykroczenia.



Związki ternarne przykład rozszerzony



BD – wykład 3 (38)

Poprzedni przykład, rozszerzony o atrybuty encji przedstawia slajd. Jak widać, encja mandat posiada swoje atrybuty, tj. kwota i data wystawienia mandatu.



Związki wyłączne

- Związki wyłączne (exclusive relationships)
 - konkretne wystąpienie encji może w danym momencie wchodzić tylko w jeden z ze związków



BD – wykład 3 (39)

Związki wyłączne (ang. exclusive relationships) stosuje się wtedy, gdy konkretne wystąpienie encji może w danym momencie wchodzić tylko w jeden z ze związków. Jako przykład takiego związku rozważmy model ze slajdu, w którym encja "Rachunek bankowy" wchodzi w związek z encją "Osoba fizyczna" lub "Osoba prawna". Oznacza to, że konkretny rachunek może być własnością albo osoby fizycznej albo osoby prawnej, ale nigdy nie będzie własnością obu tych osób jednocześnie. Związek wyłączny oznacza się w notacji Barkera jako łuk łączący związki wyłączne.



Hierarchia encji / generalizacja

- Związek generalizacji
 - określa, że pewne encje o wspólnym zbiorze atrybutów można uogólnić i stworzyć encję wyższego poziomu \Rightarrow encję generalizacji
- Encje niższego poziomu w hierarchii generalizacji \Rightarrow encje specjalizacji
- Relacja opisująca związki typu generalizacja/specjalizacja pomiędzy encjami \Rightarrow hierarchia generalizacji/specjalizacji lub hierarchia encji

BD – wykład 3 (40)

Kolejną konstrukcją modelu jest związek generalizacji, zwany również hierarchią encji, hierarchią generalizacji, lub hierarchią specjalizacji. Określa on, że pewne encje o wspólnym zbiorze atrybutów można uogólnić i stworzyć encję wyższego poziomu - encję generalizacji, zwaną często nadencją. Encje niższego poziomu w hierarchii generalizacji to tzw. encje specjalizacji, zwane również podencjami.



Hierarchia encji (1)

Dziedziczenie atrybutów

Firma zatrudnia pracowników kontraktowych i godzinowych. Wszyscy pracownicy posiadają pewien zbiór wspólnych atrybutów (PESEL, imię, nazwisko, adres). Pracownicy kontraktowi i godzinowi posiadają specyficzne dla siebie atrybuty. Dla pracowników kontraktowych jest to numer kontraktu, a dla pracowników godzinowych są to: liczba godzin pracy w tygodniu i stawka godzinowa.



BD – wykład 3 (41)

Jako przykład ilustrujący hierarchię encji rozważmy model dla poniższego opisu mikro-świata.

"Firma zatrudnia pracowników kontraktowych i godzinowych. Wszyscy pracownicy posiadają pewien zbiór wspólnych atrybutów (PESEL, imię, nazwisko, adres). Pracownicy kontraktowi i godzinowi posiadają specyficzne dla siebie atrybuty. Dla pracowników kontraktowych jest to numer kontraktu, a dla pracowników godzinowych są to: liczba godzin pracy w tygodniu i stawka godzinowa."

W proponowanym modelu wyróżnia się encję generalizacji o nazwie **Pracownik** i dwie encje specjalizacji: **Kontraktowy** i **Godzinowy**. Encja generalizacji **Pracownik** posiada atrybuty wspólne dla wszystkich pracowników, tj. i kontraktowych i godzinowych. Atrybutami wspólnymi są: PESEL, Imię, Nazwisko. Encja **Kontraktowy** posiada jeden atrybut, który jest specyficzny wyłącznie dla pracowników kontraktowych, tj. numer kontraktu. Encja **Godzinowy** posiada dwa atrybuty specyficzne tylko dla pracowników godzinowych, tj. atrybut liczba godzin i stawka.



Hierarchia encji (2)

- Interpretacja

- podencje dziedziczą wszystkie atrybuty swojej nadencji
- każde wystąpienie nadencji jest zawsze wystąpieniem jednej podencji
- semantyka związku generalizacji oznacza, że każde wystąpienie podencji JEST wystąpieniem nadencji
 - pracownik kontraktowy JEST pracownikiem
 - pracownik godzinowy JEST pracownikiem
- identyfikator nadencji jest wspólny dla wszystkich jej podencji
 - podencje nie posiadają swoich identyfikatorów

Pracownik
PESEL
* Imię
* Nazwisko
Kontraktowy
* Nr kontraktu
Godzinowy
* liczba godzin
* stawka

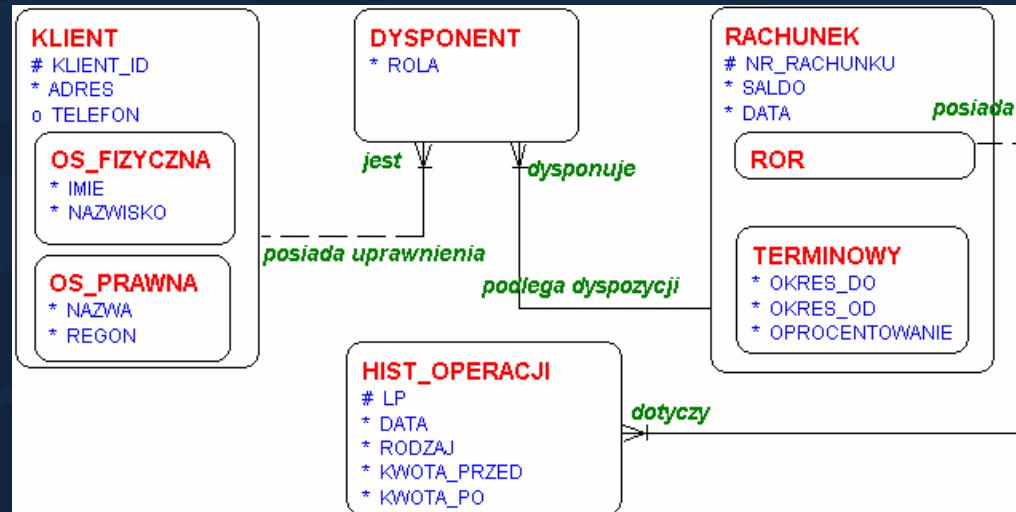
BD – wykład 3 (42)

Interpretacja hierarchii encji jest następująca.

- podencje dziedziczą wszystkie atrybuty swojej nadencji,
- każde wystąpienie nadencji jest zawsze wystąpieniem jednej podencji,
- semantyka związku generalizacji oznacza, że każde wystąpienie podencji JEST wystąpieniem nadencji; przykładowo, pracownik kontraktowy JEST pracownikiem, pracownik godzinowy JEST pracownikiem,
- identyfikator nadencji jest wspólny dla wszystkich jej podencji,
- podencje nie posiadają swoich identyfikatorów.



Hierarchia encji (3)



BD – wykład 3 (43)

Oprócz atrybutów, nadencja może posiadać związki wspólne dla wszystkich jej podencji.

Związek encji **KLIENT** z encją **DYSPONENT** dotyczy zarówno klientów osoby fizycznej jak i klientów osoby prawnej. Jest więc związkiem wspólnym dla wszystkich podencji encji **KLIENT**. Podobnie jest w przypadku związku encji **RACHUNEK** z encją **DYSPONENT**.

Ponadto, podencje mogą wchodzić w związki specyficzne wyłącznie dla siebie. Związek podencji **ROR** z encją **HIST_OPERACJI** jest związkiem specyficznym dla **ROR**, tj. obowiązuje tylko dla tej podencji.



Związki niedozwolone

Przypadek 1.



Przypadek 2.



Przypadek 3.



Przypadek 4.



BD – wykład 3 (44)

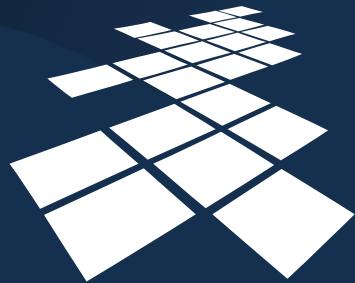
W modelu ER związki przedstawione na slajdzie nie występują. Przypadek 1, czyli związek M:N obustronnie obowiązkowy w praktyce nie występuje.

Przypadek 2, czyli związek rekursywny 1:M obustronnie obowiązkowy jest niepoprawny ponieważ modeluje hierarchię nieskończoną "w górę" i "w dół". Gdyby zamodelować hierarchię jednostek organizacyjnych w taki sposób, wówczas każda jednostka musiałaby mieć jednostkę nadzирующą i podległą. Każda z jednostek nadzorujących musiała mieć kolejną jednostkę nadzирующą itd. Podobnie byłoby w przypadku jednostek podległych.

Przypadek 3 ilustruje błędny model hierarchii nieskończonej "w dół", a przypadek 4 ilustruje błędny model hierarchii nieskończonej "w górę".

Transformacja modelu ER do modelu relacyjnego

Wykład przygotował:
Robert Wrembel



UCZELNIA
ONLINE

BD – wykład 4 (1)



Plan wykładu

- Transformacja encji
- Transformacja związków
- Transformacja hierarchii encji

BD – wykład 4 (2)

Celem wykładu jest omówienie technik transformacji modelu związków-encji do modelu relacyjnego. W ramach wykładu zostaną omówione podstawowe techniki transformacji encji do modelu relacyjnego, transformacji związków i transformacji hierarchii encji.



Pojęcia podstawowe (1)

- Schemat bazy danych
 - zbiór schematów relacji
- Relacja (tabela)
 - dwu-wymiarowa tablica
 - kolumny \Leftrightarrow atrybuty
 - wiersze \Leftrightarrow krotki, rekordy
 - każda krotka reprezentuje wystąpienie encji

BD – wykład 4 (3)

Tytułem przypomnienia podamy podstawowe definicje, do których odwołuje się wykład. Schemat bazy danych jest zbiorem schematów relacji. Relacja, zwana tabelą, jest postrzegana jako dwu-wymiarowa tablica. Kolumny tej tablicy są nazywane atrybutami, a wiersze - krotkami lub rekordami. W modelu relacyjnym każda krotka reprezentuje wystąpienie encji.



Pojęcia podstawowe (2)

- Klucz podstawowy
 - atrybut lub zbiór atrybutów - wybrany spośród kluczy potencjalnych
- Klucz obcy
 - atrybut lub zbiór atrybutów wskazujący na klucz podstawowy innej relacji
 - atrybut lub zbiór atrybutów w relacji B, będący jednocześnie kluczem podstawowym w relacji A
 - należy zaznaczyć, że klucz obcy może odnosić się do klucza podstawowego samej relacji, w której został on zdefiniowany

BD – wykład 4 (4)

Kluczem podstawowym relacji nazywamy atrybut lub zbiór atrybutów jednoznacznie identyfikujący krotkę relacji.

Kluczem obcym nazywamy atrybut lub zbiór atrybutów wskazujący na klucz podstawowy innej relacji. Innymi słowy jest to atrybut lub zbiór atrybutów w relacji B, będący jednocześnie kluczem podstawowym w relacji A. Należy zaznaczyć, że klucz obcy może odnosić się do klucza podstawowego samej relacji, w której został on zdefiniowany.



Transformacja

- Model ER \Leftrightarrow schemat relacyjny
- Transformacja
 - encji z atrybutami
 - związków
 - hierarchii encji

BD – wykład 4 (5)

Transformacja modelu ER jest konieczna, ponieważ jak pamiętamy jest on modelem abstrakcyjnym niezależnym od implementacji. Modelem do którego transformujemy jest model relacyjny. Jest on modelem implementacyjnym baz danych. Transformacji podlegają wszystkie obiekty modelu ER, czyli encje z atrybutami, związki i hierarchie encji.

Omawianie technik transformacji rozpoczęliśmy od przypadku najprostszego, czyli transformacji encji.



Reguły transformacji encji



- Encja \Rightarrow relacja
- Atrybut encji \Rightarrow atrybut relacji
- Typ danych atrybutu encji \Rightarrow typ danych atrybutu relacji
- Identyfikator encji \Rightarrow klucz podstawowy relacji
- Obowiązkowość atrybutów encji \Rightarrow ograniczenie NOT NULL
- Opcjonalność atrybutów encji \Rightarrow ograniczenie NULL
- Pozostałe ograniczenia integr. atrybutów encji \Rightarrow ograniczenia integr. atrybutów relacji

BD – wykład 4 (6)

Reguły transformacji encji są następujące.

1. Encja jest odwzorowywana w relację. Nazwa encji jest odwzorowywana w nazwę relacji. Uwaga: przyjmuje się, że nazwy relacji są rzeczownikami w liczbie mnogiej.
2. Atrybut encji jest odwzorowywany w atrybut relacji. Nazwy atrybutów encji są odwzorowywane w nazwy atrybutów relacji.
3. Typ danych atrybutu encji jest odwzorowywany w odpowiadający mu typ danych atrybutu relacji.
4. Unikalny identyfikator encji jest transformowany w klucz podstawowy relacji.
5. Obowiązkowość atrybutów encji jest reprezentowana w relacji w postaci ograniczenia NOT NULL zdefiniowanego na atrybucie relacji odpowiadającym atrybutowi encji.
6. Opcjonalność atrybutów encji jest reprezentowana w relacji w postaci ograniczenia NULL zdefiniowanego na atrybucie relacji odpowiadającym atrybutowi encji.
7. Ograniczenia integralnościowe dla atrybutów encji (unikalność, zawężenie dziedziny) są transformowane do odpowiadających im ograniczeń integralnościowych relacji. (Ograniczenia integralnościowe atrybutów encji zostały omówione w wykładzie 3, a ograniczenia integralnościowe atrybutów relacji - w wykładzie 2).



Reguły transformacji związków

- Związek binarny 1:1 \Rightarrow klucz obcy we wskazanej tabeli
- Związek unarny 1:1 \Rightarrow klucz obcy w tej samej tabeli
- Związek binarny 1:M \Rightarrow klucz obcy w tabeli po stronie "wiele"
- Związek binarny M:N \Rightarrow tabela
- Związek unarny M:N \Rightarrow tabela

BD – wykład 4 (7)

Przypadkiem bardziej złożonym jest transformacja związków.

Związek binarny 1:1 transformuje się do klucza obcego we wskazanej tabeli.

Związek unarny 1:1 transformuje się do klucza obcego w tej samej tabeli.

Związek binarny 1:M transformuje się do klucza obcego w tabeli po stronie "wiele".

Związek binarny M:N transformuje się do tabeli.

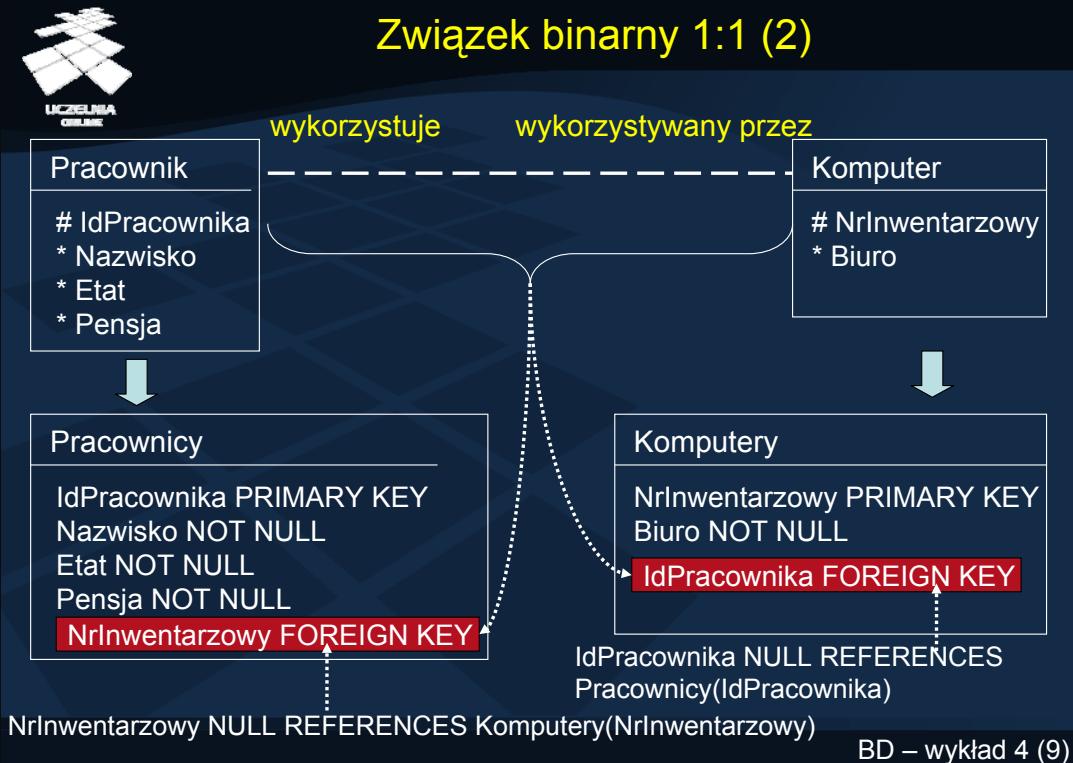
Związek unarny M:N transformuje się do tabeli.

Wymienione przypadki transformacji związków zostaną dokładnie omówione i zilustrowane przykładami w dalszej części wykładu.



Związek binarny 1:1 jednostronnie obowiązkowy transformuje się do klucza obcego w tabeli po stronie związku obowiązkowego. Ograniczenie integralnościowe jest definiowane dla atrybutu klucza obcego. Klucz ten nie może przyjmować wartości pustych.

W przykładzie ze slajdu, z encji **Pracownik** powstaje tabela **Pracownicy**, a z encji **Samochód** - tabela **Samochody**. Związek pomiędzy tymi dwoma encjami jest transformowany do klucza obcego **IdPracownika** w tabeli **Samochody**. Klucz ten wskazuje na klucz podstawowy w tabeli **Pracownicy**, czyli **IdPracownika**. Wartość atrybutu klucza obcego musi być zawsze określona ponieważ związek jest obowiązkowy od strony encji **Samochód**.



Związek binarny 1:1 obustronnie opcjonalny transformuje się do klucza obcego w tabeli o mniejszym rozmiarze. Ograniczenie integralnościowe jest definiowane dla atrybutu klucza obcego. Atrybut ten może przyjmować wartości puste, ponieważ związek jest opcjonalny.

W przykładzie ze slajdu, z encji Pracownik powstaje tabela Pracownicy, a z encji Komputer - tabela Komputery. Związek jest transformowany do klucza obcego IdPracownika w tabeli Komputery. Klucz ten wskazuje na klucz podstawowy w tabeli Pracownicy, czyli IdPracownika. Klucz obcy może przyjmować wartości puste.



Związek binarny 1:1 (3)

1.	Pracownicy IdPracownika PRIMARY KEY ... NrInwentarzowy FOREIGN KEY	Komputery NrInwentarzowy PRIMARY KEY Biuro NOT NULL IdPracownika FOREIGN KEY
2.	Pracownicy IdPracownika PRIMARY KEY ... IdPracownika FOREIGN KEY	Komputery NrInwentarzowy PRIMARY KEY Biuro NOT NULL IdPracownika FOREIGN KEY
3.	Pracownicy IdPracownika PRIMARY KEY ... NrInwentarzowy FOREIGN KEY	Komputery NrInwentarzowy PRIMARY KEY Biuro NOT NULL

BD – wykład 4 (10)

Możliwe przypadki transformacji związku 1:1 obustronnie opcjonalnego przedstawia slajd. Przypadek 1 jest stosowany niezwykle rzadko. Polega on na umieszczeniu kluczy obcych w obu tabelach wynikowych. Przypadek 2 został omówiony na poprzednim slajdzie. Przypadek trzeci polega na umieszczeniu klucza obcego w tabeli Pracownicy.



Związek 1:M (1)

- Klucz obcy dodawany do relacji po stronie "wiele"
- Ograniczenia referencyjne definiowane dla klucza obcego
- Obowiązkowość związku po stronie "wiele" \Rightarrow ograniczenie NOT NULL definiowane na kluczu obcym
- Opcjonalność związku po stronie "wiele" \Rightarrow ograniczenie NULL definiowaną na kluczu obcym relacji
- Opcjonalność lub obowiązkowość związku po stronie "jeden" nie jest odwzorowywana w modelu relacyjnym

BD – wykład 4 (11)

Reguły transformacji związku 1:M są następujące.

1. Klucz obcy jest dodawany do relacji po stronie "wiele" niezależnie od opcjonalności, czy obowiązkowości tego związku.
2. Ograniczenia integralnościowe referencyjne (tj. definiujące klucz obcy) są definiowane dla atrybutu reprezentującego klucz obcy.
3. Obowiązkowość związku po stronie "wiele" jest reprezentowana przez ograniczenie integralnościowe NOT NULL definiowane na kluczu obcym relacji.
4. Opcjonalność związku po stronie "wiele" jest reprezentowana przez ograniczenie integralnościowe NULL definiowaną na kluczu obcym relacji.
5. Opcjonalność lub obowiązkowość związku po stronie "jeden" nie jest odwzorowywana w modelu relacyjnym.



Przykład ze slajdu ilustruje sposób transformacji binarnego związku 1:M jednostronnie obowiązkowego. Z encji Pracownik powstaje tabela Pracownicy, a z encji Dział - tabela Działy. Klucz obcy jest dodawany do tabeli Pracownicy (strona "wiele") i wskazuje on na klucz podstawowy tabeli Działy, czyli IdDziału. Należy zwrócić uwagę, że klucz obcy IdDziału posiada ograniczenie NOT NULL ponieważ związek jest obowiązkowy od strony "wiele".



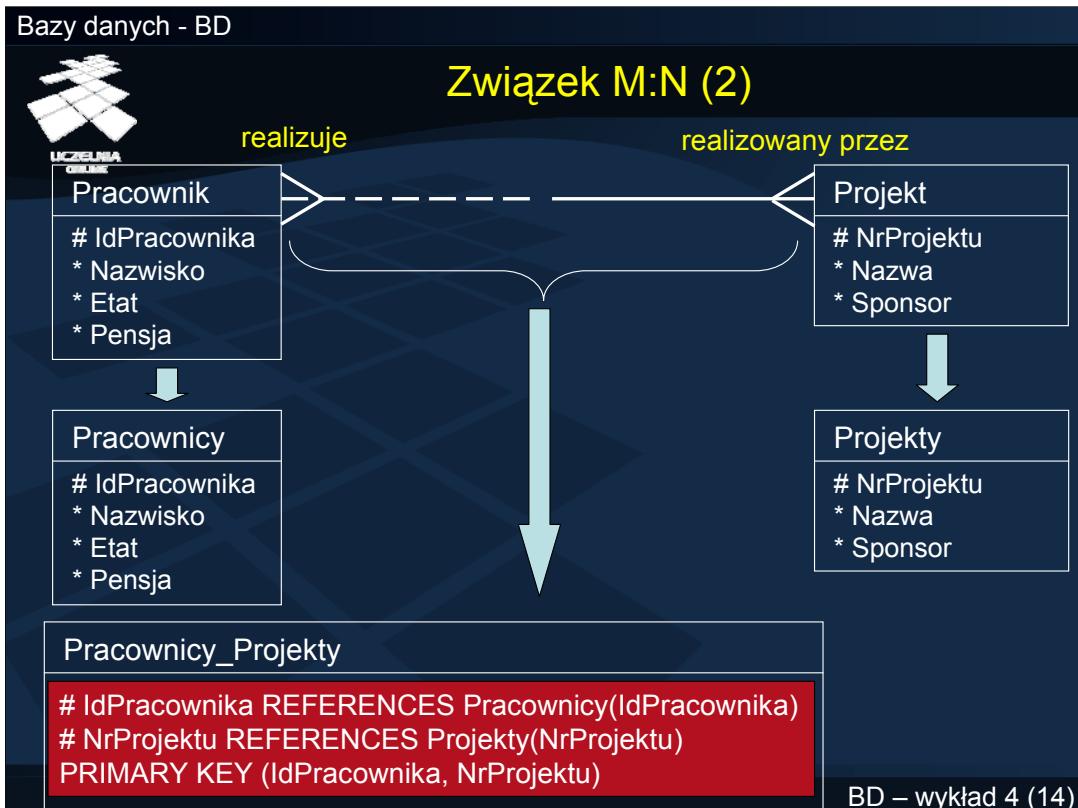
Związek M:N (1)

- Reprezentowany poprzez dodatkową relację
- Nazwa relacji jest złączeniem nazw relacji powstały z encji
- Relacja zawiera klucze obce wskazujące na klucze podstawowe relacji powstały z powiązanych encji
- Ograniczenia referencyjne definiowane dla kluczy obcych
- Klucze obce tworzą klucz podstawowy relacji

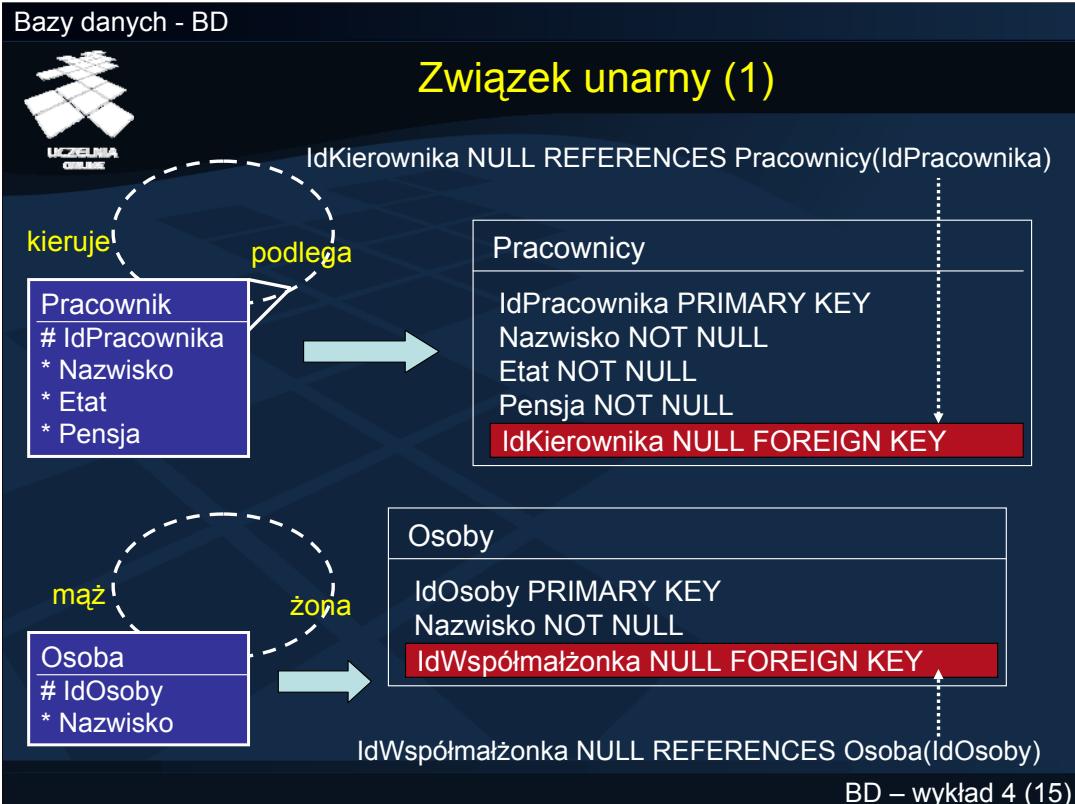
BD – wykład 4 (13)

Reguły transformacji związku M:N są identyczne zarówno dla związków jednostronne obowiązkowych, jak i obustronne opcjonalnych. Reguły te są następujące.

1. Związek M:N jest reprezentowany w modelu relacyjnym poprzez dodatkową relację.
2. Nazwa relacji reprezentującej związek M:N jest złączeniem nazw relacji powstały z encji połączonych tym związkiem.
3. Relacja dodatkowa zawiera klucze obce wskazujące na klucze podstawowe relacji powstały z powiązanych encji.
4. Ograniczenia referencyjne są definiowane dla kluczy obcych.
5. Klucze obce tworzą klucz podstawowy relacji. W konsekwencji, ich wartości nigdy nie będą puste.



Poniższy przykład ilustruje sposób transformacji binarnego związku M:N jednostronnie obowiązkowego. Z encji Pracownik powstaje tabela Pracownicy, a z encji Projekt - tabela Projekty. Ze związku powstaje tabela pośrednia o nazwie Pracownicy_Projekty. Zawiera ona dwa klucze obce. Jeden wskazuje na klucz podstawowy tabeli Pracownicy, czyli IdPracownika, a drugi - na klucz podstawowy tabeli Projekty, czyli NrProjektu. Oba klucze obce stanowią klucz podstawowy tabeli Pracownicy_Projekty. Oznacza to, że ich wartości nigdy nie mogą być puste.



Związek unarny 1:1 lub 1:M obustronnie opcjonalny transformuje się do klucza obcego w tej samej tabeli. W pierwszym przykładzie ze slajdu, z encji Pracownik powstaje tabela Pracownicy. Zawiera ona klucz obcy IdKierownika wskazujący na IdPracownika w tej samej tabeli. Należy zwrócić uwagę, że wartość klucza obcego może być pusta ponieważ związek jest opcjonalny od strony "wiele".

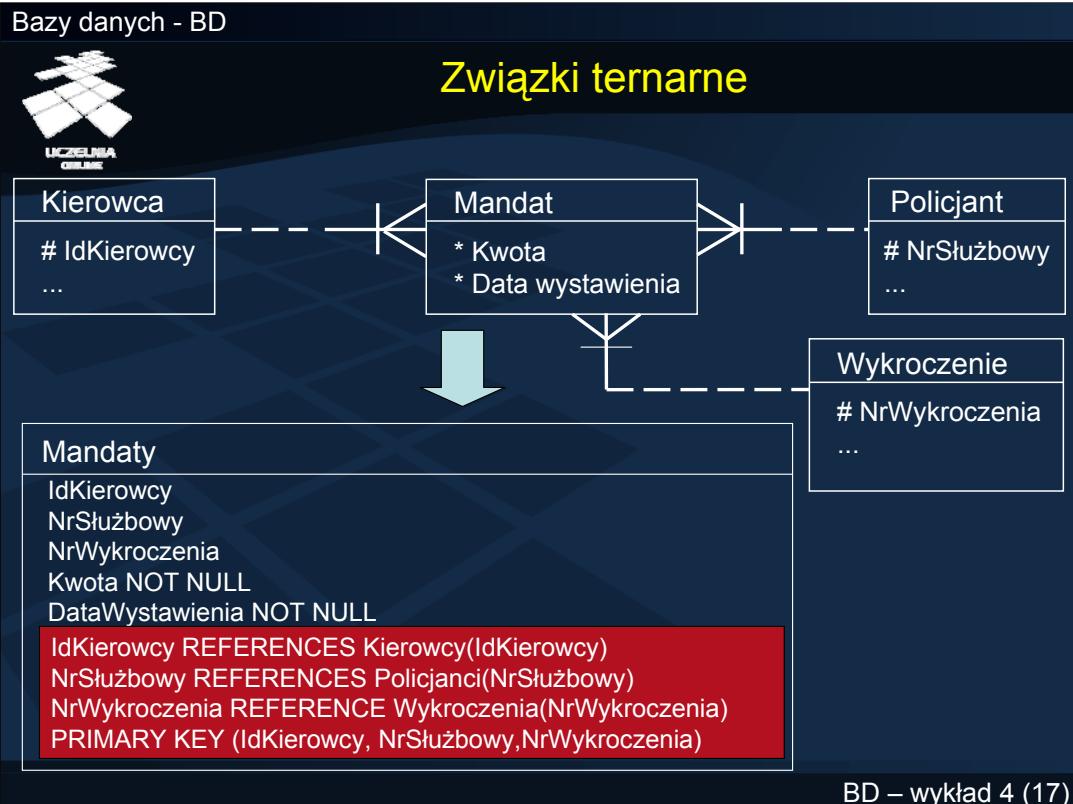
W drugim przykładzie, w tabeli Osoby powstaje klucz obcy IdWspółmałżonka wskazujący na IdOsoby w tej samej tabeli. Ponieważ związek jest opcjonalny, więc klucz obcy może przyjmować wartości puste.

Związek unarny (2)



BD – wykład 4 (16)

Związek unarny M:N obustronnie opcjonalny jest transformowany do tabeli pośredniej. W przykładzie ze slajdu, z encji Lek powstaje tabela Leki, a związek jest transformowany do tabeli o nazwie Zastępni. Tabela ta posiada dwa klucze obce (atrybut IdLeku1 i IdLeku2), oba wskazują na klucz podstawowy tabeli Leki, czyli na atrybut IdLeku. Oba klucze obce wchodzą w skład klucza podstawowego tabeli Zastępni.



Związek ternarny transformuje się w sposób identyczny jak związek 1:M. W przykładzie ze slajdu, z encji Mandat powstaje tabela Mandaty. Zawiera ona 3 klucze obce: IdKierowcy, NrSłужbowy, NrWykroczenia wskazujące odpowiednio na: IdKierowcy w tabeli Kierowcy, NrSłужbowy w tabeli Policjanci, NrWykroczenia w tabeli Wykroczenia. Te trzy klucze obce wchodzą w skład klucza podstawowego tabeli Mandaty, ponieważ transformowana encja Mandat była słaba.



Hierarchia encji

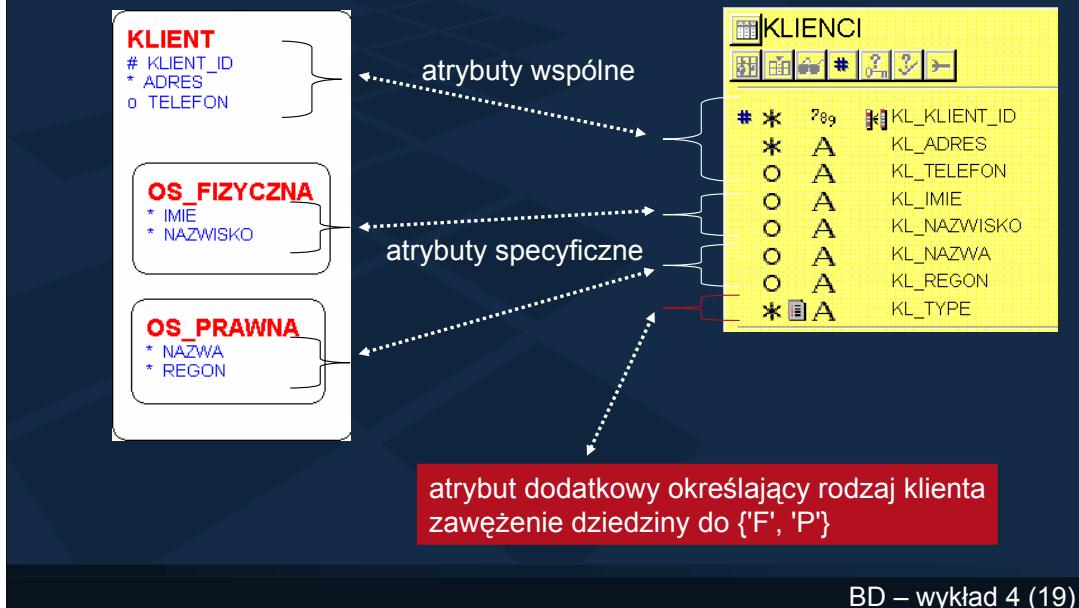
- Schemat 1: jedna wspólna tabela
- Schemat 2: dla każdej podencji tworzona tabela zawierająca atrybuty wspólne i specyficzne
- Schemat 3:
 - dla atrybutów wspólnych tworzona tabela wspólna
 - dla każdej podencji tworzona osobna tabela z kluczem i atrybutami specyficznymi
 - tabela wspólna i tabele powstałe z podencji powiązane ograniczeniami referencyjnymi

BD – wykład 4 (18)

Hierarchię encji do modelu relacyjnego można przetransformować na 3 sposoby. Sposób (schemat) 1 polega na utworzeniu jednej tabeli ze wszystkimi atrybutami i kluczami obcymi, tj. wspólnymi i specyficznymi dla podencji. Sposób (schemat) 2 polega na utworzeniu osobnej tabeli dla każdej podencji. Każda z tabel zawiera atrybuty wspólne i specyficzne dla określonej encji. Sposób (schemat) 3 polega na utworzeniu osobnej tabeli na atrybuty wspólne i osobnej tabeli dla każdej podencji. Tabele powstałe z podencji zawierają klucz podstawowy i atrybuty specyficzne. Tabela wspólna i tabele powstałe z podencji są połączone ograniczeniami referencyjnymi.



Transformacja hierarchii generalizacji - schemat 1

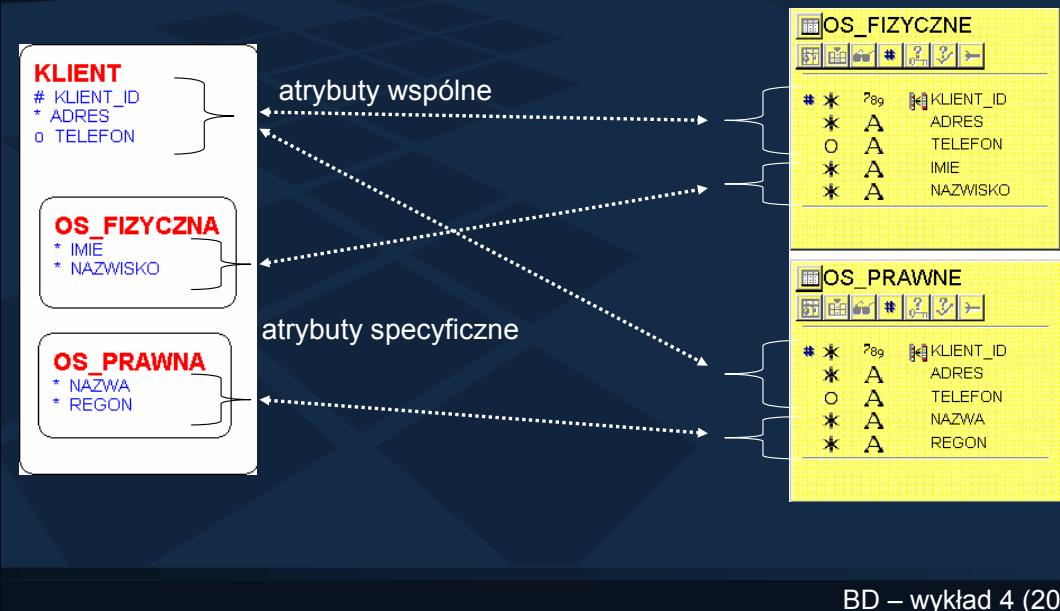


BD – wykład 4 (19)

Przykład ze slajdu ilustruje sposób transformacji hierarchii encji według schematu 1. Powstała tabela Klienci zawiera wszystkie atrybuty wspólne i wszystkie atrybuty specyficzne dla wszystkich podencji. Atrybuty specyficzne w tabeli mogą przyjmować wartości puste zawsze, niezależnie od ich definicji w pod-encjach. Wynika to z faktu, że rekord w tabeli klienci opisuje albo klienta fizycznego albo prawnego. Jeśli rekord opisuje klienta fizycznego, to atrybuty klienta prawnego pozostają puste i odwrotnie. Dodatkowo, w tabeli Klienci jest tworzony atrybut z wartością obowiązkową (KL_TYPE), którego wartościami mogą być albo 'F' albo 'P'. Wartość tego atrybutu określa czy rekord opisuje klienta fizycznego ('F'), czy prawnego ('P'). Atrybut ten jest przydatny przy wyszukiwaniu klientów określonego typu.



Transformacja hierarchii generalizacji - schemat 2

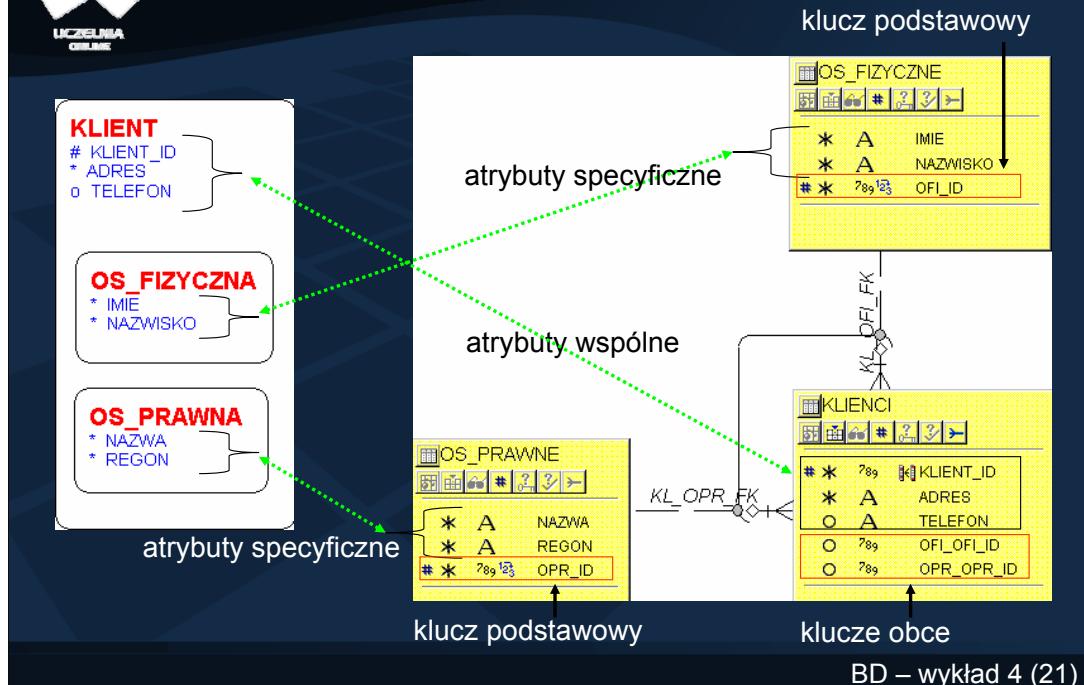


BD – wykład 4 (20)

Przykład ze slajdu ilustruje sposób transformacji hierarchii encji według schematu 2. Z encji OS_FIZYCZNA powstaje tabela OS_FIZYCZNE. Zawiera ona wszystkie atrybuty wspólne encji Klient i wszystkie atrybuty specyficzne encji OS_FIZYCZNA. Opcjonalność/obowiązkowość wartości atrybutów encji przenosi się bezpośrednio na opcjonalność/obowiązkowość atrybutów tabeli OS_FIZYCZNE. W podobny sposób jest transformowana encja OS_PRAWNA.



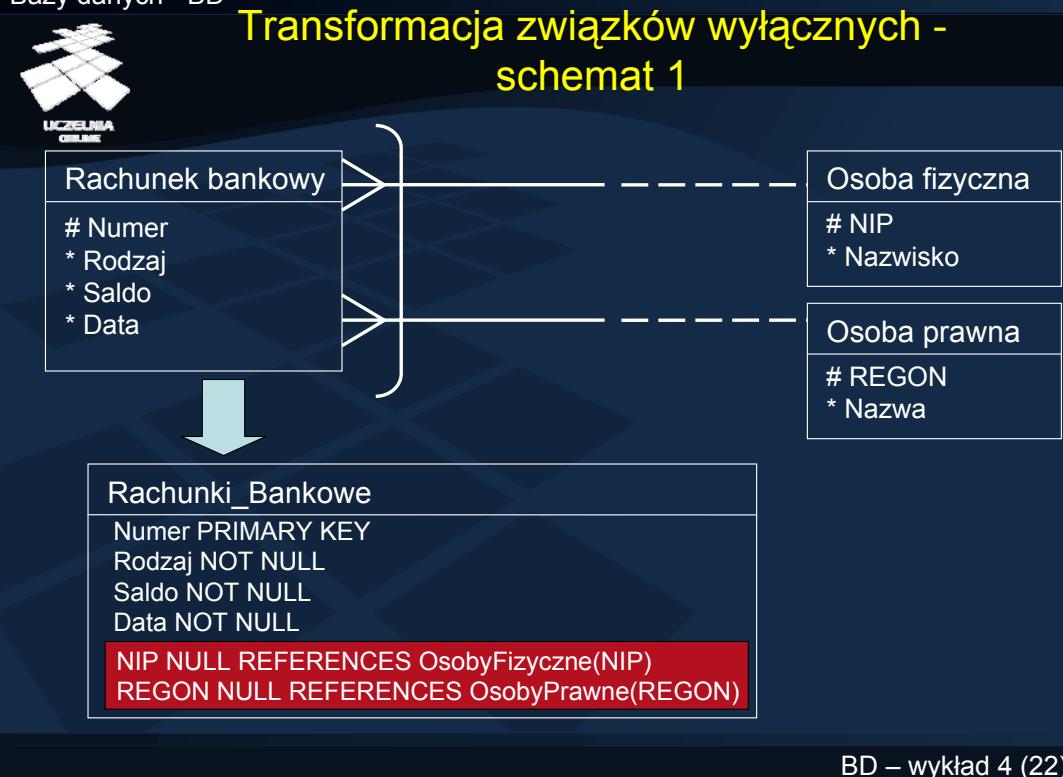
Transformacja hierarchii generalizacji - schemat 3



Przykład ze slajdu ilustruje sposób transformacji hierarchii encji według schematu 3. Z encji OS_FIZYCZNA powstaje tabela OS_FIZYCZNE, która zawiera wszystkie atrybuty specyficzne ze swojej encji i atrybut OFI_ID, który jest kluczem podstawowym tabeli OS_FIZYCZNE. Z encji OS_PRAWNA powstaje tabela OS_PRAWNE, która zawiera wszystkie atrybuty specyficzne ze swojej encji i atrybut OPR_ID, który jest kluczem podstawowym tabeli OS_PRAWNE.

Z atrybutów wspólnych encji Klient powstaje tabela KLIENCI. Dodatkowo, tabela ta posiada dwa klucze obce OFI_OFI_ID i OPR_OPR_ID, z wartościami opcjonalnymi. Pierwszy z nich wskazuje na klucz podstawowy tabeli OS_FIZYCZNE, a drugi - na klucz podstawowy tabeli OS_PRAWNE. Dla danego rekordu w tabeli KLIENCI, tylko jeden klucz obcy może przyjąć wartość, ponieważ rekord w tabeli KLIENCI opisuje albo osobę prawną albo fizyczną.

Transformacja związków wyłącznych - schemat 1

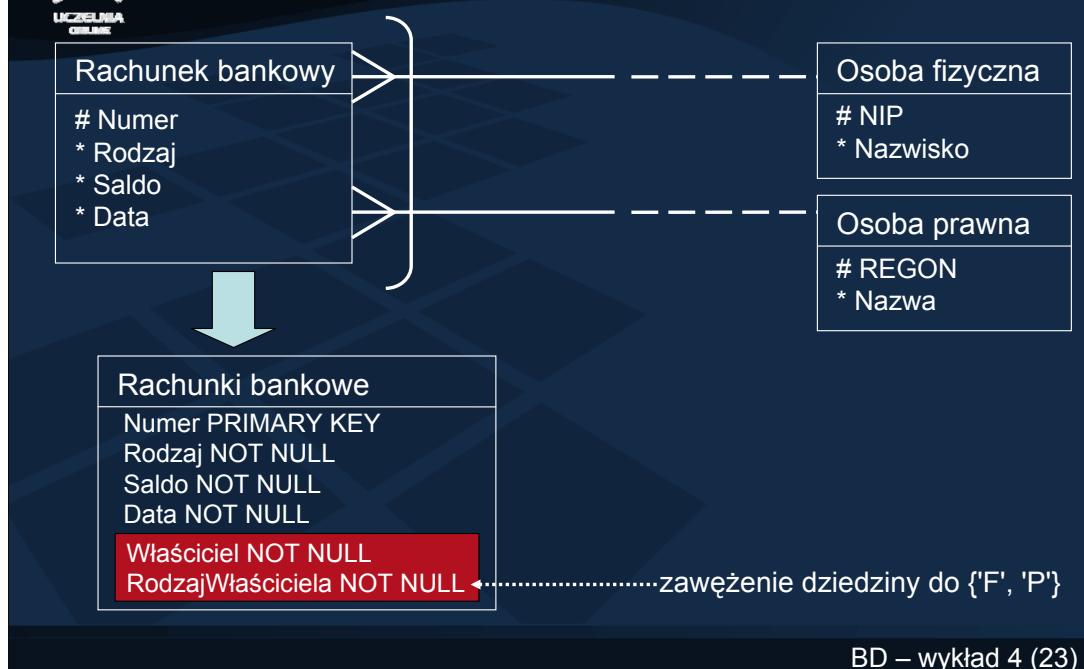


BD – wykład 4 (22)

Transformacja związków wyłącznych jest podobna do transformacji związku 1:M. Z tą tylko różnicą, że klucze obce mogą przyjmować wartości puste.

Jako przykład rozważmy encję Rachunek bankowy powiązaną związkami wyłącznymi z encją Osoba fizyczna i Osoba prawna. Z encji Rachunek bankowy powstaje tabela Rachunki_Bankowe, która posiada dwa klucze obce, jeden wskazuje na klucz podstawowy tabeli Osoby_Fizyczne, a drugi - na klucz podstawowy tabeli Osoby_Prawne. Oba klucze obce mogą przyjmować wartości puste, pomimo, że zвязki z których powstały są obowiązkowe od strony "wiele". Dzieje się tak dla tego, że dany rekord rachunku bankowego jest albo związany z osobą fizyczną albo z osobą prawną, więc tylko jeden klucz obcy przyjmie wartość.

Transformacja związków wyłącznych - schemat 2

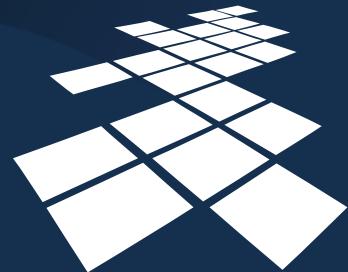


BD – wykład 4 (23)

Alternatywny sposób transformacji związków wyłącznych przedstawiono na slajdzie. Różni się on od poprzedniego tym, że w tabeli Rachunki_Bankowe jest atrybut Właściciel, który może przyjąć wartość klucza podstawowego rekordu albo z tabeli Osoby_Fizyczne albo z tabeli Osoby_Prawne; dla atrybutu tego zdefiniowanie ograniczenia referencyjnego jest niemożliwe. Ponadto, w tabeli Rachunki_Bankowe jest atrybut RodzajWłaściciela, który może przyjąć jedną z dwóch wartości 'F' lub 'P'. Służy on do określania rodzaju osoby, na którą wskazuje wartość atrybutu Właściciel. Oba atrybuty nie mogą przyjmować wartości pustych.

Normalizacja schematów logicznych relacji

Wykład przygotował:
Tadeusz Morzy



UCZELNIA
ONLINE

BD – wykład 5

Celem niniejszego wykładu jest przedstawienie i omówienie procesu normalizacji. Proces normalizacji traktujemy jako proces, podczas którego schematy relacji posiadające niepożądane cechy są dekomponowane na mniejsze schematy relacji o pożądanych własnościach.

Wykład rozpoczniemy od krótkiego przykładu motywacyjnego, ilustrującego problem. Następnie, wprowadzimy pojęcie zależności funkcyjnych stanowiących punkt wyjścia procesu normalizacji. Następnie przejdziemy do omówienia kolejnych postaci normalnych.



Motywacja (1)

- Dana jest następująca relacja Dostawcy:

Nazwisko	Adres	Produkt	Cena
Kowalski	ul. Krucza 10	chipsy	1,50
Kowalski	ul. Krucza 10	orzeszki	3,50
...
Kowalski	ul. Krucza 10	gruszki	4,50
Nowak	ul. Malwowa 4	chipsy	2,00
Nowak	ul. Malwowa 4	orzeszki	4,00
...

BD – wykład 5 (2)

Rozważmy następujący przykład. Dana jest następująca relacja Dostawcy, jak na slajdzie, składająca się z 4 atrybutów. Założymy, że atrybut nazwisko jest unikalny, to jest nie ma dwóch dostawców o tym samym nazwisku. Relacja Dostawcy zawiera informacje o dostawcach (o ich adresach), dostarczanych produktach i cenach dostarczanych produktów.



Motywacja (2)

- Założmy, że trybut **Nazwisko** jest unikalny, tj. nie ma dwóch dostawców o tym samym nazwisku.
- Cechy relacji **Dostawca**:
 - redundancja danych - problem spójności danych
 - anomalia wprowadzania danych
 - anomalia usuwania danych
 - anomalia uaktualniania danych
- Rozwiązaniem: dekompozycja relacji Dostawca na dwie relacje: Dostawca i Dostawy

BD – wykład 5 (3)

Analizując relację Dostawcy zauważmy, że relacja ta charakteryzuje się następującymi cechami. Po pierwsze, obserwujemy redundancję danych – adres dostawcy jest pamiętany tyle razy ile różnych produktów dany dostawca dostarcza. Problem redundancji danych nie sprawdza się do problemu zajętości pamięci, aktualnie pamięci są bardzo tanie, lecz problemu potencjalnej niespójności danych. W momencie zmiany adresu dostawcy, zmiana ta musi być odzwierciedlona we wszystkich krotkach zawierających adres dostawcy. W przeciwnym razie pojawi się problem spójności danych. Po drugie, obserwujemy tzw. anomalię wprowadzania danych. Założmy, że chcemy wprowadzić informację o nowym dostawcy, tj. jego nazwisko i adres. Niestety, informacji tej nie można wprowadzić do relacji Dostawca tak długo, jak długo dostawca nie dostarcza żadnych produktów. Po trzecie, obserwujemy anomalię usuwania danych. Założmy, że rezygnujemy z usług dostawcy Nowak. Usuwając informację o dostawach Nowaka mimo woli usuwamy informacje o samym dostawcy Nowak. Wreszcie, obserwujemy anomalię uaktualniania danych. Aktualizując adres dostawcy, jak już wspominaliśmy, aktualizację tę musimy wprowadzić do wszystkich krotek zawierających adres dostawcy.

Reasumując, schemat relacji Dostawca posiada szereg niepożądanych własności, które w późniejszym czasie będą utrudniały przygotowanie aplikacji operującej na tej relacji. Zauważmy, że rozwiązaniem wszystkich omówionych problemów jest dekompozycja relacji Dostawca na dwie relacje: Dostawca i Dostawy.



Motywacja (3)

Dostawca

Nazwisko	Adres
Kowalski	ul. Krucza 10
...	...
...	...
...	...
Nowak	ul. Małwowa 4
...	...

Dostawy

Nazwisko	Produkt	Cena
Kowalski	chipsy	1,50
Kowalski	orzeszki	3,50
Kowalski	gruszki	4,50
Nowak	chipsy	2,00
Nowak	orzeszki	4,00
...

Dekompozycja bez utraty informacji

BD – wykład 5 (4)

Relacja Dostawca zawiera informacje o dostawcach, natomiast relacja Dostawy zawiera informacje o dostarczanych produktach i ich cenach. Zauważmy, że w przypadku relacji Dostawca adres dostawcy jest pamiętany tylko w jednej krotce – brak redundancji danych. Zauważmy również, że dekompozycja rozwiązuje problem anomalii wstawiania – informacje o nowym dostawcy możemy wstawić do relacji Dostawca, nawet jeżeli dostawca ten nie dostarcza żadnych produktów. Dekompozycja ta rozwiązuje również problem anomalii usuwania – usunięcie informacji o dostawach z relacji Dostawy nie pociąga za sobą usunięcia informacji o samych dostawcach. Dekompozycja rozwiązuje również problem anomalii aktualizacji – zmiana adresu dostawcy dotyczy wyłącznie jednej krotki. Zauważmy, że dekompozycja relacji Dostawca na relacje Dostawca i Dostawy jest dekompozycją bez utraty informacji w tym sensie, że łącząc relację Dostawca i Dostawy wg. atrybutu połączniowego Nazwisko możemy odtworzyć oryginalną zawartość relacji Dostawca.



Zależności funkcyjne (1)

- **Zależność funkcyjna (FD)**

Dana jest relacja r o schemacie R . X, Y są podzbiorami atrybutów R . W schemacie relacji R , X wyznacza funkcjnie Y , lub Y jest funkcjnie zależny od X , co zapisujemy $X \rightarrow Y$, wtedy i tylko wtedy, jeżeli dla dwóch dowolnych krotek t_1, t_2 takich, że $t_1[X] = t_2[X]$ zachodzi zawsze $t_1[Y] = t_2[Y]$, gdzie $t_i[A]$ oznacza wartość atrybutu A krotki t_i .

- Przykłady:

- 1. Nazwisko \rightarrow Adres
- 2. {Nazwisko, Twarz} \rightarrow Cena

BD – wykład 5 (5)

Jak już wspomnieliśmy we wstępie, punktem wyjścia procesu normalizacji jest informacja o zależnościach funkcyjnych występujących w relacjach. Zależność funkcyjną definiujemy następująco:

Dana jest relacja r o schemacie R . X, Y są podzbiorami atrybutów R . W schemacie relacji R , X wyznacza funkcjnie Y , lub Y jest funkcjnie zależny od X , co zapisujemy $X \rightarrow Y$, wtedy i tylko wtedy, jeżeli dla dwóch dowolnych krotek t_1, t_2 takich, że $t_1[X] = t_2[X]$ zachodzi zawsze $t_1[Y] = t_2[Y]$, gdzie $t_i[A]$ oznacza wartość atrybutu A krotki t_i .

Przykładowo, relacja Dostawca zawiera dwie zależności funkcyjne: Nazwisko \rightarrow Adres i {Nazwisko, Twarz} \rightarrow Cena.

Z pierwszej zależności funkcyjnej wynika, że adres dostawcy jednoznacznie zależy od nazwiska dostawcy. Natomiast z drugiej zależności funkcyjnej wynika, że cena towaru zależy od kombinacji atrybutów Nazwisko i Twarz.



Zależności funkcyjne (2)

- Zależność funkcyjna określa zależność pomiędzy atrybutami. Jest to własność semantyczna, która musi być spełniona dla dowolnych wartości krotek relacji.
- Relacje które spełniają nałożone zależności funkcyjne nazywamyinstancjami legalnymi
- Zależność funkcyjna jest własnością schematu relacji R , a nie konkretnego wystąpienia relacji
- Z zależności funkcyjnej wynika, że jeżeli $t_1[X] = t_2[X]$ i $X \rightarrow Y$, to zachodzi zawsze $t_1[Y] = t_2[Y]$

BD – wykład 5 (6)

Należy podkreślić, że zależność funkcyjna określa zależność pomiędzy atrybutami. Jest to własność semantyczna, która musi być spełniona dla dowolnych wartości krotek relacji.

Relacje które spełniają nałożone zależności funkcyjne nazywamyinstancjami legalnymi. Zależność funkcyjna jest własnością schematu relacji R , a nie konkretnego wystąpienia relacji. Jeżeli zmieni się relacja, to zależność funkcyjna nadal pozostaje ważna. Zauważmy również, że z zależności funkcyjnej wynika, że jeżeli $t_1[X] = t_2[X]$ i $X \rightarrow Y$, to zachodzi zawsze $t_1[Y] = t_2[Y]$.



- Proces normalizacji relacji można traktować jako proces, podczas którego schematy relacji posiadające pewne niepożądane cechy są dekomponowane na mniejsze schematy relacji o pożądanych własnościach
- Proces normalizacji musi posiadać trzy dodatkowe własności:

Własność zachowania atrybutów - żaden atrybut nie zostanie zagubiony w trakcie procesu normalizacji

Własność zachowania informacji - dekompozycja relacji nie prowadzi do utraty informacji

Własność zachowania zależności - wszystkie zależności funkcyjne są reprezentowane w pojedynczych schematach relacji

BD – wykład 5 (7)

Przejedziemy teraz do przedstawienia procesu normalizacji.

Proces normalizacji relacji można traktować jako proces, podczas którego schematy relacji posiadające pewne niepożądane cechy są dekomponowane na mniejsze schematy relacji o pożądanych własnościach. Proces normalizacji musi posiadać trzy dodatkowe własności:

Własność zachowania atrybutów - żaden atrybut nie zostanie zagubiony w trakcie procesu normalizacji.

Własność zachowania informacji - dekompozycja relacji nie prowadzi do utraty informacji, tj. łącząc zdekomponowane relacje możemy odtworzyć oryginalną relację.

Własność zachowania zależności - wszystkie zależności funkcyjne są reprezentowane w pojedynczych schematach relacji.

Proces normalizacji schematu relacji polega na sprawdzeniu czy dany schemat jest w odpowiedniej postaci normalnej, jeżeli nie wówczas następuje dekompozycja schematu relacji na mniejsze schematy relacji. Ponownie, weryfikowana jest postać normalna otrzymanych schematów relacji. Jeżeli nie spełniają one zadanej postaci normalnej to proces dekompozycji jest kontynuowany dopóki otrzymane schematy relacji nie będą w odpowiedniej postaci normalnej.

Zanim przejdziemy do przedstawienia postaci normalnych przypomnimy podstawowe pojęcia dotyczące schematu relacji.



Pojęcia podstawowe (1)

- **Nadkluczem (superkluczem)** schematu relacji $R = \{A_1, A_2, \dots, A_n\}$ nazywamy zbiór atrybutów $S \subseteq R$, który jednoznacznie identyfikuje wszystkie krotki relacji r o schemacie R . Innymi słowy, w żadnej relacji r o schemacie R nie istnieją dwie krotki t_1, t_2 takie, że $t_1[S] = t_2[S]$
- **Kluczem K** schematu relacji R nazywamy minimalny nadklucz, to znaczy taki nadklucz, że nie istnieje $K' \subset K$ będący nadkluczem schematu R
- Kluczem schematu Dostawca:
{Nazwisko, Produkt, Cena}

BD – wykład 5 (8)

Nadkluczem (superkluczem) schematu relacji $R = \{A_1, A_2, \dots, A_n\}$ nazywamy zbiór atrybutów S będący podzbiorem zbioru R , który jednoznacznie identyfikuje wszystkie krotki relacji r o schemacie R . Innymi słowy, w żadnej relacji r o schemacie R nie istnieją dwie krotki t_1, t_2 takie, że $t_1[S] = t_2[S]$. **Kluczem K** schematu relacji R nazywamy minimalny nadklucz, to znaczy taki nadklucz, że nie istnieje żaden podzbiór zbioru K będący nadkluczem schematu R . Łatwo zauważyć, że kluczem przykładowego schematu Dostawca jest zbiór atrybutów {Nazwisko, Produkt, Cena}.



Pojęcia podstawowe (2)

- **Klucze potencjalne** (ang. *candidate keys*)

Klucz podstawowy (ang. *primary key*)

Klucze drugorzędne (ang. *secondary keys*)

- **Atrybuty:**

- atrybuty podstawowe: atrybut X jest podstawowy w schemacie R jeżeli należy do któregośkolwiek z kluczy schematu R
- atrybuty wtórne: atrybut X jest wtórny w schemacie R jeżeli nie należy do żadnego z kluczy schematu R

BD – wykład 5 (9)

Schemat relacji może posiadać wiele kluczy, które nazywamy kluczami potencjalnymi. Spośród kluczy potencjalnych wybieramy jeden klucz, tzw. klucz podstawowy. Schemat relacji może posiadać tylko jeden klucz podstawowy, definiowany za pomocą klauzuli PRIMARY KEY. System zarządzania bazą danych automatycznie weryfikuje unikalność klucza podstawowego.

Pozostałe klucze potencjalne schematu relacji, nazywane kluczami drugorzędnymi, definiujemy za pomocą klauzuli UNIQUE.

Wprowadzimy następującą klasyfikację atrybutów. Atrybuty dzielimy na atrybuty podstawowe i atrybuty wtórne. Atrybut X nazywamy atrybutem podstawowym w schemacie R jeżeli należy do któregośkolwiek z kluczy schematu R. Atrybut X nazywamy atrybutem wtórnym w schemacie R jeżeli nie należy do żadnego z kluczy schematu R. Obecnie przejdziemy do przedstawienia kolejnych postaci normalnych.



Pierwsza postać normalna 1NF (1)

- Definicja:

Schemat relacji R znajduje się w pierwszej postaci normalnej(1NF), jeżeli wartości atrybutów są atomowe (niepodzielne)

- Tablica Płeć:

Płeć	Imię
Męska	Jan, Piotr, Zenon
Żeńska	Anna, Eliza, Maria

Relacja Płeć w 1NF

Płeć	Imię
Męska	Jan
Męska	Piotr
Męska	Zenon
Żeńska	Anna
Żeńska	Eliza
Żeńska	Maria

BD – wykład 5 (10)

Mówimy, że schemat relacji R znajduje się w pierwszej postaci normalnej (1NF), jeżeli wartości atrybutów są atomowe (niepodzielne).

Rozważmy tabelę Płeć przedstawioną na slajdzie. Zauważmy, że atrybut Imię jest atrybutem typu zbiorowego. Normalizacja tabeli Płeć do 1NF polega na utworzeniu dla każdej atomowej wartości atrybutu Imię osobnej krotki. W wyniku uzyskujemy tabelę Płeć w 1NF, jak przedstawiono na slajdzie.



Pierwsza postać normalna 1 NF (2)

- Pierwsza postać normalna zabrania definiowania złożonych atrybutów, które są wielowartościowe
- Relacje, które dopuszczają definiowanie złożonych atrybutów nazywamy **relacjami zagnieżdzonymi** (ang. *nested relations*)
- W relacjach zagnieżdzonych każda krotka może zawierać inną relację
- Pracownicy (idPrac, Nazwisko, {Projekty (nr, godziny)})

BD – wykład 5 (11)

Pierwsza postać normalna zabrania definiowania złożonych atrybutów, które są wielowartościowe. Relacje, które dopuszczają definiowanie złożonych atrybutów nazywamy **relacjami zagnieżdzonymi** (ang. *nested relations*). W relacjach zagnieżdzonych każda krotka może zawierać inną relację. Rozważmy przykład relacji Pracownicy przedstawionej na kolejnym slajdzie.



Pierwsza postać normalna 1NF (3)

Pracownicy

IdPrac	Nazwisko	Projekty	
		nr	godziny
1234567	Kowalski	1	32,5
		2	7,5
6655443	Nowak	3	40,5
		1	20
4343435	Kruczak	2	20
		1	10
3333333	Morzy	2	10
		3	10
		4	10

Relacja zewnętrzna

Relacja zagnieżdzona

BD – wykład 5 (12)

Zauważmy, że relacja Pracownicy zawiera zagnieżdzoną w niej relację Projekty składającą się z atrybutów: Nr i Godziny. Trywialna normalizacja relacji Pracownicy do 1NF polegałaby na utworzeniu dla każdej krotki relacji zagnieżdzonej osobnej krotki relacji znalezionowej. W wyniku uzyskalibyśmy przykładowo, 2 krotki postaci $<1234567; \text{Kowalski}; 1; 32,5>$ i $<1234567; \text{Kowalski}; 2; 7,5>$.

Zasadniczą wadą tego sposobu normalizacji relacji Pracownicy jest duża redundancja danych, tzn. informacje dot. identyfikatora pracownika i jego nazwiska będą występować wielokrotnie w kolejnych krotkach znalezionowej relacji. Zalecany sposób normalizacji schematów relacji nie będących w 1NF opiera się na rozróżnieniu relacji zagnieżdzonej i relacji zewnętrznej. Do relacji zewnętrznej należą wszystkie atrybuty, które nie wschodzą w skład relacji zagnieżdzonej.

Przedstawiony slajd ilustruje podział relacji pracownicy na relację zewnętrzna i relację zagnieżdzoną.



Pierwsza postać normalna 1NF (4)

- Dana jest relacja R, zawierająca inną relację P
- Dekompozycja relacji R do zbioru relacji w 1NF:
 - Utwórz osobną relację dla relacji zewnętrznej
 - Utwórz osobną relację dla relacji wewnętrznej (zagnieżdżonej), do której dodaj klucz relacji zewnętrznej
 - Kluczem nowej relacji wewnętrznej (klucz relacji wewnętrznej + klucz relacji zewnętrznej)
- Dekompozycja relacji Pracownicy:
Pracownicy (IdPrac, Nazwisko)
Uczestniczy (IdPrac, Nr, Godziny)

BD – wykład 5 (13)

Zalecany sposób normalizacji schematów relacji do 1NF ma następującą postać.

Dana jest relacja R, zawierająca inną relację zagnieżdżoną P. Dekompozycja relacji R do zbioru relacji w 1NF:

- Utwórz osobną relację dla relacji zewnętrznej
- Utwórz osobną relację dla relacji wewnętrznej (zagnieżdżonej), do której dodaj klucz relacji zewnętrznej
- Kluczem nowej relacji wewnętrznej (klucz relacji wewnętrznej + klucz relacji zewnętrznej)

Przykładowo, dekompozycja relacji Pracownicy do zbioru relacji w 1NF prowadzi do 2 relacji następujących postaci: Pracownicy (IdPrac, Nazwisko) i Uczestniczy (IdPrac, Nr, Godziny).



Druga postać normalna 2NF (1)

- Pełna zależność funkcyjna

Zbiór atrybutów Y jest w pełni funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i nie istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$

Zbiór atrybutów Y jest częściowo funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje podzbiór $X' \subset X$ taki, że $X' \rightarrow Y$

- Druga postać normalna

Dana relacja r o schemacie R jest w drugiej postaci normalnej (2NF), jeżeli żaden atrybut wtórny tej relacji nie jest częściowo funkcyjnie zależny od żadnego z kluczów relacji r

BD – wykład 5 (14)

Łatwo zauważyc, że 1NF nie rozwiązuje problemu anomalii wymienionych wcześniej. Przejdziemy zatem do przedstawienia definicji drugiej postaci normalnej (2NF). W tym celu wprowadzimy definicje pełnej i częściowej zależności funkcyjnej.

Zbiór atrybutów Y jest w pełni funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i nie istnieje podzbiór X' zbioru X taki, że $X' \rightarrow Y$.

Zbiór atrybutów Y jest częściowo funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje podzbiór X' zbioru X taki, że $X' \rightarrow Y$.

Mogimy obecnie wprowadzić definicję drugiej postaci normalnej. Mówimy, że dana relacja r o schemacie R jest w drugiej postaci normalnej (2NF), jeżeli żaden atrybut wtórny tej relacji nie jest częściowo funkcyjnie zależny od żadnego z kluczów relacji r .



Druga postać normalna 2NF (2)

- **Uczestnictwo**

IdPrac	NrProj	Funkcja	Nazwisko	NazwaProj	Lokalizacja
--------	--------	---------	----------	-----------	-------------

$fd1: \{IdPrac, NrProj\} \rightarrow Funkcja$
 $fd2: \{IdPrac, NrProj\} \rightarrow Nazwisko$
 $fd3: \{IdPrac, NrProj\} \rightarrow NazwaProj$
 $fd4: \{IdPrac, NrProj\} \rightarrow Lokalizacja$
 $fd5: \{IdPrac\} \rightarrow Nazwisko$
 $fd6: \{NrProj\} \rightarrow NazwaProj$
 $fd7: \{NrProj\} \rightarrow Lokalizacja$

Zależności $fd2, fd3, fd4$ są zależnościami niepełnymi

BD – wykład 5 (15)

Rozważmy następujący przykład ilustrujący definicję drugiej postaci normalnej. Dana jest relacja Uczestnictwo składająca się z atrybutów: IdPrac, NrProj, Funkcja, Nazwisko, NazwaProj, Lokalizacja. Relacja Uczestnictwo opisuje udział pracowników o identyfikatorze (IdPrac) w realizacji projektów o numerze NrProj. Kluczem schematu relacji Uczestnictwo jest para atrybutów IdPrac i NrProj. W schemacie relacji Uczestnictwo występuje 7 zależności funkcyjnych $fd1, \dots, fd7$, z których 4 pierwsze są zależnościami od klucza. Zależność funkcyjna atrybutu od klucza oznacza, że każdy atrybut jest funkcyjnie zależny od klucza schematu relacji. Zauważmy, że zależności $fd2, fd3, fd4$ są zależnościami niepełnymi. Przykładowo, zależność funkcyjna $fd2: \{IdPrac, NrProj\} \rightarrow Nazwisko$ jest częściową zależnością funkcyjną gdyż istnieje podzbiór lewej strony zależności funkcyjnej ($IdPrac$), który wyznacza funkcyjnie prawą stronę zależności. Podobnie jest w przypadku zależności $fd3$ i $fd4$. Łatwo zauważyc, że schemat relacji uczestnictwo nie jest w 2NF, gdyż istnieją atrybuty wtórne (Nazwisko, NazwaProj, Lokalizacja), które są częściowo zależne od klucza. Zachodzi zatem konieczność dekompozycji schematu relacji Uczestnictwo na mniejsze relacje.



Druga postać normalna 2NF (3)

Uczestnictwo'

IdPrac	NrProj	Funkcja
--------	--------	---------

$fd1: \{IdPrac, NrProj\} \rightarrow Funkcja$

Pracownicy

IdPrac	ENAME
--------	-------

$fd5: \{IdPrac\} \rightarrow Nazwisko$

Projekty

NrProj	NazwaProj	Lokalizacja
--------	-----------	-------------

$fd6: \{NrProj\} \rightarrow NazwaProj$

$fd7: \{NrProj\} \rightarrow Lokalizacja$

$\{fd1, fd2, fd3, fd4, fd5, fd6, fd7\}^+ \equiv \{fd1, fd5, fd6, fd7\}^+$

bo:

$fd1 \Rightarrow fd2, fd3, fd4$, zgodnie z regułą poszerzenia

BD – wykład 5 (16)

Zależnością funkcyjną występującą w schemacie Uczestnictwo, która narusza definicję 2NF jest zależność $fd5$. W związku z tym tworzymy nowy schemat relacji Pracownicy zawierający lewą i prawą stronę zależności funkcyjnej $fd5$ i usuwamy ze schematu relacji Uczestnictwo prawą stronę zależności funkcyjnej $fd5$. Zmodyfikowany schemat Uczestnictwo nadal nie spełnia definicji 2NF ze względu na zależności funkcyjne $fd6$ i $fd7$. Podobnie jak poprzednio, tworzymy nowy schemat relacji Projekty zawierający zależności funkcyjne $fd6$ i $fd7$ i usuwamy ze schematu relacji Uczestnictwo prawie strony zależności funkcyjnych $fd6$ i $fd7$. Uzyskany schemat Uczestnictwo' składa się z atrybutów: IdPrac, NrProj, Funkcja i, co łatwo zauważyc, spełnia definicję 2NF. Ostatecznie, w wyniku dekompozycji schematu relacji Uczestnictwo otrzymujemy 3 schematy relacji: Uczestnictwo', Pracownicy, Projekty, wszystkie w 2NF.



Trzecia postać normalna 3NF (1)

Pracownicy-PP

Nazwisko	Instytut	Wydział
Brzeziński	I.Informatyki	Elektryczny
Morzy	I.Informatyki	Elektryczny
Koszlajda	I.Informatyki	Elektryczny
Królikowski	I.Informatyki	Elektryczny
...
Babij	ElektroEnerg.	Elektryczny
Kordus	ElektroEnerg.	Elektryczny
Sroczan	ElektroEnerg.	Elektryczny

Klucz: Nazwisko

Zależności funkcyjne: $\text{Nazwisko} \rightarrow \text{Instytut}$

$\text{Nazwisko} \rightarrow \text{Wydział}$

$\text{Instytut} \rightarrow \text{Wydział}$

BD – wykład 5 (17)

Rozważmy przykład relacji Pracownicy-PP przedstawiony na slajdzie. Relacja składa się z 3 atrybutów: Nazwisko, Instytut, Wydział. Założymy, że kluczem schematu relacji jest atrybut Nazwisko. Łatwo zauważyc, że schemat relacji Pracownicy-PP jest w 2NF (gdyż klucz jest jednoatrybutowy). Niestety, w schemacie relacji Pracownicy-PP występują wszystkie wymienione wcześniej typy anomalii. Fakt, że Instytut Informatyki należy do Wydziału Elektrycznego jest powielony tyle razy ilu pracowników jest zatrudnionych w instytucie (redundancja danych i anomalia aktualizacji). Występuje zjawisko anomalii wstawiania – do relacji Pracownicy-PP nie można wstawić informacji o nowoutworzonym na Wydziale Elektrycznym Instytucie Sterowania, tak długo jak długo nie zostanie zatrudniony pierwszy pracownik w tym instytucie. Wreszcie, występuje w tym schemacie również anomalia usuwania – usuwając kolejno pracowników Babij, Kordus, ..., z Instytutu Elektroenergetyki mimo woli usuniemy informacje o przypisaniu Instytutu Elektroenergetyki do Wydziału Elektrycznego.



Trzecia postać normalna 3NF (2)

- **Przechodnia zależność funkcyjna**

Zbiór atrybutów Y jest przechodnio funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje zbiór atrybutów Z , nie będący podzbiorem żadnego klucza schematu R taki, że zachodzi $X \rightarrow Z$ i $Z \rightarrow Y$

Zależność funkcyjna $X \rightarrow Y$ jest zależnością przechodnią jeżeli istnieje podzbiór atrybutów Z taki, że zachodzi $X \rightarrow Z$, $Z \rightarrow Y$ i nie zachodzi $Z \rightarrow X$ lub $Y \rightarrow Z$

BD – wykład 5 (18)

Wszystkie wymienione problemy wynikają z faktu występowania w schemacie relacji Pracownicy-PP przechodniej zależności funkcyjnej. Mówimy, że zbiór atrybutów Y jest przechodnio funkcyjnie zależny od zbioru atrybutów X w schemacie R , jeżeli $X \rightarrow Y$ i istnieje zbiór atrybutów Z , nie będący podzbiorem żadnego klucza schematu R taki, że zachodzi $X \rightarrow Z$ i $Z \rightarrow Y$. Innymi słowy, mówimy, że zależność funkcyjna $X \rightarrow Y$ jest zależnością przechodnią jeżeli istnieje podzbiór atrybutów Z taki, że zachodzi $X \rightarrow Z$, $Z \rightarrow Y$ i nie zachodzi $Z \rightarrow X$ lub $Y \rightarrow Z$.



Trzecia postać normalna 3NF (3)

- Dana relacja r o schemacie R jest w trzeciej postaci normalnej (**3NF**), jeżeli dla każdej zależności funkcyjnej $X \rightarrow A$ w R spełniony jest jeden z następujących warunków:
 - X jest nadkluczem schematu R , lub
 - A jest atrybutem podstawowym schematu R

BD – wykład 5 (19)

Wprowadzimy obecnie definicję trzeciej postaci normalnej. Dana relacja r o schemacie R jest w trzeciej postaci normalnej (**3NF**), jeżeli dla każdej zależności funkcyjnej $X \rightarrow A$ w R spełniony jest jeden z następujących warunków:

- X jest nadkluczem schematu R , lub
- A jest atrybutem podstawowym schematu R .



Trzecia postać normalna 3NF (4)

Pracownicy-PP-1

Nazwisko	Instytut
Brzeziński	I.Informatyki
Morzy	I.Informatyki
Koszlajda	I.Informatyki
Królikowski	I.Informatyki
...	...
Babij	ElektroEnerg.
Kordus	ElektroEnerg.
Sroczan	ElektroEnerg.

Pracownicy-PP-2

Instytut	Wydział
I.Informatyki	Elektryczny
...	Elektryczny
ElektroEnerg.	Elektryczny

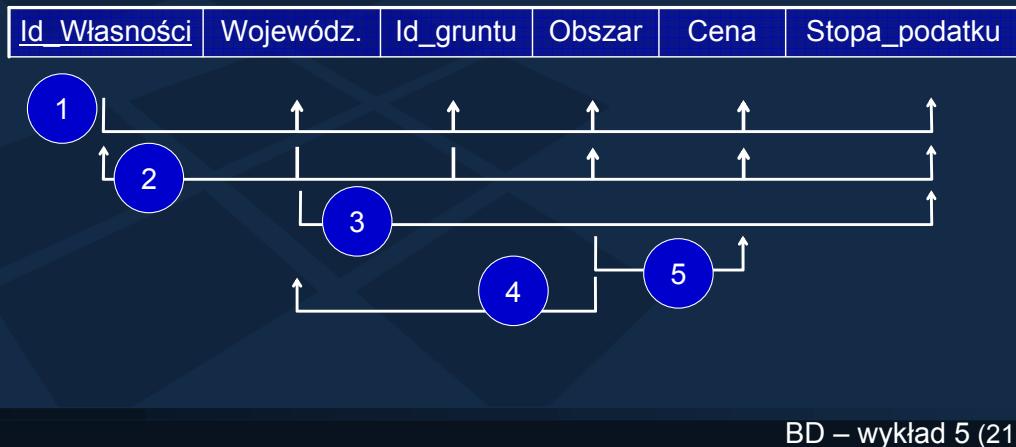
BD – wykład 5 (20)

Zauważmy, że wszystkie problemy związane z występowaniem anomalii znikną jeżeli zdekomponujemy relację Pracownicy-PP na dwie relacje Pracownicy-PP1 i Pracownicy-PP2. Relacja Pracownicy-PP1 zawiera informacje o pracownikach, natomiast relacja Pracownicy-PP2 zawiera informacje o przypisaniu instytutów do wydziałów. Zauważmy, że w przypadku relacji Pracownicy-PP2 przynależność instytutu do wydziału jest pamiętana tylko w jednej krotce – brak redundancji danych. Zauważmy również, że dekompozycja rozwiązuje problem anomalii wstawiania – informacje o nowym instytucie możemy wstawić do relacji Pracownicy-PP2, nawet jeżeli instytut ten nie zatrudnia żadnego pracownika. Dekompozycja ta rozwiązuje również problem anomalii usuwania – usunięcie informacji o pracownikach z relacji Pracownicy-PP1 nie pociąga za sobą usunięcia informacji o przypisaniu instytutów do wydziałów. Dekompozycja rozwiązuje również problem anomalii aktualizacji – zmiana przypisania instytutu do wydziału, np. Instytut Informatyki przeniesiony do Wydziału Informatyki i Zarządzania, dotyczy wyłącznie jednej krotki. Zauważmy, że dekompozycja relacji Pracownicy-PP na relacje Pracownicy-PP1 i Pracownicy-PP2 jest dekompozycją bez utraty informacji w tym sensie, że łącząc relację Pracownicy-PP1 i Pracownicy-PP2 wg. atrybutu połączniowego Instytut możemy odtworzyć oryginalną zawartość relacji Pracownicy-PP.



Postać normalna Boyce-Codd (BCNF) (1)

- Postać normalna Boyce-Codd'a stanowi warunek dostateczny 3NF, ale nie konieczny



Rozważmy przykład relacji grunty przedstawiony na slajdzie. Schemat relacji składa się z 6 atrybutów: Id_Własności, Województwo, Id_gruntu, Obszar, Cena, Stopa_podatku. Schemat relacji posiada 2 klucze. Pierwszym z nich jest atrybut Id_Własności, a drugim – para atrybutów: Województwo i Id_gruntu. Atrybutami podstawowymi relacji są: Id_Własności, Województwo i Id_gruntu. Atrybutami wtórnymi są: Obszar, Cena, Stopa_podatku.

Zbiór zależności funkcyjnych związanych ze schematem relacji został przedstawiony na slajdzie. Zależność nr 1 i nr 2 są zależnościami od klucza. Zależność nr 3 stwierdza, że atrybut Stopa_podatku zależy od atrybutu Województwo. Zależność nr 4 oznacza, że atrybut Województwo zależy od atrybutu Obszar. Zależność nr 5 oznacza, że atrybut Cena zależy od atrybutu Obszar.

Łatwo zauważyc, że schemat relacji jest w 1NF, nie jest natomiast w 2NF. Wynika to z faktu, że atrybut wtórnego Stopa_podatku jest częściowo funkcjonalnie zależny do klucza Województwo i Id_gruntu (zależność nr 3).



Postać normalna Boyce-Codd (BCNF) (3)

Grunty

<u>Id_Własności</u>	<u>Wojewódz.</u>	<u>Id-gruntu</u>	<u>Obszar</u>	<u>Cena</u>	<u>Stopa_produktu</u>

Grunty-1

<u>Id_Własności</u>	<u>Wojewódz.</u>	<u>Id-gruntu</u>	<u>Obszar</u>	<u>Cena</u>

BD – wykład 5 (22)

Dekomponujemy schemat Grunty na dwa schematy Grunty-1 i Grunty-2. Relacja Grunty-2 jest w 2NF i 3NF. Relacja Grunty-1 jest w 2NF, nie jest natomiast w 3NF ze względu na zależność funkcyjną nr 5 (Obszar → Cena).



Postać normalna Boyce-Codd (BCNF) (4)

Grunty-2

<u>Wojewódz.</u>	Stopa_produktu
------------------	----------------

Grunty-1A

<u>Id_Własności</u>	<u>Wojewódz.</u>	<u>Id-gruntu</u>	<u>Obszar</u>	<u>Obszar</u>	<u>Cena</u>

Grunty-1B

BD – wykład 5 (23)

Dokonajmy dekompozycji schematu Grunty-1 do schematów Grunty-1A i Grunty-1B.
Ta dekompozycja kończy proces normalizacji schematu Grunty do zbioru schematów relacji w 3NF.



Postać normalna Boyce-Codd (BCNF) (2)

- Założmy, że w relacji *Grunty* mamy tylko dwa województwa. Co więcej, założmy, że działki w pierwszym województwie mają rozmiar 0.5, 0.6, 0.7 h; natomiast działki w drugim województwie mają obszar 1, 1.2, 1.4 h. Ta informacja może być powielona w tysiącach krotek relacji *Grunty* oraz, po dekompozycji, w relacji *Grunty-1A*
- Relacja *Grunty-1A* jest w trzeciej postaci normalnej (*Wojewódz.* jest atrybutem podstawowym)

BD – wykład 5 (24)

Zależność funkcyjna nr 4 (Obszar -> Województwo) modeluje następującą sytuację rzeczywistą. Założmy, że w relacji *Grunty* mamy tylko dwa województwa. Co więcej, założmy, że działki w pierwszym województwie mają rozmiar 0.5, 0.6, 0.7 h; natomiast działki w drugim województwie mają obszar 1, 1.2, 1.4 h. Ta sytuacja jest opisana zależnością funkcyjną nr 4. Informacja o zależności województwa od obszaru jest powielona w tysiącach krotek relacji *Grunty* oraz, po dekompozycji, w relacji *Grunty-1A*. Relacja *Grunty-1A* jest w trzeciej postaci normalnej (Województwo jest atrybutem podstawowym). Część projektantów schematów baz danych traktuje to jako istotną wadę 3NF. Proponują oni dekompozycję schematów relacji do zmodyfikowanej 3NF, nazywanej postacią normalną Boyce'a-Codd'a. Otóż definicja postaci Boyce'a-Codd'a jest następująca:

Dana relacja r o schemacie R jest w postaci normalnej Boyce'a-Codd'a (**BCNF**), jeżeli dla każdej zależności funkcyjnej $X \rightarrow A$ w R spełniony jest następujący warunek: X jest nadkłuczem schematu R . W tym przypadku, zachodzi konieczność dekompozycji relacji *Grunty-1A* na dwa schematy relacji: *Grunty-1A1* (Id_Własności, Id_Gruntu, Obszar) oraz *Grunty-1A2* (Obszar, Województwo).



Zależności wielowartościowe (1)

Loty

Lot	Dzień_tygodnia	Typ_samolotu
106	poniedziałek	134
106	czwartek	154
106	poniedziałek	154
106	czwartek	134
206	środa	747
206	piątek	767
206	środa	767
206	piątek	747

Języki

Nazwisko	Język_obcy	Język_prog.
Nowak	angielski	Basic
Nowak	włoski	Fortran
Nowak	angielski	Fortran
Nowak	włoski	Basic
Nowak	czeski	Basic
Nowak	czeski	Fortran

BD – wykład 5 (25)

Rozważmy przykładowe relacje Loty i Języki przedstawione na slajdach. Relacja Loty składa się z 3 atrybutów: Lot, Dzień_tygodnia, Typ_samolotu. Opisuje ona typ samolotu i dzień tygodnia, w którym odbywają się określone loty. Kluczem schematu relacji Loty są wszystkie trzy wymienione atrybuty. Stąd, schemat relacji Loty jest w 3NF i BCNF. Niestety, schemat ten posiada dość istotną wadę – występuje w nim problem modyfikacji zależnej od stanu bazy danych.

Podobny problem występuje w schemacie relacji Języki składającej się również z 3 atrybutów: Nazwisko, Język_obcy, Język_programowania, które również stanowią klucz schematu relacji.



Modyfikacja relacji z zależnościami wielowartościowymi

- Lot 106 będzie dodatkowo odbywał się w Środę i na tę linię wprowadzamy, dodatkowo, nowy typ samolotu – 104

Loty	Lot	Dzień-tygodnia	Typ-samolotu
	106	poniedziałek	134
	106	czwartek	154
	106	poniedziałek	154
	106	czwartek	134
	106	poniedziałek	104
	106	czwartek	104
	106	środa	134
	106	środa	154
	106	środa	104

BD – wykład 5 (26)

Rozważmy prostą modyfikację relacji Loty. Założymy, że lot 106 będzie dodatkowo odbywał się w środę i na tę linię wprowadzamy, dodatkowo, nowy typ samolotu – 104. Zauważmy, że ta stosunkowo prosta modyfikacja wymaga wprowadzenia aż pięciu nowych krotek do relacji Loty: <106, poniedziałek, 104>, <106, czwartek, 104>, <106, środa, 134>, <106, środa, 154>, <106, środa, 104>. Dwie pierwsze krotki wiążą się z faktem, że zarówno w poniedziałek jak i czwartek lot 106 będzie obsługiwał nowy typ samolotu 104, pozostałe 3 krotki wiążą się z faktem, że lot 106 będzie dodatkowo odbywał się w środę. Liczba wprowadzanych krotek zależy od aktualnego stanu bazy danych. Ta własność schematu relacji Loty utrudnia pielęgnację tej relacji przez osoby nie będące informatykami.

Podobny problem występuje w odniesieniu do relacji języki. Założymy, że Nowak nauczył się języka obcego francuskiego i języka programowania C++. Wprowadzenie tej modyfikacji do relacji języki wymaga wprowadzenia 6 nowych krotek.



Dekompozycja

Lot-1

Lot	Dzień-tygodnia
106	poniedziałek
106	czwartek
206	środa
206	piątek
...	...
106	środa

Lot-2

Lot	Typ-samolotu
106	134
106	154
206	747
206	767
...	...
106	104

Język-1

Nazwisko	Język_obcy
Nowak	angielski
Nowak	włoski
Nowak	czeski

Język-2

Nazwisko	Język_prog.
Nowak	Basic
Nowak	Fortran

BD – wykład 5 (27)

Wymieniony wyżej problem modyfikacji relacji Loty i Języki znika jeżeli oba schematy zdekomponujemy odpowiednio na: Lot-1 i Lot-2 oraz Język-1 i Język-2. Przykładowo, wprowadzenie modyfikacji „lot 106 będzie dodatkowo odbywał się w Środę i na tę linię wprowadzamy, dodatkowo, nowy typ samolotu – 104” wymaga, po dekompozycji, wprowadzenia jednej krotki $\langle 106, \text{środa} \rangle$ do relacji Lot-1 oraz jednej krotki $\langle 106, 104 \rangle$ do relacji Lot-2. Zauważmy, że teraz modyfikacja ta nie zależy od stanu bazy danych.

Podobnie jest w przypadku modyfikacji relacji Języki. Wprowadzenie modyfikacji „Nowak nauczył się języka obcego francuskiego i języka programowania C++” wymaga wprowadzenia jednej krotki $\langle \text{Nowak, francuski} \rangle$ do relacji Język-1 i $\langle \text{Nowak, C++} \rangle$ do relacji Język-2.



Zależności wielowartościowe (2)

- Zależności wielowartościowe są konsekwencją wymagań pierwszej postaci normalnej, która nie dopuszcza, aby krotki zawierały atrybuty wielowartościowe
- Zależność wielowartościowa występuje w relacji r(R) nie dlatego, że na skutek zbiegu okoliczności tak ułożyły się wartości krotek, lecz występuje ona dla dowolnej relacji r o schemacie R dlatego, że odzwierciedla ona ogólną prawidłowość modelowanej rzeczywistości

Lot →→ Dzień-tygodnia

Lot →→ Typ-samolotu

Nazwisko →→ Język-obcy

Nazwisko →→ Język-programowania

BD – wykład 5 (28)

Powyższe problemy z modyfikacją zależną od stanu bazy danych wynikają z faktu występowania w schemacie relacji Loty i Języki tzw. zależności wielowartościowych. Zależności wielowartościowe są konsekwencją wymagań pierwszej postaci normalnej, która nie dopuszcza, aby krotki zawierały atrybuty wielowartościowe. Zależność wielowartościowa jest własnością semantyczną schematu relacji.

Zależność wielowartościowa występuje w relacji r(R) nie dlatego, że na skutek zbiegu okoliczności tak ułożyły się wartości krotek, lecz występuje ona dla dowolnej relacji r o schemacie R dlatego, że odzwierciedla ona ogólną prawidłowość modelowanej rzeczywistości. W przykładowych relacjach Loty i Języki występują 4 zależności wielowartościowe:

Lot->->Dzień-tygodnia

Lot->->Typ-samolotu

Nazwisko->->Język-obcy

Nazwisko->->Język-programowania



Zależności wielowartościowe (3)

- Wystąpienie zależności wielowartościowej $X \rightarrow\rightarrow Y$ w relacji o schemacie $R = XYZ$ wyraża dwa fakty:
- Związek pomiędzy zbiorami atrybutów X i Y ;
- Niezależność zbiorów atrybutów Y , Z . Zbiory te są związane ze sobą pośrednio poprzez zbiór atrybutów X

Lot-3

Lot	Dzień-tygodnia	Typ-samolotu
106	poniedziałek	134
106	czwartek	154
106	czwartek	134
206	środa	747
206	piątek	767

BD – wykład 5 (29)

Wystąpienie zależności wielowartościowej $X \rightarrow\rightarrow Y$ w relacji o schemacie $R = XYZ$ wyraża dwa fakty:

- związek pomiędzy zbiorami atrybutów X i Y ;
- niezależność zbiorów atrybutów Y , Z ; zbiory te są związane ze sobą pośrednio poprzez zbiór atrybutów X .

W relacji Lot-3 przedstawionej na slajdzie występuje jedna zależność wielowartościowa: $\text{Lot} \rightarrow\rightarrow \{\text{Dzień_tygodnia}, \text{Typ_samolotu}\}$. Dekompozycja schematu Lot-3, podobnie jak w przypadku relacji Loty, prowadziłaby do utraty informacji, że lot 206 w środę jest obsługiwany przez typ samolotu 747 i lot 206 w piątek jest obsługiwany przez typ samolotu 767. Innymi słowy, schemat Lot-3 jest niedekomponowalny bez utraty informacji.



Definicja własności zależności wielowartościowych

- Niech R oznacza schemat relacji, natomiast X, Y są rozłącznymi zbiorami atrybutów schematu R i $Z = R - (XY)$
- Relacja $r(R)$ spełnia zależność wielowartościową $X \rightarrow\rightarrow Y$, jeżeli dla dwóch dowolnych krotek t_1 i t_2 z $r(R)$ takich, że $t_1[X] = t_2[X]$, zawsze istnieją w $r(R)$ krotki t_3, t_4 takie, że spełnione są następujące warunki:

$$\begin{aligned}t_1[X] &= t_2[X] = t_3[X] = t_4[X] \\t_3[Y] &= t_1[Y] \text{ i } t_4[Y] = t_2[Y] \\t_3[R - X - Y] &= t_2[R - X - Y] \\t_4[R - X - Y] &= t_1[R - X - Y]\end{aligned}$$

- Z symetrii powyższej definicji wynika, że jeżeli w relacji $r(R)$ zachodzi $X \rightarrow\rightarrow Y$, to zachodzi również: $X \rightarrow\rightarrow [R - X - Y]$
- Ponieważ $R - X - Y = Z$, to powyższy fakt zapisujemy czasami w postaci: $X \rightarrow\rightarrow Y / Z$

BD – wykład 5 (30)

Teraz krótko scharakteryzujemy własności zależności wielowartościowych. Niech R oznacza schemat relacji, natomiast X, Y są rozłącznymi zbiorami atrybutów schematu R i $Z = R - (XY)$.

Relacja $r(R)$ spełnia zależność wielowartościową $X \rightarrow\rightarrow Y$, jeżeli dla dwóch dowolnych krotek t_1 i t_2 z $r(R)$ takich, że $t_1[X] = t_2[X]$, zawsze istnieją w $r(R)$ krotki t_3, t_4 takie, że spełnione są następujące warunki, przedstawione na slajdzie:

$$\begin{aligned}t_1[X] &= t_2[X] = t_3[X] = t_4[X] \\t_3[Y] &= t_1[Y] \text{ i } t_4[Y] = t_2[Y] \\t_3[R - X - Y] &= t_2[R - X - Y] \\t_4[R - X - Y] &= t_1[R - X - Y]\end{aligned}$$

Z symetrii powyższej definicji wynika, że jeżeli w relacji $r(R)$ zachodzi $X \rightarrow\rightarrow Y$, to zachodzi również: $X \rightarrow\rightarrow [R - X - Y]$. Ponieważ $R - X - Y = Z$. Powyższy fakt zapisujemy czasami w postaci: $X \rightarrow\rightarrow Y / Z$.



Trywialna zależność wielowartościowa (1)

- Zależność wielowartościowa $X \rightarrow\rightarrow Y$ w relacji $r(R)$ nazywamy zależnością **trywialną**, jeżeli
 - zbiór Y jest podzbiorem X , lub
 - $X \cup Y = R$
- Zależność nazywamy **trywialną**, gdyż jest ona spełniona dla dowolnej instancji r schematu R

BD – wykład 5 (31)

Zanim przejdziemy do zdefiniowania czwartej postaci normalnej wprowadźmy pojęcie trywialnej zależności wielowartościowej. Zależność wielowartościowa $X \rightarrow\rightarrow Y$ w relacji $r(R)$ nazywamy zależnością trywialną, jeżeli zbiór Y jest podzbiorem X , lub $X \cup Y = R$. Zależność nazywamy trywialną, gdyż jest ona spełniona dla dowolnej instancji r schematu R .



Czwarta postać normalna 4NF

- Relacja r o schemacie R jest w czwartej postaci normalnej (**4NF**) względem zbioru zależności wielowartościowych MVD jeżeli jest ona w $3NF$ i dla każdej zależności wielowartościowej $X \rightarrow\rightarrow Y \in MVD$ zależność ta jest trywialna lub X jest nadkluczem schematu

BD – wykład 5 (32)

Obecnie wprowadzimy definicję czwartej postaci normalnej. Mówimy, że relacja r o schemacie R jest w czwartej postaci normalnej (**4NF**) względem zbioru zależności wielowartościowych MVD jeżeli jest ona w $3NF$ i dla każdej zależności wielowartościowej $X \rightarrow\rightarrow Y \in MVD$ zależność ta jest trywialna lub X jest nadkluczem schematu.

Jak łatwo zauważyc, przedstawione uprzednio schematy relacji Loty i Języki nie są w 4NF. Przykładowo, schemat relacji Loty nie jest w 4NF gdyż zależność wielowartościowa, np. Lot->->Typ_samolotu nie jest trywialna jak również Lot nie jest nadkluczem schematu Loty. Równie łatwo zauważyc, że schematy relacji Lot-1 i Lot-2, uzyskane w wyniku dekompozycji schematu Loty, są w 4NF gdyż każdy z tych schematów zawiera trywialną zależność wielowartościową. Podobnie jest w przypadku relacji Języki.



Dekompozycja relacji na relacje bez utraty informacji (1)

- **Dekompozycja na relacje w 3NF**

Dana jest relacja r o schemacie R , i dany jest zbiór F zależności funkcyjnych dla R . Niech relacje r_1 i r_2 o schematach, odpowiednio, R_1 i R_2 , oznaczają dekompozycję relacji $r(R)$. Dekompozycja ta jest dekompozycją bez utraty informacji, jeżeli co najmniej jedna z poniższych zależności funkcyjnych jest spełniona:

$$\begin{aligned} R_1 \cap R_2 &\rightarrow R_1 \\ R_1 \cap R_2 &\rightarrow R_2 \end{aligned}$$

BD – wykład 5 (33)

Na zakończenie podamy twierdzenia dotyczące dekompozycji schematów relacji na mniejsze schematy relacji, bez utraty informacji. Pierwsze twierdzenie dotyczy dekompozycji schematu relacji R na schematy relacji w 3NF.

Dana jest relacja r o schemacie R , i dany jest zbiór F zależności funkcyjnych dla R . Niech relacje r_1 i r_2 o schematach, odpowiednio, R_1 i R_2 , oznaczają dekompozycję relacji $r(R)$. Dekompozycja ta jest dekompozycją bez utraty informacji, jeżeli co najmniej jedna z poniższych zależności funkcyjnych jest spełniona:

- $R_1 \text{ ILOCZYN } R_2 \rightarrow R_1$,
- $R_1 \text{ ILOCZYN } R_2 \rightarrow R_2$.



Dekompozycja relacji na relacje bez utraty informacji (1)

- **Dekompozycja na relacje w 4NF**

Dana jest relacja r o schemacie R. Niech relacje r1 i r2 o schematach, odpowiednio, R1 i R2, oznaczają dekompozycję relacji r(R). Dekompozycja ta jest dekompozycją bez utraty informacji, jeżeli co najmniej jedna z poniższych zależności wielowartościowych jest spełniona:

$$\begin{aligned} R_1 \cap R_2 &\rightarrow\rightarrow (R_1 - R_2) \\ R_1 \cap R_2 &\rightarrow\rightarrow (R_2 - R_1) \end{aligned}$$

BD – wykład 5 (34)

Drugie twierdzenie dotyczy dekompozycji schematu relacji R na schematy relacji w 4NF.

Dana jest relacja r o schemacie R. Niech relacje r1 i r2 o schematach, odpowiednio, R1 i R2, oznaczają dekompozycję relacji r(R). Dekompozycja ta jest dekompozycją bez utraty informacji, jeżeli co najmniej jedna z poniższych zależności wielowartościowych jest spełniona:

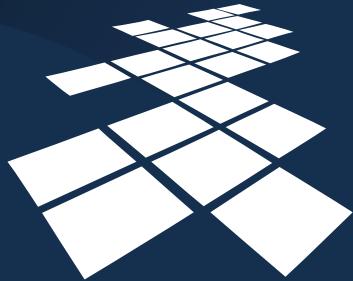
- R₁ ILOCZYN R₂ ->-> (R₁ - R₂),
- R₁ ILOCZYN R₂ ->-> (R₂ - R₁).

Przykładowo, dekompozycja schematu relacji Loty na schematy Lot-1 i Lot-2 w 4NF jest dekompozycją bez utraty informacji, gdyż:

- Lot-1 ILOCZYN Lot-2 = {Lot} wyznacza wielowartościowo zarówno (Lot-1 – Lot-2) jak i (Lot-2 – Lot-1). (Lot-1 – Lot-2) = {Dzień_tygodnia}, natomiast (Lot-2 – Lot-1) = {Typ_samolotu}.

Organizacja plików

Wykład przygotował:
Robert Wrembel



**UCZELNIA
ONLINE**

BD – wykład 5 (1)



Plan wykładu

- Struktura przechowywania danych i organizacja rekordów w blokach
- Rodzaje organizacji plików
 - pliki nieuporządkowane
 - pliki uporządkowane
 - pliki haszowe

BD – wykład 5 (2)

Celem wykładu jest przedstawienie podstawowych organizacji plików. W ramach wykładu zostaną omówione następujące zagadnienia:

- struktura przechowywania danych i organizacja rekordów w blokach,
- rodzaje organizacji plików, czyli pliki nieuporządkowane, uporządkowane i haszowe.

Każda z organizacji plików zostanie scharakteryzowana strukturą, mechanizmami dostępu i kosztami dostępu. Ponadto, dla plików haszowych zostaną przedstawione podstawowe techniki rozwiązywania kolizji.



Wprowadzenie (1)

- Organizacja pliku
 - sposób uporządkowania rekordów w pliku przechowywanym na dysku
 - wspiera wykonywanie operacji na pliku
- Wybór odpowiedniej organizacji zależy od sposobu użytkowania danego pliku
 - wyszukiwanie rekordów opisujących zatrudnionych pracowników w porządku alfabetycznym ⇒ sortowanie pliku według nazwisk
 - wyszukiwanie rekordów opisujących zatrudnionych, których zarobki są w podanym zakresie ⇒ sortowanie nie jest właściwe
 - wybór odpowiedniej organizacji dla każdego pliku
⇒ administrator

BD – wykład 5 (3)

Organizacja pliku określa sposób uporządkowania rekordów w pliku przechowywanym na dysku. Wybór właściwej organizacji zależy od sposobu użytkowania danego pliku.

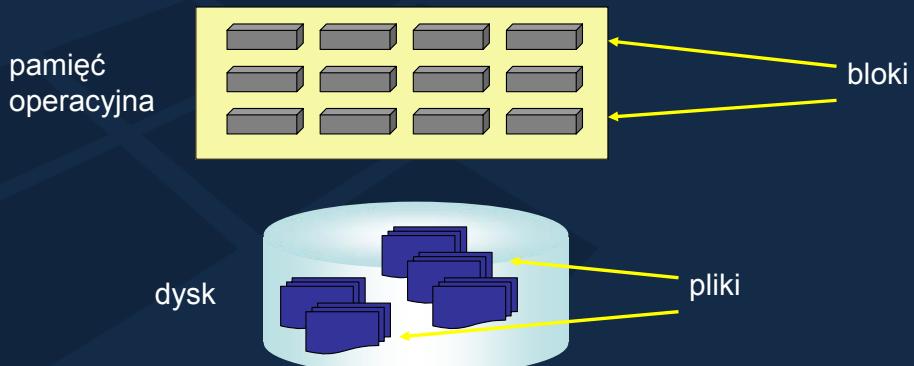
Przykładowo, jeśli chcemy wyszukiwać rekordy opisujące zatrudnionych pracowników w porządku alfabetycznym, sortowanie pliku według nazwisk jest dobrą organizacją pliku. Z drugiej strony, jeśli chcemy wyszukiwać rekordy opisujące zatrudnionych, których zarobki są w podanym zakresie, sortowanie rekordów pracowników według nazwisk nie jest właściwą organizacją pliku.

Wybranie właściwej organizacji dla każdego pliku jest zadaniem administratora BD.



Wprowadzenie (2)

- Media fizyczne tworzą hierarchię pamięci składającą się z:
 - pamięci operacyjnej o organizacji blokowej
 - pamięci zewnętrznej o organizacji plikowej



BD – wykład 5 (4)

Media fizyczne tworzą hierarchię pamięci składającą się z pamięci operacyjnej i pamięci zewnętrznej. Pamięć zewnętrzna ma organizację plikową, oznacza to, że jednostką alokacji na dysku jest plik. Natomiast pamięć operacyjna ma organizację blokową. Oznacza to, że jednostką alokacji jest blok. Blok alokowany w pamięci operacyjnej jest wielokrotnością rozmiaru fizycznego bloku dyskowego.



Wprowadzenie (3)

- Trwałe dane w BD są przechowywane w pamięci zewnętrznej:
 - ze względu na rozmiar danych
 - odporność pamięci zewnętrznej na awarie
 - niski koszt przechowywania
- Buforowanie bloków dyskowych

BD – wykład 5 (5)

Trwałe dane w bazie danych są przechowywane w pamięci zewnętrznej z trzech powodów:

- ze względu na rozmiar bazy danych
- odporność pamięci zewnętrznej na awarie
- koszt jednostkowy

W czasie pracy bazy danych, poszukiwane dane są odczytywane z plików dyskowych i umieszczone/buforowane w blokach systemu operacyjnego. Bloki te są często nazywane buforami bazy danych. Stąd dane są następnie udostępniane użytkownikom BD. Zapis danych na dysk również odbywa się za pośrednictwem buforów bazy danych.

Użytkownicy modyfikują dane w buforach. Zawartość tych buforów jest następnie zapisywana do plików.



Wprowadzenie (4)

- Alokacja danych w plikach \Rightarrow rekordy
 - rekordy proste / złożone
 - rekordy o stałej / zmiennej długości
- Na poziomie dyskowym rekordy przechowywane w blokach dyskowych

BD – wykład 5 (6)

Dane w plikach są reprezentowane w postaci rekordów. Każdy rekord składa się z pól przechowujących wartości.

Wyróżnia się rekordy o strukturze prostej i złożonej (zagnieżdżonej). W pierwszym przypadku, wartością każdego pola rekordu jest wartość elementarna (liczba, łańcuch znaków, data, ciąg bitów). W drugim przypadku, wartością pola rekordu jest inny rekord.

Rekordy mogą mieć stałą długość lub zmienną. Stała długość oznacza, że rekord zawsze zajmuje tyle samo miejsca na dysku, niezależnie od rzeczywistych rozmiarów przechowywanych w nim danych. Rekordy o stałej długości przyjmują zawsze rozmiar będący sumą maksymalnych zadeklarowanych rozmiarów ich atrybutów. Rekordy o zmiennej długości przyjmują taki rozmiar jaki faktycznie przyjmują przechowywane w nich dane.

Na poziomie dyskowym, rekordy są przechowywane w blokach dyskowych. Rozmiar tych bloków jest określany przez system operacyjny. Często jest to 512B.



Współczynnik blokowania

- W bloku dyskowym lub bloku pamięci operacyjnej mieści się niecałkowita liczba rekordów
- Współczynnik blokowania $\Leftrightarrow bfr$
 - rozmiar bloku B
 - rozmiar rekordu R
 - współczynnik blokowania $bfr = \lfloor (B/R) \rfloor$ 1
 - wolny obszar w bloku: $B - (bfr * R)$ 2

BD – wykład 5 (7)

W praktyce, w jednym bloku dyskowym lub bloku pamięci operacyjnej mieści się niecałkowita liczba rekordów. Wynika to z faktu, że rozmiar bloku nigdy nie jest wielokrotnością rozmiaru rekordu. Maksymalna liczba rekordów, która może się zmieścić w bloku jest określana za pomocą tzw. współczynnika blokowania (blocking factor) - bfr. Jest on definiowany następująco.

Niech B oznacza rozmiar bloku dyskowego, R - oznacza rozmiar rekordu. Dla uproszczenia zakłada się że albo rozważane są rekordy o stałej długości R bajtów, albo dla rekordów o zmiennej długości R oznacza maksymalny rozmiar rekordu.

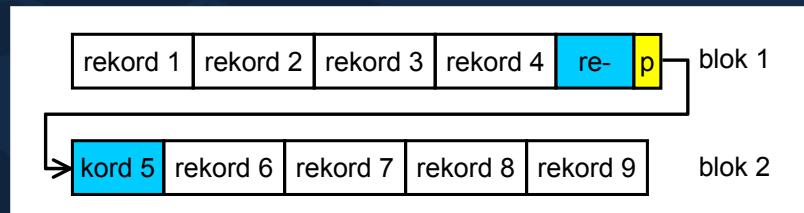
Współczynnik blokowania jest określany jako największa całkowita mniejsza od ilorazu rozmiaru bloku i rozmiaru rekordu. Wyraża to wzór nr 1 ze slajdu.

Przyjmując taką definicję współczynnika blokowania, łatwo jest obliczyć wolny obszar jaki pozostaje w każdym z bloków. Jest on wyrażony wzorem: $B - (bfr * R)$. W przypadku rekordów o zmiennej długości jest to obszar minimalny.



Organizacja rekordów w blokach (1)

- Dzielona (spanned)



BD – wykład 5 (8)

Rekordy w blokach są zorganizowane albo jako dzielone (ang. spanned) albo jako niepodzielne (ang. unspanned). W pierwszym przypadku, rekord, który cały nie mieści się w bloku jest dzielony, a jego części są przechowywane w kilku blokach.

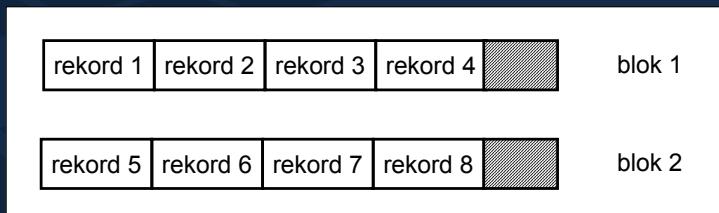
Poglądowy schemat dzielonej organizacji rekordów przedstawiono na slajdzie. Rekord 5 nie mieści się w bloku 1, więc jest dzielony na 2 części. Początek rekordu znajduje się w bloku 1, a koniec rekordu - w bloku 2.

Organizacja dzielona zapewnia lepsze wykorzystanie miejsca w blokach - wszystkie bloki są w całości wypełnione.



Organizacja rekordów w blokach (2)

- Niepodzielna (unspanned)



BD – wykład 5 (9)

W przypadku organizacji niepodzielnej rekord, który nie mieści się w całości w bloku jest przenoszony do takiego bloku, w którym zmieści się cały.

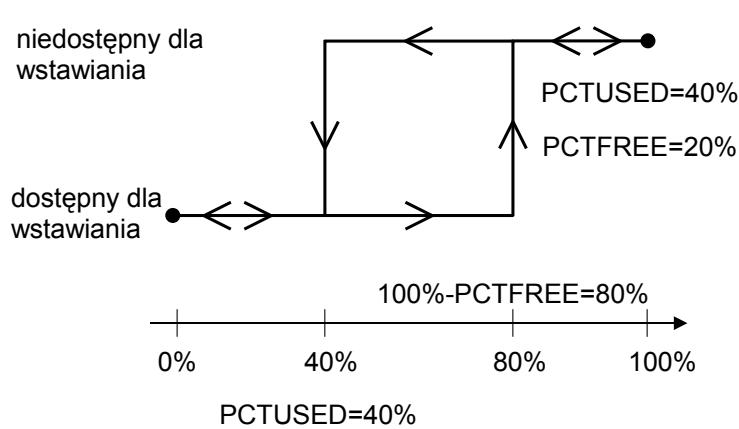
Poglądowy schemat niepodzielnej organizacji rekordów przedstawia slajd. W tym przykładzie, rekord 5 w całości został przeniesiony do bloku 2.

Organizacja niepodzielna powoduje, że w blokach pozostaje niewykorzystane miejsce.



Zarządzanie rozmiarem bloku danych

- Utrzymywanie wolnej pamięci w bloku dla potencjalnych modyfikacji



BD – wykład 5 (10)

W praktyce, w każdym bloku, niezależnie od organizacji rekordów, pozostawia się część wolnego miejsca. Miejsce to jest wykorzystywane na rozszerzanie się rekordów na skutek modyfikowania wartości ich pól. Gdyby nie było wolnego miejsca w bloku, wówczas po modyfikacji rekord mógłby się nie zmieścić w bloku i musiałby zostać przesunięty do innego bloku.

Tę technikę stosuje m.in. SZBD Oracle. W systemie tym, dla bloków definiuje się dwa parametry PCTFREE i PCTUSED. Pierwszy z nich określa ile procent rozmiaru bloku pozostanie wolne. PCTUSED określa kiedy do bloku można wstawiać rekordy. Parametr ten jest wyrażony również procentowym rozmiarem bloku. Jeśli zajętość bloku przewyższa wartość PCTUSED, wówczas blok nie jest wykorzystywany do wstawiania nowych rekordów. Jeśli natomiast zajętość bloku jest niższa niż PCTUSED, wówczas do bloku można wstawiać rekordy.

W przykładzie na slajdzie PCTFREE=20%, a PCTUSED=40%. Oznacza to, że w bloku zawsze pozostaje przynajmniej 20% wolnego miejsca. Do bloku wolno wstawić dane jeżeli aktualna jego zajętość nie przekracza 40% jego rozmiaru. Jeżeli na skutek wstawiania nowych rekordów do bloku, zajętość wzrośnie powyżej 40%, SZBD przestaje wstawiać rekordy do tego bloku. Jeżeli na skutek usuwania rekordów zajętość bloku spadnie poniżej 40%, blok ten ponownie zostanie wykorzystany do wstawiania rekordów.



Operacje na plikach

- Operacje dostępu do pojedynczego rekordu (record-at-a-time)
 - wyszukiwanie: Find, FindFirst, FindNext
 - usuwanie: Delete
 - aktualizacja: Update
 - wstawianie: Insert
- Operacje dostępu do zbioru rekordów (set-at-a-time)
 - wyszukiwanie wszystkich rekordów spełniających kryterium: FindAll
 - wyszukiwanie z sortowaniem: FindOrdered
 - reorganizacja pliku: Reorganize

BD – wykład 5 (11)

Ponieważ rekordy są fizycznie przechowywane w plikach, więc dostęp do rekordów musi być realizowany za pomocą dedykowanych i zoptymalizowanych operacji.

Wyróżnia się operacje na pojedynczym rekordzie (ang. record-at-a-time)

i operacje na zbiorze rekordów (ang. set-at-a-time). Do pierwszej grupy zalicza się:

- wyszukiwanie rekordu spełniającego kryterium: Find, FindFirst, FindNext,
- usuwanie: Delete,
- aktualizację: Update,
- wstawianie: Insert.

Do drugiej grupy zalicza się:

- wyszukiwanie wszystkich rekordów spełniających kryterium: FindAll,
- wyszukiwanie rekordów spełniających kryterium z sortowaniem wyników: FindOrdered,
- reorganizację pliku: Reorganize (np. posortowanie rekordów wg nowego kryterium).



Model kosztów

- Notacja i założenia:
 - $N \Rightarrow$ liczba bloków
 - każdy blok zawiera R rekordów
 - średni czas odczytu/zapisu bloku dyskowego wynosi D
 - średni czas przetwarzania rekordu wynosi C
 - dla plików haszowych czas obliczenia wartości funkcji haszowej wynosi H
- Typowe wartości wynoszą:
 - $D = 15 \text{ ms}$
 - $C \text{ i } H$ od 1 do $10 \mu\text{s}$
 - dominuje koszt I/O

BD – wykład 5 (12)

Każda operacja na pliku posiada swój tzw. koszt, który jest oczywiście zależny od organizacji wewnętrznej pliku. Koszt jest konkretną wartością, której miarą może być np. czas wykonania, liczba dostępów do dysku. Koszt jest wartością wynikającą z tzw. modelu kosztów (ang. cost model).

W celach analizy kosztów dostępu do plików nieuporządkowanych, uporządkowanych i haszowych przyjmiemy następujący model kosztów.

Niech:

- N oznacza liczbę bloków;
- każdy blok zawiera R rekordów;
- średni czas odczytu/zapisu bloku dyskowego wynosi D ;
- średni czas przetwarzania rekordu (np., porównanie wartości atrybutu ze stałą) wynosi C ;
- w przypadku plików haszowych stosujemy funkcję haszową odwzorowującą wartości rekordów na liczby naturalne; czas obliczenia wartości funkcji haszowej wynosi H ;

Typowe wartości wymienionych parametrów wynoszą $D = 15 \text{ ms}$, $C \text{ i } H$ od 1 do $10 \mu\text{s}$. Jak widać, czas dostępu do dysku (I/O) jest tu dominującym.



Rodzaje organizacji plików

- Pliki nieuporządkowane (unordered files, heap files)
- Pliki uporządkowane (ordered files)
- Pliki haszowe (hash files)

BD – wykład 5 (13)

Omówione operacje na plikach są implementowane i optymalizowane w różny sposób, zależny od organizacji wewnętrznej samego pliku. Ze względu na organizację wewnętrzną wyróżnia się: pliki nieuporządkowane (ang. unordered files, heap files), pliki uporządkowane (ang. ordered files) i pliki haszowe (ang. hash files).



Plik nieuporządkowany

- Nagłówek pliku zawierający wskaźnik do bloku danych
- Blok danych zawiera wskaźnik do bloku następnego i poprzedniego
- Rekordy wstawiane na koniec pliku



BD – wykład 5 (14)

W pliku nieuporządkowanym rekordy są przechowywane w kolejności ich wstawiania. Nagłówek pliku zawiera wskaźnik do pierwszego bloku danych. Bloki danych tworzą listę dwu-kierunkową. Oznacza to, że blok poprzedni posiada wskaźnik do bloku następnego, a blok następny posiada wskaźnik do bloku poprzedniego. W pliku nieuporządkowanym rekordy są wstawiane zawsze na koniec pliku.



Plik nieuporządkowany - operacje (1)

Podstawową organizacją pliku nieuporządkowanego jest stóg (ang. heap)

- Operacje:

- wstawianie rekordu

- rekord jest wstawiany do ostatniego bloku pliku;
blok ten jest zapisywany na dysk

- $\text{koszt} = 2D+C$

- wyszukiwanie rekordu:

- konieczność liniowego przeszukiwania wszystkich bloków

- średni koszt = $0.5N(D+RC)$. 1

- maksymalny koszt = $N(D+RC)$ (przejrzenie całego pliku) 2

BD – wykład 5 (15)

Podstawową organizacją pliku nieuporządkowanego jest stóg (ang. heap).

W przypadku operacji wstawiania rekordu, rekord trafia do ostatniego bloku pliku. Przed wstawieniem, blok ten musi zostać odczytany z dysku do bufora pamięci operacyjnej. Tu rekord jest wstawiany i blok ten jest z powrotem zapisywany na dysk. W konsekwencji, koszt wstawienia rekordu wynosi $2D+C$. Składnik $2D$ to odczyt ostatniego bloku z dysku i jego ponowny zapis. C to koszt przetworzenia rekordu w buforze.

W przypadku operacji wyszukania rekordu spełniającego wskazane kryterium, zachodzi konieczność liniowego przeszukiwania wszystkich bloków. Średni koszt liniowego przeszukiwania jest wyrażony wzorem nr 1 ze slajdu ($0.5N(D+RC)$). Średnio, odszukanie rekordu wymaga odczytu połowy bloków - stąd składnik kosztu $0.5ND$. Odczytane rekordy są następnie przetwarzane w buforze (sprawdzane jest czy rekordy spełniają kryterium poszukiwania), stąd składnik kosztu $0.5NRC$.

W przypadku najgorszym, tj. albo jeżeli nie ma rekordów spełniających warunek selekcji albo poszukiwane rekordy znajdują się w ostatnim bloku, system musi przejrzeć cały plik. W tym przypadku koszt jest wyrażony wzorem nr 2 ($N(D+RC)$).



Plik nieuporządkowany - operacje (2)

- przeglądanie pliku
 - koszt = $N(D + RC)$ 1
 - odczyt N stron z kosztem D
 - dla każdej strony przetworzyć R rekordów z kosztem C
- wyszukiwanie z przedziałem wartości
 - odczyt wszystkich bloków
 - koszt = $N(D + RC)$ 2

BD – wykład 5 (16)

W przypadku operacji przeglądania całego pliku koszt jest wyrażony wzorem nr 1 ze slajdu ($N(D + RC)$) ponieważ trzeba odczytać N stron (D koszt odczytu strony) i dla każdej strony trzeba przetworzyć R rekordów (C koszt przetworzenia pojedynczego rekordu).

Wyszukiwanie z przedziałem wartości również wymaga przeszukania całego pliku. W związku z tym, koszt tak jak poprzednio jest wyrażony wzorem nr 2 ($N(D + RC)$).



Plik nieuporządkowany - operacje (3)

– usuwanie rekordu

- liniowe przeszukiwanie i zapis bloku na dysk
- koszt wyszukania + koszt ($C + D$)
 - średnio: $0.5N(D+RC)+(C+D)$
 - maksymalnie: $N(D+RC)+(C+D)$
- pozostaje zwolniona pamięć \Rightarrow konieczność okresowej reorganizacji pliku

1

2

– sortowanie

- trudne
- stosowane tzw. sortowanie zewnętrzne
 - plik sortowany fragmentami mieszczącymi się w pamięci operacyjnej
 - fragmenty są łączone w większe w kolejnych przebiegach algorytmu sortowania

BD – wykład 5 (17)

Usunięcie rekordu wymaga najpierw jego odszukania, stąd konieczne jest liniowe przeszukiwanie pliku, odczyt do bufora bloku zawierającego usuwany rekord i zapis tego bloku na dysk już po usunięciu rekordu. Stąd, na całkowity koszt operacji usunięcia rekordu składa się koszt wyszukania rekordu oraz koszt przetworzenia (czyli usunięcia) rekordu i koszt zapisu bloku na dysk. Średni koszt usunięcia rekordu jest wyrażony wzorem nr 1: $0.5N(D+RC) + (C + D)$, a maksymalny koszt - wzorem nr 2: $N(D+RC) + (C + D)$.

Ponadto, przy częstym usuwaniu rekordów, w blokach pozostaje zwolniona pamięć. Konieczna jest zatem okresowa reorganizacja pliku w celu odzyskania tej pamięci.

Ze względów efektywnościowych, sortowanie należy wykonywać w pamięci operacyjnej. Sortowanie dużych plików jest zagadnieniem trudnym implementacyjnie. W praktyce bowiem, rozmiar pliku jest znacznie większy niż rozmiar dostępnej pamięci operacyjnej. Techniką sortowania stosowaną najczęściej jest tzw. sortowanie zewnętrzne (ang. external sorting). Koncepcyjnie, polega ono na sortowaniu pliku fragmentami, które mieszczą się w pamięci operacyjnej. Każdy posortowany fragment jest w drugiej fazie sortowania łączony z innymi fragmentami. Łączenie to wymaga zwykle wielu przebiegów.



Plik nieuporządkowany - charakterystyka

- Efektywne wstawianie pojedynczych rekordów i dużych zbiorów rekordów
- Efektywne pozostałe operacje w przypadku plików o rozmiarze kilku bloków
- Struktura właściwa dla odczytu wszystkich rekordów
- Struktura stosowana z innymi strukturami dostępu do danych (np. indeksy)

BD – wykład 5 (18)

Charakterystyka dostępu do pliku nieuporządkowanego jest następująca:

Plik taki umożliwia efektywne wstawianie pojedynczych rekordów i dużych zbiorów rekordów. Pliki o rozmiarze kilku bloków są również efektywne dla pozostałych operacji tj. wyszukiwania rekordów, modyfikowania i usuwania. Plik nieuporządkowany można stosować w przypadkach, gdy są często czytane wszystkie rekordy. Ponadto, jest to struktura stosowana z innymi strukturami dostępu do danych (np. indeksami).



Plik uporządkowany

- Rekordy pliku są porządkowane według pola porządkującego (ang. ordering field)

Aaron Brian			
Abbott Jim			blok 1
Acosta Phil			
<hr/>			
Adams Jim			
Adams Robin			blok 2
Adler John			
<hr/>			
...			
Winslet Kathy			
Zobel Mick			blok N
Zezula Pavol			

BD – wykład 5 (19)

Drugim omawianym rodzajem organizacji plików jest plik uporządkowany. W pliku takim, rekordy w nim składowane są uporządkowane wg wartości tzw. pola porządkującego (ang. ordering field). W przykładowym pliku ze slajdu rekordy zostały uporządkowane wg wartości nazwiska i imienia.



Plik uporządkowany - operacje (1)

- Operacje:
 - przeglądanie pliku
 - koszt = $N(D+RC)$ 1
 - ponieważ wszystkie strony muszą być odczytane
 - wyszukiwanie rekordu
 - wyszukiwanie binarne
 - koszt = $D \cdot \log_2 B + C \cdot \log_2 R$ 2
 - wyszukiwanie z przedziałem wartości
 - koszt wyszukania pierwszego rekordu + koszt selekcji zbioru rekordów

BD – wykład 5 (20)

Operacja przeglądnięcia całego pliku wymaga odczytu wszystkich bloków danych, stąd jej koszt jest wyrażony wzorem 1: $N(D+RC)$, podobnie jak dla pliku nieuporządkowanego.

Wyszukanie rekordu spełniającego warunek selekcji jest realizowane z wykorzystaniem algorytmu wyszukiwania binarnego (połowienia binarnego). Stąd jego koszt jest wyrażony wzorem 2: $D \cdot \log_2 B + C \cdot \log_2 R$.

Operacja wyszukania rekordów spełniających warunek z zadanego przedziału wymaga odszukania pierwszego rekordu spełniającego warunek lewej strony przedziału, a następnie odczytania kolejnych rekordów aż do ostatniego rekordu spełniającego warunek prawej strony przedziału. Koszt odszukania pierwszego rekordu to: $D \cdot \log_2 B + C \cdot \log_2 R$. Koszt odczytania kolejnych rekordów to: ND , gdzie N jest liczbą bloków, w których te rekordy się znajdują.



Plik uporządkowany - operacje (2)

– wstawianie rekordu

- koszt wyszukania miejsca wstawienia rekordu
- koszt przepisania średnio połowy rekordów
- koszt całkowity = $2*(0.5N(D+RC))$ 1

– usuwanie rekordu

- koszt znalezienia usuwanego rekordu
- koszt przepisania średnio połowy rekordów
- identyczny jak koszt wstawiania
- koszt całkowity = $2*(0.5N(D+RC))$; 2

BD – wykład 5 (21)

Wstawianie i usuwanie rekordu są operacjami kosztownymi, ponieważ po zakończeniu tych operacji rekordy muszą pozostać posortowane.

Wstawienie rekordu wymaga: po pierwsze, znalezienia właściwej pozycji w pliku na wstawienie rekordu, zgodnie z wartością atrybutu porządkującego. Po drugie, wymaga utworzenia pustego miejsca w pliku, w które zostanie wstawiony nowy rekord. Operacja utworzenia pustego miejsca wymaga średnio przesunięcia (przepisania) połowy rekordów w miejsce o 1 dalej w pliku. Koszt całkowity operacji wstawienia rekordu jest wyrażony wzorem 1.

Operacja usunięcia rekordu wymaga: po pierwsze znalezienia usuwanego rekordu, i po drugie, oznaczenia tego rekordu jako usunięty. Miejsce po usuniętym rekordzie pozostaje w pliku i nie jest wykorzystywane. W wyniku wielu operacji usunięcia, w pliku będzie wiele pustych miejsc, co ze względów efektywnościowych nie będzie dobre. Z tego względu, plik należy okresowo reorganizować, co jest operacją bardzo kosztowną. W skrajnym przypadku wymaga to przesunięcia wszystkich rekordów. Koszt całkowity operacji usunięcia rekordu jest wyrażony wzorem 2.



Plik uporządkowany - rozwiązywanie problemu wstawiania rekordów

- Przesuwanie średnio połowy rekordów \Leftrightarrow nieefektywne
- Pozostawienie wolnej pamięci w każdym bloku na wstawiane rekordy
 - wstawienie rekordu wymaga przesunięcia rekordów tylko w ramach bieżącego bloku
- Tworzenie nieuporządkowanego pliku rekordów, a następnie łączenie go z plikiem głównym
 - efektywne wstawianie
 - nieefektywne wyszukiwanie

BD – wykład 5 (22)

Problem wstawiania rekordów można rozwiązać na cztery sposoby. Pierwszym jest omówione wcześniej przesuwanie średnio połowy rekordów w pliku, w celu zagwarantowania właściwego porządku rekordów w pliku.

Drugie rozwiązanie zakłada pozostawienie w każdym bloku dyskowym pustego miejsca na wstawiane rekordy. W tym przypadku przesunięcia rekordów należy dokonać tylko w tym bloku, do którego jest wstawiany rekord.

Trzeci sposób wykorzystuje nieuporządkowany tymczasowy plik, zwany nadmiarowym (ang. overflow file) lub transakcyjnym (ang. transaction file). Plik uporządkowany jest nazywany plikiem master (ang. master file) lub głównym (ang. main file). Wstawiane rekordy są umieszczane w pliku nadmiarowym, na jego końcu. Okresowo, plik nadmiarowy jest sortowany i scalany z plikiem głównym. Przy takim podejściu wstawianie jest efektywne, ale wyszukiwanie jest kosztowne. Wymaga ono przeszukania pliku głównego metodą połowienia binarnego. Następnie plik nadmiarowy jest przeszukiwany z wykorzystaniem pełnego jego odczytu. Dla aplikacji, które nie wymagają danych aktualnych, plik nadmiarowy może być ignorowany.



Plik uporządkowany - modyfikowanie rekordu

- Znalezienie i odczyt modyfikowanego rekordu do bufora
 - połowienie binarne dla warunku na pole porządkującym
 - przeszukanie całego pliku dla innego warunku
- Modyfikowanie wartości atrybutu nieporządkującego
 - zmodyfikowanie w buforze i zapis na dysk w to samo miejsce
- Modyfikowanie wartości atrybutu porządkującego
 - zmiana pozycji rekordu w pliku
 - usunięcie rekordu + wstawienie nowego

BD – wykład 5 (23)

Modyfikowanie rekordu wymaga jego znalezienia w pliku i odczytania do bufora. Jeżeli modyfikowany rekord spełnia warunek nałożony na pole porządkujące, wówczas wyszukanie rekordu realizuje się metodą połowienia binarnego. Jeśli natomiast wyspecyfikowano warunek nałożony na inne pole, wówczas wyszukanie rekordu wymaga w najgorszym przypadku odczytania całego pliku.

Jeżeli w znalezionym rekordzie jest modyfikowana wartość pola nieporządkującego, wówczas jest on modyfikowany w buforze i zapisywany na dysk w to samo miejsce. Jeśli natomiast modyfikacji ulega wartość pola porządkującego, wówczas rekord zmieni swoją pozycję w pliku. Implementacyjnie oryginalny rekord jest usuwany i wstawiany jest nowy rekord ze zmodyfikowaną wartością atrybutu porządkującego.



Plik uporządkowany - zalety

- Efektywny odczyt rekordów w kolejności pola porządkującego \Rightarrow posortowane
- Znalezienie następnego rekordu, według określonego porządku, jest bardzo proste
- Metoda połowienia binarnego do wyszukiwania rekordu z warunkiem opartym o pole porządkujące

BD – wykład 5 (24)

Operacja odczytu i sortowania rekordów wg pola porządkującego jest efektywna ponieważ odczytywane rekordy mają już właściwy porządek wynikający z organizacji pliku.

Znalezienie następnego rekordu, według określonego porządku, jest bardzo proste i wymaga odczytania następnego rekordu. Rekord ten znajduje się albo w tym samym bloku albo w bloku następnym.

Jeżeli warunek wyszukiwania rekordu jest oparty na polu porządkującym, wówczas w celu wyszukiwania danych można zastosować szybką metodę wyszukiwania binarnego (połowienia binarnego).



Plik uporządkowany - wady

- Uporządkowanie pliku jest nieprzydatne, gdy wyszukiwanie jest realizowane według wartości pola nie porządkującego pliku danych,
- Kosztowne wstawianie i usuwanie rekordów ze względu na konieczność zachowania porządku w pliku

BD – wykład 5 (25)

Uporządkowanie pliku jest nieprzydatne, gdy wyszukiwanie jest realizowane według wartości pola nie porządkującego pliku danych. W takim przypadku trzeba w najgorszym razie odczytać cały plik.

Wstawianie i usuwanie rekordów jest kosztowne ze względu na konieczność zachowania porządku w pliku.



Plik haszowy

- Plik o strukturze wykorzystującej technikę haszowania \Rightarrow pliku haszowy (bezpośredni)
- Porządek rekordów w pliku określony na podstawie tzw. pola haszowego
- Koncepcja
 - zdefiniowanie funkcji haszowej (ang. hash function) z argumentem, którym jest wartość pola haszowego
 - wynikiem funkcji haszowej jest adres bloku dyskowego, w którym powinien znaleźć się dany rekord
- Haszowanie
 - wewnętrzne
 - zewnętrzne

BD – wykład 5 (26)

Plik o strukturze wykorzystującej technikę haszowania nosi nazwę pliku haszowego (ang. hash file) lub bezpośredniego.

Podstawą określającą porządek rekordów w pliku jest jedno z pól rekordu zwane polem haszowym (ang. hash field).

Koncepcja organizacji tego pliku polega na zdefiniowaniu funkcji haszowej (ang. hash function), która dla argumentu równego wartości pola haszowego wyznacza pewien wynik. Wynik ten jest adresem bloku dyskowego, w którym powinien znaleźć się dany rekord.

W praktyce stosuje się dwie techniki haszowania, tj. haszowanie wewnętrzne i haszowanie zewnętrzne. Obie zostaną omówione w niniejszym wykładzie.



Haszowanie wewnętrzne

Dana jest tablica rekordów o M szczelinach, których adresy odpowiadają indeksom tablicy haszowej

Indeks tablicy: 0 \Rightarrow M-1

	Id_prac	Nazwisko	Etat	Płaca
0	450	Nowak	kierownik	2000
1	120	Kuzio	portier	1000
2	220	Misiek	z-ca kier.	1800
M-2	100	Dziubek	dyrektor	5000
M-1	110	Gulczas	portier	800

Funkcja haszowa $H(K=\text{wartość pola haszowego}) \rightarrow \{0, \dots, M-1\}$

Najczęściej spotykana funkcja haszowa $H(K)=K \bmod M$

BD – wykład 5 (27)

Koncepcja haszowania wewnętrznego jest następująca.

Przyjmijmy, że dana jest tablica rekordów o M szczelinach. Adresy szczelin odpowiadają indeksom tablicy haszowej. Indeksy te przyjmują wartości od 0 do M-1.

Przyjmijmy funkcję haszową postaci: $H(K=\text{wartość pola haszowego}) \rightarrow \{0, \dots, M-1\}$. Funkcja ta transformuje wartość pola haszowego do liczby całkowitej z zakresu od 0 do M-1.

Najczęściej spotykaną funkcją haszową jest funkcja $h(K)=K \bmod M$.

W przykładzie ze slajdu polem haszowym mógłby być Id_prac lub nazwisko. Wynikiem działania funkcji haszowej byłby numer odpowiedniej szczeliny, w której umieszczony zostałby konkretny rekord.



Haszowanie wewnętrzne - przykład

Id_Prac	Nazwisko	Etat
16	Kociołek	portier
14	Misiek	kierownik
12	Pantarka	sekretarka
11	Oleks	portier
13	Bamczyk	kierowca
15	Paker	ochroniaż
10	Karczek	ochroniaż

szczelina
0 10; Karczek; ochroniaż
1 11; Oleks; protier
2 12; Pantarka; sekretarka
3 13; Bamczyk; kierowca
4 14; Misiek; kierownik
5 15; Paker; ochroniaż
6 16; Kociołek; portier

$$H(Id_Prac) = Id_Prac \bmod 10$$

BD – wykład 5 (28)

Jako przykład ilustrujący haszowanie wewnętrzne rozważmy rekordy pracowników z polami: Id_Prac, Nazwisko, Etat. Przyjmijmy tablicę haszową ze szczelinami o numerach od 0 do 6. Zdefiniujmy funkcję haszową postaci: $H(Id_Prac)=Id_Prac \bmod 10$. Funkcja ta oblicza modulo 10 z wartością atrybutu Id_Prac. Wynikiem działania funkcji są numery szczelin w tablicy haszowej. W szczelinach tych są następnie umieszczane rekordy.



Proces haszowania

- Proces haszowania składa się z dwóch kroków:
 - transformacji
 - pola nienumeryczne są transformowane do liczb całkowitych
 - haszowania
- Przykładowy algorytm transformacji

```
temp := 1;  
for i = 1 to 20 do  
    temp := temp * code(K[i])  
hash_address := temp MOD M;
```

BD – wykład 5 (29)

Argumentem funkcji modulo musi być wartość całkowita. Oznacza to, że przed zastosowaniem haszowania należy przetransformować wartości nienumeryczne do numerycznych. Na slajdzie przedstawiono prosty algorytm, który transformuje 20 znaków zapisanych w tablicy K do wartości numerycznych i tak przetransformowane wartości transformuje za pomocą funkcji MOD M.



Kolizja

- Kolizja \Leftrightarrow wartość funkcji haszowej dla danej wartości pola haszowego nowego rekordu odpowiada zajętemu już adresowi szczeliny
- Źródło kolizji \Leftrightarrow funkcja haszowa generująca te same adresy szczeliny dla różnych wartości atrybutu haszowego
- Rozwiążanie kolizji \Leftrightarrow procedura znajdowania innej lokalizacji dla wprowadzanego rekordu

BD – wykład 5 (30)

Zasadniczym problemem w przypadku haszowej organizacji pliku jest tzw. problem kolizji. Kolizja występuje wtedy, gdy wartość funkcji haszowej dla danej wartości pola haszowego nowego rekordu odpowiada adresowi szczeliny, w której znajduje się już inny rekord. Kolizja jest spowodowana tym, że żadna funkcja haszowa nie gwarantuje, że różnym wartościom pola haszowego będą odpowiadały różne adresy szczelin.

Rozwiązaniem problemu kolizji jest zastosowanie procedury znajdowania innej lokalizacji dla wprowadzanego rekordu.



Metody rozwiązywania kolizji

- Adresowanie otwarte (open addressing)
 - następna wolna lokalizacja
- Łącuchowanie (chaining)
 - wskaźniki do nowych lokalizacji
- Haszowanie wielokrotne (multiple hashing)
 - nowa funkcja haszowa
- Każda metoda rozwiązywania kolizji wymaga własnych algorytmów wstawiania, wyszukiwania i usuwania rekordów

BD – wykład 5 (31)

Wyróżnia się trzy podstawowe metody rozwiązywania kolizji, mianowicie: adresowanie otwarte (ang. open addressing), łańcuchowanie (ang. chaining) i haszowanie wielokrotne (ang. multiple hashing). Każda z tych metod wymaga własnych algorytmów wstawiania, wyszukiwania i usuwania rekordów.

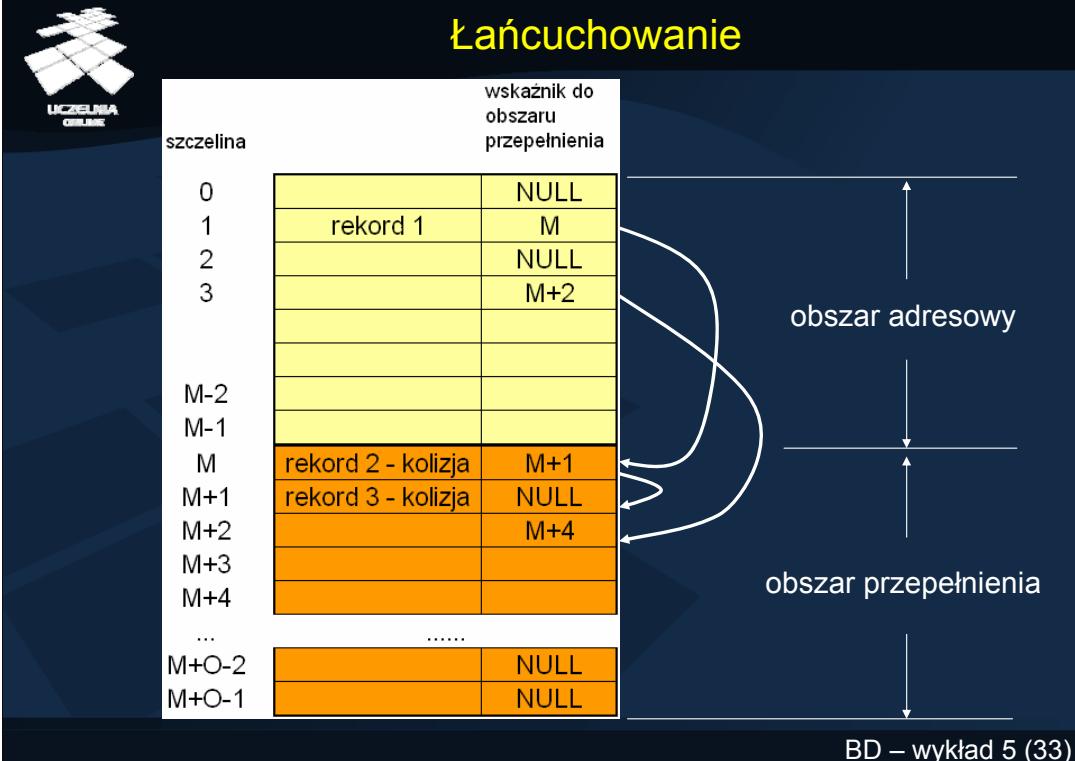


Adresowanie otwarte

```
i := adres_szczeliny --pierwotny adres szczeliny
a := i;
if lokalizacja i zajęta then
    begin
        i := (i + 1) mod M;
        while (i ≠ a) and lokalizacja i zajęta
            do
                i := (i + 1) mod M;
                if (i = a)
                    then wszystkie pozycje są zajęte;
                else nowy_adres := i;
    end;
```

BD – wykład 5 (32)

Adresowanie otwarte polega na znalezieniu następnej najbliższej wolnej szczeliny.
Pseudo-kod algorytmu adresowania otwartego przedstawiono na slajdzie.



Technika łańcuchowania polega na przechowywaniu w szczelinie dodatkowo wskaźnika do tzw. obszaru przepelnienia (ang. overflow space). Służy on do przechowywania wszystkich rekordów ulegających kolizji w danej szczelinie. Rekordy w obszarze przepelnienia tworzą listę.

Rozważmy rysunek ze slajdu. Przyjmijmy, że pierwszy rekord ("rekord 1") jest umieszczany w szcelinie o adresie 1, zgodnie z pewną funkcją haszową. Kolejny rekord ulega kolizji w szcelinie 1 i jest umieszczany w obszarze przepelnienia o adresie M ("rekord 2 - kolizja"). We wskaźniku do obszaru przepelnienia szcelyn 1 jest umieszczany adres szcelyn przechowujacej pierwszy rekord z kolizji, czyli adres M. Przyjmijmy dalej, że kolejny rekord ulega kolizji w szcelinie 1 i jest on umieszczany w kolejnej wolnej szcelinie obszaru przepelnienia. W naszym przypadku jest to "rekord 3 - kolizja" umieszczony w szcelinie M+1. Adres tej szcelyn jest zapisywany w polu wskaźnika do obszaru przepelnienia szcelyn M. NULL w polu wskaźnika do obszaru przepelnienia oznacza brak wskaźnika.



Haszowanie wielokrotne

- Jeżeli występuje kolizja z wykorzystaniem funkcji haszowej H1, to stosowana jest druga funkcja haszowa H2
- Jeżeli H2 również powoduje kolizję to stosuje się trzecią funkcję haszową lub haszowanie otwarte

BD – wykład 5 (34)

Ostatnią omawianą techniką rozwiązywania kolizji jest haszowanie wielokrotne. Jeżeli występuje kolizja z wykorzystaniem funkcji haszowej H1, to w haszowaniu wielokrotnym stosowana jest druga/inna funkcja haszowa H2, której rezultatem jest inny adres. Jeśli funkcja H2 również powoduje kolizję to stosuje się trzecią funkcję haszową lub haszowanie otwarte.



Haszowanie zewnętrzne (1)

- Wartościami tablicy haszowej są adresy logicznych obszarów dyskowych (LOD) (block buckets)
- Liczba LOD jest stała i równa liczbie szczelin w tablicy haszowej \Rightarrow jest to tzw. **haszowanie statyczne** (static hashing)
- LOD jest albo pojedynczym blokiem dyskowym albo zbiorem kolejnych (leżących obok siebie) bloków dyskowych
- Funkcja haszowa odwzorowuje wartość atrybutu haszowego w numer LOD
- Plik dyskowy zawiera tablicę konwersji numerów LOD w fizyczne adresy bloków dyskowych

BD – wykład 5 (35)

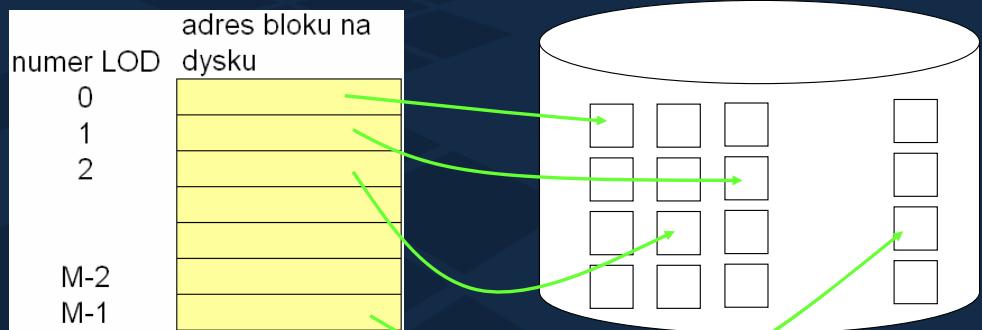
W haszowaniu zewnętrznym wartościami tablicy haszowej są adresy logicznych obszarów dyskowych (LOD) (ang. block buckets). Liczba LOD jest stała i jest równa liczbie szczelin w tablicy haszowej. Ponieważ liczba LOD jest stała, więc technika ta nazywa się haszowaniem statycznym.

Każdy z LOD jest albo pojedynczym blokiem dyskowym albo zbiorem kolejnych (leżących obok siebie) bloków dyskowych. Funkcja haszowa odwzorowuje wartość atrybutu haszowego w numer LOD. Plik dyskowy zawiera tablicę konwersji numerów LOD w fizyczne adresy bloków dyskowych.



Haszowanie zewnętrzne (2)

tablica konwersji



BD – wykład 5 (36)

Omówioną koncepcję haszowania zewnętrznego ilustruje slajd.



W haszowaniu zewnętrznym kolizje występują rzadziej niż w omawianej wcześniej technice haszowania wewnętrznego. Dzieje się tak dla tego, że ten sam LOD, którego numer jest wynikiem działania funkcji haszowej, może pomieścić wiele rekordów. Kolizja może jednak wystąpić po zapełnieniu się wszystkich bloków dyskowych wchodzących w skład danego LOD. W takiej sytuacji, alokowany jest obszar nadmiarowy. LOD zawiera wskaźnik do pierwszego rekordu tego obszaru. Kolejny rekord ulegający kolizji w tym samym LOD będzie zapisany w obszarze nadmiarowym, a wskaźnik do tego rekordu znajdzie się w poprzednim rekordzie obszaru nadmiarowego. Koncepcję tę ilustruje slajd.

LOD0 został w całości zapełniony rekordami 1 do l. Kolejny rekord m powinien trafić do LOD0. Z braku miejsca jest on umieszczany w obszarze nadmiarowym. W LOD0 jest umieszczany wskaźnik do tego rekordu. Następnie rekord m+1 również powinien trafić do LOD0 i jest umieszczany w obszarze nadmiarowym. Do rekordu m+1 prowadzi wskaźnik z rekordu m.



Haszowanie zewnętrzne - operacje (1)

- Poszukiwanie rekordu z warunkiem nałożonym na pole inne niż haszowe
 - przeszukanie całego pliku i obszaru nadmiarowego
- Poszukiwanie rekordu z warunkiem nałożonym na pole haszowe
 - funkcja haszowa
 - w przypadku kolizji przeszukanie obszaru nadmiarowego

BD – wykład 5 (38)

Poszukiwanie w pliku haszowym rekordu z warunkiem nałożonym na pole inne niż haszowe wymaga przeszukanie całego pliku i obszaru nadmiarowego. Natomiast poszukiwanie rekordu z warunkiem nałożonym na pole haszowe jest realizowane z wykorzystaniem funkcji haszowej, która generuje adres szczeliny z poszukiwanym rekordem. W przypadku kolizji, wymagane jest dodatkowo przeszukanie obszaru nadmiarowego z wykorzystaniem listy łączonej wskaźników do rekordów w tym obszarze.



Haszowanie zewnętrzne - operacje (2)

- Usunięcie rekordu z LOD
 - wyszukanie rekordu (funkcja haszowa + tablica konwersji) i zwolnienie szczeliny
 - przesunięcie pierwszego rekordu z obszaru nadmiarowego do LOD
- Usunięcie rekordu z obszaru nadmiarowego
 - wyszukanie rekordu (funkcja haszowa) + przeszukanie listy rekordów w obszarze nadmiarowym
 - zwolnienie szczeliny
 - utrzymywanie listy wolnych szczelin

BD – wykład 5 (39)

Usunięcie rekordu z pliku haszowego przebiega w dwóch wariantach. W wariantie pierwszym, usuwany rekord znajduje się w logicznym obszarze dyskowym. Jego usunięcie polega na znalezieniu szczeliny (z wykorzystaniem funkcji haszowej i tablicy konwersji) i jej zwolnieniu. Dodatkowo, pierwszy rekord z obszaru nadmiarowego może zostać przesunięty do zwolnionej szczeliny.

W wariantie drugim, usuwany rekord znajduje się w obszarze nadmiarowym. Jego usunięcie polega na znalezieniu szczeliny (z wykorzystaniem funkcji haszowej i tablicy konwersji) w LOD. Następnie za pomocą wskaźnika do pierwszego rekordu w obszarze nadmiarowym przeszukuje się listę rekordów w tym obszarze. Znaleziony rekord jest usuwany, a jego szczelina zwalniana. Dokonywana jest też zmiana wartości wskaźnika w rekordzie, który adresował usunięty rekord. Nowa wartość wskazuje na następny rekord wskazywany z rekordu usuniętego. W celu zoptymalizowania alokowania rekordów w obszarze nadmiarowym jest utrzymywana lista wolnych szczelin.



Haszowanie zewnętrzne - operacje (3)

- Wstawienie rekordu
 - odczyt adresu szczeliny (funkcja haszowa)
 - w przypadku kolizji zaalokowanie szczeliny w obszarze nadmiarowym
- Zmodyfikowanie wartości pola haszowego
 - odczytanie rekordu (funkcja haszowa)
 - rekord zmienia szczelinę
 - usunięcie rekordu + wstawienie rekordu z nową wartością
- Zmodyfikowanie wartości pola nie-haszowego
 - odczytanie rekordu (funkcja haszowa)
 - zmodyfikowanie wartości
 - rekord nie zmienia szczeliny

BD – wykład 5 (40)

Wstawienie rekordu do pliku haszowego polega na odczytaniu adresu szczeliny z wykorzystaniem funkcji haszowej i zapisaniu rekordu do tej szczeliny. W przypadku kolizji, alokowana jest szczelina w obszarze nadmiarowym i aktualniany jest wskaźnik prowadzący do tej szczeliny.

Zmodyfikowanie wartości pola haszowego polega na odczytaniu rekordu z wykorzystaniem funkcji haszowej. Ponieważ zmiana wartości pola haszowego wymaga przeniesienia rekordu do innej szczeliny, fizycznie rekord stary jest usuwany i wstawiany jest nowy rekord do właściwej szczeliny.

Zmodyfikowanie wartości pola nie-haszowego polega na odczytaniu rekordu, zmodyfikowaniu wartości pola i zapisaniu rekordu do tej samej szczeliny.



Funkcja haszowa

- Cechą dobrej funkcji haszowej jest zapewnienie równomiernego rozkładu rekordów w obrębie przestrzeni adresowej tablicy haszowej
- Zalecany rozmiar tablicy haszowej

$$r/M \in (0,7 \div 0,9)$$

$r \Rightarrow$ liczba rekordów

$M \Rightarrow$ liczba bloków

BD – wykład 5 (41)

Dobra funkcja haszowa to taka, która odwzorowuje wartości kluczy rozłożone nierównomiernie w obrębie dużej dziedziny, w przestrzeń adresową rekordów o znacznie mniejszym rozmiarze w taki sposób, że wynikowe adresy są równomiernie rozłożone w obrębie przestrzeni adresowej tablicy haszowej.

Praktyka pokazuje, że tablica haszowa powinna być zajęta w 70 do 90%, tak aby zapewnić niewielką liczbę kolizji i nie powodować zbyt dużego marnowania obszaru dyskowego. Stąd zalecany rozmiar tablicy haszowej jest wyrażony wzorem

r/M between 0.7 and 0.9, gdzie r jest liczbą rekordów, a M jest liczbą szczelin adresowych tablicy haszowej.



Charakterystyka plików haszowych

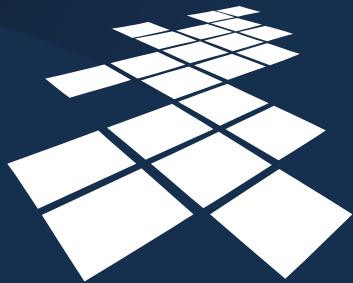
- Problem porządkowania pliku oraz wyszukiwania rekordów w porządku wartości pola haszowego
- Problem stałego rozmiaru przestrzeni adresowej przydzielonej plikowi
 - częste kolizje

BD – wykład 5 (42)

Pliki haszowe są nieefektywne dla operacji odczytu danych z ich porządkowaniem zgodnie z wartością pola haszowego. Jest to konsekwencją działania funkcji haszowej, która burzy porządek wartości pola haszowego rozmieszczając rekordy sąsiednie w różnych blokach. Ponadto, haszowanie statyczne zakłada stały rozmiar przestrzeni adresowej przydzielonej plikowi. Konsekwencją tego ograniczenia jest duże prawdopodobieństwo występowania kolizji.

Indeksy

Wykład przygotował:
Robert Wrembel



UCZELNIA
ONLINE

BD – wykład 7 (1)



Plan wykładu

- Problematyka indeksowania
- Podział indeksów i ich charakterystyka
 - indeks podstawowy, zgrupowany, wtórny
 - indeks rzadki, gęsty
- Indeks wielopoziomowy statyczny (ISAM)
- Indeks wielopoziomowy dynamiczny (B^+ -drzewo)
- Algorytm wstawiania danych do indeksu B^+ -drzewo

BD – wykład 7 (2)

Celem wykładu jest omówienie podstawowych koncepcji indeksowania danych i struktur indeksowych. W ramach wykładu zostaną omówione:

- wprowadzenie do problematyki indeksowania danych,
- charakterystyka różnego rodzaju indeksów (podstawowy, zgrupowany, wtórny, rzadki i gęsty),
- indeks wielopoziomowy statyczny (ISAM),
- indeks wielopoziomowy dynamiczny (B^+ -drzewo),
- algorytm wstawiania danych do indeksu B^+ -drzewo.



Wprowadzenie (1)

- Problem:
 - Dany jest plik zawierający uporządkowane lub nieuporządkowane rekordy danych
 - W jaki sposób efektywnie zrealizować wyszukanie rekordu lub rekordów z zadanego zakresu wartości wybranego pola?

BD – wykład 7 (3)

Rozważmy plik zawierający uporządkowane lub nieuporządkowane rekordy danych. Jak pamiętamy z poprzedniego wykładu wyszukiwanie danych w plikach nie jest efektywne. Z tego względu chcielibyśmy znaleźć sposób efektywnego wyszukiwania danych.



Wprowadzenie (2)

- Odpowiedź:
 - Utworzyć drugi plik, zdefiniowany na atrybucie wykorzystanym do specyfikacji kryterium poszukiwania
 - Plik ten zawiera rekordy odpowiadające poszukiwanym wartościom pierwszych rekordów w poszczególnych blokach pliku danych
 - Rekordy w dodatkowym pliku mają postać:
 \langle pierwszy klucz w bloku, wskaźnik do bloku \rangle
 - Plik dodatkowy jest uporządkowany według wartości poszukiwanych

BD – wykład 7 (4)

Rozwiążanie tego problemu bazuje na wykorzystaniu drugiego pliku zdefiniowanego na atrybucie wykorzystanym do specyfikowania kryterium przeszukiwania. Plik ten zawierałby rekordy odpowiadające poszukiwanym wartościom pierwszych rekordów w poszczególnych blokach pliku danych. Rekordy w dodatkowym pliku miałyby postać: \langle pierwszy klucz w bloku, wskaźnik do bloku \rangle , a plik dodatkowy byłby uporządkowany według wartości poszukiwanych.

Taki plik dodatkowy nazywa się indeksem.



Wprowadzenie (3)

Indeks - dodatkowa struktura fizyczna

- Cel stosowania - przyśpieszenie dostępu do danych
- Zakładane na pojedynczych atrybutach lub zbiorach atrybutów relacji
 - atrybuty te są nazywane indeksowymi
- Model fizyczny indeksu
 - uporządkowany plik rekordów indeksu (ang. data entry) o stałej długości
 - rekord indeksu zawiera dwa pola
 - **klucz** reprezentujący jedną z wartości występujących w atrybutach indeksowych relacji
 - **wskaźnik** do bloku danych zawierający krotkę, której atrybut indeksowy równy jest kluczowi

BD – wykład 7 (5)

Indeks zdefiniowany na pliku jest dodatkową strukturą fizyczną, której celem jest przyspieszenie wykonywania operacji, które nie są wystarczająco efektywnie wspierane przez podstawowe organizacje plików i struktury logiczne danych.

Indeksy są zakładane na pojedynczych atrybutach lub zbiorach atrybutów relacji. Atrybuty te noszą nazwę atrybutów indeksowych.

Indeks jest uporządkowanym plikiem rekordów indeksu (ang. data entry) o stałej długości. Rekordy indeksu zawierają dwa pola: klucz reprezentujący jedną z wartości występujących w atrybutach indeksowych relacji oraz wskaźnik do bloku danych zawierający krotkę, której atrybut indeksowy równy jest kluczowi.



Wprowadzenie (4)

- Rekord indeksu - k^*
 - zawiera dostateczną informację umożliwiającą wyszukanie (jednego lub więcej) rekordów danych o wartości klucza k
- Pytanie:
 - W jaki sposób rekordy indeksu powinny być zorganizowane, aby efektywnie wspierać wyszukiwanie rekordów o danej wartości klucza?
 - Co powinien zawierać rekord indeksu?

BD – wykład 7 (6)

Każdy rekord indeksu, oznaczony k^* , zawiera dostateczną informację umożliwiającą wyszukanie (jednego lub więcej) rekordów danych o wartości klucza k .

Projektując strukturę indeksu należy odpowiedzieć na dwa pytania:

Po pierwsze, w jaki sposób rekordy indeksu powinny być zorganizowane, aby efektywnie wspierać wyszukiwanie rekordów o danej wartości klucza? Po drugie, co powinien zawierać rekord indeksu?



Rekordy indeksu

- Typy rekordów indeksu:
 1. Rekord indeksu **k*** jest rekordem danych (o wartości klucza k)
 2. Rekord indeksu jest parą **<k, rid>**, gdzie **rid** jest identyfikatorem rekordu danych o wartości klucza **k**
 3. Rekord indeksu jest parą **<k, rid-list>**, gdzie **rid-list** jest listą identyfikatorów rekordów danych o wartości klucza k
 4. Rekord indeksu jest parą **<k, bitmapa>**, gdzie **bitmapa** jest wektorem 0 i 1 reprezentującym zbiór rekordów danych

BD – wykład 7 (7)

Rekord indeksu **k*** umożliwia wyszukanie rekordów danych o wartości klucza k.

Wyróżnia się cztery typy rekordów indeksu:

1. Rekord indeksu **k*** jest rekordem danych (o wartości klucza k).
2. Rekord indeksu jest parą **<k, rid>**, gdzie **rid** jest identyfikatorem rekordu danych o wartości klucza k.
3. Rekord indeksu jest parą **<k, rid-list>**, gdzie **rid-list** jest listą identyfikatorów rekordów danych o wartości klucza k.
4. Rekord indeksu jest parą **<k, bitmapa>**, gdzie **bitmapa** jest wektorem 0 i 1 reprezentującym zbiór rekordów danych.



Rodzaje indeksów - charakterystyka atrybutu indeksowego

- Indeks podstawowy (primary index)
 - założony na atrybucie porządkującym unikalnym
- Indeks zgrupowany (clustering index)
 - założony na atrybucie porządkującym nieunikalnym
- Indeks wtórny (secondary index)
 - założony na atrybucie nieporządkującym

BD – wykład 7 (8)

Z punktu widzenia charakterystyki atrybutu indeksowego wyróżnia się trzy rodzaje indeksów:

- indeks podstawowy (ang. primary index)
- indeks zgrupowany (ang. clustering index)
- indeks wtórny (ang. secondary index).

Indeks podstawowy jest założony na atrybucie porządkującym indeksowanego pliku. Atrybut porządkujący określa porządek rekordów w pliku. Dla indeksu podstawowego atrybut porządkujący przyjmuje wartości unikalne.

Indeks zgrupowany jest zakładany również na atrybucie porządkującym, ale w tym przypadku jego wartości nie są unikalne.

Indeks wtórny jest zakładany na atrybucie, który nie jest atrybutem porządkującym pliku.



Rodzaje indeksów - wskazania do pliku danych

- Indeks gęsty (dense)
 - posiada rekord indeksu dla każdego rekordu indeksowanego pliku danych
- Indeks rzadki (sparse)
 - posiada rekordy tylko dla wybranych rekordów indeksowanego pliku danych

BD – wykład 7 (9)

Z punktu widzenia liczby wskazań indeksu do pliku danych wyróżnia się indeksy gęste (ang. dense) i indeksy rzadkie (ang. sparse). Indeks gęsty posiada rekord indeksu dla każdego rekordu indeksowanego pliku danych. Indeks rzadki posiada rekordy tylko dla wybranych rekordów indeksowanego pliku danych.



Rodzaje indeksów - liczba poziomów

- Indeksy jednopoziomowe
 - jeden plik indeksu dla jednego pliku danych
- Indeksy wielopoziomowe
 - indeks do indeksu

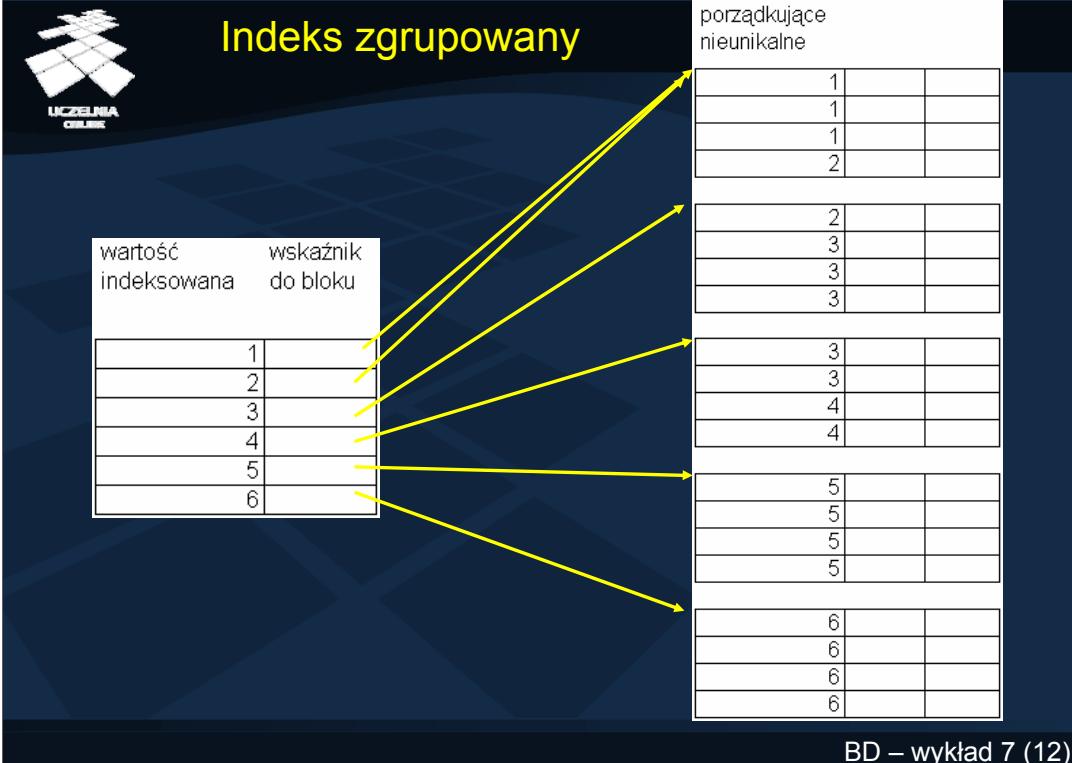
BD – wykład 7 (10)

Z punktu widzenia liczby poziomów indeksu wyróżnia się indeksy jednopoziomowe i wielopoziomowe. W pierwszym przypadku, dla pliku danych jest tworzony tylko jeden indeks (plik indeksu). W drugim przypadku, dla pliku danych tworzy się indeks (jak w przypadku pierwszym). Dodatkowo, do indeksu tworzy się dodatkowy indeks.



Indeks podstawowy jest indeksem rzadkim ponieważ nie wszystkie rekordy pliku danych posiadają rekordy indeksowe. Rekord indeksowy indeksu podstawowego dla wartości X zawiera adres bloku danych, w którym znajduje się rekord danych z wartością atrybutu indeksowego równą X.

Poglądową strukturę indeksu podstawowego przedstawia slajd. Jak widać, wskazania z rekordów indeksowych prowadzą do bloków danych.



Indeks zgrupowany jest również indeksem rzadkim ponieważ nie wszystkie rekordy pliku danych posiadają rekordy indeksowe. Rekord indeksowy indeksu zgrupowanego dla wartości X zawiera adres bloku danych, w którym znajduje się pierwszy rekord danych z wartością atrybutu indeksowego równą X. Widać to wyraźnie dla wartości 2 rekordu indeksu. Rekord ten posiada wskazanie do pierwszego bloku danych - w bloku tym znajduje się pierwszy rekord z wartością 2 indeksowanego atrybutu.

Taka organizacja pliku powoduje problemy z wstawianiem i rekordów, ponieważ porządek rekordów po wstawieniu musi pozostać zachowany.



Indeks zgrupowany z blokami nadmiarowymi

wartość indeksowana	wskaźnik do bloku
1	
2	
3	
4	
5	
6	

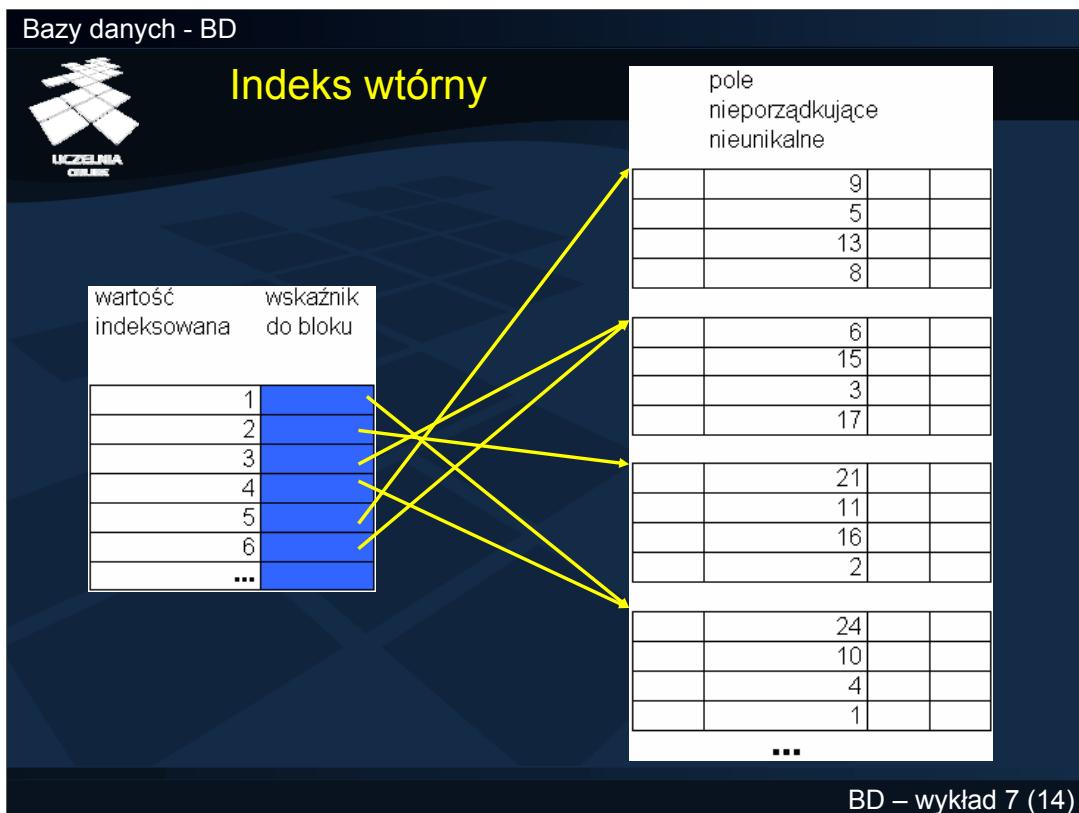
pole porządkujące nieunikalne	1	
wskaźnik do bl. nadmiarowego		
	2	
	2	
wskaźnik do bl. nadmiarowego		
	3	
	3	
	3	
	3	
wskaźnik do bl. nadmiarowego		
	3	
	3	
	3	
wskaźnik do bl. nadmiarowego		

blok nadmiarowy

BD – wykład 7 (13)

Rozwiązaniem tego problemu jest po pierwsze, zarezerwowanie całego bloku na rekordy z tą samą (jedną) wartością. Po drugie, zastosowanie bloków nadmiarowych, jak pokazuje slajd. Wstawiane rekordy są umieszczane w wolnych szczelinach bloku głównego, a po jego zapełnieniu - w odpowiednim bloku nadmiarowym, do którego prowadzi wskaźnik z bloku głównego.

W przykładzie ze slajdu, rekord z wartością 3 nie mieści się w bloku głównym, więc jest składowany w bloku nadmiarowym. Z bloku głównego przechowującego rekordy z wartością atrybutu indeksowego równą 3 prowadzi wskaźnik do bloku nadmiarowego.



Indeks wtórny jest również uporządkowany. Jest on zakładany na atrybutie indeksowym pliku danych, który nie jest atrybutem porządkującym tego pliku. Każdy rekord pliku danych posiada swój odpowiednik w rekordzie indeksu. Stąd, indeks wtórny jest indeksem gęstym.

Rekord indeksu wtórnego składa się z dwóch pól: wartości pola indeksowego i wskaźnika albo do rekordu albo do bloku danych zawierającego ten rekord.

Na slajdzie przedstawiono gęsty indeks wtórny założony na polu nieporządkującym, którego wartości są unikalne. Rekordy indeksu posiadają wskaźniki do bloków danych.



Indeks o kluczu złożonym

- Klucz indeksu może zawierać kilka atrybutów - klucz złożony

Indeks<wiek; pensja>

32; 4300
35; 4500
42; 5000
46; 5800

Indeks<pensja; wiek>

4300; 32
4500; 35
5000; 42
5800; 46

Indeks o kluczu złożonym

Indeks<wiek>

32
35
42
46

Indeks<pensja>

4300
4500
5000
5800

BD – wykład 7 (15)

Klucz indeksu może zawierać kilka atrybutów – taki klucz nazywamy kluczem złożonym. W przykładzie ze slajdu zdefiniowano cztery różne indeksy. Pierwszy na kluczu złożonym <wiek, pensja>, drugi na kluczu złożonym <pensja, wiek>, trzeci na kluczu <wiek>, a czwarty na kluczu <pensja>.

Indeksy na kluczach złożonych stosuje się do wyszukiwania rekordów spełniających warunki równościowe (tzw. zapytania punktowe) nałożone jednocześnie na pola występujące w kluczu złożonym.

Indeks <wiek, pensja> będzie przydatny do wyszukiwania rekordów spełniających warunki równościowe nałożone jednocześnie na pole wiek i na pole pensja. Może też zostać wykorzystany przy warunkach nałożonych na inne pola, razem z warunkami nałożonymi na wiek i pensję. Ponadto, indeks <wiek, pensja> może zostać wykorzystany do wyszukiwania rekordów z warunkiem nałożonym tylko na pole wiek.

Indeks ten będzie jednak nieprzydatny do poszukiwania rekordów z warunkami nałożonymi na atrybut pensja, ponieważ atrybutem wiodącym tego indeksu jest wiek. W takim przypadku, mógłby zostać wykorzystany indeks <pensja, wiek>.

Z tych powodów, bardziej uniwersalne są indeksy zakładane na pojedynczych atrybutach. Są one jednak mniej efektywne od indeksów złożonych w przypadku warunków poszukiwania jednocześnie nałożonych na indeksowane atrybuty.



Indeks wielopoziomowy - ISAM

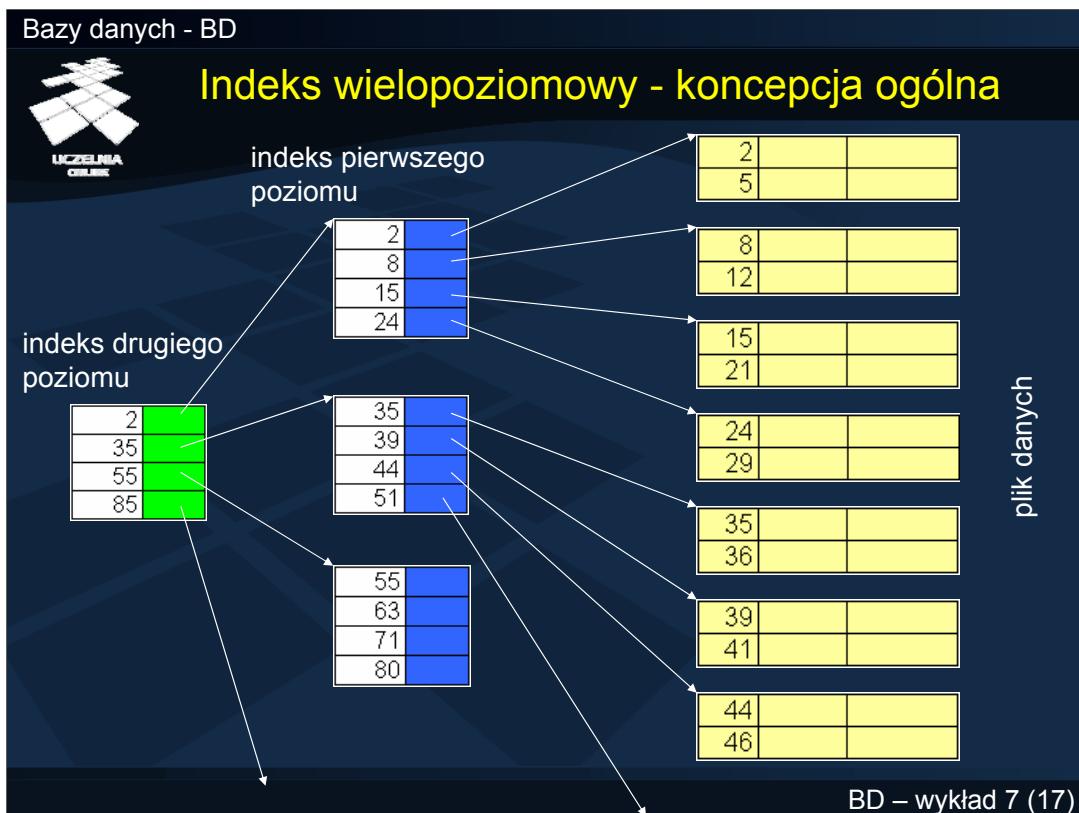
- ISAM - Indexed Sequential Access Method (IBM)
 - poziom pierwszy:
 - indeks cylindrów <klucz, adres indeksu ścieżki>
 - poziom drugi:
 - indeks ścieżki <klucz, adres ścieżki>
 - struktura silnie związana ze sprzętem
- VSAM - Virtual Sequential Access Method
 - rozwinięcie ISAM
 - niezależne od sprzętu

BD – wykład 7 (16)

Wyszukanie danych z wykorzystaniem indeksu jednopoziomowego wymaga przeszukiwania pliku indeksu. Z wykorzystaniem znalezionych rekordów indeksu następuje odczytanie rekordów danych. Należy zwrócić uwagę na fakt, że plik indeksu jest przeszukiwany algorytmem połownienia binarnego ponieważ jest to plik uporządkowany. Algorytm ten nie należy do efektywnych. Z tego względu wprowadzono indeksy wielopoziomowe, których efektywność przeszukiwania jest większa.

Jedną z fundamentalnych koncepcji indeksu wielopoziomowego jest struktura ISAM (ang. Indexed Sequential Access Method), oryginalnie opracowana przez IBM. Koncepcyjnie struktura ta jest zbudowana z dwóch poziomów. Poziom pierwszy indeksuje cylindry. Rekordy indeksu na tym poziomie zawierają pary wartości: poszukiwany klucz i adres do indeksu ścieżki dyskowej. Poziom drugi indeksuje ścieżki. Jego rekordy zawierają pary wartości: poszukiwany klucz i adres ścieżki. Jak widać, jest to struktura silnie związana z zastosowanym sprzętem komputerowym.

Rozwinięciem struktury ISAM jest VSAM (ang. Virtual Sequential Access Method). VSAM jest już niezależna od rozwiązań sprzętowych.



BD – wykład 7 (17)

Ogólną koncepcję indeksu wielopoziomowego przedstawiono na slajdzie. Indeks na pierwszym poziomie adresuje bloki danych. Każdy rekord tego indeksu zawiera wartość pola indeksowego i adres bloku danych, w którym ta wartość się znajduje. Rekordy tego indeksu są przechowywane w pliku uporządkowanym zgodnie z wartościami klucza indeksu pierwszego poziomu. Rekordy indeksu drugiego poziomu zawierają adresy bloków indeksu pierwszego poziomu.



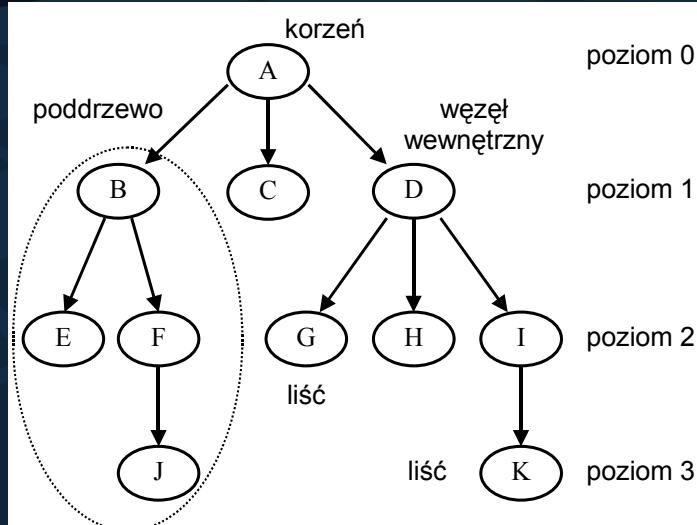
- Indeks statyczny
 - modyfikowanie zawartości pliku degeneruje indeks - spada efektywność dostępu do danych
 - powstają puste obszary po usuniętych rekordach
 - wstawiane rekordy trafiają do bloków nadmiarowych

BD – wykład 7 (18)

ISAM jest indeksem statycznym, co oznacza, że nie posiada zaawansowanych mechanizmów modyfikowania struktury w sytuacji zmodyfikowania zawartości indeksowanego pliku (dodanie, zmodyfikowanie, usunięcie rekordu). Usunięcie rekordu powoduje powstanie pustego miejsca w bloku indeksu. Nowe rekordy są dodawane do bloków przepełnienia. W konsekwencji struktura indeksu typu ISAM staje się nieefektywna.

Rozwiązaniem tego problemu jest wprowadzenie indeksów dynamicznych. Najpowszechniej stosowanymi indeksami dynamicznymi są indeksy drzewiaste, S-drzewa, B-drzewa, B⁺-drzewa.

Struktura drzewiasta



BD – wykład 7 (19)

Na slajdzie przedstawiono ogólną strukturę drzewiastą. Wyróżnia się w niej tzw. korzeń, będący wierzchołkiem (punktem wejścia) całej struktury. Na slajdzie korzeniem jest węzeł A. Z korzenia prowadzą łuki, czyli wskazania albo do węzłów wewnętrznych (węzły B i D na rysunku) albo do liści (węzeł C). Węzeł wewnętrzny posiada wskazania do innych węzłów. Liść nie posiada wskazań do innych węzłów. Jest więc elementem końcowym całej struktury. Przykładowy indeks ze slajdu składa się z 4 poziomów. Przy czym korzeń znajduje się na poziomie 0.



Indeks B⁺-drzewo

- Zrównoważona struktura drzewiasta
 - wierzchołki wewnętrzne służą do wspomagania wyszukiwania
 - wierzchołki liści zawierają rekordy indeksu ze wskaźnikami do rekordów danych
- W celu zapewnienia odpowiedniej efektywności realizacji zapytań przedziałowych wierzchołki liści stanowią listę dwukierunkową

BD – wykład 7 (20)

Najpowszechniej stosowanym drzewiastym indeksem dynamicznym jest B⁺-drzewo. Jest on implementowany we wszystkich komercyjnych i niekomercyjnych SZBD. Ponadto stanowi podstawę implementacji innych indeksów, tj. indeksów bitmapowych i połączeniowych.

Indeks B⁺-drzewo jest zrównoważoną strukturą drzewiastą, w której wierzchołki wewnętrzne służą do wspomagania wyszukiwania, natomiast wierzchołki liści zawierają rekordy indeksu ze wskaźnikami do rekordów w plikach danych. Zrównoważenie struktury oznacza, że odległość (liczba poziomów) od korzenia do dowolnego liścia jest zawsze taka sama.

W celu zapewnienia odpowiedniej efektywności realizacji zapytań przedziałowych wierzchołki liści stanowią listę dwukierunkową.



Indeks B⁺-drzewo - charakterystyka

- Operacje wstawiania i usuwania rekordów indeksu pozostawiają indeks zrównoważony
- Każdy wierzchołek jest wypełniony w co najmniej 50% (za wyjątkiem korzenia)
 - usuwanie rekordów może skutkować mniejszym wypełnieniem niż 50%
- Wyszukanie rekordu wymaga przejścia od korzenia do liścia
 - długość ścieżki od korzenia do dowolnego liścia nazywamy wysokością drzewa indeksu
- Indeks zrównoważony

BD – wykład 7 (21)

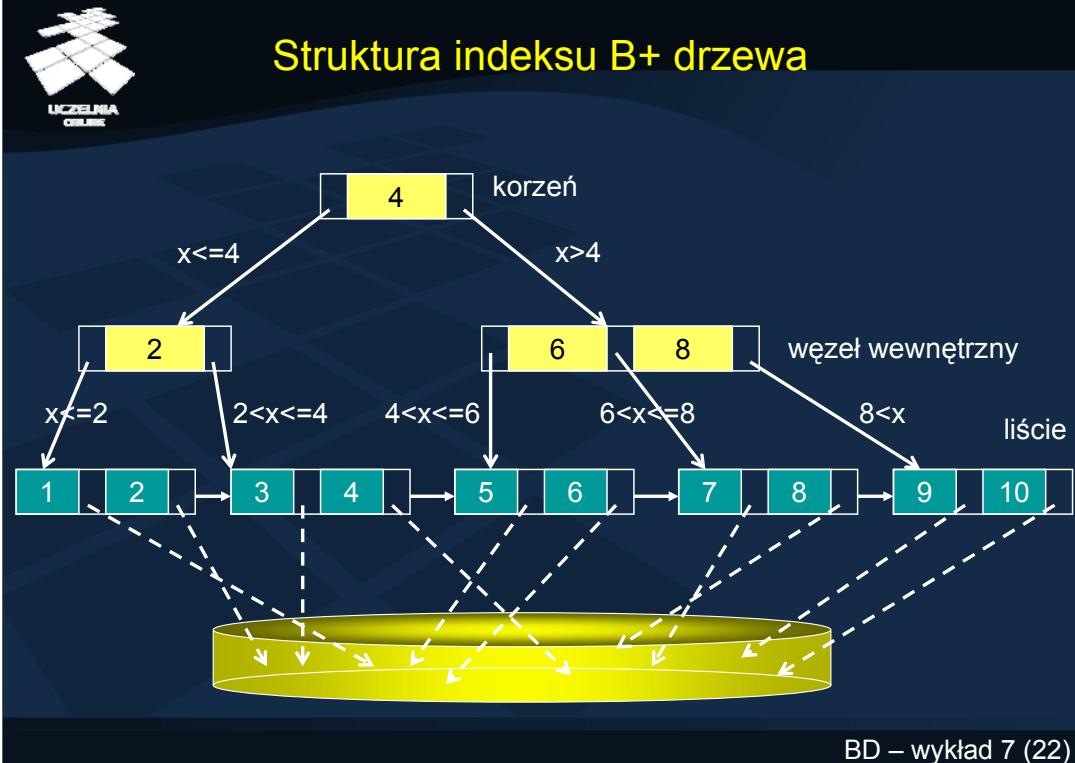
Najważniejsza charakterystyka indeksu B⁺-drzewo jest następująca:

Po pierwsze, operacje wstawiania i usuwania rekordów indeksu pozostawiają indeks zrównoważonym.

Po drugie, każdy wierzchołek jest wypełniony w co najmniej 50% (za wyjątkiem korzenia). Odstępstwo od tej reguły może być spowodowane operacjami usuwania rekordów. Dla operacji usuwania, rekord indeksu jest usuwany z indeksu ale wolne miejsce pozostaje w liściu. W konsekwencji, wierzchołki liści mogą być wypełnione w mniej niż 50%.

Po trzecie, wyszukanie rekordu wymaga przejścia od korzenia do liścia. Długość ścieżki od korzenia do dowolnego liścia nazywamy wysokością drzewa indeksu.

Po czwarte, jak wspomniano, B⁺-drzewo jest indeksem zrównoważonym.



BD – wykład 7 (22)

Przykładowy indeks B⁺-drzewo przedstawiono na slajdzie. Składa się on z trzech poziomów: korzenia, węzłów wewnętrznych i liści. W korzeniu jest przechowywana pewna wartość graniczna klucza indeksu i wskaźniki do węzłów wewnętrznych. W naszym przykładzie wartością graniczną jest 4, a korzeń zawiera wskaźniki do dwóch węzłów wewnętrznych. W węzłach wewnętrznych są przechowywane również pewne wartości graniczne klucza indeksu i wskaźniki do liści. Liście z kolei przechowują pary: <wartość klucza indeksu, wskaźnik do rekordu na dysku>.



Węzeł wewnętrzny (1)

- Struktura węzła wewnętrznego indeksu B^+ -drzewo rzędu p jest następująca
1. Węzeł wewnętrzny ma postać:
 $\langle P_1, K_1, P_2, \dots, P_{Q-1}, K_{Q-1}, P_Q \rangle$
 2. Dla każdego wierzchołka wewnętrznego zachodzi
 $K_1 < K_2 < \dots < K_{Q-1}$



BD – wykład 7 (23)

Struktura węzła wewnętrznego indeksu B^+ -drzewo rzędu p jest następująca.

1. Węzeł wewnętrzny ma następującą postać: wskaźnik do węzła, wartość klucza indeksu, kolejny wskaźnik, kolejna wartość, itd. Liczba wskaźników jest o jeden większa od liczby wartości klucza.

$$\langle P_1, K_1, P_2, \dots, P_{Q-1}, K_{Q-1}, P_Q \rangle$$

gdzie $Q \leq p$; P_i jest wskaźnikiem do poddrzewa; K_i jest wartością klucza indeksu.

Maksymalna liczba wskaźników jaka może zostać zapisana w węźle jest nazywana rzędem B^+ -drzewa i jest oznaczana jako p .

2. Dla każdego wierzchołka wewnętrznego zachodzi:

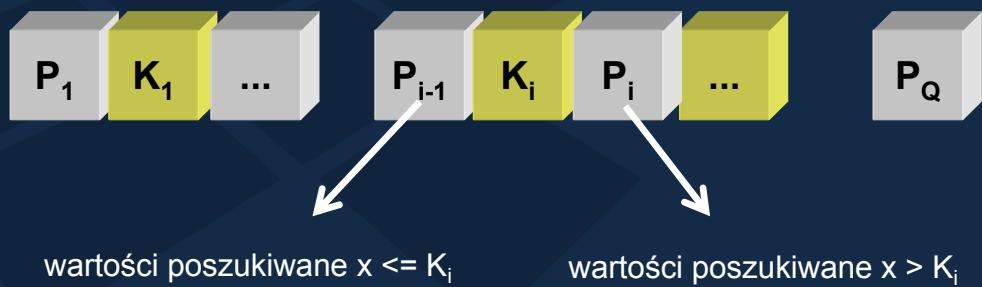
$$K_1 < K_2 < \dots < K_{Q-1}$$

Oznacza to, że wartości klucza indeksowego są uporządkowane (od lewej - wartości najmniejsze do prawej - wartości największe).



Węzeł wewnętrzny (2)

3. Dla danej wartości K_i klucza w węźle zewnętrzny
- lewy wskaźnik prowadzi do poddrzewa zawierającego wartości poszukiwane $\leq K_i$
 - prawy wskaźnik prowadzi do poddrzewa zawierającego wartości poszukiwane $> K_i$



BD – wykład 7 (24)

3. Dla danej wartości K_i klucza w węźle wewnętrzny, lewy wskaźnik prowadzi do poddrzewa zawierającego wartości poszukiwane $\leq K_i$, a prawy wskaźnik prowadzi do poddrzewa zawierającego wartości poszukiwane $> K_i$, jak pokazano na slajdzie.



Węzeł wewnętrzny (3)

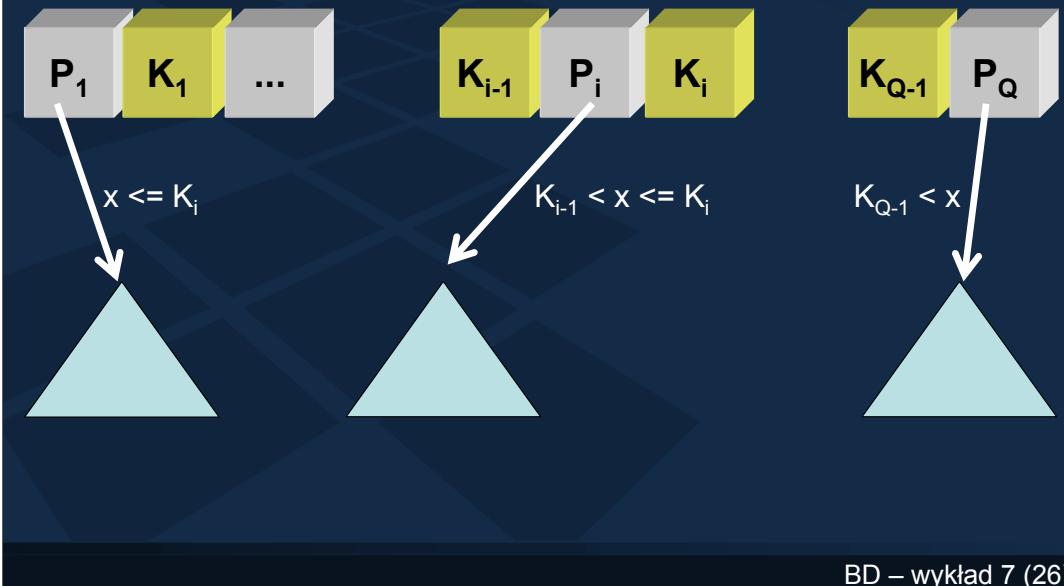
4. Każdy wierzchołek wewnętrzny posiada co najwyżej p wskaźników do poddrzew
5. Każdy wierzchołek wewnętrzny, za wyjątkiem korzenia, posiada co najmniej $\lceil (p/2) \rceil$ wskaźników do poddrzew
 - korzeń posiada co najmniej 2 wskaźniki do poddrzew
6. Każdy wierzchołek wewnętrzny o Q wskaźnikach posiada $Q-1$ wartości kluczy

BD – wykład 7 (25)

4. Każdy wierzchołek wewnętrzny posiada co najwyżej p wskaźników do poddrzew.
5. Dla każdego wierzchołka wewnętrznego liczba wskaźników do poddrzew jest określona jako najmniejsza liczba całkowita większa lub równa połowie rzędu drzewa. Korzeń posiada co najmniej 2 wskaźniki do poddrzew.
6. Każdy wierzchołek wewnętrzny o Q wskaźnikach posiada $Q-1$ wartości kluczy.



Wierzchołek wewnętrzny (4)



BD – wykład 7 (26)

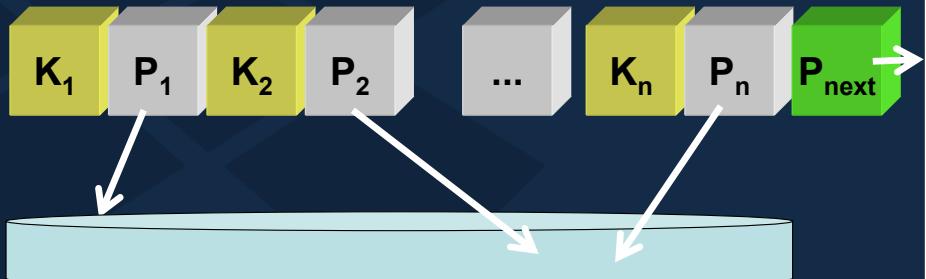
Podsumowanie struktury wierzchołka wewnętrznego przedstawiono na slajdzie.



Liść (1)

Struktura liścia indeksu B⁺-drzewo rzędu p jest następująca:

1. Liść ma postać:
 $\langle K_1, P_1 \rangle, \langle K_2, P_2 \rangle, \dots, \langle K_k, P_k \rangle, P_{\text{next}}$
2. Dla każdego wierzchołka liścia zachodzi:
 $K_1 < K_2 < \dots < K_k$



BD – wykład 7 (27)

Struktura liścia indeksu B⁺-drzewo rzędu p jest następująca:

1. Liść ma postać:

- zbiór par: wartość klucza indeksu, wskaźnik do rekordu (bloku danych) z tą wartością klucza (oznaczone jako K_k, P_k)

$\langle K_1, P_1 \rangle, \langle K_2, P_2 \rangle, \dots, \langle K_k, P_k \rangle, P_{\text{next}}$

K_1, K_2, \dots, K_k są wartościami klucza indeksu;

P_1, P_2, \dots, P_k są wskaźnikami do rekordów na dysku lub do bloków danych;

- wskaźnik do następnego liścia.

W indeksach typu B^{*}-drzewo, każdy liść ma dodatkowo wskaźnik do poprzedniego liścia.

2. Dla każdego wierzchołka liścia zachodzi:

$$K_1 < K_2 < \dots < K_{Q-1}$$

Oznacza to, że wartości klucza indeksowego są uporządkowane (od lewej - wartości najmniejsze do prawej - wartości największe).



Liść (2)

3. Każdy liść posiada co najmniej $\lfloor(p/2)\rfloor$ wartości kluczy
4. Wszystkie liście znajdują się na tym samym poziomie (tej samej wysokości)

BD – wykład 7 (28)

3. Dla każdego liścia minimalna liczba wartości kluczy indeksu jest określona jako największa liczba całkowita mniejsza lub równa połowie rzędu drzewa.
4. Wszystkie liście znajdują się na tym samym poziomie (tej samej wysokości).



Obliczenie rzędu indeksu

- Dane: rozmiar klucza V ; rozmiar wskaźnika do bloku P ; rozmiar bloku B ; liczba rekordów w indeksowanym pliku danych r ; liczba bloków pliku b
- Węzły wewnętrzne zawierają maksymalnie p wskaźników i $p-1$ kluczy
- Każdy z węzłów musi zmieścić się w pojedynczym bloku dyskowym - rząd B^+ -drzewa p jest największą liczbą całkowitą, dla której spełniona jest nierówność:

$$(p*P)+((p-1)*V) \leq B \quad 1$$
- Minimalna wysokość indeksu rzadkiego: $h = \lceil \log_p b \rceil \quad 2$
- Minimalna wysokość indeksu gęstego: $h = \lceil \log_p r \rceil \quad 3$

BD – wykład 7 (29)

Przedstawiony zostanie teraz tok rozumowania prowadzący do obliczenia rzędu indeksu. Przypominamy, że rząd indeksu oznaczamy jako p .

Przyjmijmy, że: V oznacza rozmiar klucza indeksu; P oznacza rozmiar wskaźnika do bloku; B oznacza rozmiar bloku danych (dyskowego); r oznacza liczbę indeksowanych rekordów w pliku danych; b oznacza liczbę bloków pliku danych.

Pamiętamy, że węzły wewnętrzne zawierają maksymalnie p wskaźników i $p-1$ kluczy. Ponieważ każdy węzeł musi zmieścić się w pojedynczym bloku dyskowym, więc rząd B^+ -drzewa jest największą liczbą całkowitą spełniającą nierówność ze slajdu (oznaczoną symbolem 1).

Z innych przydatnych wzorów wymienić należy wzory obliczające minimalną wysokość indeksu rzadkiego (wzór 2) i minimalną wysokość indeksu gęstego (wzór 3), przedstawione na slajdzie.



Obliczenie rzędu indeksu - przykład (1)

Dane:

Rozmiar pliku: $r = 30\ 000$ rekordów

Rozmiar bloku: $B = 1024B$

Rozmiar rekordu: $R = 100$ bajtów

Rozmiar klucza: $V = 9$

Rozmiar wskaźnika: $P = 6$

Rekordy mają stałą długość i nie są dzielone między bloki

Indeks wtórny

$$\text{Liczba rekordów w bloku: } 1 \quad rbl = \left\lfloor \frac{B}{R} \right\rfloor = \left\lfloor \frac{1024}{100} \right\rfloor = 10$$

$$\text{Liczba bloków danych: } 2 \quad b = \left\lceil \frac{r}{rbl} \right\rceil = \left\lceil \frac{30000}{10} \right\rceil = 3000$$

BD – wykład 7 (30)

Rozważmy następujący przykład ilustrujący sposób obliczania rzędu B^+ -drzewa.

Niech: liczba indeksowanych rekordów danych wynosi $r=30\ 000$, rozmiar bloku wynosi $B=1kB$, rozmiar rekordu wynosi $R=100B$, rozmiar klucza indeksu wynosi $V=9B$ i rozmiar wskaźnika wynosi $P=6B$.

Przyjmujemy ponadto, że rekordy mają stałą długość i nie są dzielone między bloki. Na pliku danych jest zakładany indeks wtórny.

Ponieważ rekordy mają stałą długość i nie są dzielone między bloki, liczbę rekordów w bloku obliczymy za pomocą wzoru 1.

Liczbę bloków danych do składowania 30000 rekordów obliczymy za pomocą wzoru 2.



Obliczenie rzędu indeksu - przykład (2)

Rząd węzła

3

$$(p * P) + [(p - 1) * V] \leq B$$

$$p \leq \frac{V + B}{P + V}$$

4

$$p \leq \frac{9 + 1024}{6 + 9}$$

$$p = 68$$

Minimalna wysokość indeksu:

$$5 \quad h = \lceil \log_p r \rceil$$



$$h = \lceil \log_{68} 30000 \rceil = 3$$

BD – wykład 7 (31)

Rząd węzła obliczamy przekształcając wzór 3 ze slajdu do wzoru 4. Po podstawieniu wartości do wzoru 4 otrzymujemy wynik: rząd węzła wewnętrznego $p=68$. Z wykorzystaniem obliczonej wartości rzędu węzła możemy następnie obliczyć minimalną wysokość indeksu niezbędną do poindeksowania 30000 rekordów danych. Wykorzystujemy do tego celu wzór 5. Po podstawieniu danych, otrzymujemy wynik - indeks musi mieć przynajmniej 3 poziomy.



Przykład

- Koszt wyszukania rekordu bez pomocy indeksu
 - średnio: **liczba bloków danych/2** = 1500 dostępów
- Koszt wyszukiwania rekordu za pomocą indeksu typu B⁺-drzewo:
 - równy: **wysokość drzewa + 1** = 3 + 1 = 4 dostępów

BD – wykład 7 (32)

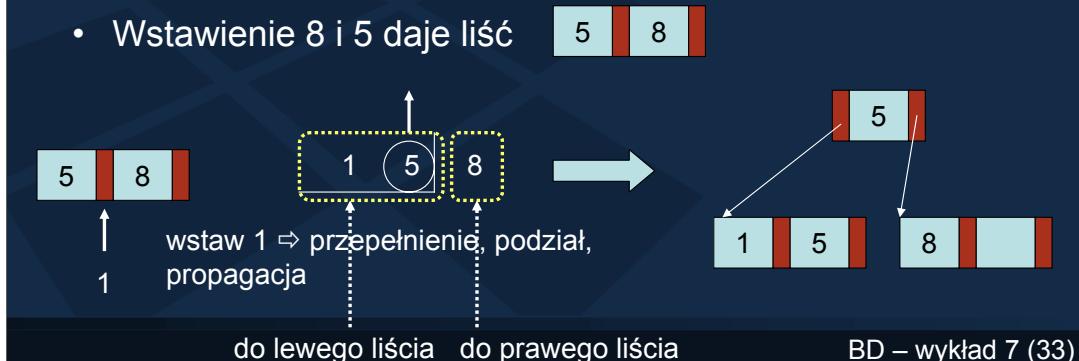
Jak łatwo dowieźć analitycznie, indeks B⁺-drzewo znakomicie przyspiesza wyszukiwanie rekordu z pliku, w porównaniu z dostępem do pliku bez indeksu. Jeżeli w pliku jest wyszukiwany rekord z kryterium nałożonym na atrybut nieporządkujący, wówczas należy przeszukać cały plik. Średnio będzie to wymagało odczytania połowy bloków przechowujących rekordy danych, czyli w naszym przykładzie 1500.

W przypadku indeksu B⁺-drzewo należy znaleźć odpowiedni liść i sięgnąć po rekord do dysku. Znalezienie liścia wymaga odczytania 3 bloków indeksu - korzenia, węzła wewnętrznego i liścia. Korzystając ze wskaźnika w liściu, należy odczytać jeden blok danych zawierający poszukiwany rekord. Znalezienie rekordu będzie więc wymagało 4 dostępów do dysku.



Wstawianie danych do indeksu - przykład (1)

- Założenia
 - indeks B⁺-drzewo
 - rząd drzewa: 3
 - sekwencja danych wstawianych do indeksu:
 - 8, 5, 1, 7, 3, 12, 9, 6
- Wstawienie 8 i 5 daje liść



Zarządzanie strukturą indeksu B⁺-drzewo w sytuacjach wstawiania, usuwania i modyfikowania wartości atrybutu indeksowego rekordów danych jest realizowane za pomocą specjalizowanych algorytmów. Na kilku kolejnych slajdach przedstawimy idee działania algorytmu modyfikowania struktury indeksu na skutek wstawiania rekordów danych.

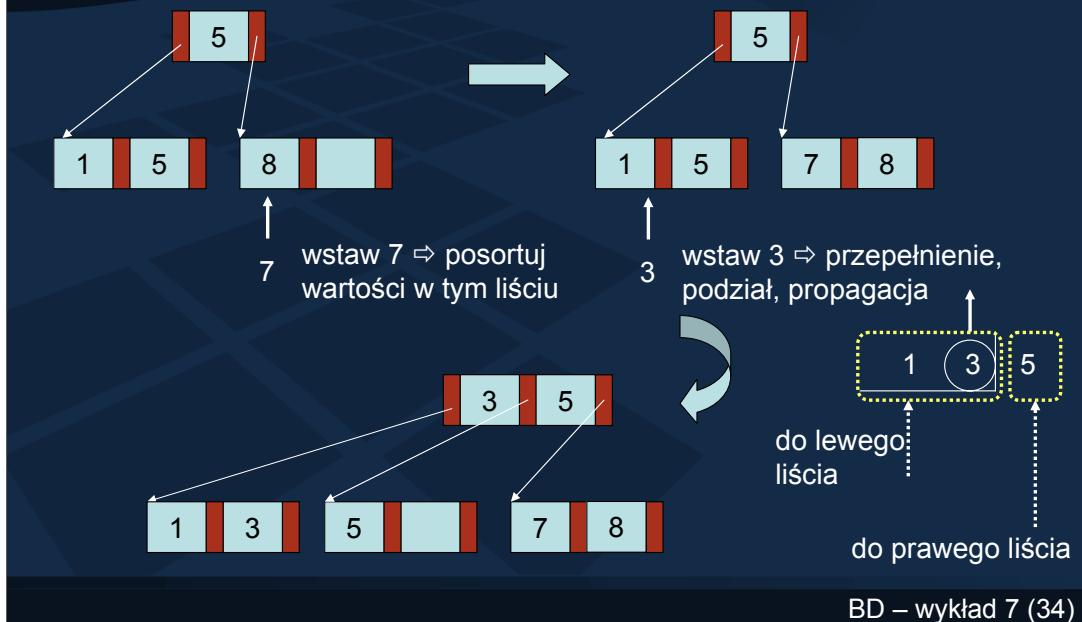
Dla uproszczenia przyjmijmy, że indeks jest rzędu 3. Oznacza to, że każdy węzeł posiada minimalnie 2 i maksymalnie 3 wskaźniki. Liść posiada od 1 do dwóch wartości atrybutu indeksowego. Do indeksu są wstawiane następujące wartości w kolejności ich wymienienia: 8, 5, 1, 7, 3, 12, 9, 6.

Wstawione wartości 8 i 5 mieścią się w jednym bloku indeksu, więc wystarczy je przechować jako liść.

Wstawienie wartości 1 do węzła spowodowałoby jego przepelnienie. Z tego powodu węzeł jest rozbijany na 2. W tym celu porządkujemy wartości istniejące w węźle i wartość wstawianą, od lewej (najmniejsza) do prawej (największa). Wartość środkową, czyli 5, przenosimy do poziomu wyższego - staje się ona wartością w korzeniu. Wartości 1 i 5 trafiają do lewego liścia, a 8 - do prawego, jak pokazano na slajdzie.

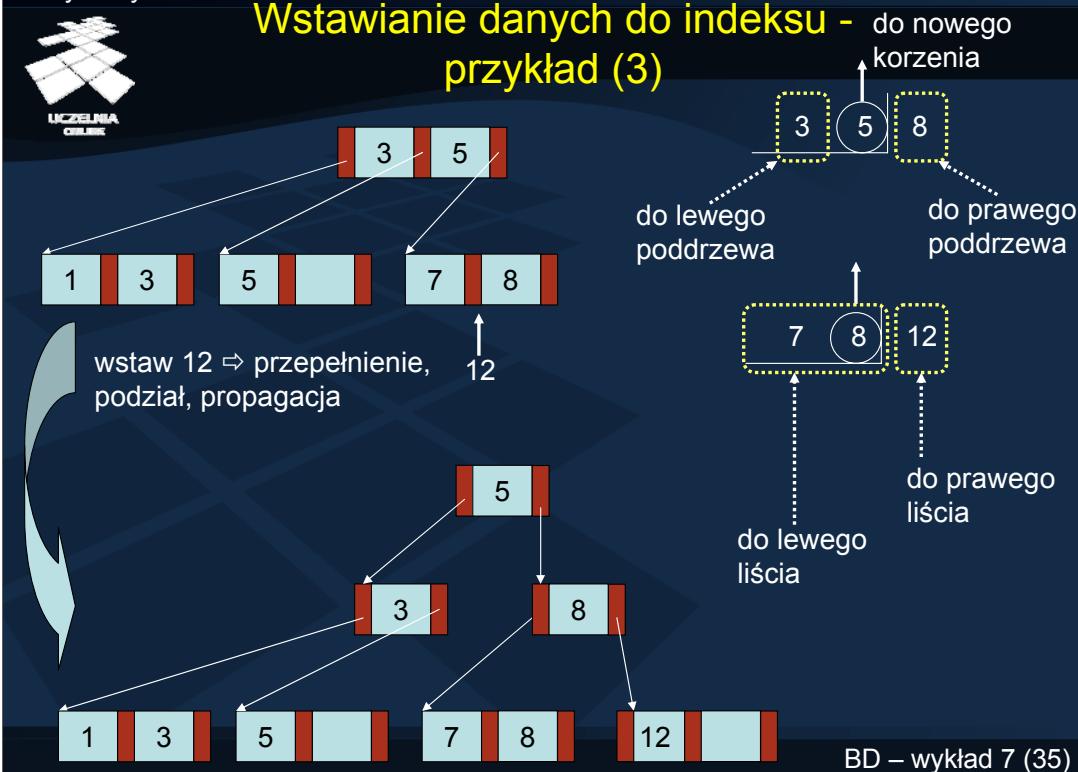


Wstawianie danych do indeksu - przykład (2)



Wartość 7 musi być wstawiona do prawego liścia, który posiada miejsce na jedną wartość. Wartości w tym liściu należy posortować.

Wartość 3 musi być wstawiona do lewego liścia. Jednak jest on już w całości zajęty. Z tego powodu węzeł jest rozbijany na 2. W tym celu porządkujemy wartości istniejące w węźle i wartość wstawianą od lewej (najmniejsza) do prawej (największa). Wartość środkową, czyli 3, przenosimy do korzenia. Wartości 1 i 3 trafiają do lewego liścia, a 5 - do prawego, jak pokazano na slajdzie.

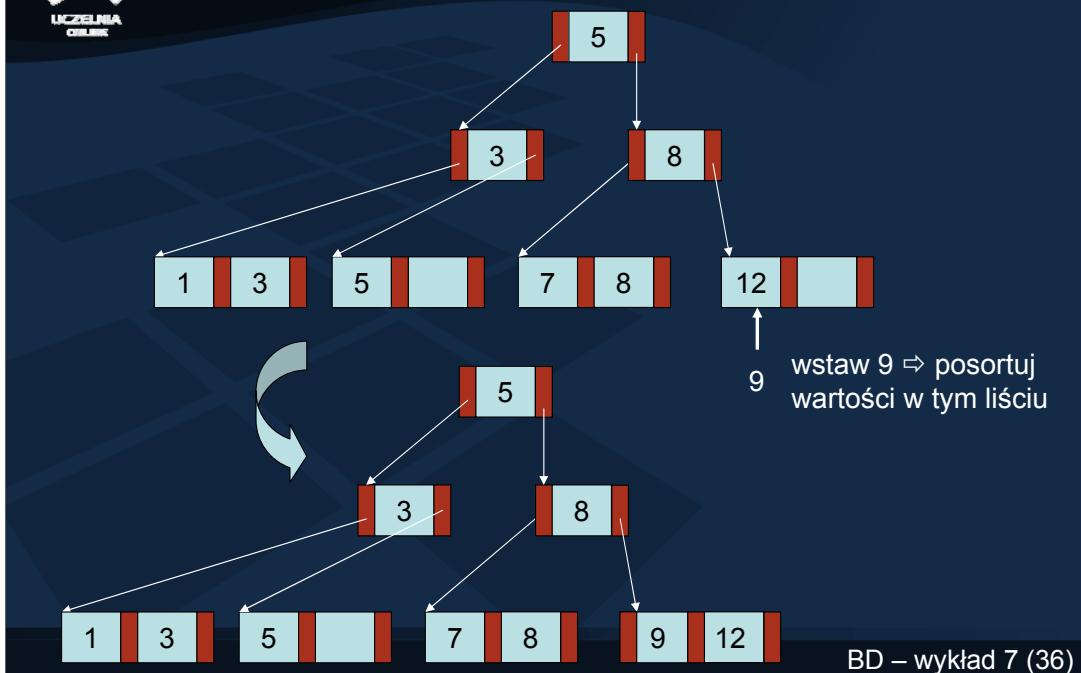


Wartość 12 musi być wstawiona do skrajnie prawego liścia. Jednak jest on już w całości zajęty. Z tego powodu węzeł jest rozbijany na 2. W tym celu porządkujemy wartości istniejące w węźle i wartość wstawianą, od lewej (najmniejsza) do prawej (największa). Wartość środkową, czyli 8, przenosimy do korzenia. Wartości 7 i 8 trafiają do lewego liścia, a 12 - do prawego, jak pokazano na slajdzie.

Ponadto, w tym przypadku wartość 8 powinna trafić do korzenia, który jest już w całości wypełniony. Z tego względu ulega on rozbiciu na nowy korzeń i dwa węzły wewnętrzne. W tym celu porządkujemy wartości już przechowywane w korzeniu i wstawianą do niego wartość, jak pokazano na slajdzie. Wartość środkowa, czyli 5 trafia do nowego korzenia. Wartość 3 trafia do lewego poddrzewa, a wartość 8 - do prawego, jak pokazano na slajdzie.



Wstawianie danych do indeksu - przykład (4)



Wartość 9 musi być wstawiona do skrajnie prawego liścia, który posiada miejsce na jedną wartość. Wartości w tym liściu należy posortować.



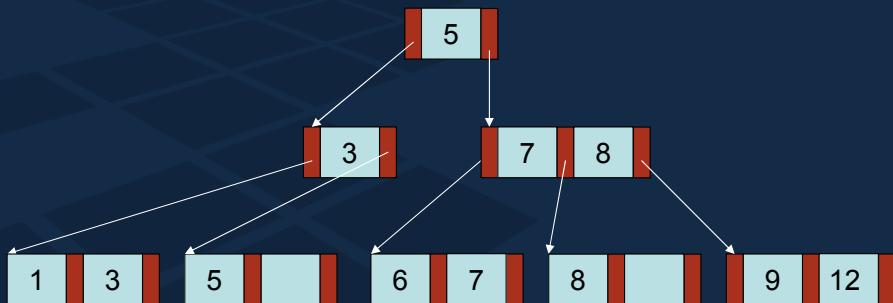
Wstawianie danych do indeksu - przykład (5)



Wartość 6 musi być wstawiona do trzeciego od lewej liścia. Jednak jest on już w całości zajęty. Z tego powodu węzeł jest rozbijany na 2. W tym celu porządkujemy wartości istniejące w węźle i wartość wstawianą, od lewej (najmniejsza) do prawej (największa). Wartość środkową, czyli 7, przenosimy do węzła wewnętrznego. Wartości 6 i 7 trafiają do lewego liścia, a 8 - do prawego, jak pokazano na slajdzie.



Wstawianie danych do indeksu - przykład (6)

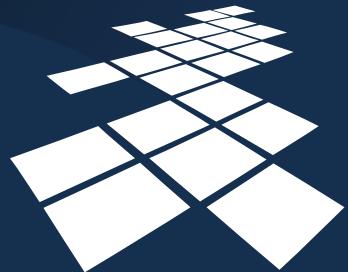


BD – wykład 7 (38)

Ostateczna struktura indeksu została przedstawiona na slajdzie.

Przetwarzanie transakcyjne

Wykład przygotował:
Tadeusz Morzy



UCZELNIA
ONLINE

BD – wykład 8

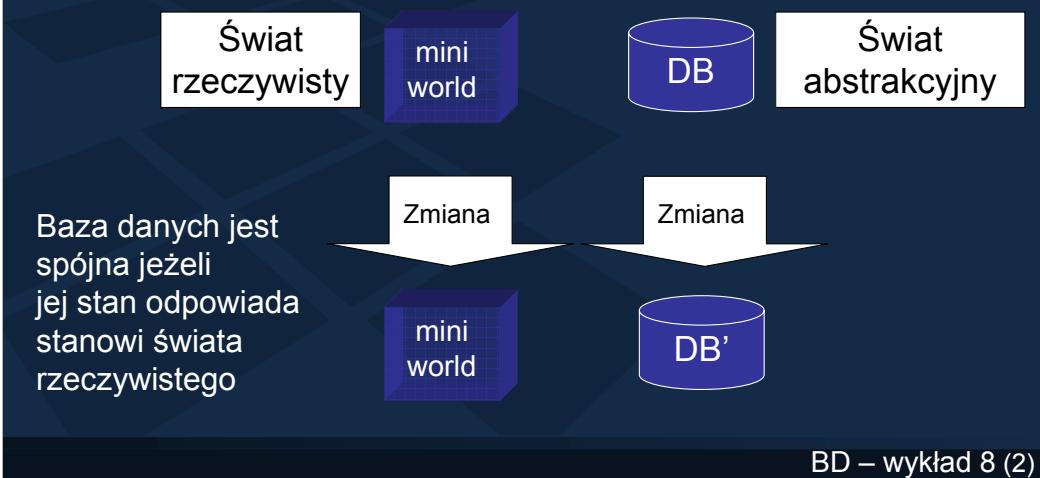
Celem niniejszego wykładu jest omówienie problematyki związanej z transakcjami w bazie danych. W szczególności zostaną omówione:

- transakcja i jej własności,
- formalny model transakcji,
- sekwencyjne i współbieżne realizacje zbioru transakcji,
- uszeregowalność transakcji.



Wprowadzenie (1)

- Baza danych – jest abstrakcyjnym odzwierciedleniem wybranego fragmentu rzeczywistości (ang. *miniworld*)



BD – wykład 8 (2)

Jak wspomnialiśmy w jednym z pierwszych wykładów, baza danych jest abstrakcyjnym odzwierciedleniem wybranego fragmentu rzeczywistości. Fragment ten powinien być wiernie odzwierciedlany w bazie danych. Mówimy, że baza danych jest spójna jeżeli jej stan odpowiada stanowi świata rzeczywistego.



Wprowadzenie (2)

- Zmiany zachodzące w świecie rzeczywistym muszą być zakodowane w postaci programu, który będzie transformował bazę danych z jednego stanu spójnego do innego stanu spójnego
- Niebezpieczeństwa związane z realizacją programu transformującego bazę danych
 - Awaryjność środowiska sprzętowo-programowego
 - Współbieżny dostęp do danych
 - Rozproszenie baz danych

BD – wykład 8 (3)

W celu zapewnienia tej spójności, zmiany zachodzące w świecie rzeczywistym muszą być zakodowane w postaci programu, który będzie transformował bazę danych z jednego stanu spójnego do innego stanu spójnego. Wykonanie tego programu powinno być odporne na wszelkiego rodzaju awarie sprzętowo-programowe. Ponadto, baza danych jest przeznaczona do użytkowania przez wielu równocześnie pracujących użytkowników. Taka równoległa praca może również wpływać na poprawność danych w bazie danych. W przypadku rozproszenia bazy danych na wiele węzłów sieci, należy zapewnić poprawność danych we wszystkich węzłach.



Problemy przygotowania aplikacji

- **Przykład:** Napisać aplikację przelewu kwoty N z konta A na konto B

- **Problem 1 – awaria systemu**

Po pobraniu kwoty N z konta A, i zapisaniu tej aktualizacji do bazy danych, wystąpiła awaria systemu. W wyniku awarii systemu wykonana została jedynie część operacji składających się na daną aplikację

- **Problem 2 – współbieżny dostęp do danych**

Operacje współbieżnie wykonywanych transakcji mogą naruszać spójność bazy danych, lub generować niepoprawne wyniki

BD – wykład 8 (4)

Jako przykład rozważmy system bankowy i aplikację przelewającą kwotę N z konta A na konto B. Założymy, że w czasie realizowania tej operacji, po pobraniu kwoty N z konta A, i zapisaniu tej aktualizacji do bazy danych, wystąpiła awaria systemu. W wyniku tej awarii wykonana została jedynie pierwsza część operacji przelewu, tj. kwota N została zdjęta z konta A, ale nie zdążyła ona wpłynąć na konto B.

Jeżeli w systemie bankowym będzie równocześnie działać wiele aplikacji przelewu (co jest typowe w rzeczywistości), wówczas ich równoczesna praca może powodować powstawanie danych niespójnych, czyli nieprawdziwych - mogą się pojawiać stany kont w rzeczywistości niewystępujące.



Transakcja (1)

- **Problem 3 - utrata danych w wyniku awarii**

Wyniki zakończonych aplikacji, buforowane w pamięci operacyjnej, mogą zostać utracone w wyniku awarii systemu

- Rozwiązaniem problemu awaryjności, rozproszenia i wielodostępności środowiska systemu bazy danych – koncepcja **transakcji**

Transakcja jest sekwencją logicznie powiązanych operacji na bazie danych, która przeprowadza bazę danych z jednego stanu spójnego w inny stan spójny.

Typy operacji na bazie danych obejmują: odczyt i zapis danych oraz zakończenie i akceptację (zatwierdzenie), lub wycofanie transakcji

BD – wykład 8 (5)

Kolejnym problemem jest niebezpieczeństwo utraty danych w wyniku awarii systemu. Jeżeli dane zmodyfikowane i wprowadzone przez zakończone aplikacje są buforowane w pamięci operacyjnej, to oznacza, że są one ulotne. Jakkolwiek awaria systemu spowoduje utratę tych danych.

Rozwiązaniem omówionych problemów jest wprowadzenie mechanizmu tzw. transakcji. Transakcja jest sekwencją logicznie powiązanych operacji na bazie danych, która przeprowadza bazę danych z jednego stanu spójnego w inny stan spójny. Typy operacji na bazie danych obejmują: odczyt i zapis danych oraz zakończenie i akceptację (zatwierdzenie), lub wycofanie transakcji.



Transakcja (2)

Transakcja przelewu kwoty N z konta A na konto B:

```
begin
    // odejmij kwotę N z konta A;
    update konta
        SET stan = stan - N
        where id_konta = A;
    // dodaj do konta B kwotę N;
    update konta
        SET stan = stan + N
        where id_konta = B;
commit;
```

BD – wykład 8 (6)

Jako przykład, rozważmy transakcję przelewu kwoty N z konta A na konto B. Transakcja ta składa się z następujących operacji:

1. rozpoczęcie transakcji - begin,
2. pomniejszenie stanu konta A o kwotę N,
3. powiększenie stanu konta B o kwotę N,
4. zatwierdzenie transakcji - commit.



Właściwości transakcji (1)

A(atomicity)C(onsistency)I(isolation)D(urability)

- **Atomowość (A)**

Zbiór operacji wchodzących w skład transakcji jest niepodzielny: albo zostaną wykonane wszystkie operacje transakcji albo żadna. Dotyczy to również wszystkich operacji transakcji wykonywanych na obiektach rzeczywistych (tak zwane akcje rzeczywiste) – np. wypłata gotówki z bankomatu

- **Spójność (C)**

Transakcja przeprowadza bazę danych z jednego stanu spójnego do innego stanu spójnego. W trakcie wykonywania transakcji baza danych może być przejściowo niespójna. Transakcja nie może naruszać ograniczeń integralnościowych

BD – wykład 8 (7)

Każda transakcja posiada cztery cechy, tj. atomowość (ang. **Atomicity**), spójność (ang. **Consistency**), izolacja (ang. **Isolation**) i trwałość (ang. **Durability**). Cechy te są najczęściej oznaczane jako ACID, od angielskich nazw.

Atomowość oznacza, że zbiór operacji wchodzących w skład transakcji jest niepodzielny, to znaczy albo zostaną wykonane wszystkie operacje transakcji albo żadna. Dotyczy to również wszystkich operacji transakcji wykonywanych na obiektach rzeczywistych (tak zwane akcje rzeczywiste) – np. wypłata gotówki z bankomatu.

Spójność oznacza, że transakcja przeprowadza bazę danych z jednego stanu spójnego do innego stanu spójnego. W trakcie wykonywania transakcji baza danych może być przejściowo niespójna. Transakcja nie może naruszać ograniczeń integralnościowych.



Właściwości transakcji (2)

- **Izolacja (I)**

Transakcje są od siebie logicznie odseparowane. Transakcje oddziałują na siebie poprzez dane. Mimo współbieżnego wykonywania, transakcje widzą stan bazy danych tak, jak gdyby były wykonywane w sposób sekwencyjny

- **Trwałość (D)**

Wyniki zatwierdzonych transakcji nie mogą zostać utracone w wyniku wystąpienia awarii systemu. Zatwierdzone dane w bazie danych, w przypadku awarii, muszą być odtwarzalne

BD – wykład 8 (8)

Izolacja oznacza, że transakcje są od siebie logicznie odseparowane. Transakcje oddziałują na siebie poprzez dane. Mimo współbieżnego wykonywania, transakcje widzą stan bazy danych tak, jak gdyby były wykonywane w sposób sekwencyjny.

Trwałość oznacza, że wyniki zatwierdzonych transakcji nie mogą zostać utracone w wyniku wystąpienia awarii systemu. Zatwierdzone dane w bazie danych, w przypadku awarii, muszą być odtwarzalne.



Transakcja (3)

- Transakcja jest:
- **Atomowa**: jeżeli pieniądze zostaną poprawnie przetransferowane z konta **A** do **B**
- **Spójna**: jeżeli kwota odjęta z konta **A** jest równa kwocie dodanej do konta **B**
- **Izolowana**: jeżeli inne transakcje wykonywane współbieżnie, czytające i modyfikujące konta A i B, nie mają wpływu na transakcję
- **Trwała**: jeżeli po zakończeniu transakcji, baza danych trwale odzwierciedla nowe stany kont **A** i **B**

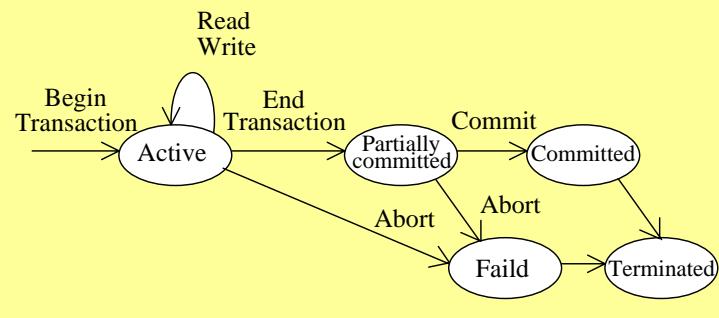
BD – wykład 8 (9)

Transakcja przelewu z naszego przykładu jest:

- atomowa jeżeli pieniądze zostaną poprawnie przetransferowane z konta A do B;
- spójna jeżeli kwota odjęta z konta A jest równa kwocie dodanej do konta B;
- izolowana jeżeli inne transakcje wykonywane współbieżnie, czytające i modyfikujące konta A i B, nie mają wpływu na tę transakcję;
- trwała jeżeli po zakończeniu transakcji, baza danych trwale odzwierciedla nowe stany kont A i B.



Diagram stanów transakcji



Begin_transaction: początek transakcji.

Read, Write: operacje odczytu i zapisu danych w bazie danych.

End_transaction: koniec transakcji:

Commit: zatwierdzenie (akceptacja) wyników transakcji.

Rollback: wycofanie wyników transakcji

BD – wykład 8 (10)

Każda realizowana transakcja posiada zbiór ścisłe określonych stanów i zbiór ścisłe określonych przejść z jednego stanu do drugiego. Stany te są następujące:

- Active: transakcja jest aktywna, jest w czasie realizowania swoich operacji;
- Partially committed: transakcja jest częściowo zatwierdzona;
- Committed: transakcja została zatwierdzona;
- Failed: transakcja została wycofana;
- Terminated: transakcja zakończyła się zatwierdzeniem lub wycofaniem.

Przejścia z jednego stanu do drugiego są opisane tzw. diagramem stanów transakcji przedstawionym na slajdzie. Rozpoczęcie transakcji (Begin Transaction) uruchamia transakcję, która jest aktywna. Każda operacja zapisu lub odczytu danych w ramach tej transakcji dokonuje się w stanie aktywnym transakcji. Kończenie transakcji z jej wycofaniem (Abort) przeprowadza transakcję ze stanu Active do stanu Failed, a następnie Terminate. Kończenie transakcji z jej zatwierdzeniem przeprowadza ją ze stanu Active do Partially committed - transakcja jest gotowa do zatwierdzenia. Z tego stanu można jeszcze transakcję wycofać, np. w sytuacji awarii systemu. Ostateczne zatwierdzenie transakcji przeprowadza ją do stanu Committed, a następnie do Terminated, co kończy działanie transakcji.



Zakończenie transakcji

- **End_transaction:** koniec transakcji oznacza, że wszystkie operacje odczytu i/lub zapisu transakcji zostały wykonane. W tym momencie, zachodzi konieczność podjęcia decyzji, czy zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych (zatwierdzenie transakcji) czy też mają być wycofane z bazy danych
- **Commit:** zatwierdzenie (akceptacja transakcji) oznacza pomyślne zakończenie transakcji - zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych
- **Rollback:** wycofanie transakcji oznacza niepoprawne zakończenie transakcji i konieczność wycofania z bazy danych wszystkich ewentualnych zmian wprowadzonych przez transakcję

BD – wykład 8 (11)

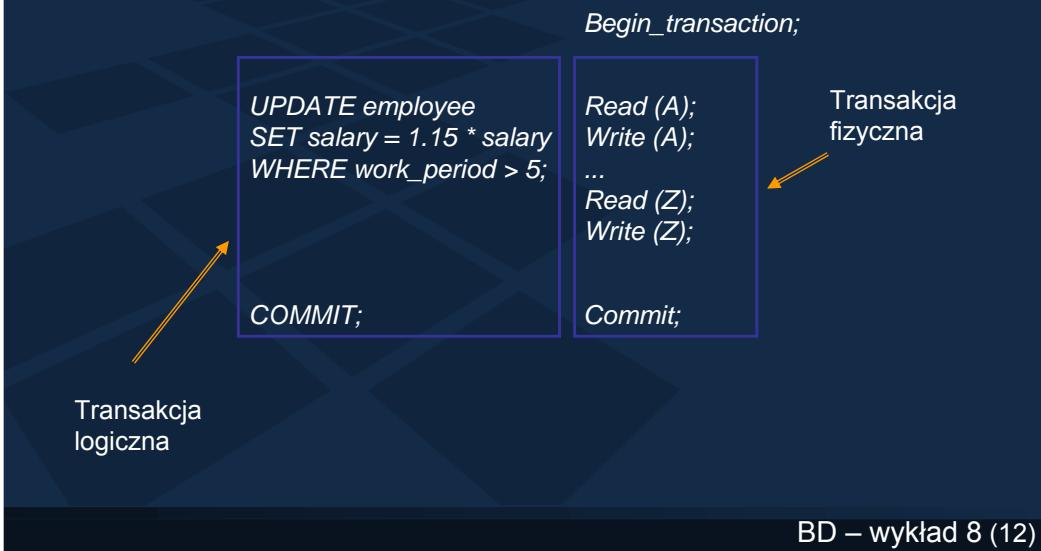
End_transaction oznacza, że wszystkie operacje odczytu i/lub zapisu transakcji zostały wykonane. W tym momencie, zachodzi konieczność podjęcia decyzji, czy zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych (zatwierdzenie transakcji) czy też mają być wycofane z bazy danych.

Commit oznacza zatwierdzenie (akceptację transakcji), czyli pomyślne zakończenie transakcji - zmiany wprowadzone przez transakcję mają być wprowadzone do bazy danych.

Rollback oznacza wycofanie transakcji, czyli niepoprawne zakończenie transakcji i konieczność wycofania z bazy danych wszystkich ewentualnych zmian wprowadzonych przez transakcję.



Transakcja logiczna a transakcja fizyczna



Z punktu widzenia użytkownika, transakcja jest zbiorem poleceń języka SQL, tj. select, insert, update, delete, commit, rollback. Mówimy tu o tzw. transakcji logicznej. Na poziomie systemu zarządzania bazą danych mówimy o tzw. transakcji fizycznej, która jest zarządzana przez odpowiedni moduł SZBD. Transakcja fizyczna składa się z elementarnych operacji rozpoczęcia transakcji, operacji zaalokowania zasobów systemowych dla transakcji, blokowania danych (przy pewnych rozwiązaniach synchronizacji transakcji), operacji na samych danych, końca transakcji i zwalniania zasobów systemowych.



Model transakcji (1)

- **Transakcją** T_i nazywamy uporządkowaną parę:

gdzie:

$$T_j = (\bar{T}_i < T_j)$$

$\bar{T}_i = \{ o_j : 1 \leq j \leq n \}$, oznacza zbiór operacji na bazie danych: { **R** - odczyt, **W** - zapis, **C** – zatwierdzenie transakcji, **A** - wycofanie}
 $\langle T_j \rangle$ jest relacją częściowego porządku na zbiorze \bar{T}_i

Przyjmiemy następującą notację:

- $r_i(x)$ lub $r_i(x, \text{wartość})$
- $w_i(x)$ lub $w_i(x, \text{wartość})$
- c_i lub a_i

BD – wykład 8 (13)

Formalny model transakcji przedstawiono na slajdzie. Transakcją T_i nazywamy uporządkowaną parę: <zbiór operacji na bazie danych, relacja częściowego porządku na zbiorze tych operacji>. Zbiór operacji zawiera: odczyt (R), zapis (W), zatwierdzenie transakcji (C), wycofanie transakcji (A).

W dalszej części wykładu będziemy stosowali następującą notację:

- $r_i(x)$ oznacza odczyt danej x przez i -tą transakcję;
- $r_i(x, \text{wartość})$ oznacza odczyt danej x przez i -tą transakcję, przy czym 'wartość' jest aktualnie odczytaną wartością danej x ;
- $w_i(x)$ oznacza zapis danej x przez i -tą transakcję;
- $w_i(x, \text{wartość})$ oznacza zapis danej x przez i -tą transakcję, przy czym 'wartość' jest aktualnie zapisaną wartością danej x ;
- c_i oznacza zatwierdzenie i -tej transakcji;
- a_i oznacza wycofanie i -tej transakcji.



Model transakcji (2)

- Każda transakcja może być reprezentowana przez graf skierowany:
 $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, gdzie:
 - \mathbf{V} jest zbiorem węzłów odpowiadających operacjom transakcji T_i ,
 - \mathbf{A} jest zbiorem krawędzi reprezentujących porządek na zbiorze operacji
- Przykład:

a) $r_I(x) \longrightarrow w_I(x) \longrightarrow r_2(y) \longrightarrow w_2(y) \longrightarrow c_I$

b) $r_I(x) \longrightarrow w_I(x) \longrightarrow w_2(y) \longrightarrow c_I$

$r_2(y)$

BD – wykład 8 (14)

Każda transakcja może być reprezentowana przez graf skierowany: $G = (V, A)$, gdzie:

- V jest zbiorem węzłów odpowiadających operacjom transakcji T_i ;
- A jest zbiorem krawędzi reprezentujących porządek na zbiorze operacji.

Dwa przykłady grafu transakcji przedstawiono na slajdzie. W pierwszym z nich, pierwsza operacja transakcji pierwszej odczytuje daną x ($r_1(x)$), następnie zapisuje/modyfikuje tę daną ($w_1(x)$). Pierwszą operacją drugiej transakcji jest operacja odczytu danej y ($r_2(y)$), następną operacją jest zapis danej y ($w_2(y)$) przez transakcję drugą. Ostatnią jest operacja zatwierdzenia transakcji pierwszej (c_1). Pierwszy przykład reprezentuje sekwencyjnie wykonywane transakcje. Drugi przykład reprezentuje współbieżnie wykonywane transakcje.



Klasyfikacja transakcji

- **Ze względu na porządek operacji:**

- transakcja sekwencyjna

- transakcja współbieżna

- **Ze względu na zależność operacji:**

- transakcja zależna od danych

- transakcja niezależna od danych

- **Ze względu na typy operacji:**

- zapytania lub transakcja odczytu (read only)

- transakcja aktualizująca - transakcja (read/write)

BD – wykład 8 (15)

Transakcje można podzielić ze względu na wiele kryteriów. Na potrzeby wykładu wprowadzimy trzy kryteria podziału:

- porządek operacji,
- zależność operacji,
- typ operacji.

Zgodnie z pierwszym kryterium wyróżnia się transakcje realizowane sekwencyjnie (tj. jedna po drugiej) i transakcje realizowane współbieżnie (tj. równocześnie). Zgodnie z drugim kryterium wyróżnia się transakcje zależne od danych i transakcje niezależne od danych. W transakcji zależnej do danych, zbiór danych adresowanych przez transakcję może nie być w całości znany w momencie rozpoczęcia transakcji. Zbiór ten jest określany dynamicznie w trakcie pracy transakcji, zależnie od danych przetworzonych przez wcześniejsze polecenia. Ze względu na trzecie kryterium wyróżnia się transakcje wyłącznie odczytujące dane i transakcje modyfikujące dane.



Realizacje transakcji (1)

- Częściowo uporządkowaną sekwencją operacji należących do zbioru wspólnie wykonywanych transakcji nazywamy realizacją (historią). Realizacja modeluje, formalnie, wspólnie wykonanie zbioru transakcji
- Formalnie, **realizacją S** zbioru n transakcji T_1, T_2, \dots, T_n nazywamy takie uporządkowanie operacji wspólnie wykonywanych transakcji, w którym, dla każdej transakcji T_i w realizacji S , porządek wykonania operacji transakcji T_i jest taki sam jak porządek $\langle T_i$

BD – wykład 8 (16)

W praktyce, w jednym systemie bazy danych działa równocześnie wiele transakcji. Należy zapewnić, aby transakcje te były wykonane w takiej kolejności, która nie wprowadzi danych niepoprawnych.

Częściowo uporządkowaną sekwencją operacji należących do zbioru wspólnie wykonywanych transakcji nazywamy realizacją (historią). Realizacja modeluje, formalnie, wspólnie wykonanie zbioru transakcji.

Formalnie, realizacją S zbioru n transakcji T_1, T_2, \dots, T_n nazywamy takie uporządkowanie operacji wspólnie wykonywanych transakcji, w którym, dla każdej transakcji T_i w realizacji S , porządek wykonania operacji transakcji T_i jest taki sam jak częściowy porządek w zbiorze operacji transakcji T_i .



Realizacje transakcji (2)

$$S(\tau) = (\bar{T}_r(\tau), < r)$$

- gdzie:

1. $\bar{T}_r(\tau)$ zbiór operacji wszystkich transakcji należących do zbioru τ
2. $< r$ relacja częściowego porządku na zbiorze $\bar{T}_r(\tau)$,
3. Dla dowolnej pary operacji $o_i, o_j \in \bar{T}_r(\tau)$, takich, że żądają one dostępu do tej samej danej i co najmniej jedna z nich jest operacją zapisu, zachodzi $o_i <_r o_j$ lub $o_j <_r o_i$

BD – wykład 8 (17)

Formalną notację realizacji S zbioru transakcji (oznaczonego jako TAU) przedstawiono na slajdzie. Jest to para: zbiór operacji wszystkich transakcji należących do zbioru TAU i relacja częściowego porządku na zbiorze operacji należących do transakcji ze zbioru TAU. Dla dowolnej pary operacji o_i, o_j należących do zbioru operacji transakcji ze zbioru τ takich, że żądają one dostępu do tej samej danej i co najmniej jedna z nich jest operacją zapisu, zachodzi $o_i <_r o_j$ lub $o_j <_r o_i$.



Realizacje transakcji (3)

- Realizacja zawierająca tylko operacje zatwierdzonych transakcji nazywana jest **zaakceptowaną projekcją**

(Dalsze rozważania dotyczyć będą tyko realizacji spełniających powyższy warunek)

Przykład:

$r: w_0(x), w_0(y), c_0, r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), c_2, w_1(y), c_1, r_f(x), r_f(y), c_f;$

BD – wykład 8 (18)

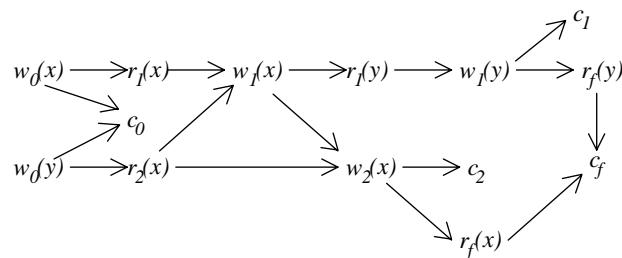
Realizacja zawierająca tylko operacje zatwierdzonych transakcji nazywana jest zaakceptowaną projekcją. Dalsze rozważania dotyczyć będą tyko realizacji spełniających ten warunek.

Jako przykład zaakceptowanej projekcji rozważmy realizację przedstawioną na slajdzie. Transakcja T_0 zapisuje daną x ($w_0(x)$), następnie daną y ($w_0(y)$) i zatwierdza te operacje (c_0). Następnie transakcje T_1 i T_2 są realizowane współbieżnie. T_1 odczytuje daną x ($r_1(x)$), następnie T_2 ($r_2(x)$) odczytuje tę samą daną x . Kolejne operacje to: zapis danej x przez T_1 ($w_1(x)$), odczyt danej y przez T_1 ($r_1(y)$), zapis danej x przez T_2 ($w_2(x)$), zatwierdzenie T_2 (c_2), zapis danej y przez T_1 ($w_1(y)$), zatwierdzenie T_1 , odczyt danej x przez T_f , odczyt danej y przez T_f i zatwierdzenie T_f .



Realizacje transakcji (4)

- Dowolną realizację można przedstawić w postaci grafu skierowanego, nazywanego grafem realizacji, $GR(s(\tau)) = (V, A)$. Węzły grafu odpowiadają operacjom ze zbioru $\bar{T}_r(\tau)$, natomiast krawędzie grafu reprezentują relację częściowego porządku $< r$
- Przykład:**



BD – wykład 8 (19)

Dowolną realizację można przedstawić w postaci grafu skierowanego, nazywanego grafem realizacji. Węzły grafu odpowiadają operacjom ze zbioru operacji należących do transakcji natomiast krawędzie grafu reprezentują relację częściowego porządku na zbiorze tych operacji.

Przykład grafu realizacji dla transakcji T_1, T_2, T_f przedstawiono na slajdzie.



Realizacje sekwencyjne i współbieżne

- Mówimy, że dana realizacja jest **sekwencyjna** jeżeli, dla każdych dwóch transakcji, wszystkie operacje jednej z nich poprzedzają wszystkie operacje drugiej
- W przeciwnym wypadku realizacja jest **współbieżna**

BD – wykład 8 (20)

Mówimy, że dana realizacja jest sekwencyjna jeżeli, dla każdych dwóch transakcji, wszystkie operacje jednej z nich poprzedzają wszystkie operacje drugiej. W przeciwnym wypadku realizacja jest współbieżna.



Stan i obraz bazy danych

- **Stan bazy danych**

zbiór wartości wszystkich danych w bazie danych

- **Obraz bazy danych** widziany przez transakcję T_i

zbiór wartości danych odczytywanych
przez transakcję T_i

BD – wykład 8 (21)

W kontekście zarządzania transakcjami należy wprowadzić pojęcie stanu bazy danych i obrazu bazy danych.

Stan bazy danych reprezentuje zbiór wartości wszystkich danych w bazie. Obraz bazy danych widziany przez transakcję T_i jest zbiorem wartości danych odczytywanych przez transakcję T_i .



Uszeregowalność realizacji (1)

- **Założenie 1:**

Każda realizacja sekwencyjna jest poprawna

- **Założenie 2:**

Każda realizacja współbieżna równoważna dowolnej realizacji sekwencyjnej tego samego zbioru transakcji jest również poprawna

BD – wykład 8 (22)

Przejdziemy teraz do omówienia problematyki uszeregowalności realizacji.
Przyjmiemy następujące założenia:

- każda realizacja sekwencyjna jest poprawna;
- każda realizacja współbieżna równoważna dowolnej realizacji sekwencyjnej tego samego zbioru transakcji jest również poprawna.



Uszeregowalność realizacji (2)

Przykład:

Dane (początkowe wartości): $a=50$; $b=50$

Transakcja T1: sumuje konta a i b

Transakcja T2: przelewa 30 z konta a na konto b

Dana realizacja postaci:

s: ...r2(a, 50) w2(a, 20) r1(a,20) r1(b, 50) r2(b,50)
w2(b, 80) c1 c2

Czy dana realizacja jest poprawna?

BD – wykład 8 (23)

Jako przykład rozważmy transakcję T1, która sumuje wartości konta a i konta b i transakcję T2, która przelewa 30 z konta a na konto b. Założymy, że początkowa wartość konta $a=50$ i konta $b=50$. Przedstawiona na slajdzie realizacja nie jest poprawna ponieważ obraz bazy danych widziany przez transakcję T1 to $a+b=70$, zamiast 100.



Uszregowalność realizacji (3)

Realizacje sekwencyjne transakcji T1 i T2:

```
s1:...r1(a, 50) r1(b, 50) c1 r2(a, 50) w2(a, 20)  
r2(b, 50) w2(b, 80) c2 .....
```

końcowy stan bazy danych: a= 20; b= 80

obraz bazy danych widziany przez T2: a = 50; b = 50
obraz bazy danych widziany przez T1: a = 50; b = 50

BD – wykład 8 (24)

W przykładzie ze slajdu transakcje T1 i T2 są realizowane sekwencyjnie. W tym przypadku obraz bazy danych widziany przez obie transakcje jest poprawny.



Konflikt (1)

Dwie **operacje** $o_i(x)$, $o_j(y)$ współbieżnej realizacji są **konfliktowe**, wtedy i tylko wtedy, gdy są spełnione następujące trzy warunki:

1. $x = y$ Operacje na różnych danych nigdy nie są konfliktowe
2. $i \neq j$ Operacje konfliktowe muszą należeć do różnych transakcji
3. Jedna z dwóch operacji o_i lub o_j musi być operacją zapisu

BD – wykład 8 (25)

Jeżeli przynajmniej dwie operacje należące do różnych transakcji realizują dostęp do tej samej danej i przynajmniej jedna z tych operacji jest modyfikacją/zapisem danej, wówczas występuje konflikt w dostępie do tej danej. Bardziej formalnie: mówimy, że dwie operacje $o_i(x)$, $o_j(y)$ współbieżnej realizacji są konfliktowe, wtedy i tylko wtedy, gdy są spełnione następujące trzy warunki.

Po pierwsze, gdy dotyczą tej samej danej. Innymi słowy, operacje na różnych danych nigdy nie są konfliktowe.

Po drugie, operacje konfliktowe muszą należeć do różnych transakcji.

Po trzecie, jedna z dwóch operacji o_i lub o_j musi być operacją zapisu.



Konflikt (2)

- Dwie **transakcje** T_i, T_j są **konfliktowe**, jeżeli zawierają wzajemnie konfliktowe operacje
- Mówimy, że **operacja** $o_i(x)$ **poprzedza operację** $o_j(y)$ w realizacji $r(\tau)$, co zapisujemy jako $o_i(x) o_j(y)$, jeżeli operacje te są konfliktowe i $o_i(x) < r o_j(y)$
- Następujące pary operacji mogą znajdować się w konflikcie:
 - $r_i(x) w_j(x)$
 - $w_i(x) r_j(x)$
 - $w_i(x) w_j(x)$

BD – wykład 8 (26)

Pojęcie konfliktu można rozszerzyć na zbiór transakcji. Mówimy, że dwie transakcje T_i, T_j są konfliktowe, jeżeli zawierają wzajemnie konfliktowe operacje. Wprowadzimy obecnie pojęcie relacji poprzedzania operacji w realizacji $r(TAU)$. Mówimy, że operacja $o_i(x)$ znajduje się w relacji poprzedzania z operacją $o_j(y)$ w realizacji $r(TAU)$, co zapisujemy jako $o_i(x) \rightarrow o_j(y)$, jeżeli operacje te są konfliktowe i operacja $o_i(x)$ poprzedza w realizacji $r(TAU)$ operację $o_j(y)$.

Łatwo zauważyc, że następujące pary operacji mogą znajdować się w konflikcie:

- $r_i(x)$ i $w_j(x)$,
- $w_i(x)$ i $r_j(x)$,
- $w_i(x)$ i $w_j(x)$.



Konfliktowa równoważność

- Mówimy, że **transakcja T_i poprzedza transakcję T_j** w realizacji $r(\tau)$, co zapisujemy jako $T_i \rightarrow T_j$, jeżeli zawierają odpowiednio operacje $o_i(x)$ i $o_j(x)$, między którymi zachodzi związek poprzedzania
- Mówimy, że dwie realizacje $r(\tau) = (\overline{T}_r(\tau), < r)$ i $r'(\tau) = (\overline{T}_r(\tau), < r')$ są **konfliktowo równoważne**, jeżeli dla każdej pary operacji $o_i(x)$ i $o_j(y)$ w realizacji $r(\tau)$, takich, że $o_i(x) \rightarrow o_j(x)$, zachodzi również $o_i(x) \rightarrow o_j(y)$ w realizacji $r'(\tau)$

BD – wykład 8 (27)

Relacje poprzedzania można również rozszerzyć na zbiór transakcji. Mówimy, że transakcja T_i jest w relacji poprzedzania z transakcją T_j w realizacji $r(TAU)$, co zapisujemy jako $T_i \rightarrow T_j$, jeżeli transakcje te zawierają odpowiednio operacje $o_i(x)$ i $o_j(x)$, między którymi zachodzi relacja poprzedzania. Przypomnijmy, że zgodnie z założeniem 2, każda realizacja współbieżna równoważna dowolnej realizacji sekwencyjnej tego samego zbioru transakcji jest poprawna. Jak już wspominaliśmy, kluczowe, w powyższej definicji, jest pojęcie równoważności.

Obecnie, po wprowadzeniu relacji poprzedzania, możemy formalnie zdefiniować pojęcie równoważności dwóch realizacji. Mówimy, że dwie realizacje $r(TAU) = (Tr(TAU), < r)$ i $r'(TAU) = (Tr(TAU), < r')$ są **konfliktowo równoważne**, jeżeli dla każdej pary operacji $o_i(x)$ i $o_j(x)$ w realizacji $r(TAU)$, takich, że $o_i(x) \rightarrow o_j(y)$, zachodzi również $o_i(x) \rightarrow o_j(y)$ w realizacji $r'(TAU)$. Obecnie, sformułujemy kryterium poprawności współbieżnej realizacji zbioru transakcji nazywane kryterium konfliktowej uszeregowalności.



Kryterium konfliktowej uszeregowalności

Kryterium konfliktowej uszeregowalności

Realizacja $r(\tau)$ zbioru transakcji τ jest **konfliktowo uszeregowalna** wtedy i tylko wtedy, gdy jest ona konfliktowo równoważna dowolnej sekwencyjnej realizacji τ

Grafem konfliktowej-uszeregowalności realizacji $r(\tau)$

nazywamy skierowany graf $\text{CSRG}(r(\tau)) = (V, A)$, taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru T , natomiast zbiór krawędzi $A = \{(T_i, T_j) : T_i \rightarrow T_j\}$

BD – wykład 8 (28)

Definicja kryterium konfliktowej uszeregowalności brzmi następująco. Realizacja $r(\text{TAU})$ zbioru transakcji T jest **konfliktowo uszeregowalna** wtedy i tylko wtedy, gdy jest ona konfliktowo równoważna dowolnej sekwencyjnej realizacji zbioru TAU .

W jaki sposób można zweryfikować czy dana realizacja współbieżna spełnia kryterium konfliktowej uszeregowalności? W celu weryfikacji konfliktowej uszeregowalności realizacji konstruujemy graf konfliktowej uszeregowalności realizacji. Grafem konfliktowej uszeregowalności realizacji $r(\text{TAU})$ nazywamy skierowany graf $\text{CSRG}(r(\text{TAU})) = (V, A)$, taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru T , natomiast zbiór krawędzi zawiera relacje poprzedzania transakcji T_i i T_j : $A = \{(T_i, T_j) : T_i \rightarrow T_j\}$.



Twierdzenie o konfliktowej uszeregowalności

Realizacja $r(\tau)$ zbioru transakcji jest
konfliktowo-uszeregowalna

wtedy i tylko wtedy, gdy jej graf konfliktowej uszeregowalności CSRG($r(\tau)$) jest acykliczny

BD – wykład 8 (29)

Możemy obecnie, korzystając z grafu konfliktowej uszeregowalności sformułować twierdzenie, pozwalające w sposób algorytmiczny weryfikować, czy dana realizacja współbieżna jest poprawna, tj. konfliktowo-uszeregowalna. Realizacja $r(TAU)$ zbioru transakcji T jest konfliktowo-uszeregowalna wtedy i tylko wtedy, gdy jej graf konfliktowej uszeregowalności CSRG($r(TAU)$) jest acykliczny.

Poprawność powyższego twierdzenia wynika bezpośrednio z własności spójności transakcji. Zgodnie z własnością spójności, każda transakcja transformuje bazę danych z jednego stanu spójnego do innego stanu spójnego. Stąd wynika, że każda realizacja sekwencyjna zbioru transakcji zachowuje spójność bazy danych, gdyż jest ona sekwencją transformacji odwzorowujących bazę danych z jednego do innego stanu spójnego. Z definicji grafu konfliktowej uszeregowalności wynika, że graf ten, dla dowolnej realizacji sekwencyjnej, musi być acykliczny. Z definicji kryterium konfliktowej uszeregowalności wynika, że dowolna realizacja współbieżna jest poprawna, jeżeli jest ona równoważna dowolnej realizacji sekwencyjnego tego samego zbioru transakcji. Z definicji równoważności realizacji wynika, że graf konfliktowej uszeregowalności realizacji współbieżnej musi być również acykliczny, jeżeli realizacja ta jest konfliktowo-uszeregowalna. Co kończy skrótny dowód poprawności podanego twierdzenia.



Realizacje odtwarzalne (1)

- Czy własność uszeregowalności gwarantuje wolność od anomalii ?

Przykład:

$$H = r_1[x] \ w_1[x] \ r_1[y] \ r_2[x] \ w_1[y] \ r_2[y] \ c_2 \ r_1[z] \ w_1[z] <\text{crash}> c_1$$

- Historia H jest uszeregowalna, ale nie jest wolna od anomalii (brudny odczyt). Po restarcie systemu transakcja T2 nie zostanie poprawnie odtworzona

BD – wykład 8 (30)

Można sformułować następujące pytanie: Czy własność uszeregowalności gwarantuje poprawność dowolnej realizacji transakcji, w szczególności, czy gwarantuje wolność od anomalii współbieżnego wykonywania transakcji? Odpowiedź na to pytanie jest twierdząca, jeżeli rozważamy wyłącznie realizacje będące zaakceptowanymi projekcjami, tj. realizacje zawierające tylko operacje zatwierdzonych transakcji. W rzeczywistości, realizacje zawierają nie tylko operacje zatwierdzonych transakcji. Transakcje są wycofywane przez użytkowników, podlegają awariom, są wycofywane przez system np. na skutek wystąpienia zakleszczenia. Jeżeli rozważamy realizacje, które zawierają operacje wycofywanych transakcji, to odpowiedź na postawione na wstępie pytanie jest negatywna. Ilustruje to przykładowa realizacja przedstawiona na slajdzie. Realizacja ta jest uszeregowalna – po usunięciu operacji transakcji T1, której wykonanie zostało przerwane na skutek wystąpienia awarii w systemie, pozostała realizacja zawiera operacje zatwierdzonej transakcji T1. Niestety, realizacja ta, mimo, że uszeregowalna, nie jest wolna od anomalii brudnego odczytu.

Transakcja T2 odczytuje wartości danych x i y zapisane przez transakcję T1, która następnie, jest wycofywana. Zauważmy, że po restarcie systemu, transakcja T2 nie zostanie poprawnie odtworzona, gdyż powinna ona odczytać stan bazy danych przed wykonania transakcji T1.



- Potrzebna jest definicja nowych własności realizacji wykluczających anomalie będące wynikiem awarii systemu
- Mówimy, że transakcja T_i **czyta** daną x z transakcji T_j w realizacji H jeżeli Mówimy, że transakcja T_i czyta daną x z transakcji T_j w realizacji H jeżeli:

$$w_j[x] < r_i[x]$$

$$a_j < r_i[x]$$

jeżeli istnieje operacja $w_k[x]$ taka, że $w_j[x] < w_k[x] < r_i[x]$,
wtedy $a_k < r_i[x]$

- Mówimy, że transakcja T_i **czyta** z transakcji T_j w realizacji H , jeżeli T_i czyta jakąś daną z transakcji T_j w realizacji H

BD – wykład 8 (31)

Jeżeli rozważamy szerszą klasę realizacji, które zawierają operacje zatwierdzonych jak i wycofywanych, na skutek awarii, transakcji, potrzebne są definicje nowych własności realizacji, wykluczających anomalie będące wynikiem awarii systemu. Aby zdefiniować nowe niezbędne własności realizacji, konieczne jest wprowadzenie dodatkowych definicji.

Mówimy, że transakcja T_i czyta daną x z transakcji T_j w realizacji H jeżeli:

1. $w_j[x] < r_i[x]$;
2. $a_j < r_i[x]$;
3. jeżeli istnieje operacja $w_k[x]$ taka, że $w_j[x] < w_k[x] < r_i[x]$, wtedy $a_k < r_i[x]$.

Mówimy, że transakcja T_i czyta z transakcji T_j w realizacji H , jeżeli T_i czyta dowolną daną z transakcji T_j w realizacji H .



Realizacje odtwarzalne (2)

- Realizacja H jest **odtwarzalna** (ang. *recoverable*) (*RC*) wówczas, jeżeli transakcja T_i czyta z transakcji T_j ($i \neq j$) w realizacji H i $c_i \in H$, to $c_j < c_i$
- Realizacja H unika **kaskadowych wycofań** (ang. *avoids cascading aborts*) (*ACA*) wówczas, jeżeli transakcja T_i czyta z transakcji T_j ($i \neq j$), to $c_j < r_i[x]$
- Realizacja H jest **ścisła** (ang. *strict*) (*ST*) wówczas, jeżeli $w_j[x] < o_i[x]$ ($i \neq j$), zachodzi $a_j < o_i[x]$ lub $c_j < o_i[x]$, gdzie $o_i[x]$ jest jedną z operacji $r_i[x]$ lub $w_i[x]$

BD – wykład 8 (32)

Korzystając z wprowadzonych pojęć, możemy obecnie zdefiniować nowe klasy realizacji transakcji.

Mówimy, że realizacja H jest odtwarzalna (ang. *recoverable*) (*RC*) wówczas, jeżeli transakcja T_i czyta z transakcji T_j (i różne od j) w realizacji H i c_i należy do realizacji H , to $c_j < c_i$

Mówimy, że realizacja H unika kaskadowych wycofań (ang. *avoids cascading aborts*) (*ACA*) wówczas, jeżeli transakcja T_i czyta z transakcji T_j (i różne od j), to $c_j < r_i[x]$

Mówimy, że realizacja H jest ścisła (ang. *strict*) (*ST*) wówczas, jeżeli $w_j[x] < o_i[x]$ (i różne od j), zachodzi $a_j < o_i[x]$ lub $c_j < o_i[x]$, gdzie $o_i[x]$ jest jedną z operacji $r_i[x]$ lub $w_i[x]$



Realizacje odtwarzalne (3)

- Przykład:

$$\begin{aligned} T_1 &= w_1[x] \ w_1[y] \ w_1[z] \ c_1 \\ T_2 &= r_2[u] \ w_2[x] \ r_2[y] \ w_2[y] \ c_2 \end{aligned}$$

$$\begin{aligned} H_1 &= w_1[x] \ w_1[y] \ r_2[u] \ w_2[x] \ r_2[y] \ w_2[y] \ c_2 \ w_1[z] \ c_1 \\ H_2 &= w_1[x] \ w_1[y] \ r_2[u] \ w_2[x] \ r_2[y] \ w_2[y] \ w_1[z] \ c_1 \ c_2 \\ H_3 &= w_1[x] \ w_1[y] \ r_2[u] \ w_2[x] \ w_1[z] \ c_1 \ r_2[y] \ w_2[y] \ c_2 \\ H_4 &= w_1[x] \ w_1[y] \ r_2[u] \ w_1[z] \ c_1 \ w_2[x] \ r_2[y] \ w_2[y] \ c_2 \end{aligned}$$

BD – wykład 8 (33)

Dla ilustracji nowo wprowadzonych klas realizacji transakcji rozważmy przykładowe realizacje przedstawione na slajdzie. Dane są dwie transakcje T1 i T2 przedstawione na slajdzie.

Rozważmy realizację H1. Realizacja H1 jest realizacją konfliktowo-uszeregowalną, ale nie jest realizacją odtwarzalną. Transakcja T2 czyta daną y z transakcji T1, ale $c_2 < c_1$. Łatwo również zauważyć, że realizacja H1 nie należy do klasy realizacji ACA jak również ST.

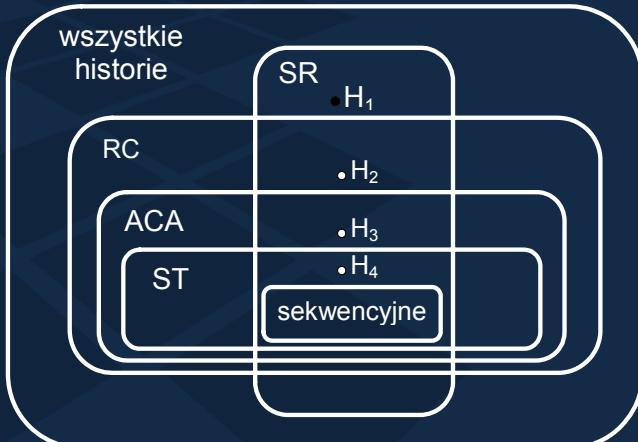
Rozważmy realizację H2. Realizacja H2 jest realizacją konfliktowo-uszeregowalną. Ponadto, jest również realizacją odtwarzalną. Transakcja T2 czyta daną y z transakcji T1, ale tym razem $c_1 < c_2$. Realizacja H2 nie należy do klasy realizacji ACA jak również ST.

Rozważmy realizację H3. Realizacja H3 jest realizacją konfliktowo-uszeregowalną i odtwarzalną. Ponadto, jest ona również realizacją unikającą kaskadowych wycofań (należy do klasy ACA). Transakcja T2 czyta daną y z transakcji T1 i spełniony jest warunek $c_1 < r_2[y]$. Realizacja H3 nie jest natomiast realizacją ścisłą.

Wreszcie, rozważmy realizację H4. Realizacja H4 jest realizacją konfliktowo-uszeregowalną, odtwarzalną unikającą kaskadowych wycofań oraz realizacją ścisłą.

Zależności między zbiorami realizacji RC, ACA i ST

Twierdzenie: $ST \subset ACA \subset RC$



BD – wykład 8 (34)

Można sformułować i udowodnić następujące twierdzenie określające zależności pomiędzy zbiorami realizacji RC, ACA i ST.

Zbiór realizacji ścisłych zawiera się w zbiorze realizacji unikających kaskadowych wycofań, który, z kolei, zawiera się w zbiorze realizacji odtwarzalnych. Diagram przedstawiony na slajdzie ilustruje zależności pomiędzy zbiorami realizacji RC, ACA i ST. Na diagramie zaznaczono również przynależność przykładowych realizacji H1, H2, H3 i H4.



Izolacja = Uszeregowalność \cap ST

BD – wykład 8 (35)

Mówiąc o własnościach transakcji, stwierdziliśmy, że jedną z czterech własności transakcji jest własność izolacji. Korzystając z wprowadzonych dotychczas pojęć, własność izolacji transakcji możemy zdefiniować formalnie. Własność izolacji transakcji oznacza, że transakcja jest konfliktowo-uszeregowalna i ścisła.



Realizacje uszeregowalne

- Realizacja r zbioru transakcji jest **poprawna** (uszeregowalna) jeżeli jest ona obrazowo i stanowo równoważna jakiejkolwiek sekwencyjnej realizacji tego zbiuru transakcji. Realizację taką nazywamy **realizacją uszeregowalną (SR)**

BD – wykład 8 (36)

Przedstawiona, na poprzednich slajdach, definicja kryterium konfliktowej uszeregowalności stanowi zmodyfikowaną wersję podstawowego kryterium poprawności współbieżnej realizacji transakcji, które nosi nazwę kryterium uszeregowalności. Zasadnicza różnica pomiędzy definicją kryterium uszeregowalności a kryterium konfliktowej uszeregowalności kryje się w definicji równoważności realizacji transakcji. Formalnie, definicja kryterium uszeregowalności brzmi następująco: realizacja $r(T)$ zbiuru transakcji T jest poprawna (uszeregowalna), jeżeli jest ona obrazowo i stanowo równoważna jakiejkolwiek sekwencyjnej realizacji zbiuru transakcji T . Realizację $r(T)$, spełniającą zdefiniowane powyżej kryterium, nazywamy realizacją uszeregowalną (należy do klasy SR). Jak łatwo zauważyc, równoważność konfliktowa realizacji została zastąpiona w kryterium uszeregowalności równoważnością obrazową i stanową realizacji. Kryterium uszeregowalności ma charakter bardziej egzystencjalny, jest natomiast mało przydatne w sensie konstrukcyjnym.



Graf uszeregowalności (1)

- **Grafem uszeregowalności** realizacji $r(\tau)$ nazywamy skierowany graf $SG(r(\tau)) = (V, A)$, taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru τ , natomiast zbiór krawędzi jest zdefiniowany następująco:
 - Jeżeli istnieje dana x , i operacje $T_i : r(x), T_j : w(x) \in T_r(\tau)$, takie, że $T_i : r(x)$ czyta wartość danej x zapisanej przez operację $T_j : w(x)$, to:
 1. $(T_j, T_i) \in A$
 2. Jeżeli $T_j \neq T_0, T_i \neq T_f$ i istnieje operacja $T_k : w(x) \in T_r(\tau)$, $T_k \neq T_0$, to $(T_k, T_j) \in A$ lub $(T_i, T_k) \in A$
 3. Jeżeli $T_j \neq T_0$, to $(T_0, T_j) \in A$

BD – wykład 8 (37)

W celu weryfikacji uszeregowalności realizacji konstrujemy graf uszeregowalności realizacji. Grafem uszeregowalności realizacji $r(T)$ nazywamy skierowany graf $SG(r(T)) = (V, A)$, taki, w którym zbiór wierzchołków V odpowiada transakcjom ze zbioru T , natomiast definicja zbioru krawędzi została przedstawiona na prezentowanych slajdach.



Graf uszeregowalności (2)

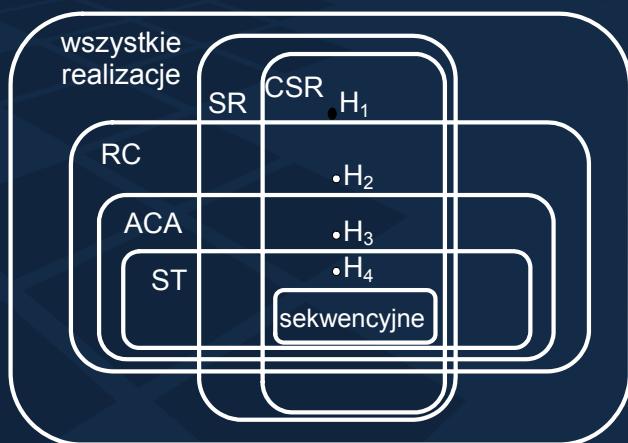
4. Jeżeli $T_j = T_0$, $T_i \neq T_f$ i istnieje operacja $T_k : w(x) \in T_r(\tau)$, $T_k \neq T_0$, to $(T_i, T_k) \in A$;
5. Jeżeli $T_i = T_f$, i istnieje operacja $Tk : w(x) \in T_r(\tau)$, to $(T_k, T_j) \in A$
 - Dana realizacja $r(\tau)$ jest uszeregowalna wtedy i tylko wtedy, gdy można skonstruować dla niej acykliczny skierowany graf uszeregowalności $SG(r(\tau))$

Dana realizacja $r(\tau)$ jest uszeregowalna wtedy i tylko wtedy, gdy można skonstruować dla niej acykliczny skierowany graf uszeregowalności $SG(r(\tau))$

BD – wykład 8 (38)

Można pokazać, że dana realizacja $r(T)$ jest uszeregowalna wtedy i tylko wtedy, gdy można skonstruować dla niej acykliczny skierowany graf uszeregowalności $SG(r(T))$. Problem weryfikacji, czy dana realizacja jest uszeregowalna, jest problemem NP.-zupełnym, to znaczy, problemem obliczeniowo trudnym. Trudność weryfikacji kryterium uszeregowalności wynika z konstrukcji tego grafu. Z definicji grafu uszeregowalności wynika, że wynikowy graf jest poligrafem (patrz warunek 2). Z teorii grafów wynika, że weryfikacja czy dany poligraf zawiera cykl jest problemem NP.-zupełnym. Stąd, w praktyce, kryterium uszeregowalności zastąpiono łatwiejszym do weryfikacji kryterium konfliktowej uszeregowalności. Problem weryfikacji, czy dana realizacja jest realizacją konfliktowo-uszeregowalną jest problemem obliczeniowo łatwym (problem należy do klasy problemów P).

Uszeregowalność transakcji - klasyfikacja

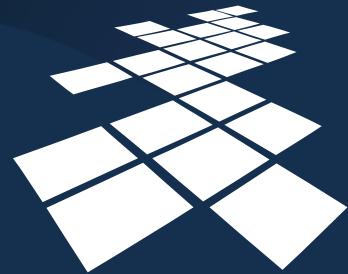


BD – wykład 8 (39)

Diagram przedstawiony na slajdzie ilustruje zależności pomiędzy zbiorami realizacji RC, ACA i ST oraz zbiorami realizacji uszeregowalnych, konfliktowo uszeregowalnych oraz realizacji sekwencyjnych. Jak widać z diagramu, zbiór realizacji konfliktowo uszeregowalnych jest podzbiorem zbioru realizacji uszeregowalnych.

Algorytmy zarządzania współbieżnym wykonywaniem transakcji część I

Wykład przygotował:
Tadeusz Morzy



UCZELNIA
ONLINE

BD – wykład 9

Celem wykładu jest przedstawienie i omówienie podstawowych algorytmów zarządzania współbieżnym wykonywaniem transakcji. Rozpoczniemy od przedstawienia algorytmów blokowania. Omówimy podstawowy algorytm blokowania – algorytm blokowania dwu-fazowego. Następnie przedstawimy zjawisko zakleszczenia i omówimy podstawowe algorytmy rozwiązywania zakleszczenia. Na zakończenie wykładu, przedstawimy i omówimy problem duchów.



Klasyfikacja algorytmów

Algorytmy zarządzania współbieżnym wykonywaniem transakcji możemy sklasyfikować następująco:

1. **algorytmy blokowania** - uszeregowanie transakcji wynika z kolejności uzyskiwanych blokad (algorytm blokowania dwufazowego – 2PL)
2. **algorytmy znaczników czasowych** - uszeregowanie transakcji wynika z wartości znaczników czasowych związanych z transakcjami
3. **algorytmy optymistyczne** - walidacja poprawności uszeregowania

BD – wykład 9 (2)

Algorytmy zarządzania współbieżnym wykonywaniem transakcji możemy sklasyfikować następująco:

algorytmy blokowania - uszeregowanie transakcji wynika z kolejności uzyskiwanych blokad (algorytm blokowania dwufazowego – 2PL);

algorytmy znaczników czasowych - uszeregowanie transakcji wynika z wartości znaczników czasowych związanych z transakcjami;

algorytmy optymistyczne - walidacja poprawności uszeregowania.



Algorytmy blokowania (1)

- Blokada jest zmienną skojarzoną z każdą daną w bazie danych, określającą dostępność danej ze względu na możliwość wykonania na niej określonych operacji
- Ogólnie, z każdą daną mamy skojarzoną jedną blokadę

Ze względu na proces blokowania, dane w bazie danych mogą występować w jednym z trzech stanów:

- dana nie zablokowana (**O**)
- dana zablokowana dla odczytu **R** (współdzielona **S**)
- dana zablokowana dla zapisu **W** (wyłączna **X**)

BD – wykład 9 (3)

Podstawową grupą algorytmów zarządzania współbieżnym wykonywaniem transakcji są algorytmy blokowania. Algorytmy te opierają się na mechanizmie blokad zakładanych przez transakcje. Blokada jest zmienną skojarzoną z każdą daną w bazie danych, określającą dostępność danej ze względu na możliwość wykonania na niej określonych operacji. Ogólnie, z każdą daną mamy skojarzoną jedną blokadę.

Ze względu na proces blokowania, dane w bazie danych mogą występować w jednym z trzech stanów:

- dana nie zablokowana (**O**),
- dana zablokowana dla odczytu **R** (współdzielona **S**),
- dana zablokowana dla zapisu **W** (wyłączna **X**).



Algorytmy blokowania (2)

- System zarządzania bazą danych musi realizować trzy dodatkowe operacje na bazie danych:
 - Blokowanie danej x do odczytu (LR(x))
 - Blokowanie danej x do zapisu (LW(x))
 - Odblokowanie danej x (UNL(x))
- Operacje blokowania muszą poprzedzać wykonanie operacji odczytu oraz zapisu danej

BD – wykład 9 (4)

Wprowadzenie blokad pociąga za sobą konieczność modyfikacji zbioru operacji wykonywanych na bazie danych. Podstawowy zbiór operacji transakcji (odczyt, zapis, zatwierdzenie, wycofanie) rozszerzamy o 3 dodatkowe operacje:

- Blokowanie danej x do odczytu (LR(x)),
- Blokowanie danej x do zapisu (LW(x)),
- Odblokowanie danej x (UNL(x)).

Operacje blokowania muszą poprzedzać wykonanie operacji odczytu oraz zapisu danej. Z każdą operacją blokowania danej X skojarzona jest operacja odblokowania danej X.



Kompatybilność blokad

- Dwie blokady są kompatybilne jeżeli mogą być założone na tej samej danej przez dwie różne transakcje

Blokada uzyskana	R	W
Blokada żądana		
R	✓	-
W	-	-

BD – wykład 9 (5)

Wprowadzimy obecnie pojęcie kompatybilności blokad. Mówimy, że dwie blokady są kompatybilne jeżeli mogą być założone na tej samej danej przez dwie różne transakcje. Macierz kompatybilności blokad przedstawiono na slajdzie. Jak łatwo zauważyc, kompatybilne są blokady do odczytu, natomiast pozostałe kombinacje blokad (RW, WR, WW) są niekompatybilne.



Konwersja blokad

- Transakcja posiadająca blokadę określonego typu na danej może dokonać jej konwersji w blokadę innego typu

Blokada uzyskana	R	W
Blokada żądana		
R	✓	-
W	✓	✓

BD – wykład 9 (6)

Wprowadzimy obecnie pojęcie konwersji blokad. Transakcja posiadająca blokadę określonego typu na danej może dokonać jej konwersji w blokadę innego typu zgodnie z przyjętą macierzą konwersji blokad. Macierz konwersji blokad przedstawiono na slajdzie. Jak łatwo zauważyc, dopuszczalna jest konwersja blokady typu R na blokadę typu W (nazywamy to operacją eskalacji blokad). Natomiast niedopuszczalna jest konwersja z blokady do zapisu (W) na blokadę do odczytu (R).



Implementacja algorytmów blokowania (1)

- Struktury danych:

dana	tid	blokada
x1	T1	W
x2	T1	R
x2	T2	R
x3	T2	W



dana	tid	blokada	kolejka
x1	T2	R	1
x1	T3	W	2
x2	T4	W	1

BD – wykład 9 (7)

Obecnie przedstawimy implementację algorytmów blokowania. Implementacja ta obejmuje struktury danych wspierające blokowanie oraz algorytmy zakładania i zdejmowania blokad.

Z każdą daną jest związana blokada tej danej oraz dwie kolejki transakcji: kolejka transakcji, które uzyskały dostęp do danej oraz kolejka transakcji oczekujących na dostęp do danej. Dla ilustracji rozważmy przykład przedstawiony na slajdzie.

Dana x1 jest zablokowana przez transakcję T1 do zapisu (blokada W). W kolejce transakcji oczekujących na dostęp do x1 znajdują się dwie transakcje, tj. T2 i T3, które żądają odpowiednio blokady do odczytu (R) i zapisu (W). Dana x2 jest zablokowana do odczytu przez transakcje T1 i T2 (blokady do odczytu są kompatybilne dla obu tych transakcji). W kolejce transakcji oczekujących na dostęp do x2 znajduje się transakcja T4, która żąda blokady do zapisu (W). Dana x3 jest zablokowana przez transakcję T2 do zapisu, natomiast kolejka transakcji oczekujących na dostęp do x3 jest pusta.



Implementacja algorytmów blokowania (2)

Operacje: **LOCK, R_lock, W_lock, Unlock**

```
LOCK(X, tid) {O, R, W}  
R_lock(X, tid) begin  
    B: if (LOCK(X, tid)=O or LOCK(X, tid)=R)  
        then LOCK(X, tid) ← R;  
    else begin  
        < insert into queue(X) and wait  
        until lock  
        manager wakes up the transaction>;  
        go to B;  
    end;  
end R_lock;
```

BD – wykład 9 (8)

Przedstawimy obecnie algorytmy zakładania i zdejmowania blokad. Rozpoczniemy od algorytmu zakładania blokady do odczytu, przedstawionego na slajdzie.

Algorytm przedstawia założenie blokady do odczytu przez transakcję tid na danej X. Jeżeli dana X jest niezablokowana lub dana X jest zablokowana do odczytu to transakcja tid zakłada blokadę do odczytu (R) danej X. W przeciwnym razie, transakcja tid jest umieszczana w kolejce transakcji oczekujących na dostęp do danej X i oczekuje tam do momentu zwolnienia blokady na danej X.



Implementacja algorytmów blokowania (3)

```
W_lock(X, tid) begin
    B: if LOCK(X, tid) = 0
        then LOCK(X, tid) ← W;
        else begin
            < insert into queue(X) and wait
                until lock manager
                wakes up the transaction>;
            go to B;
        end;
    end W_lock;
```

BD – wykład 9 (9)

Algorytm zakładania blokady do zapisu przedstawiono na slajdzie. Algorytm przedstawia założenie blokady do zapisu przez transakcję tid na danej X. Jeżeli dana X jest niezablokowana to transakcja tid zakłada blokadę do zapisu (W) danej X. W przeciwnym razie, transakcja tid jest umieszczana w kolejce transakcji oczekujących na dostęp do danej X i oczekuje tam do momentu zwolnienia blokady na danej X.



Implementacja algorytmów blokowania (4)

```
Unlock(X, tid) begin
    if LOCK(X, tid) = W
    then begin
        LOCK(X, tid) ← 0;
        < wake up one of the waiting
            transactions, if any >;
    end;
    else if LOCK(X, tid) = R
    then begin
        LOCK(X, tid) ← 0;
        if (number_of_read_locks_on_X=0)then
        begin
            < wake up one of the waiting
                transactions, if any >;
        end;
    end;
end Unlock;
```

BD – wykład 9 (10)

Niniejszy slajd przedstawia algorytm odblokowania danej X przez transakcję tid. Jeżeli dana X była blokowana w trybie do zapisu (W) przez transakcję tid, wówczas, następuje zdjęcie blokady do zapisu (dana X jest niezablokowana) i dostęp do danej X uzyskuje pierwsza transakcja z kolejki transakcji oczekujących na dostęp do danej X. Jeżeli dana X była blokowana w trybie do odczytu (R) przez transakcję tid, wówczas następuje zdjęcie blokady do odczytu dla tej transakcji. Jeżeli transakcja tid była jedyną transakcją blokującą daną X do odczytu, wówczas, po zdjęciu blokady przez transakcję tid, dana X znajdzie się w stanie – nie zablokowana. Jeżeli natomiast, równolegle z transakcją tid dana X była blokowana do odczytu przez inne transakcje, wówczas, po zdjęciu blokady przez transakcję tid, dana X znajdzie się w stanie - zablokowana do odczytu.



Algorytm blokowania (1)

T_1	T_2
R_lock(T_1 , Y)	R_lock(T_2 , X)
read(Y)	read(X)
unlock(Y)	unlock(X)
W_lock(T_1 , X)	W_lock(T_2 , Y)
read(X)	read(Y)
$X := X + Y;$	$Y := Y + X;$
write(X);	write(Y);
unlock(X)	unlock(Y)

Wartości początkowe:

$$X = 20, Y = 30$$

Wyniki sekwencyjnych realizacji transakcji T_1 i T_2 :
 $(T_1 \rightarrow T_2) : X = 50, Y = 80; (T_2 \rightarrow T_1) : X = 70, Y = 50;$

BD – wykład 9 (11)

Rozważmy następujący przykład przedstawiony na slajdzie ilustrujący działanie mechanizmu zakładania i zdejmowania blokad. Dane są transakcje T_1 i T_2 . Transakcja T_1 odczytuje wartości danych X i Y . Następnie sumuje te wartości i zapisuje do danej X . Transakcja 2 odczytuje wartości danych X i Y . Następnie sumuje te wartości i zapisuje do danej Y .

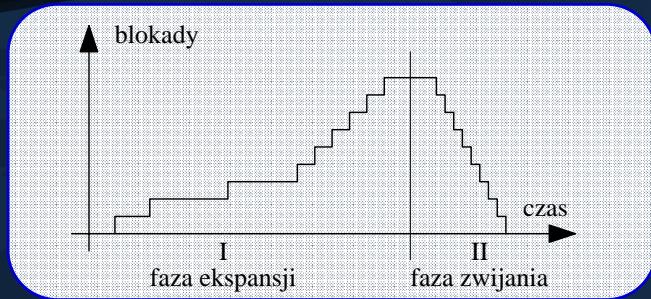
Załóżmy, że wartości początkowe danych X i Y wynoszą, odpowiednio, 20 i 30. Współbieżne wykonanie transakcji T_1 i T_2 przebiega następująco. Transakcja T_1 przed odczytem Y zakłada blokadę do odczytu tej danej. Po wykonaniu operacji odczytu, zdejmuje blokadę danej Y . Współbieżnie, transakcja T_2 przed odczytem X zakłada blokadę do odczytu tej danej. Po wykonaniu operacji odczytu, zdejmuje blokadę danej X . Następnie, transakcja T_1 zakłada blokadę do zapisu danej X , wykonuje odczyt danej X , aktualizuje wartość danej X , zapisuje zmodyfikowaną wartość danej X i zdejmuje blokadę do zapisu danej X . Następnie, transakcja T_2 zakłada blokadę do zapisu danej Y , wykonuje odczyt danej Y , aktualizuje wartość danej Y zapisując zmodyfikowaną wartość danej Y i zdejmuje blokadę do zapisu danej Y .

Końcowy stan bazy danych uzyskany w wyniku przedstawionego współbieżnego wykonania transakcji T_1 i T_2 wynosi $X=50$ i $Y=50$. Zauważmy, że stan bazy danych uzyskany w wyniku sekwencyjnej realizacji transakcji T_1 i T_2 różni się od stanu bazy danych uzyskanego w wyniku przedstawionego współbieżnego wykonania transakcji T_1 i T_2 . Dla realizacji sekwencyjnej, w której transakcja T_1 poprzedza transakcję T_2 końcowy stan bazy danych wynosi $X=50$ i $Y=80$, natomiast dla realizacji sekwencyjnej, w której transakcja T_2 poprzedza transakcję T_1 końcowy stan bazy danych wynosi $X=70$ i $Y=50$. Oznacza to, że przedstawiona realizacja współbieżna transakcji jest nieuszeregowalna.

Stosowanie blokad na danych nie gwarantuje automatycznie uszeregowalności realizacji zbioru transakcji.



Algorytm blokowania dwufazowego (1)



Algorytm podstawowy:

1. Każda operacja $read(X)$ danej transakcji T musi być poprzedzona operacją $R_lock(X, T)$ lub $W_lock(X, T)$
2. Każda operacja $write(X)$ danej transakcji T musi być poprzedzona operacją $W_lock(X, T)$
3. Operacje $unlock(x, T)$ dla danej transakcji T są wykonywane po zakończeniu wszystkich operacji $read$ i $write$

BD – wykład 9 (12)

Jak widać z poprzedniego przykładu stosowanie blokad na danych nie gwarantuje automatycznie uszeregowalności realizacji zbioru transakcji. Okazuje się, że istotna jest kolejność zakładania i zdejmowania blokad. Ilustruje to podstawowy algorytm blokowania, nazywany algorytmem blokowania dwu-fazowego (2PL). Podstawowa wersja algorytmu 2PL ma następującą postać:

1. Każda operacja odczytu danej X przez transakcję T ($read(X)$) musi być poprzedzona operacją zablokowania danej X w trybie do odczytu ($R_lock(X, T)$) lub w trybie do zapisu ($W_lock(X, T)$).
2. Każda operacja zapisu danej X przez transakcję T ($write(X)$) musi być poprzedzona operacją $W_lock(X, T)$.
3. Operacje odblokowania danej X ($unlock(x, T)$) dla danej transakcji T są wykonywane po zakończeniu wszystkich operacji $read$ i $write$.

Jak widać z przedstawionego schematu, realizacja transakcji, zgodnie z algorytmem 2PL, przebiega w dwóch fazach (stąd nazwa algorytmu): w fazie ekspansji oraz w fazie zwijania. W fazie ekspansji transakcja zakłada blokady kolejnych danych, nie zwalniając żadnej z uzyskanych blokad, aż do osiągnięcia punktu akceptacji. W fazie zwijania transakcja zwalnia blokady wszystkich danych, nie może żądać natomiast założenia żadnej blokady.



Algorytm blokowania dwufazowego (2)

- **Algorytm statyczny:** (1., 2., 3.)
Wszystkie blokady muszą być uzyskane przed rozpoczęciem transakcji (przez predeklarowanie zbioru odczytywanych i modyfikowanych danych)
- **Algorytm restryktywny:** (1., 2.)
Operacje $unlock(x, T)$ dla danej transakcji T są wykonywane po operacji $commit$ lub $rollback$

BD – wykład 9 (13)

Istnieje wiele wariantów podstawowego algorytmu blokowania dwu-fazowego. W algorytmie statycznym 2PL, składającym się z podstawowych kroków algorytmu 2PL z poprzedniego slajdu, wszystkie blokady muszą być uzyskane przed rozpoczęciem transakcji (przez predeklarowanie zbioru odczytywanych i modyfikowanych danych).

W algorytmie restryktywnym 2PL, kroki 1 i 2 są identyczne z algorytmem podstawowym 2PL, natomiast w kroku trzecim operacje $unlock(x, T)$ dla danej transakcji T są wykonywane po operacji $commit$ lub $rollback$.



Algorytm blokowania dwufazowego (3)

T₁	T₂
R_lock(T ₁ , Y)	
read(Y)	
W_lock(T ₁ , X)	
	R_lock(T ₂ , X)
read(X)	(wait)
X := X + Y;	(wait)
write(X);	(wait)
unlock(Y)	(wait)
unlock(X)	(wait)

	read(X)
	W_lock(T ₂ , Y)
	read(Y)
	Y := Y + X;
	write(Y);
	unlock(X)
	unlock(Y)

BD – wykład 9 (14)

Dla ilustracji działania algorytmu blokowania dwu-fazowego rozważmy ponownie przykładową realizację transakcji T1 i T2 ze slajdu nr 11. Transakcja T1 po uzyskaniu blokady do odczytu danej Y i uzyskaniu blokady do zapisu danej X wykonuje odczyt obu danych, aktualizuje i zapisuje daną X. Następnie, zdejmuję blokady danych X i Y. Współbieżnie, transakcja T2 żąda założenia blokady do odczytu danej X. Ponieważ dana X jest zablokowana przez transakcję T1 do zapisu, żądanie transakcji T2 nie może być zrealizowane i transakcja T2 zostaje umieszczona w kolejce transakcji oczekujących na uwolnienie danej X. Po odblokowaniu danej X przez transakcję T1, realizacja transakcji T2 jest wznowiona. T2 realizuje odczyt danych X i Y, aktualizuje i zapisuje nową wartość danej Y.

Końcowy stan bazy danych uzyskany w wyniku przedstawionego współbieżnego wykonania transakcji T1 i T2 wynosi tym razem X=50 i Y=80. Zauważmy, że tym razem stan końcowy bazy danych jest równoważny stanowi bazy danych uzyskanemu w wyniku sekwencyjnej realizacji transakcji T1 i T2. Oznacza to, że przedstawiona realizacja współbieżna transakcji jest uszeregowalna.



Zakleszczenie transakcji (1)

T_1	T_2
R_lock(T_1 , Y)	
read(Y)	
	R_lock(T_2 , X)
	read(X)
	W_lock(T_2 , Y)
W_lock(T_1 , X)	(wait)
(wait)	(wait)
(wait)	(wait)
dead	lock

Dwa podejścia do problemu zakleszczenia transakcji:

- Wykrywanie i rozwiązywanie zakleszczenia
- Zapobieganie wystąpieniu zakleszczenia

BD – wykład 9 (15)

Rozważmy przykład współbieżnej realizacji transakcji T1 i T2, zgodnie z algorytmem 2PL, przedstawiony na slajdzie. Transakcja T1 zakłada blokadę do odczytu danej Y, a następnie żąda założenia blokady do zapisu danej X. Współbieżnie transakcja T2 zakłada blokadę do odczytu danej X, a następnie żąda założenia blokady do zapisu danej Y. Zauważmy, że realizacja obu transakcji ulega zawieszeniu. Transakcja T1 oczekuje na uwolnienie danej X przez transakcję T2, natomiast transakcja T2 oczekuje na uwolnienie danej Y blokowanej przez transakcję T1. Stan wzajemnego oczekiwania transakcji na uwolnienie zasobów nazywamy zakleszczeniem transakcji.

W systemach baz danych stosowane są dwa podejścia do problemu zakleszczenia transakcji:

- wykrywanie i rozwiązywanie zakleszczenia,
- zapobieganie wystąpieniu zakleszczenia.



Algorytmy zapobiegania zakleszczeniom (1)

- Algorytmy wykorzystujące **znaczniki czasowe transakcji** - $TS(T)$, nadawane w momencie inicjacji transakcji:
- wait-die:** Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_i będzie czekać na zwolnienie blokady. W przeciwnym wypadku T_i będzie wycofana i restartowana z tym samym znacznikiem czasowym
- wound-wait:** Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_i będzie wycofana i restartowana z tym samym znacznikiem czasowym. W przeciwnym wypadku T_i będzie czekać na zwolnienie blokady

BD – wykład 9 (16)

Przedstawimy obecnie krótko dwa podstawowe algorytmy zapobiegania zakleszczeniom. Oba algorytmy do rozwiązywania problemu zakleszczenia wykorzystują tzw. znaczniki czasowe transakcji. Znacznik czasowy transakcji T , oznaczany jako $TS(T)$, jest nadawany transakcji w momencie jej inicjacji i stanowi konkatenację stanu zegara fizycznego (lub logicznego) oraz identyfikatora stanowiska. Cechą charakterystyczną znacznika czasowego transakcji jest jego unikalność.

Pierwszy z wymienionych algorytmów zapobiegania zakleszczeniom, algorytm wait-die, ma następującą postać:

Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_i będzie czekać na zwolnienie blokady. W przeciwnym wypadku T_i będzie wycofana i restartowana z tym samym znacznikiem czasowym.

Drugi z wymienionych algorytmów zapobiegania zakleszczeniom, algorytm wound-wait, ma podobną filozofię działania ale nadaje inne priorytety działania transakcjom starszym i transakcjom młodszym. Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Jeżeli $TS(T_i) < TS(T_j)$ (T_i jest starsza T_j) wtedy transakcja T_j będzie wycofana i restartowana z tym samym znacznikiem czasowym. W przeciwnym wypadku T_i będzie czekać na zwolnienie blokady.

Oba algorytmy zapobiegają wystąpieniu zakleszczenia. Ich wadą jest to, że czasami prowadzą do wycofania transakcji nawet jeżeli nie występuje niebezpieczeństwo zakleszczenia.



Algorytmy zapobiegania zakleszczeniom (2)

- Algorytmy nie korzystające ze **znaczników czasowych**.
- **no waiting**: Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Transakcja T_i będzie wycofana i restartowana z pewnym opóźnieniem czasowym.
- **cautious waiting**: Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Jeżeli transakcja T_j nie czeka na uzyskanie innej blokady, T_i będzie czekać na zwolnienie blokady przez T_j . W przeciwnym wypadku T_i będzie wycofana i restartowana

BD – wykład 9 (17)

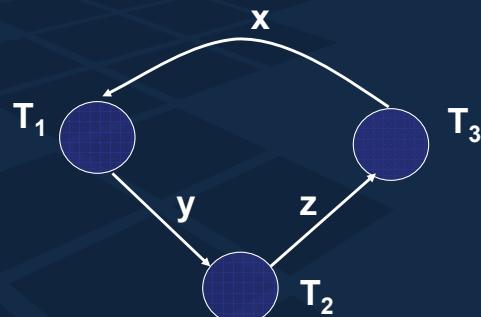
Drugą grupą algorytmów zapobiegania zakleszczeniom są algorytmy nie korzystające ze znaczników czasowych.

no waiting: Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Transakcja T_i będzie wycofana i restartowana z pewnym opóźnieniem czasowym.

cautious waiting: Transakcja T_i próbuje uzyskać blokadę na danej X , tymczasem dana ta jest już zablokowana przez transakcję T_j . Jeżeli transakcja T_j nie czeka na uzyskanie innej blokady, T_i będzie czekać na zwolnienie blokady przez T_j . W przeciwnym wypadku T_i będzie wycofana i restartowana.

Metody wykrywania i rozwiązywania zakleszczeń (1)

- **Graf oczekiwania (waits-for graph – WFG)**



Zakleszczenie jest zjawiskiem stosunkowo rzadkim i, najczęściej, obejmuje niewiele transakcji. Stąd, taniej jest wykrywać zakleszczenia i je rozwiązywać w momencie wystąpienia.

BD – wykład 9 (18)

Drugą z popularnych metod rozwiązywania problemu zakleszczenia jest metoda wykrywania i rozwiązywania zakleszczeń. Idea tej metody polega na wykrywaniu zakleszczenia i następnie na jego rozwiązywaniu.

Idea algorytmu wynika ze spostrzeżenia, że zakleszczenie jest zjawiskiem stosunkowo rzadkim i, najczęściej, obejmuje niewiele transakcji. Stąd, taniej jest wykrywać zakleszczenia i je rozwiązywać w momencie wystąpienia, niż zapobiegać wystąpieniu zakleszczeń. Do wykrywania zakleszczeń metody te wykorzystują graf oczekiwania transakcji (waits-for-graphs – WFG), który reprezentuje stan wzajemnego oczekiwania transakcji na uwolnienie zasobów. Węzłami grafu są transakcje, natomiast łuki reprezentują stan oczekiwania transakcji na uwolnienie zasobu.

Przykładowo, na grafie FWG przedstawionym na slajdzie transakcja T1 oczekuje na zwolnienie danej Y blokowanej przez transakcję T2. Transakcja T2 oczekuje na uwolnienie danej Z blokowanej przez transakcję T3, która, z kolei oczekuje na zwolnienie danej X blokowanej przez transakcję T1.

Cykl w grafie FWG oznacza wystąpienie zakleszczenia w systemie.



Metody wykrywania i rozwiązywania zakleszczeń (2)

- Graf WFG jest okresowo sprawdzany, czy wystąpił w nim cykl. Zakleszczenie jest rozwiązywane przez wycofanie jednej z transakcji należących do cyklu
- Mechanizm timeout-u: jeżeli transakcja czeka zbyt długo na założenie blokady (przekroczyła czas timeout-u), możemy założyć, że wystąpiło zakleszczenie

BD – wykład 9 (19)

Algorytm wykrywania i rozwiązywania zakleszczeń konstruuje okresowo graf FWG i sprawdza, czy wystąpił w nim cykl. Zakleszczenie jest rozwiązywane przez wycofanie jednej z transakcji należących do cyklu. Graf FWG jest konstruowany przez specjalny proces systemowy w oparciu o mechanizm timeout-u. Jeżeli transakcja czeka zbyt długo na założenie blokady (przekroczyła limit czasu przydzielony jej przez system), możemy założyć, że wystąpiło zakleszczenie.



Procedura wykrywania zakleszczenia (1)

- Do budowy grafu WFG wykorzystujemy struktury opisujące blokady. Z każdą blokadą są związane dwie listy: lista transakcji, które uzyskały blokadę, oraz lista transakcji oczekujących na przydział blokady. Obie listy mają postać $((T_i, m_i), \dots)$, gdzie T_i oznacza transakcję, natomiast m_i oznacza rodzaj blokady. Do grafu WFG dodajemy łuk $T_i \rightarrow T_j$, jeżeli zachodzi warunek:
 - transakcja T_j należy do listy transakcji, które uzyskały blokadę, natomiast T_i jest na liście transakcji oczekujących, lub
 - transakcja T_j jest przed transakcją T_i na liście transakcji oczekujących
 - blokady m_i i m_j są niekompatybilne

BD – wykład 9 (20)

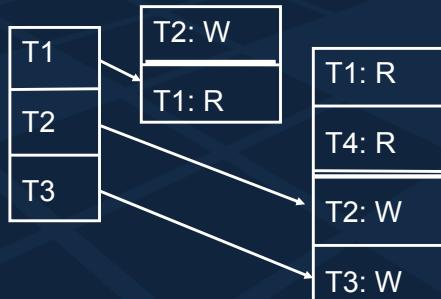
Do budowy grafu WFG wykorzystujemy struktury opisujące blokady. Z każdą blokadą są związane dwie listy: lista transakcji, które uzyskały blokadę, oraz lista transakcji oczekujących na przydział blokady. Obie listy mają postać $((T_i, m_i), \dots)$, gdzie T_i oznacza transakcję, natomiast m_i oznacza rodzaj blokady. Do grafu WFG dodajemy łuk $T_i \rightarrow T_j$, jeżeli zachodzi warunek:

- transakcja T_j należy do listy transakcji, które uzyskały blokadę, natomiast T_i jest na liście transakcji oczekujących, lub
- transakcja T_j jest przed transakcją T_i na liście transakcji oczekujących,
- blokady m_i i m_j są niekompatybilne.

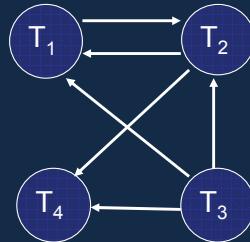


Procedura wykrywania zakleszczenia (2)

Transaction
wait list Lock
lists



WFG



BD – wykład 9 (21)

Dla ilustracji działania procedury wykrywania zakleszczenia rozważmy przykład przedstawiony na slajdzie. Mamy dwie dane oraz dwie kolejki transakcji, które uzyskały dostęp do tych danych. Pierwsza z danych jest blokowana przez transakcję T2 do zapisu (W), natomiast druga jest blokowana przez transakcje T1 i T4 do odczytu (R). Transakcja T1 oczekuje na transakcję T2 na uwolnienie pierwszego z zasobów, natomiast transakcje T2 i T3 oczekują na uwolnienie drugiego z zasobów.

Graf WFG składa się z 4 wierzchołków reprezentujących transakcje T₁, T₂, T₃, T₄. Łuk (T₁,T₂) reprezentuje oczekiwanie transakcji T₁ na T₂ w odniesieniu do pierwszego zasobu. Łuk (T₂,T₁) reprezentuje oczekiwanie transakcji T₂ na T₁ w odniesieniu do drugiego zasobu. Łuk (T₂,T₄) reprezentuje oczekiwanie transakcji T₂ na T₄ w odniesieniu do drugiego zasobu. Podobnie, łuki T₃T₄ i T₃T₁. Łuk T₃T₂ reprezentuje sytuację, gdy obie transakcje T₃ i T₂ znajdują się kolejce transakcji oczekujących na uwolnienie zasobu, ale ich blokady są niekompatybilne. Zauważmy, że w przykładowym grafie występuje cykl obejmujący wierzchołki T₁ i T₂.



Problem „duchów” (1)

Zakładaliśmy dotychczas, że baza danych jest zbiorem stałych i niezależnych obiektów. Rodzi to pewien problem

```
T1: select * from emp  
      where eyes="blue" and hair="red";  
T2: insert into emp  
      where eyes="blue" and hair="red";
```

Zakładając, że blokowania jest realizowane na poziomie rekordów bazy danych, pojawia się niebezpieczeństwo nieuszeregowalności realizacji: Zauważmy, że transakcja T1 nie widzi rekordu wprowadzanego przez transakcję T2. W odniesieniu do innej relacji, kolejność operacji transakcji T1 i T2 może być odwrotna

BD – wykład 9 (22)

Przedstawimy obecnie problem duchów, który jest konsekwencją przyjęcia określonej jednostki blokowania. Zakładaliśmy dotychczas, że baza danych jest zbiorem stałych i niezależnych obiektów. Rodzi to pewien problem.

Rozważmy transakcje T1 i T2 przedstawione na slajdzie. Założymy, że transakcje T1 i T2 są wykonywane sekwencyjnie: najpierw T1, a później T2. Zakładając, że blokowanie jest realizowane na poziomie rekordów bazy danych, pojawia się niebezpieczeństwo nieuszeregowalności realizacji. Zauważmy, że transakcja T1 nie widzi rekordu wprowadzanego przez transakcję T2. Stąd, transakcja T1 nie mogła założyć blokady odczytu tego rekordu. W odniesieniu do innej relacji, kolejność operacji transakcji T1 i T2 może być odwrotna. W konsekwencji, współbieżna realizacja transakcji T1 i T2 będzie nieuszeregowalna. W grafie uszeregowalności transakcja T1 będzie poprzedzała transakcję T2 w odniesieniu do relacji emp przedstawionej na slajdzie, natomiast transakcja T2 będzie poprzedzała T1 w odniesieniu do drugiej relacji. Łatwo zauważyc, że w grafie uszeregowalności wystąpi cykl świadczący o nieuszeregowalności realizacji.



Problem „duchów” (2)

- W jaki sposób zapobiec, aby transakcja T2 nie wprowadzała nowych rekordów do relacji *emp* w trakcie realizacji transakcji T1?
- Łatwo zauważyc, że współbieżne wykonanie transakcji T1 i T2 może być nieuszeregowalne
- Takie nowowprowadzane lub usuwane rekordy nazywamy **duchami**
- Nie istnieje żaden mechanizm blokowania, realizowany na poziomie blokad rekordów, który gwarantowałby rozwiązanie problemu „duchów”
- Rozwiązanie problemu „duchów” wymaga wprowadzenia blokad hierarchicznych

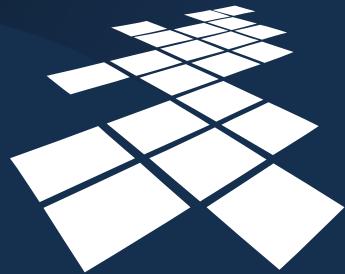
BD – wykład 9 (23)

Takie nowowprowadzane lub usuwane rekordy nazywamy „duchami”.

W jaki sposób zapobiec, aby transakcja T2 nie wprowadzała „duchów” do relacji *emp* w trakcie realizacji transakcji T1? Jak pokazaliśmy na poprzednim slajdzie pojawienie się rekordów duchów może prowadzić do nieuszeregowalności transakcji. Okazuje się, że nie istnieje żaden mechanizm blokowania, realizowany na poziomie blokad rekordów, który gwarantowałby rozwiązanie problemu „duchów”. Rozwiązanie problemu „duchów” wymaga wprowadzenia blokad hierarchicznych.

Algorytmy zarządzania współbieżnym wykonywaniem transakcji część II

**Wykład przygotował:
Tadeusz Morzy**



**UCZELNIA
ONLINE**

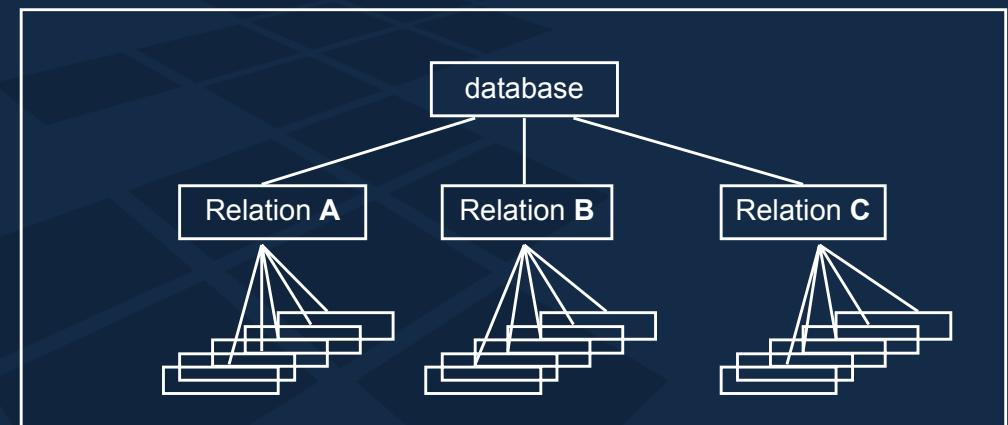
BD – wykład 9

Kontynuujemy prezentację i omówienie algorytmów zarządzania współbieżnym wykonywaniem transakcji. Przedstawimy hierarchiczny algorytm blokowania dwufazowego, będący podstawowym algorytmem zarządzania współbieżnym wykonywaniem transakcji w komercyjnych systemach baz danych, a następnie, omówimy algorytm porządkowania transakcji według etykiet czasowych i algorytm optymistyczny.



Ziarnistość blokad

• Hierarchia ziarnistości danych



BD – wykład 9 (2)

Jak już wspomnieliśmy na zakończenie poprzedniego wykładu, nie istnieje żaden mechanizm blokowania, realizowany na poziomie blokad rekordów, który gwarantowałby rozwiązanie problemu rekordów „duchów”. Rozwiązanie tego problemu wymaga wprowadzenia mechanizmu, który pozwalałby zakładać blokady na różnych poziomach organizacji danych – innymi słowy, pozwalałby zakładać blokady o różnej ziarnistości (baza danych, relacja, strona, rekord). Z drugiej strony może to rodzić pewien problem. Założymy, że dana jest relacja *Pracownicy* i że założyliśmy blokadę dla odczytu rekordu opisującego pracownika o nazwisku „Dziandziak”. Z drugiej strony, inny użytkownik postanowił założyć blokadę do zapisu całej relacji *Pracownicy*, gdyż zamierza uaktualnić adres zamieszkania pracownika o nazwisku „Nowak”. Czy te dwie blokady są kompatybilne czy też nie? Rozwiązaniem tego problemu jest połączenie mechanizmu blokad fizycznych, zakładanych na poziomie rekordów (krotek) bazy danych, z blokadami intencyjnymi, nazywanymi również blokadami predykatorowymi, zakładanymi na poziomie bazy danych i relacji. Algorytm, który łączy te dwa typy blokad nazywamy hierarchicznym algorytmem blokowania dwufazowego. Wykorzystuje on immanentną cechę organizacji bazy danych jaką jest hierarchia ziarnistości danych. Zauważmy, że baza danych posiada strukturę drzewa. Korzeniem tej struktury jest baza danych, która składa się ze zbioru relacji, które, z kolei, składają się ze zbioru krotek.

Hierarchiczna struktura bazy danych pozwala na stosunkowo efektywną implementację hierarchicznego algorytmu blokowania.



Jednostka blokowania

- Wybór jednostki blokowania jest kompromisem między stopniem współbieżności systemu, a narzutem systemowym
 - Współbieżność pracy systemu rośnie wraz ze zwiększaniem precyzyji blokowania (zmniejszaniem liczby zablokowanych danych i zwiększeniem liczby blokad)
 - Precyzyjne blokowanie jest kosztowne dla złożonych transakcji, które wymagają długiego czasu utrzymywania dużej liczby blokad
 - Blokowanie dużych jednostek danych wspiera złożone transakcje o dużej liczbie operacji, kosztem prostych transakcji o niewielkiej liczbie operacji
- **Konkluzja:** potrzebny jest protokół, który będzie wspierał obydwa rodzaje transakcji, to znaczy taki, który będzie umożliwiał równoczesne zakładanie blokad na różnych jednostkach danych

BD – wykład 9 (3)

Jeszcze innym problemem związanym z wyborem jednostki blokowania jest problemem efektywności działania systemu. Podstawową miarą oceny działania systemu bazy danych jest przepustowość systemu mierzona liczbą transakcji na sekundę. Rozważmy dwa przykładowe scenariusze wykonywania transakcji na relacji Pracownicy. Założmy, że relacja Pracownicy zawiera 50 tys. krotek. Dana jest transakcja, która aktualizuje zarobki wszystkich pracowników o 1%. Ta transakcja odczytuje wszystkie krotki relacji Pracownicy, a następnie, aktualizuje wartość atrybutu zarobki dla każdej krotki. Oznacza to konieczność założenia 50 tys. blokad do odczytu, które, następnie, są konwertowane do blokad do zapisu (50 tys. konwersji blokad). Po wykonaniu transakcji, system wykonuje 50 tys. operacji odblokowania. Gdyby jednostką blokowania była cała relacja, wówczas wykonanie transakcji wymagałoby założenia jednej blokady na całej relacji, jednej operacji konwersji blokady, i jednej operacji zdjęcia blokady. Z drugiej strony, rozważmy współbieżne wykonanie dwóch transakcji, z których pierwsza odczytuje krotkę r1 relacji Pracownicy, natomiast druga aktualizuje krotkę r2 tej relacji. Gdyby obie transakcje zakładały blokady na poziomie całej relacji, to, oczywiście, wystąpiłby konflikt blokad pomiędzy transakcjami, co pociągnęłoby konieczność zawieszenia wykonywania jednej transakcji do czasu zdjęcia blokady przez drugą transakcję. Z punktu widzenia tego scenariusza, lepszym rozwiązaniem byłoby, przyjęcie jako jednostki blokowania, pojedynczej krotki bazy danych. Wybór jednostki blokowania jest zatem kompromisem między stopniem współbieżności systemu, a narzutem systemowym związanym z implementacją algorytmu blokowania:

- przepustowość systemu rośnie wraz ze zwiększaniem precyzyji blokowania (zmniejszaniem liczby zablokowanych danych i zwiększeniem liczby blokad),
- precyzyjne blokowanie jest kosztowne dla złożonych transakcji, które wymagają długiego czasu utrzymywania dużej liczby blokad,
- blokowanie dużych jednostek danych wspiera złożone transakcje o dużej liczbie operacji, kosztem prostych transakcji o niewielkiej liczbie operacji.

Reasumując, potrzebny jest protokół, który będzie wspierał obydwa typy transakcji, to znaczy taki, który będzie umożliwiał równoczesne zakładanie blokad na różnych jednostkach danych.



Blokady intencyjne (1)

- **Idea hierarchicznego protokołu blokowania:**

Zablokowanie pośredniego węzła w hierarchii ziarnistości danych jest równoznaczne z pośrednim zablokowaniem wszystkich węzłów potomnych

Wymaga ono jednak wcześniejszego uzyskania blokad na wszystkich węzłach rodzicielskich

BD – wykład 9 (4)

Idea hierarchicznego algorytmu blokowania dwufazowego jest następująca. Zablokowanie pośredniego węzła w hierarchii ziarnistości danych jest równoznaczne z pośrednim zablokowaniem wszystkich węzłów potomnych. Wymaga ono jednak wcześniejszego uzyskania blokad na wszystkich węzłach rodzicielskich.



Blokady intencyjne (2)

- ***IR*** - transakcja zamierza uzyskiwać blokady typu ***R*** na poszczególnych obiektach składowych
- ***IW*** - transakcja zamierza uzyskiwać blokady typu ***W*** na poszczególnych obiektach składowych
- ***RIW*** - transakcja blokuje wszystkie obiekty składowe w blokadą typu ***R***, a na niektórych zamierza uzyskiwać blokady typu ***W***

Blokada uzyskana \ Blokada żądana	<i>IR</i>	<i>IW</i>	<i>R</i>	<i>RIW</i>	<i>W</i>
<i>IR</i>	✓	✓	✓	✓	-
<i>IW</i>	✓	✓	-	-	-
<i>R</i>	✓	-	✓	-	-
<i>RIW</i>	✓	-	-	-	-
<i>W</i>	-	-	-	-	-

BD – wykład 9 (5)

Hierarchiczny algorytm blokowania dwufazowego wprowadza dwa rodzaje blokad: blokady fizyczne oraz blokady intencyjne. Wyróżniamy trzy blokady intencyjne:

IR - transakcja zamierza uzyskać blokady typu ***R*** na poszczególnych obiektach składowych;

IW - transakcja zamierza uzyskać blokady typu ***W*** na poszczególnych obiektach składowych;

RIW - transakcja blokuje wszystkie obiekty składowe blokadą typu ***R***, a na niektórych elementach składowych zamierza uzyskać blokady typu ***W***. (blokada wspierająca aktualizację).

Znaczenie blokad fizycznych pozostaje bez zmian. Macierz kompatybilności rozszerzonego zbioru blokad przedstawiono na slajdzie.



Reguły blokowania intencyjnego

- Żądanie uzyskania blokady **W** na danym obiekcie jest pośrednim żądaniem ustawienia blokad **W** na wszystkich obiektach składowych
- Żądanie uzyskania blokady **R** lub **RIW** na danym obiekcie jest pośrednim żądaniem ustawienia blokad **R** na wszystkich obiektach składowych
- Zanim transakcja uzyska blokadę typu **IR** lub **R** na danym obiekcie, musi uzyskać blokadę typu **IR** na co najmniej jednym zawierającym go obiekcie wyższego poziomu
- Zanim transakcja uzyska blokadę typu **IW**, **RIW** lub **W** na danym obiekcie, musi uzyskać blokadę typu **IW** na wszystkich zawierających go obiektach wyższego poziomu
- Zanim transakcja zwolni blokadę na danym obiekcie musi wpierw zwolnić wszystkie blokady z obiektów składowych

BD – wykład 9 (6)

Reguły blokowania hierarchicznego algorytmu 2PL można zdefiniować następująco:

Żądanie uzyskania blokady **W** na danym obiekcie jest pośrednim żądaniem ustawienia blokad **W** na wszystkich obiektach składowych.

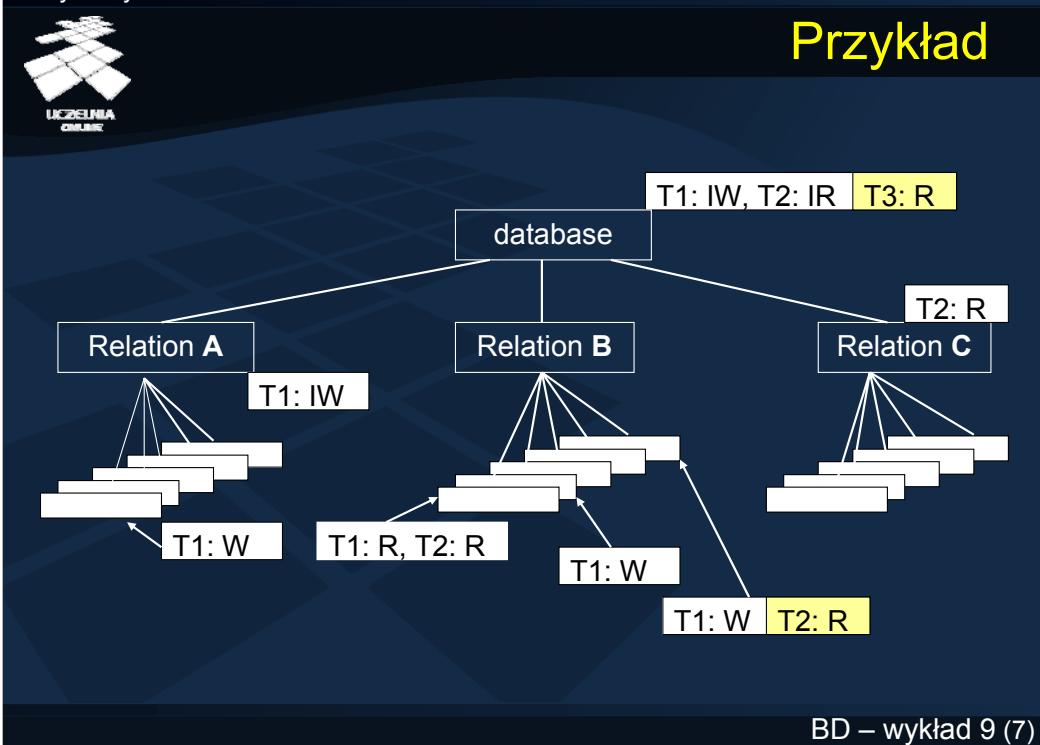
Żądanie uzyskania blokady **R** lub **RIW** na danym obiekcie jest pośrednim żądaniem ustawienia blokad **R** na wszystkich obiektach składowych.

Zanim transakcja uzyska blokadę typu **IR** lub **R** na danym obiekcie, musi uzyskać blokadę typu **IR** na co najmniej jednym zawierającym go obiekcie wyższego poziomu.

Zanim transakcja uzyska blokadę typu **IW**, **RIW** lub **W** na danym obiekcie, musi uzyskać blokadę typu **IW** na wszystkich zawierających go obiektach wyższego poziomu.

Zanim transakcja zwolni blokadę na danym obiekcie musi wpierw zwolnić wszystkie blokady z obiektów składowych.

Przykład



BD – wykład 9 (7)

Dla ilustracji działania hierarchicznego algorytmu 2PL rozważmy prosty przykład przedstawiony na slajdzie. Dane są trzy transakcje T1, T2 i T3. Transakcja T1 zapisuje pierwszą krotkę relacji A, odczytuje pierwszą krotkę relacji B, i aktualizuje drugą i ostatnią krotkę relacji B. Transakcja T2 jest zapytaniem, które odczytuje pierwszą i ostatnią krotkę relacji B, oraz wszystkie krotki relacji C. Transakcja T3 odczytuje wszystkie relacje A, B i C. Zgodnie z algorymem hierarchicznego blokowania, transakcja T1 zakłada blokadę intencyjną zapisu IW na bazie danych, oraz intencyjne blokady do zapisu IW na relacjach A i B. Następnie, zakłada fizyczna blokadę W na zapisywanej krotce relacji A, blokadę R na pierwszej krotce relacji B oraz blokadę W na drugiej i ostatniej krotce relacji B. Transakcja T2 zakłada blokadę intencyjną do odczytu IR na bazie danych. Blokady IW i IR są kompatybilne. Następnie, transakcja T2 zakłada blokadę R na pierwszej krotce relacji B oraz próbuje założyć blokadę R na ostatniej krotce relacji B. Ponieważ krotka ta jest zablokowana do zapisu przez T1, operacja transakcji T2 założenia blokady do odczytu tej krotki zostaje umieszczona w kolejce transakcji oczekujących na uwolnienie blokady tej krotki. Ponieważ transakcja T2 odczytuje wszystkie krotki relacji C, zakłada fizyczną R całej relacji C.

Transakcja T3 zamierza odczytać wszystkie relacje A, B i C, stąd zakłada fizyczną R na całej bazie danych. Blokada fizyczna R jest niekompatybilna z blokadą intencyjną do zapisu IW, stąd żądanie transakcji T3 założenia blokady R nie może być spełnione.



Tryb blokady aktualizacji (1)

- Blokada aktualizacji (update mode) została wprowadzona w celu minimalizacji prawdopodobieństwa wystąpienia zakleszczeń

```
update emp  
set salary = salary * 1.1  
where name = "Morzy";
```

- Aktualizacja jest wykonywana następująco: blokada dla odczytu jest zakładana na wszystkich rekordach odczytywanych przez zapytanie (rekordy relacji *emp*), a następnie, na wszystkich rekordach, które spełniają predykat aktualizacji, jest zakładana blokada dla zapisu

BD – wykład 9 (8)

W późniejszym czasie, zbiór typów blokad fizycznych został poszerzony o blokadę aktualizacji (update mode). Blokada aktualizacji została wprowadzona w celu minimalizacji prawdopodobieństwa wystąpienia zakleszczeń. Rozważmy następującą transakcję aktualizacji, która podnosi zarobki pracownikowi „Morzy” o 10%.

```
update emp  
set salary=salary*1.1  
where name="Morzy";
```



Tryb blokady aktualizacji (2)

- Problem **hotspots** – rekordów często aktualizowanych
- Analiza systemu R pokazała, że większość zakleszczeń pojawia się w wyniku wykonywania aktualizacji na danych typu **hotspots**
- Wprowadzenie blokady aktualizacji minimalizuje liczbę zakleszczeń
- Blokada aktualizacji jest asymetryczna

BD – wykład 9 (9)

Realizacja tej transakcji, zgodnie z hierarchicznym algorytmem 2PL, przebiega następująco. Na relacji *Emp* jest zakładana blokada intencyjna RIW, następnie, na wszystkich rekordach odczytywanych przez transakcję jest zakładana blokada do odczytu, a następnie, na wszystkich rekordach, które spełniają predykat aktualizacji, jest zakładana blokada do zapisu. Jeżeli, współbieżnie, inna transakcja aktualizuje zarobki pracownika „Dziandziak”, istnieje duże prawdopodobieństwo wystąpienia zakleszczenia. Prawdopodobieństwo wystąpienia zakleszczenia rośnie dla rekordów często aktualizowanych. Takie rekordy w bazie danych nazywamy „hotspots”. Jak pokazała analiza działania systemu R, prototypowej wersji systemu DB2, większość zakleszczeń pojawia się w wyniku wykonywania aktualizacji na danych typu hotspots. W celu rozwiązania tego problemu wprowadzono blokadę aktualizacji. Cechą charakterystyczną tej blokady jest jej asymetria w stosunku do blokady R. Jeżeli transakcja T1 założyła blokadę do odczytu danej X, to inna transakcja T2 może założyć blokadę aktualizacji U tej danej. Jeżeli natomiast, transakcja T2 założyła blokadę U danej Y, to transakcja T1 nie może założyć już blokady R danej Y.



Tryb blokady aktualizacji (3)

- Pełna macierz kompatybilności blokad ma następującą postać:

Blokada żądana	Blokada uzyskana	IR	IW	R	RIW	U	W
IR		✓	✓	✓	✓	-	-
IW		✓	✓	-	-	-	-
R		✓	-	✓	-	-	-
RIW		✓	-	-	-	-	-
U		-	-	✓	-	-	-
W		-	-	-	-	-	-

BD – wykład 9 (10)

Pełną macierz kompatybilności blokad, z uwzględnieniem blokady aktualizacji, przedstawiono na slajdzie.



Algorytmy znaczników czasowych (ang. timestamp ordering)

- **Znacznik czasowy** (ang. *timestamp*) transakcji T - $TS(T)$, jest jej unikalnym identyfikatorem. Znaczniki są przydzielone transakcjom w kolejności, w której transakcje pojawiają się w systemie zarządzania bazą danych
- Z każdą daną (x) w bazie danych związane są dwie wartości znaczników czasowych:
 - **Read_TS(x)** - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie odczytały tę daną
 - **Write_TS(x)** - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie zapisały tę daną

BD – wykład 9 (11)

Przejdziemy obecnie do przedstawienia drugiej ważnej grupy algorytmów zarządzania współbieżnym wykonywaniem transakcji, a mianowicie, algorytmów znaczników czasowych (algorytmy T/O) (inna popularna nazwa tej grupy algorytmów to – algorytmy porządkowania transakcji wg. etykiet czasowych). W algorytmach blokowania, uszeregowanie transakcji będących w konflikcie wynika z kolejności, w jakiej te transakcje uzyskały blokady danych. Mechanizm dwufazowości zakładania blokad gwarantuje uszeregowalność realizacji transakcji. W przypadku algorytmów znaczników czasowych, uszeregowanie transakcji będących w konflikcie wynika z porządku znaczników czasowych przydzielanych transakcjom w momencie ich inicjacji. Porządek jest więc predefiniowany, a nie ustalany dynamicznie, jak w przypadku algorytmów blokowania.

Znacznik czasowy (ang. *timestamp*) transakcji T ($TS(T)$), jest unikalnym identyfikatorem transakcji. Znaczniki są przydzielone transakcjom w kolejności, w której transakcje pojawiają się w systemie bazy danych.

Z każdą daną (x) w bazie danych związane są dwie wartości znaczników czasowych:
Read_TS(x) - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie odczytały tę daną.

Write_TS(x) - największy znacznik czasowy spośród wszystkich transakcji, które pomyślnie zapisały tę daną.



Implementacja algorytmu znaczników czasowych (1)

```
Read(Ti, x) begin
    if (TS(Ti) < Write_TS(x)) then
        < abort Ti and restart it with
        a new timestamp >;
    else begin
        < read x >;
        Read_TS(x) = max ( Read_TS(x) , TS(Ti) );
    end;
end Read;
```

BD – wykład 9 (12)

Implementacja podstawowego algorytmu znaczników czasowych składa się z dwóch procedur: procedury odczytu danej i procedury zapisu danej. Prezentowany slajd przedstawia procedurę odczytu danej x przez transakcję T_i . Jeżeli znacznik czasowy transakcji T_i ($TS(T_i)$) jest mniejszy od znacznika czasowego zapisu danej x ($Write_TS(x)$), wówczas transakcja T_i jest wycofywana i restartowana ponownie z nowym, późniejszym, znacznikiem czasowym. W przeciwnym razie, transakcja odczytuje wartość danej x i, ewentualnie, aktualnia znacznik czasowy odczytu danej x ($Read_TS(x)$).



Implementacja algorytmu znaczników czasowych (2)

```
Write(Ti, x) begin
    if (TS(Ti) < Read_TS(x) or
        TS(Ti) < Write_TS(x)) then
        < abort Ti and restart it with
        a new timestamp >;
    else begin
        < write x >;
        Write_TS(x) TS(Ti);
    end;
end Write;
```

BD – wykład 9 (13)

Kolejny slajd przedstawia procedurę zapisu danej x przez transakcję T_i . Jeżeli znacznik czasowy transakcji T_i ($TS(T_i)$) jest mniejszy od znacznika czasowego zapisu danej x ($Write_TS(x)$) lub znacznika czasowego odczytu danej x ($Read_TS(x)$), wówczas transakcja T_i jest wycofywana i restartowana ponownie z nowym, późniejszym, znacznikiem czasowym. W przeciwnym razie, transakcja zapisuje nową wartość danej x i aktualizuje jej znacznik czasowy zapisu ($Write_TS(x)$).



Algorytm znaczników czasowych

- Algorytm T/O, w swojej podstawowej wersji, jest wolny od zakleszczeń

S: T1: r(x) T2: r(y) T1: w(y) T2: w(x) T1: c T2: c

- Algorytm T/O, w swojej podstawowej wersji, nie zapewnia odtwarzalności realizacji!!!

S: T1: w(x) T2: r(x) T2: w(x) T2: c T1: abort

BD – wykład 9 (14)

Algorytm znaczników czasowych, w swojej podstawowej wersji, jest wolny od zakleszczeń. Rozważmy realizację transakcji T1 i T2 przedstawioną na slajdzie.

S: T1: r(x) T2: r(y) T1: w(y) T2: w(x) T1: c T2: c

W przypadku algorytmu blokowania, przedstawiona realizacja prowadzi do wystąpienia zakleszczenia. W przypadku algorytmu znaczników czasowych, zakładając, że $TS(T1) < TS(T2)$, transakcja T1 zostanie wycofana i restartowana ponownie (na skutek odrzucenia operacji zapisu $T1: w(y)$). Zasadniczą wadą algorytmu jest to, że, w swojej podstawowej wersji, algorytm nie zapewnia odtwarzalności realizacji!!! Rozważmy realizację transakcji T1 i T2 przedstawioną na slajdzie:

S: T1: w(x) T2: r(x) T2: w(x) T2: c T1: abort

Jak łatwo zauważyc, przedstawiona realizacja jest nie odtwarzalna, gdyż transakcja T2 odczytała stan danej x zapisany przez transakcję T1, która została wycofana.



Algorytm znaczników czasowych z buforowaniem operacji

- Aby zapewnić odtwarzalność realizacji generowanych przez algorytm T/O konieczna jest modyfikacja algorytmu polegająca na buforowaniu operacji odczytu i zapisu danych, aż do momentu zatwierdzenia transakcji
- Transakcja T1 zapisuje daną x: aktualizowana jest wartość znacznika Write_TS(x), zmiana wartości danej x jest odsunięta w czasie do momentu zatwierdzenia transakcji T1
- Transakcja T2 odczytuje daną x aktualizowaną przez transakcję T1: znacznik transakcji TS(T2) jest porównywany ze znacznikiem Write_TS(x), jeżeli warunek odczytu jest spełniony, to odczyt jest odsunięty w czasie do momentu akceptacji transakcji T1, w przeciwnym razie transakcja T2 jest wycofywana

Efekt buforowania jest podobny do efektu zakładania blokad dla zapisu na danych !!!

BD – wykład 9 (15)

Aby zapewnić odtwarzalność realizacji generowanych przez algorytm znaczników czasowych konieczna jest modyfikacja algorytmu polegająca na buforowaniu operacji odczytu i zapisu danych, aż do momentu zatwierdzenia transakcji.

Transakcja T1 zapisuje daną x: aktualizowana jest wartość znacznika Write_TS(x), zmiana wartości danej x jest odsunięta w czasie do momentu zatwierdzenia transakcji T1.

Transakcja T2 odczytuje daną x aktualizowaną przez transakcję T1: znacznik transakcji TS(T2) jest porównywany ze znacznikiem Write_TS(x), jeżeli warunek odczytu jest spełniony, to odczyt jest odsunięty w czasie do momentu akceptacji transakcji T1, w przeciwnym razie transakcja T2 jest wycofywana.

Efekt buforowania jest podobny do efektu zakładania blokad dla zapisu na danych !!!



Algorytmy optymistyczne (1)

- **Algorytmy optymistyczne** zarządzania współbieżnością transakcji przeprowadzają transakcje przez trzy stany:
 - **Faza odczytu:** Transakcje czytają dane z bazy danych. Wprowadzane modyfikacje są przechowywane w lokalnych obszarach roboczych transakcji
 - **Faza walidacji:** Wykonywana jest walidacja uszeregowalności transakcji. Transakcje niespełniające kryterium uszeregowalności są wycofywane i restartowane
 - **Faza zapisu:** Jeżeli faza walidacji zakończy się pomyślnie, modyfikacje transakcji są wprowadzane do bazy danych

BD – wykład 9 (16)

Na zakończenie prezentacji algorytmów zarządzania współbieżnym wykonywaniem transakcji, przedstawimy, krótko, ostatnią z wymienionych wcześniej metod – metodę optymistyczną.

Algorytmy blokowania nazywane są często algorytmami pesymistycznymi zarządzania współbieżnym wykonywaniem transakcji, gdyż zakładają one pesymistyczny scenariusz konfliktów pomiędzy transakcjami – wystąpienie konfliktu jest traktowane jako potencjalne źródło nieuszeregowalności realizacji. Stąd, konflikt pomiędzy transakcjami jest rozwiązywany poprzez blokady lub wycofanie transakcji. Jednakże, w systemach, w których obciążenia danych są niskie, prawdopodobieństwo, że wystąpi konflikt pomiędzy współbieżnie wykonywanymi transakcjami jest niewielkie, a jeszcze mniejsze jest prawdopodobieństwo, że konflikt ten spowoduje nieuszeregowalność realizacji. Z drugiej strony narzuca systemowy związek z mechanizmem zakładania i zdejmowania blokad jest wysoki. Podstawowym założeniem metod optymistycznych jest założenie, że konflikty pomiędzy transakcjami występują stosunkowo rzadko (stąd nazwa metody), stąd algorytm powinien być maksymalnie mało restryktywny jeżeli chodzi o ograniczenie dostępu do danych dla współbieżnie wykonywanych transakcji.

Wykonywanie transakcji w metodzie optymistycznej przebiega w trzech fazach:

Fazie odczytu: Transakcje czytają dane z bazy danych. Wprowadzane modyfikacje są przechowywane w lokalnych obszarach roboczych transakcji.

Fazie walidacji: Wykonywana jest walidacja uszeregowalności transakcji. Transakcje nie spełniające kryterium uszeregowalności są wycofywane i restartowane.

Fazie zapisu: Jeżeli faza walidacji zakończy się pomyślnie, modyfikacje transakcji są wprowadzane do bazy danych.



Algorytmy optymistyczne (2)

- Z każdą transakcją T_i związane są czasy rozpoczęcia poszczególnych faz transakcji oraz utrzymywane są zbiory danych odczytywanych R_i i modyfikowanych W_i
- W fazie walidacji transakcji T_i , dla każdej transakcji T_j , która została zatwierdzona lub znajduje się w fazie walidacji sprawdzane jest czy spełniony jest jeden z warunków:
 1. Transakcja T_j zakończyła swoją fazę zapisu zanim transakcja T_i rozpoczęła swoją fazę odczytu
 2. $(R_i \cup W_i) \cap W_j = \emptyset$ i T_i zakończyła swoją fazę odczytu zanim T_j zakończyła swoją fazę odczytu

BD – wykład 9 (17)

Z każdą transakcją T_i związane są czasy rozpoczęcia poszczególnych faz transakcji oraz utrzymywane są zbiory danych odczytywanych R_i i modyfikowanych W_i .

W fazie walidacji transakcji T_i , dla każdej transakcji T_j , która została zatwierdzona lub znajduje się w fazie walidacji sprawdzane jest czy spełniony jest jeden z warunków:

1. Warunek pierwszy: transakcja T_j zakończyła swoją fazę zapisu zanim transakcja T_i rozpoczęła swoją fazę odczytu.
2. Warunek drugi: wyrażenie nr 2 ze slajdu daje w wyniku zbiór pusty ($(R_i \text{ SUMA } W_i) \text{ ILOCZYN } W_j = \text{ZBIÓR PUSTY}$) i T_i zakończyła swoją fazę odczytu zanim T_j zakończyła swoją fazę odczytu.

Jeżeli spełniony jest którykolwiek z powyższych warunków, transakcja T_i jest zatwierdzana i przechodzi do fazy zapisu. W przeciwnym przypadku, transakcja T_i jest wycofywana i restartowana ponownie. Łatwo zauważyc, że test walidacji jest stosunkowo prosty i może być efektywnie zaimplementowany. Algorytm optymistyczny został zastosowany jako mechanizm zarządzania współbieżnością, dla specyficznych danych, w systemie DB2.



Poziomy izolacji (1)

- Z każdą transakcją związane są trzy parametry systemowe: **diagnostics_size**, **access_mode**, i **isolation_level**

Diagnostics_size:

Parametr "diagnostics_size" określa liczbę opisów błędów, które system zapamiętuje dla danej transakcji

Access_mode:

**Transakcja może występować w dwóch trybach:
READ ONLY i READ WRITE**

BD – wykład 9 (18)

Paradoksalnie, komercyjne systemy zarządzania bazami danych jak i standard SQL, nie zapewniają, automatycznie, uszeregowalności realizacji transakcji. Wprowadzają one koncepcję tak zwanych poziomów izolacji. Zanim przedstawimy koncepcję poziomów izolacji transakcji przedstawioną w standardzie SQL, krótko omówimy inne elementy specyfikacji transakcji w tym standardzie.

Z każdą transakcją, zgodnie ze standardem SQL, związane są trzy parametry systemowe: diagnostics_size, access_mode, i isolation_level.

Parametr "diagnostics_size" określa liczbę opisów błędów, które system zapamiętuje dla danej transakcji. Parametr „access_mode” określa tryb wykonywania transakcji. Transakcja może występować w dwóch trybach: READ ONLY i READ WRITE.



Poziomy izolacji (2)

- Jeżeli transakcja występuje w trybie READ ONLY oznacza to, że transakcja jest zapytaniem i nie może modyfikować bazy danych. Dla transakcji w trybie READ ONLY, system zakłada wyłącznie blokady dla odczytu - prowadzi to do zwiększenia stopnia współbieżności.
- Domyślnie, trybem wykonywania transakcji jest tryb READ WRITE, który pozwala na wykonywanie wszystkich typów operacji na bazie danych

BD – wykład 9 (19)

Jeżeli transakcja występuje w trybie READ ONLY oznacza to, że transakcja jest zapytaniem i nie może modyfikować bazy danych. Dla transakcji w trybie READ ONLY, system zakłada wyłącznie blokady dla odczytu - prowadzi to do zwiększenia stopnia współbieżności.

Domyślnie, trybem wykonywania transakcji jest tryb READ WRITE, który pozwala na wykonywanie wszystkich typów operacji na bazie danych.



Poziomy izolacji (3)

Większość systemów zarządzania bazami danych nie zapewnia automatycznie uszeregowalności realizacji!!

Celem prowadzenia tak zwanych poziomów izolacji jest znalezienie kompromisu pomiędzy poprawnością wykonywania transakcji a zapewnieniem jak największego stopnia współbieżności wykonywanych transakcji.

Standard SQL-92 definiuje cztery poziomy izolacji:

- **READ UNCOMMITTED**
- **READ COMMITTED**
- **REPEATABLE READ**
- **SERIALIZABLE**

BD – wykład 9 (20)

Jak już wspomnieliśmy, systemy zarządzania bazami danych nie zapewniają automatycznie uszeregowalności realizacji!! Wprowadzają one koncepcję tak zwanych poziomów izolacji transakcji.

Celem wprowadzenia poziomów izolacji transakcji jest znalezienie kompromisu pomiędzy poprawnością wykonywania transakcji a zapewnieniem jak największego stopnia współbieżności wykonywanych transakcji.

Standard SQL-92 definiuje cztery poziomy izolacji:

- **READ UNCOMMITTED** (poziom izolacji 0)
- **READ COMMITTED** (poziom izolacji 1)
- **REPEATABLE READ** (poziom izolacji 2)
- **SERIALIZABLE** (poziom izolacji 3)

Jedynie poziom izolacji 3 zapewnia pełną uszeregowalność realizacji transakcji. W kolejnych slajdach, krótko omówimy poszczególne poziomy izolacji.



Poziomy izolacji (4)

- **SERIALIZABLE**

ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, odczytywanej lub zapisywanej przez T , nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T . Ten poziom izolacji gwarantuje uszeregowalność wszystkich realizacji

BD – wykład 9 (21)

Poziom izolacji 3 - SERIALIZABLE. Ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, odczytywanej lub zapisywanej przez T , nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T . Ten poziom izolacji gwarantuje uszeregowalność wszystkich realizacji transakcji T .



Poziomy izolacji (5)

- **REPEATABLE READ**

ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, odczytywanej lub zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Niestety, ten poziom izolacji nie gwarantuje rozwiązania problemu „duchów” i, stąd, nie gwarantuje uszeregowalności wszystkich realizacji

BD – wykład 9 (22)

Poziom izolacji 2 - REPEATABLE READ. Ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, odczytywanej lub zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Niestety, ten poziom izolacji nie gwarantuje rozwiązania problemu rekordów „duchów” i, stąd, nie gwarantuje uszeregowalności wszystkich realizacji.



Poziomy izolacji (6)

- **READ COMMITTED** (ang. *cursor stability*)

ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Jednakże, poziom ten nie gwarantuje, że wartość danej odczytywanej przez transakcję T nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Ten poziom izolacji nie gwarantuje również rozwiązyania problemu „duchów” i, stąd, nie gwarantuje uszeregowalności wszystkich realizacji

BD – wykład 9 (23)

Poziom izolacji 1 - READ COMMITTED. Ten poziom izolacji gwarantuje, że każda transakcja T odczytuje dane utworzone wyłącznie przez zatwierdzone transakcje i że wartość żadnej danej, zapisywanej przez T, nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Jednakże, poziom ten nie gwarantuje, że wartość danej odczytywanej przez transakcję T nie zostanie zmieniona przez inną transakcję do momentu zakończenia transakcji T. Ten poziom izolacji nie gwarantuje również rozwiązyania problemu „duchów” i, stąd, nie gwarantuje uszeregowalności wszystkich realizacji



Poziomy izolacji (7)

- **READ UNCOMMITTED**

ten poziom izolacji dopuszcza odczyt przez transakcję T zmian dokonywanych przez aktywne jeszcze transakcje. Ten poziom izolacji jest dostępny tylko dla transakcji wykonywanych w trybie READ ONLY

BD – wykład 9 (24)

Poziom izolacji 0 - READ UNCOMMITTED. Ten poziom izolacji dopuszcza odczyt przez transakcję T zmian dokonywanych przez aktywne jeszcze transakcje. Ten poziom izolacji jest dostępny tylko dla transakcji wykonywanych w trybie READ ONLY



Poziomy izolacji (8)

- W standardzie SQL-92 poziom izolacji i tryb wykonywania transakcji definiuje się za pomocą komendy SET TRANSACTION.
- Przykładowa komenda deklaruje transakcję jako transakcję tylko do odczytu i realizację uszeregowalną

```
SET TRANSACTION ISOLATION LEVEL  
SERIALIZABLE READ ONLY
```

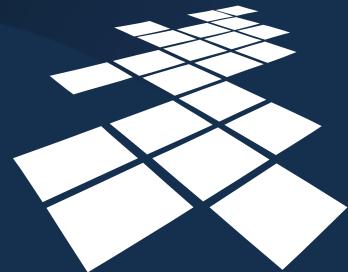
BD – wykład 9 (25)

W standardzie SQL-92 poziom izolacji i tryb wykonywania transakcji definiuje się za pomocą polecenia systemowego SET TRANSACTION.

Przykładowe polecenie, przedstawione na slajdzie, deklaruje transakcję jako transakcję tylko do odczytu i realizację uszeregowalną.

Recovery – Transakcyjne odtwarzanie bazy danych po awarii

Wykład przygotował:
Tadeusz Morzy



UCZELNIA
ONLINE

BD – wykład 11

Tematem wykładu jest problem odtwarzania spójnego stanu bazy danych po awarii. Rozpoczniemy od krótkiego wprowadzenia do modeli awarii. Następnie, przejdziemy do omówienia miary dostępności, jako podstawowej miary oceny efektywności mechanizmów odtwarzania, oraz do przedstawienia ogólnej architektury modułu odtwarzania. W kolejnej części, przedstawimy i omówimy generowanie rekordów logu oraz procedury Rollback oraz Roll Forward algorytmu odtwarzania spójnego stanu bazy danych. Przedyskutujemy problem poprawności przedstawionych procedur i omówimy strategię WAL, gwarantującą poprawność algorytmu odtwarzania. Na zakończenie, powiemy krótko o tak zwanych punktach kontrolnych pliku logu.



Cel odtwarzania

- Podstawowym **celem** mechanizmów transakcyjnego odtwarzania bazy danych po awarii jest odtworzenie spójnego stanu bazy danych
- **Definicja odtwarzania** – wiąże się bezpośrednio z problemem efektywnej implementacji własności atomowości i trwałości transakcji (własności ACID)

BD – wykład 11 (2)

Podstawowym celem algorytmów transakcyjnego odtwarzania bazy danych po awarii jest odtworzenie spójnego stanu bazy danych sprzed awarii. Pierwsze pytanie, które się nasuwa, to jaki stan spójny chcemy odtworzyć? W trakcie wykonywania setek czy tysiący transakcji, stan bazy danych, praktycznie, nigdy nie jest spójny! Odpowiedź na postawione pytanie wynika z definicji transakcji. Stan spójny bazy danych, który będzie odtworzony po awarii, to taki stan, który gwarantuje własności atomowości i trwałości wykonywanych transakcji (własności ACID). Do sformułowania poprawnego stanu bazy danych po awarii wróćmy później na wykładzie.



Modele awarii

- **Miękkie model awarii** – awaria nie niszczy danych w pamięci zewnętrznej
- **Twardy model awarii** - awaria niszczy dane w pamięci zewnętrznej
- „Niedeterministyczny” charakter błędów prowadzących do wystąpienia awarii systemu – błędy w oprogramowaniu systemowym (tzw. „Heisenbugs”)
- **Podejście „Fail-stop crash”:**
 - Wyłącz serwer
 - Odtwórz spójny stan bazy danych
 - Restartuj system

BD – wykład 11 (3)

Zanim przejdziemy do dyskusji i prezentacji procedury odtwarzania bazy danych po awarii, wprowadzimy klasyfikację modeli awarii. Klasyfikacji awarii jest wiele – ta wprowadzona dla potrzeb wykładu wyróżnia dwa podstawowe typy awarii – tak zwane awarie miękkie i awarie twarde. Awaria miękka to awaria, która nie niszczy danych w pamięci zewnętrznej. Innymi słowy, awaria miękka to awaria, która nie niszczy mediów zewnętrznych. Awaria twarda to awaria, która niszczy dane w pamięci zewnętrznej. W dalszej części wykładu zakładamy model awarii miękkich, dla których opracowano systemowe mechanizmy przeciwdziałania. W przypadku awarii twardych, typowe rozwiązania sprowadzają się do replikacji danych i oprogramowania oraz sprzętu. Do grupy awarii miękkich zaliczamy te awarie, które prowadzą do utraty danych w pamięci operacyjnej. Są to, głównie, błędy oprogramowania systemowego i aplikacyjnego, które prowadzą do błędego działania systemu, oraz zaniki napięcia, których skutki, z punktu wiedzenia działania systemu są podobne. Mówiąc o awariach, najczęściej, jako źródło awarii podaje się zanik napięcia. W praktyce ten rodzaj awarii występuje bardzo rzadko. Najczęstszym typem awarii są błędy w oprogramowaniu. Błędy te mają charakter niedeterministyczny. W przypadku, gdy występuje determinizm zdarzeń prowadzących do awarii, to ich źródło można usunąć. W przypadku błędów o charakterze niedeterministycznym mówimy często o tak zwanych błędach typu „Heisenbug” od nazwiska słynnego niemieckiego fizyka Wernera Heisenberga. Najlepszym sposobem rozwiązania problemów z błędami typu „Heisenbugs” jest podejście nazwane „fail_stop crash”. Polega ono na: wyłączeniu serwera, odtworzeniu poprawnego (spójnego) stanu systemu w oparciu o mechanizm redo/undo w celu zapewnienia własności ACID, i na restarcie serwera.



Efektywność odtwarzania

- Konieczność minimalizacji czasu odtwarzania po awarii - w trakcie odtwarzania serwer i dane są niedostępne
- Miara efektywności odtwarzania – dostępność systemu, tj. prawdopodobieństwo, że próbując losowo użytkownik znajduje system gotowy do realizacji żądań dostępu

$$\frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

MTTF – średni czas do awarii

MTTR – średni czas odtworzenia po awarii

BD – wykład 11 (4)

Mechanizm odtwarzania stanu spójnego powinien być poprawny, ale powinien być również efektywny. W trakcie odtwarzania systemu serwer i dane są niedostępne, stąd, konieczność minimalizacji czasu odtwarzania. Miarą efektywności procedury odtwarzania jest tak zwana dostępność systemu. Dostępność systemu definiuje się jako prawdopodobieństwo, że próbując losowo system użytkownik znajduje ten system gotowy do realizacji żądań dostępu.



Dostępność

- Przykład: raz w miesiącu ma miejsce awaria serwera, odtworzenie stanu serwera zajmuje 2 godz., czas niedostępności serwera – 26 godz./rocznie
$$\text{dostępność} = (720/722) = 99,7 \%$$
- Przykład: awaria serwera ma miejsce raz na 48 godz., odtworzenie stanu serwera zajmuje 30 sek.
$$\text{dostępność} = 99,98 \%$$
- Wniosek: efektywność odtwarzania ma kluczowe znaczenie z punktu widzenia dostępności systemu
- Inna miara efektywności odtwarzania: ilość zasobów niezbędnych, podczas normalnego działania systemu, do tego, aby w razie awarii poprawnie odtworzyć stan systemu

BD – wykład 11 (5)

Od czego zależy dostępność systemu? Łatwo zauważyc, że podstawowym czynnikiem determinującym dostępność systemu jest czas odtworzenia systemu po awarii. Dla ilustracji rozważmy dwa przypadki. Założmy, że awaria serwera ma miejsce raz w miesiącu, natomiast czas odtworzenia poprawnego stanu serwera zajmuje 2 godz. Czas niedostępności serwera wynosi 26 godz. rocznie. Stąd, dostępność systemu wynosi = 99,7 %. Rozważmy inny przypadek. Założmy, że awaria serwera ma miejsce raz na 48 godz., odtworzenie stanu serwera zajmuje 30 sek. W takim przypadku dostępność wynosi= 99,98 %. Wniosek: efektywność odtwarzania ma kluczowe znaczenie z punktu widzenia dostępności systemu. Czasami przyjmuje się dodatkowe kryteria oceny efektywności procedury odtwarzania. Przykładowo - ilość zasobów niezbędnych, podczas normalnego działania systemu, do tego, aby w razie awarii poprawnie odtworzyć stan systemu.



Architektura modułu odtwarzania (1)

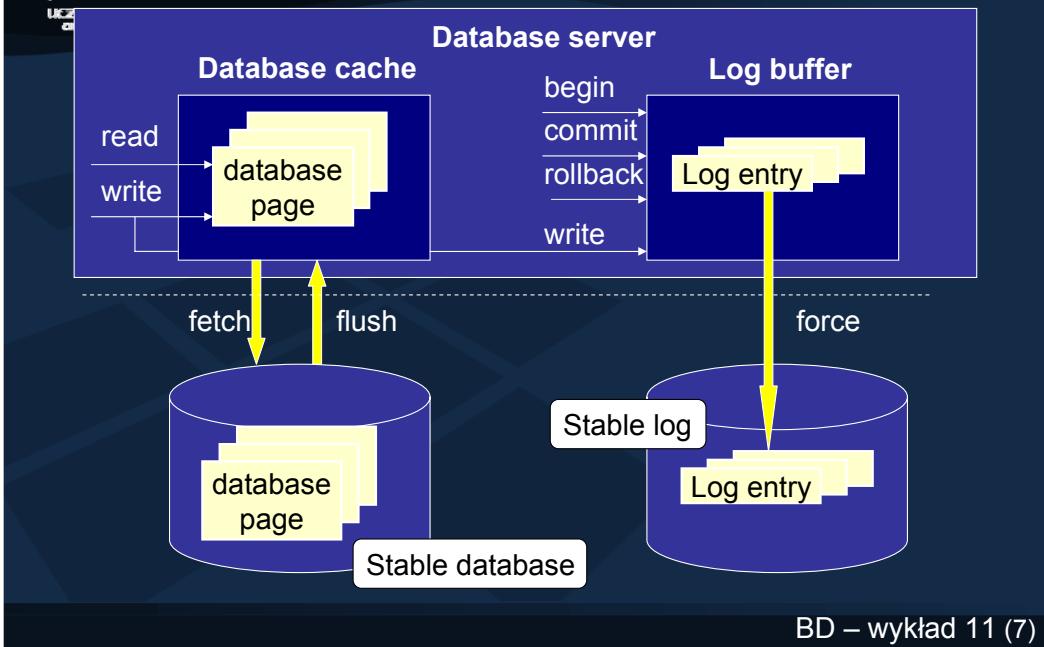
- Elementy składowe modułu odtwarzania :
 - Baza danych
 - Bufor bazy danych
 - Plik logu
 - Bufor logu
- Zawartość wymienionych składowych modułu odtwarzania całkowicie określa stan systemu
- Plik logu jako plik sekwencyjny typu „append-only”,
- Alternatywa – plik logu o dostępie swobodnym przechowujący zbiór wersji stron bazy danych (ang. **shadow database**)

BD – wykład 11 (6)

Przejdziemy obecnie do przedstawienia podstawowych elementów składowych systemu odtwarzania. Zaliczamy do nich: bazę danych, bufor bazy danych, plik logu, oraz bufor logu.

Stan wymienionych składowych systemu odtwarzania całkowicie określa stan systemu. Podstawowym elementem składowym systemu odtwarzania jest plik logu. Plik logu jest implementowany, najczęściej, jako plik sekwencyjny typu „append-only”, jednakże prezentowany model odtwarzania jest na tyle ogólny, że dopuszcza również alternatywne implementacje logu (np. wersje stron bazy danych przechowywane w pliku o dostępie swobodnym – tzw. **shadow database**)

Architektura modułu odtwarzania (2)



BD – wykład 11 (7)

System odtwarzania składa się z dwóch komplementarnych par elementów: bazy danych i bufora bazy danych, z jednej strony, z drugiej, z pliku logu i bufora logu. Baza danych jest przechowywana w pamięci zewnętrznej, która jest pamięcią nie ulotną. Nieformalnie, baza danych jest zbiorem plików, z których każdy składa się z sekwencji stron. Bufor bazy danych jest fragmentem pamięci operacyjnej (pamięci ulotnej), o ograniczonym rozmiarze. Jest to zbiór tzw. ramek, z których każda może pomieścić jedną stronę. Procesy systemu zarządzania bazą danych realizują dostęp do danych (odczyt i zapis rekordów znajdujących się na stronach) via bufor danych. Żądanie procesu Q dostępu do danej strony jest przesyłane do zarządcy bufora, który sprawdza, czy dana strona jest w buforze. Jeżeli tak, to adres strony w buforze jest zwracany do procesu. Jeżeli nie, to zarządcy bufora sprowadza daną stronę z bazy danych do bufora (operacja fetch) i, podobnie jak poprzednio, zwraca adres strony w buforze do procesu. Ściagnięcie strony z bazy danych do bufora wymaga, najczęściej, usunięcia jakieś strony z bufora (operacja flush) – tak zwana wymiana stron na żądanie. Usuwana strona z bufora jest zapisywana na dysk do bazy danych. Każda operacja aktualizacji zawartości strony w buforze danych, oraz operacje begin_transaction, commit_transaction oraz rollback_transaction, generują rekord logu. Rekordy logu są umieszczane w buforze logu, który, co pewien czas, jest zapisywany do pliku logu.



Operacje (1)

- Operacje transakcji:
 - Begin(T)
 - Commit(T)
 - Rollback(T)
 - Save(T)
 - Restore (T,s)
- Operacje na stronach bazy danych:
 - read[pageno, T]
 - write[pageno, T]
 - full-write[pageno, T] (blind-write, real action)
 - exec[op, obj, T]

BD – wykład 11 (8)

Zbiór operacji wykonywanych w ramach systemu odtwarzania podzieliliśmy na operacje poszczególnych elementów składowych systemu. Operacje transakcji, które powodują generacje rekordów logu to: Begin(T) (początek transakcji), Commit(T) (akceptacja transakcji), Rollback(T) (wycofanie transakcji), Save(T) (zawieszenie wykonywania transakcji), oraz Restore (T,s) (odwieszenie wykonywania transakcji).

Operacje, które są wykonywane na stronach bazy danych w buforze to: read[pageno, T] (odczyt strony o zadanym numerze pageno), write[pageno, T] (zapis strony o zadanym numerze pageno), full-write[pageno, T] (zapis nowej strony o numerze pageno – strona pusta), oraz exec[op, obj, T] (wykonanie wskazanej operacji op na wskazanym obiekcie obj, który znajduje się na stronie w buforze).



Operacje (2)

- Operacje na buforze danych:
 - **Fetch** [pageno] : pobierz stronę o identyfikatorze *pageno* z bazy danych do bufora danych,
 - **Flush** [pageno] : zapisz stronę o identyfikatorze *pageno* z bufora danych do bazy danych
- Operacje na pliku logu:
 - **Force ()** : zapisz rekordy logu z bufora logu do pliku logu

BD – wykład 11 (9)

Operacje na buforze danych:

Fetch [pageno] - pobierz stronę o identyfikatorze *pageno* z bazy danych do bufora danych,

Flush [pageno] - zapisz stronę o identyfikatorze *pageno* z bufora danych do bazy danych. Wreszcie, operacja na buforze logu: Force () : zapisz rekordy logu z bufora logu do pliku logu.



Bufor danych (1)

- Bufor: zmniejszenie liczby operacji I/O
- Tablica bufora (ang. *lookaside table*)
- Strategia LRU (ang. *least recently used*) – usuwamy z bufora stronę najmniej używaną w ostatnim czasie

Idea: Nie chcemy zapisywać strony na dysk za każdym razem, gdy strona jest aktualizowana przez transakcję

- Wniosek: Często aktualizowane strony pozostają w buforze aż:
 - zostaną usunięte w wyniku zastosowania strategii LRU
 - zostaną zapisane na dysku po zadany czasie

BD – wykład 11 (10)

Jak już wspomnieliśmy, bufor bazy danych stanowi interfejs pomiędzy procesami zarządzającymi transakcjami a bazą danych. Dostęp do danych jest realizowany wyłącznie w obszarze bufora bazy danych. Ponieważ cała komunikacja transakcji z bazą danych jest realizowana poprzez bufor, efektywność procedur zarządzania buforem ma istotny wpływ na efektywność całego systemu bazy danych. Podstawową miarą oceny efektywności zarządzania buforem jest liczba operacji we/wy. Minimalizując liczbę operacji we/wy optymalizujemy efektywność działania systemu. Jaka jest zatem podstawowa strategia zarządzania buforem? Żądanie procesu Q dostępu do danej strony jest przesyłane do zarządcy bufora, który sprawdza, czy dana strona jest w buforze. W tym celu wykorzystuje tablicę haszową bufora (tzw. *lookaside table*). Jeżeli żądana strona jest w buforze, to adres strony w buforze jest zwracany do procesu. Jeżeli nie, to zarządca bufora sprowadza daną stronę z bazy danych do bufora (operacja *fetch*) i, podobnie jak poprzednio, zwraca adres strony w buforze do procesu. Jak wspomnieliśmy, wczytanie strony do bufora wymaga, najczęściej, usunięcia innej strony z bufora (mówimy o wymianie stron na żądanie). Bufor w trakcie normalnego działania pracuje w nasyceniu, co oznacza, że wszystkie ramki bufora są zajęte przez strony. Najpopularniejszą strategią wymiany stron na żądanie jest strategia LRU (ang. *least recently used*) – usuwamy z bufora stronę najmniej używaną w ostatnim czasie. Idea strategii LRU bazuje na lokalności odwołań – zarządca bufora nie usuwa stron z bufora po zakończeniu aktualizacji tej strony w nadziei, że kolejny proces może zażądać dostępu do tej strony. Pozostawiając zaktualizowaną stronę w buforze minimalizujemy liczbę operacji we/wy. W konsekwencji często aktualizowane strony pozostają w buforze bardzo długo, aż: zostaną usunięte w wyniku zastosowania strategii LRU lub zostaną zapisane na dysku po zadany czasie.



Bufor danych (2)

- Mówimy, że strona w buforze jest „brudna”, jeżeli została uaktualniona przez transakcję od czasu ostatniego zapisu na dysk
- „Brudne” strony mogą pozostać w buforze jeszcze długo po tym, jak uaktualniające je transakcje zostały zaakceptowane

Problem: awaria. Jeżeli aktualizowane strony nie były zapisywane na dysk, to tracimy informacje o dokonanych aktualizacjach

- W momencie aktualizacji, system zapisuje ten fakt w postaci tzw. rekordu logu (ang. *log entry*) w buforze logu
- Okresowo, bufor logu jest zapisywany na dysk do pliku logu (ang. *log file*)

BD – wykład 11 (11)

Wprowadźmy pojęcie tak zwanej „brudnej” strony. Mówimy, że strona w buforze jest „*brudna*”, jeżeli została uaktualniona przez transakcję od czasu ostatniego zapisu na dysk. Z tego co powiedzieliśmy poprzednio wynika, że „brudne” strony mogą pozostać w buforze jeszcze długo po tym, jak uaktualniające je transakcje zostały zaakceptowane. Jest to ważny element mechanizmu zapewniającego odpowiednią efektywność działania systemu bazy danych. Jednakże rodzi to bardzo poważny problem: w przypadku wystąpienia awarii, jeżeli aktualizowane strony nie były zapisywane w międzyczasie na dysk, to tracimy informacje o dokonanych aktualizacjach. Jak rozwiązać ten problem? Okazuje się również, że rozwiązanie polegające na natychmiastowym zapisie uaktualnionych stron na dysk nie rozwiązuje poprawnie problemu utarty informacji. Rozwiązaniem jest mechanizm logu.

W momencie aktualizacji dowolnej strony, system zapisuje ten fakt w postaci tzw. *rekordu logu* (ang. *log entry*) w *buforze logu*. Okresowo, bufor logu jest zapisywany na dysk do *pliku logu* (ang. *log file*). Współdziałanie bufora danych, bazy danych, bufora logu i pliku logu pozwala na poprawną implementację procedur odtwarzania spójnego stanu bazy danych po awarii.



Format logu (1)

- Rozważmy następującą realizację transakcji:

```
r1 = T1:r(a, 50)  T1:w(a, 20)  T2:r(c, 100)  
T2:w(c,50) T2:c  T1:r(b, 50)  T1:w(b, 80)  T1:c.....
```

- Założmy, że w bazie danych zdefiniowano ograniczenie integralnościowe postaci: $a + b = 100$ i założmy, że zapis na dysk jest realizowany zgodnie ze strategią LRU
- Wystąpiła awaria – założmy, że aktualne wartości danych na dysku są następujące: $a = 50$, $b = 80$, $c = 100$
- Dane niespójne

BD – wykład 11 (12)

Przejdzmy obecnie do formalnego zdefiniowania problemu odtwarzania bazy danych po awarii. Zaczniemy od fundamentalnego pytania: czy można podać taką strategię zarządzania buforem danych, aby dane w bazie danych były zawsze spójne? Innymi słowy, czy konieczny jest osobny mechanizm gwarantujący poprawne odtworzenie bazy danych po awarii? Odpowiedź na pierwsze pytanie brzmi – NIE, i, oczywiście, odpowiedź na pytanie drugie brzmi – TAK. Rozpoczniemy od prostego przykładu. Dana jest wspólnie realizacja zbioru dwóch transakcji T1 i T2 przedstawiona na slajdzie. Założmy dodatkowo, że w bazie danych zdefiniowano ograniczenie integralnościowe postaci: $a + b = 100$ i założmy, że zapis na dysk jest realizowany zgodnie ze strategią LRU. Przyjmijmy, że początkowo, stan bazy danych wynosił: $a=50$, $b=50$ i $c=100$. Strony a i c, uaktualnione przez transakcje T1 i T2, pozostały w buforze (nowe wartości $a=20$ i $c=50$), natomiast strona b, uaktualniona przez transakcję T1, została zapisana na dysk w wyniku zadziałania strategii LRU. Jeżeli po wykonaniu operacji T1:w(b, 80) wystąpi awaria w systemie, to wartości danych a, b i c będą następujące: a=50 (poprzednia wartość), b=80 (nowa wartość zapisana przez T1), c=100 (poprzednia wartość). Łatwo zauważyć, że stan bazy danych jest niepoprawny – dane są niespójne. Wniosek: zakładając, że zapis stron z bufora danych na dysk jest realizowany wyłącznie w oparciu o strategię LRU, nie możemy zapewnić poprawnego stanu bazy danych.



Format logu (2)

- Założmy, że strategia zapisu jest następująca: strona jest zapisywana na dysk natychmiast po jej aktualizacji w buforze
- Realizacja jak wyżej - awaria wystąpiła po wykonaniu T2:c
- Wartości danych na dysku są następujące: $a = 20$, $b = 80$, $c = 50$ – dane niespójne
- Wniosek: strategia LRU jak i strategia zapisu natychmiastowego aktualizacji na dysk może prowadzić do niespójności bazy danych

BD – wykład 11 (13)

Przyjmijmy w takim razie inną strategię działania bufora danych. Założmy mianowicie, że strategia zapisu stron na dysk jest następująca: strona jest zapisywana na dysk natychmiast po jej aktualizacji w buforze. Być może to zapewni poprawność działania systemu? Odpowiedź jest negatywna. Rozważmy realizację przedstawioną na poprzednim slajdzie. Założmy, że awaria wystąpiła po wykonaniu operacji T2:c. Zakładając, że każda strona, po aktualizacji, jest natychmiast zapisywana na dysk, otrzymujemy następujący stan bazy danych: $a = 20$, $b = 80$, $c = 50$. Dane są niespójne. Wniosek: strategia LRU jak i strategia natychmiastowego zapisu aktualizacji na dysk może prowadzić do niespójności bazy danych.



Stan spójny bazy danych (1)

- Celem odtwarzania jest zapewnienie własności atomowości i trwałości transakcji
- Procedura odtwarzania bazy danych (ang. database recovery) po awarii, korzystając z informacji zawartej w rekordach logu i stanu bazy danych przechowywanego na dysku, przeprowadza bazę danych do stanu, który odzwierciedla wszystkie lub żadną operacje aktualizacji transakcji aktywnych w momencie wystąpienia awarii
- System transakcyjny, inicjowany ponownie po awarii, nie pamięta intencji logiki transakcji, które były aktywne w momencie awarii - stąd, system musi wycofać wszystkie zmiany w bazie danych wprowadzone przez aktywne transakcje

BD – wykład 11 (14)

Celem odtwarzania jest zapewnienie własności atomowości i trwałości transakcji. Procedura odtwarzania bazy danych po awarii (ang. database recovery), korzystając z informacji zawartej w rekordach logu i stanu bazy danych przechowywanego na dysku, przeprowadza bazę danych do stanu, który odzwierciedla wszystkie lub żadną operacje aktualizacji transakcji aktywnych w momencie wystąpienia awarii. System transakcyjny, inicjowany ponownie po awarii, nie pamięta intencji logiki transakcji, które były aktywne w momencie awarii - stąd, system musi wycofać wszystkie zmiany w bazie danych wprowadzone przez aktywne transakcje.



Stan spójny bazy danych (2)

- Dla realizacji (awaria w momencie realizacji operacji T1:c):

```
r1 = T1:r(a, 50)  T1:w(a, 20)  T2:r(c, 100)  
T2:w(c,50) T2:c  T1:r(b, 50)  T1:w(b, 80)  T1:c.....
```

- transakcja T2 zaakceptowana; T1 wycofana. Stąd, stan spójny po awarii: a = 50, b = 50, c = 50
- **Rekordy logu** – rekordy logu są generowane i zapisywane do bufora logu dla każdej operacji aktualizacji obiektu bazy danych i operacji inicjacji transakcji

BD – wykład 11 (15)

Reasumując, stan spójny po awarii to taki stan, który odzwierciedla wszystkie aktualizacje wykonane przez zaakceptowane transakcje; natomiast zmiany w bazie danych, wprowadzone do niej, przez transakcje aktywne w momencie wystąpienia awarii, muszą zostać wycofane.

Dla realizacji przedstawionej na slajdzie (awaria w momencie realizacji operacji T1:c), poprawny stan bazy danych po awarii powinien odzwierciedlać w bazie danych aktualizacje transakcji T2 (T2 zaakceptowana); oraz powinien wycofać z bazy danych zmiany wprowadzone, być może, przez transakcję (T1 aktywna w momencie awarii). Stąd, stan spójny po awarii powinien być następujący: a = 50, b = 50, c = 50.



Rekordy logu (1)

- Format rekordu pliku logu:
 - **(S, i)**: rekord inicjacji transakcji Ti;
 - **(W, i, a, x, y)**: rekord zapisu danej a przez transakcję Ti, poprzednia wartość danej x (before image), nowa wartość danej y (after image);
 - **(C, i)**: rekord akceptacji transakcji Ti;
- Realizacja:

```
r1 = T1:r(a, 50)  T1:w(a, 20)  T2:r(c, 100)  
T2:w(c,50) T2:c  T1:r(b, 50)  T1:w(b, 80)  T1:c.....
```

BD – wykład 11 (16)

Jak już wspomnieliśmy, mechanizm odtwarzania po awarii opiera się na współdziałaniu czterech elementów. Kluczowymi elementami systemu odtwarzania są bufor i plik logu. Rekordy logu są generowane i zapisywane do bufora logu dla każdej operacji aktualizacji obiektu bazy danych i operacji inicjacji transakcji.

W dalszej części wykładu przyjmiemy następującą notację zapisu formatu rekordów logu:

- (S, i): oznacza rekord inicjacji transakcji Ti;
- (W, i, a, x, y): oznacza rekord zapisu danej a przez transakcję Ti, poprzednia wartość danej x (before image), nowa wartość danej y (after image);
- (C, i): oznacza rekord akceptacji transakcji Ti;



Rekordy logu (2)

- Rekordy logu wygenerowane przez moduł odtwarzania dla przedstawionej realizacji:

Operacja	Rekord logu
1. T1:r(a, 50)	(S, 1) rekord logu początku transakcji T1, nie wpisuje się rekordów logu dla operacji odczytu, w tym przypadku to operacja początku transakcji
2. T1:w(a, 20)	(W, 1, a, 50, 20) – rekord logu dla operacji aktualizacji atrybutu a. Wartość a=50 before image, a=20 – after image

BD – wykład 11 (17)

Dla rozważanej realizacji ze slajdu 15 sekwencja rekordów logu generowana przez zarządcę bufora logu zostanie przedstawiona na kolejnych slajdach.

Operacja

1. T1:r(a, 50) – generowany rekord (S, 1) - rekord logu początku transakcji T1 (generalnie, system nie generuje rekordów logu dla operacji odczytu, w tym przypadku jest to operacja początku transakcji)

2. T1:w(a, 20) – generowany rekord (W, 1, a, 50, 20) – rekord logu dla operacji aktualizacji atrybutu a. Wartość a=50 before image, a=20 – after image



Rekordy logu (3)

3. T2:r(c, 100)

4. T2:w(c, 50)

5. T2:c

6. T1:r(b, 50)

(S, 2) – rekord logu początku transakcji T2

(W, 2, c, 100, 50) – rekord logu dla operacji aktualizacji atrybutu c.
Wartość c=100 (before image),
c=50 – after image

(C, 2) – rekord akceptacji transakcji T2
– (**zapis bufora logu na dysk**)

BD – wykład 11 (18)

Operacja

3. T2:r(c, 100) - generowany rekord (S, 2) - rekord logu początku transakcji T2
4. T2:w(c, 50) - generowany rekord (W, 2, c, 100, 50) – rekord logu dla operacji aktualizacji atrybutu c. Wartość c=100 (before image), c=50 – after image
5. T2:c - generowany rekord (C, 2) – rekord akceptacji transakcji T2 (następuje **zapis bufora logu na dysk**)
6. T1:r(b, 50) brak generacji rekordu logu (operacja odczytu)



Rekordy logu (4)

7. T1:w(b, 80)

(W, 1, b, 50, 80) – rekord logu dla operacji aktualizacji atrybutu b.
Wartość b=50 – before image,
b=80 – after image

8. T1:c

(C, 2) – rekord akceptacji transakcji T1
– (**zapis bufora logu na dysk**)

- Bufor logu jest zapisywany do pliku logu w dwóch sytuacjach:
 - w momencie akceptacji transakcji
 - w momencie przepełnienia bufora (*double-buffered disk write*)

BD – wykład 11 (19)

Operacja

7. T1:w(b, 80) - generowany rekord (W, 1, b, 50, 80) – rekord logu dla operacji aktualizacji atrybutu b. Wartość b=50 (before image), b=80 – after image

8. T1:c - generowany rekord (C, 2) – rekord akceptacji transakcji T1 (następuje **zapis bufora logu na dysk**)

Zauważmy, że bufor logu jest zapisywany do pliku logu w momencie akceptacji transakcji. Oczywiście, bufor logu jest również zapisywany do pliku logu w momencie przepełnienia bufora. Stosowana jest architektura tzw. double-buffered disk write, tj. w momencie, gdy część bufora logu jest zapisywana na dysk do pliku logu, druga część bufora jest dostępna dla nowo generowanych rekordów logu.



Poprawność odtwarzania (1)

- Czy informacje zapisane w buforze logu są wystarczające do odtworzenia spójnego stanu bazy danych?
- Mogą wystąpić dwa problemy:
 - może się okazać, że zapisaliśmy na dysk strony uaktualnione przez nie zaakceptowane transakcje (**UNDO**)
 - może się okazać, że nie zapisaliśmy na dysk stron uaktualnionych przez zaakceptowane transakcje (**REDO**)

BD – wykład 11 (20)

Czy informacje zapisane w buforze logu są wystarczające do odtworzenia spójnego stanu bazy danych po awarii? Aby odpowiedzieć na to pytanie, należy rozważyć dwa przypadki, które mogą stwarzać potencjalnie pewne problemy.

–może się okazać, że zapisaliśmy na dysk strony uaktualnione przez nie zaakceptowane transakcje – musimy wówczas wycofać wszystkie zmiany wprowadzone przez nie zaakceptowane transakcje,

–może się okazać, że nie zapisaliśmy na dysk stron uaktualnionych przez zaakceptowane transakcje – musimy zagwarantować, że zmiany wprowadzone przez zaakceptowane transakcje znajdą się w bazie danych.



Poprawność odtwarzania (2)

- Realizacja (scenariusz nr 1)

$$r1 = T1:r(a, 50) \quad T1:w(a, 20) \quad T2:r(c, 100) \\ T2:w(c, 50) \quad T2:c \quad T1:r(b, 50) \quad T1:w(b, 80) \quad T1:c.....$$

- Założmy, że wystąpiła awaria systemu po wykonaniu operacji $T1:w(b, 80)$ ($(W, 1, b, 50, 80)$ ostatni zapis do bufora logu)
- Bufor logu został zapisany na dysk w momencie akceptacji transakcji T2
- Transakcja T2 została zaakceptowana; transakcja T1 nie.
- Po ponownej inicjacji systemu po awarii, operacje aktualizacji wykonane przez transakcję T1 muszą zostać wycofane! (brak danych umożliwiających wycofanie)

BD – wykład 11 (21)

Aby zilustrować pierwszy z wymienionych problemów rozważmy ponownie przykładową realizację ze slajdu 15. Założmy, że awaria systemu wystąpiła po wykonaniu operacji $T1:w(b, 80)$ ($(W, 1, b, 50, 80)$ - ostatni zapis do bufora logu). Przypomnijmy, że bufor logu został zapisany na dysk w momencie akceptacji transakcji T2. W momencie wystąpienia awarii transakcja T2 była zaakceptowana; natomiast transakcja T1 była aktywna. Należy pamiętać, że awaria systemu niszczy nie tylko dane znajdujące się w buforze danych, ale również bufor logu. Jedyna informacja, które może być wykorzystana do odtworzenia stanu spójnego bazy danych znajduje się na dysku zewnętrznym w pliku logu. Po ponownej inicjacji systemu po awarii, operacje aktualizacji wykonane przez transakcję T1 muszą zostać wycofane! Jeżeli na skutek działania strategii LRU strona b, zmodyfikowana przez transakcję T1, została zapisana na dysk, brak jest danych umożliwiających wycofanie tej aktualizacji. Rekord logu ($W, 1, b, 50, 80$) nie został zapisany do pliku logu. Do tego problemu wróćmy w późniejszej części wykładu.



Poprawność odtwarzania (3)

- Procedura odtwarzania po awarii jest wykonywana w dwóch fazach: **ROLLBACK** i **ROLL FORWARD**
- W fazie **ROLLBACK**, rekordy pliku logu są odczytywane w odwrotnej kolejności (od końca). W fazie tej procedura odtwarzania wykonuje operacje UNDO wszystkich operacji aktualizacji transakcji, które nie zostały zaakceptowane
- W fazie **ROLL FORWARD**, rekordy pliku logu są odczytywane w oryginalnej kolejności (od początku). W fazie tej procedura odtwarzania wykonuje operacje REDO wszystkich operacji aktualizacji transakcji, które zostały zaakceptowane

BD – wykład 11 (22)

Procedura odtwarzania bazy danych po awarii jest wykonywana w dwóch fazach: fazie ROLLBACK i fazie ROLL FORWARD. W fazie ROLLBACK, rekordy pliku logu są odczytywane w odwrotnej kolejności (od końca) pliku logu. W fazie tej procedura odtwarzania wykonuje operacje UNDO wszystkich operacji aktualizacji transakcji, które nie zostały zaakceptowane. W fazie ROLL FORWARD, rekordy pliku logu są odczytywane w oryginalnej kolejności (od początku) pliku logu. W fazie tej procedura odtwarzania wykonuje operacje REDO wszystkich operacji aktualizacji transakcji, które zostały zaakceptowane.



Faza Rollback

• Rekord logu	Rollback
5. (C, 2)	wstaw T2 do listy transakcji zaakceptowanych
4. (W,2,c,100,50)	T2 na liście transakcji zaakceptowanych – nie rób nic
3. (S, 2)	T2 na liście transakcji zaakceptowanych – nie rób nic
2. (W,1,a,50,20)	T1 – aktywna, wykonaj UNDO operacji aktualizacji – dana a przyjmuje wartość 50
1. (S, 1)	Transakcja T1 pasywna. Zakończ fazę ROLLBACK

BD – wykład 11 (23)

Przedstawimy obecnie, nieco dokładniej, działanie procedury Rollback w odniesieniu do przykładu ze slajdu 21. Następujące rekordy logu zostały zapisane do pliku logu w momencie akceptacji transakcji T2 (od początku pliku logu): .(S, 1), .(W,1,a,50,20), .(S, 2), .(W,2,c,100,50), .(C, 2). Procedura Rollback odczytuje rekordy pliku logu w odwrotnej kolejności. Omówimy kolejno operacje wykonywane w fazie Rollback:

- | | |
|-------------------|---|
| 5. (C, 2) | wstaw T2 do listy transakcji zaakceptowanych |
| 4. (W,2,c,100,50) | T2 na liście transakcji zaakceptowanych – nie rób nic |
| 3. (S, 2) | T2 na liście transakcji zaakceptowanych – nie rób nic |
| 2. (W,1,a,50,20) | T1 – aktywna, wykonaj UNDO operacji aktualizacji – dana a przyjmuje wartość 50 (before image) |
| 1.(S, 1) | Transakcja T1 pasywna. Zakończ fazę ROLLBACK |

W fazie Rollback została wykonana operacja UNDO dla transakcji T1, która była aktywna w momencie wystąpienia awarii. Przejdziemy do przedstawienia fazy Roll Forward.



Faza Roll Forward

• Rekord logu	Rollback
1. (S, 1)	brak akcji (T1 aktywna)
2. (w,1,a,50,20)	brak akcji (T1 aktywna)
3. (S, 2)	brak akcji
4. (w,2,c,100,50)	Transakcja T2 na liście transakcji zaakceptowanych. Wykonaj REDO operacji aktualizacji – dana c przyjmuje wartość 50
5. (C,2)	brak akcji Zakończono przeglądanie pliku logu – zakończ procedurę odtwarzania

BD – wykład 11 (24)

Podobnie jak poprzednio, działanie procedury ROLL FORWARD przedstawimy w odniesieniu do przykładu ze slajdu 21. W fazie ROLL FORWARD, rekordy pliku logu są odczytywane w oryginalnej kolejności (od początku) pliku logu. W fazie tej procedura odtwarzania wykonuje operacje REDO wszystkich operacji aktualizacji transakcji, które zostały zaakceptowane.

1. (S, 1) - dla pierwszego rekordu logu transakcja T1 jest inicjowana i system nie podejmuje żadnej akcji;
2. (w,1,a,50,20) - dla drugiego rekordu logu transakcja T1 jest aktywna i system nie podejmuje żadnej akcji;
3. (S, 2) - podobnie jest dla trzeciego rekordu logu, który opisuje inicjowanie transakcji T2;
4. (w,2,c,100,50) - dla czwartego rekordu logu, transakcja T2 znajduje się na liście transakcji zaakceptowanych; system wykonuje operację REDO operacji aktualizacji – dana c przyjmuje wartość 50 (after image);
5. (C,2) - piąty rekord logu oznacza zatwierdzenie transakcji drugiej, a system nie podejmuje żadnej akcji. W tym momencie kończy się przeglądanie pliku logu, co kończy procedurę odtwarzania.

W fazie Roll Forward została wykonana operacja REDO dla transakcji T2, która została zaakceptowana wcześniej, przed wystąpieniem awarii.



Poprawność odtwarzania (4)

- W jaki sposób możemy zapewnić, że wszystkie rekordy logu konieczne do zapewnienia poprawności procedury odtwarzania zostaną zapisane na dysku?
- Założenia dodatkowe:
 - zapis na dysk jest realizowany w sposób atomowy - „read after write”
 - akceptacja transakcji + zapis bufora logu - akcja atomowa
- Poprawność procedur ROLLBACK i ROLL FORWARD

BD – wykład 11 (25)

Wróćmy obecnie do dyskusji nad poprawnością zaproponowanego mechanizmu odtwarzania spójności bazy danych po awarii. W jaki sposób możemy zapewnić, że wszystkie rekordy logu konieczne do zapewnienia poprawności procedury odtwarzania zostaną zapisane na dysku? System zarządzania bazą danych musi zapewnić dodatkowe mechanizmy:

–zapis na dysk jest realizowany w sposób atomowy (np. mechanizm „read after write”)
–akceptacja transakcji + zapis bufora logu = akcja atomowa, tj. jeżeli z jakiś względów nie udało się zapewnić zapisu bufora logu do pliku logu, to oznacza to brak akceptacji transakcji.

Wróćmy do pytania, czy mechanizm odtwarzania oparty o procedury Rollback i ROLL FORWARD, uzupełniony o dwa wymienione wyżej mechanizmy, gwarantuje poprawność odtwarzania?



Poprawność odtwarzania (5)

ROLL FORWARD - operacje REDO

Transakcja nie jest zaakceptowana jeżeli nie zostanie zapisany bufor logu. Jeżeli bufor logu zostanie zapisany, to wszystkie rekordy logu, zapisywane do bufora logu, znajdą się na dysku co gwarantuje poprawność realizacji procedury ROLL FORWARD

ROLLBACK - operacje UNDO

Musimy zagwarantować, że wszystkie aktualizacje wprowadzone przez nie zaakceptowane transakcje zostaną cofnięte - ale strony zmodyfikowane przez nie zaakceptowane transakcje mogą być zapisane na dysk (np. przez LRU)

BD – wykład 11 (26)

Rozpoczniemy od poprawności procedury ROLL FORWARD. Czy procedura ta pozwala zagwarantować, że wszystkie zmiany wprowadzone do bazy danych przez zaakceptowane transakcje znajdują się na dysku? Odpowiedź jest twierdząca. Transakcja nie jest zaakceptowana, jeżeli nie zostanie zapisany bufor logu. Jeżeli bufor logu zostanie zapisany, to wszystkie rekordy logu, zapisywane do bufora logu, znajdują się na dysku. To gwarantuje poprawność realizacji procedury ROLL FORWARD, tj. gwarantuje, że wszystkie zmiany wprowadzone do bazy danych przez zaakceptowane transakcje znajdują się na dysku (mechanizm REDO w fazie ROLL FORWARD). Drugie pytanie brzmi: czy procedura ROLLBACK gwarantuje, że wszystkie aktualizacje wprowadzone przez nie zaakceptowane transakcje zostaną cofnięte ? Niestety, odpowiedź jest negatywna - strony zmodyfikowane przez nie zaakceptowane transakcje mogą być zapisane na dysk (np. przez LRU), natomiast tych zmian nie można wycofać w oparciu o rekordy logu zapisane w pliku logu.



Poprawność odtwarzania (6)

- Czy może to stwarzać problemy przy odtwarzaniu? TAK
- Rozwiązania problemu:
- **Rozwiązanie A:** zawiesić działanie procedury LRU

Wszystkie brudne strony pozostają w buforze do momentu akceptacji transakcji - procedura UNDO nie jest wówczas potrzebna

- **Rozwiązanie B:** zmodyfikować LRU

Rozwiązaniem jest technika „*write ahead log*” (WAL) i sekwencyjne numery logu LSN (ang. *Log sequence numbers*)

BD – wykład 11 (27)

Możliwe są dwa rozwiązania tego problemu.

Rozwiązanie A: zawiesić działanie procedury LRU, to znaczy, przyjąć założenie, że wszystkie brudne strony pozostają w buforze do momentu akceptacji transakcji - procedura UNDO nie jest wówczas potrzebna.

Rozwiązanie B: zmodyfikować strategię LRU.

Rozwiązaniem jest technika nazywana „*write ahead log*” (WAL).



- System bazy danych utrzymuje licznik, który generuje rosnącą sekwencję LSN
- **LSN** jest liczbą całkowitą, przypisywaną do każdego rekordu logu zapisywanego do bufora logu
- **LSN_BUFFMIN** - SBD przechowuje również najmniejszą wartość LSN strony, od czasu ostatniego zapisu bufora na dysk
- **LSN_PGMAX** - dla każdej strony w buforze danych, system pamięta wartość ostatniej LSN operacji, która aktualizowała daną na tej stronie

BD – wykład 11 (28)

System bazy danych utrzymuje specjalny licznik, który generuje rosnącą sekwencję tzw. log sequence number (LSN). LSN jest liczbą całkowitą, przypisywaną do każdego rekordu logu zapisywanego do bufora logu. SBD przechowuje również najmniejszą wartość LSN strony, od czasu ostatniego zapisu bufora na dysk. Oznaczamy ją LSN_BUFFMIN. Jest to unikalna liczba dla całego systemu. Ponadto, dla każdej strony w buforze danych, system pamięta wartość ostatniej operacji LSN, która aktualizowała daną na tej stronie. Oznaczamy ją przez LSN_PGMAX.



- Reguła modyfikacji LRU: dana strona w buforze może zostać zapisana na dysk (zgodnie z LRU), wtedy i tylko wtedy gdy jej **LSN_PGMAX < LSN_BUFFMIN**
- Reguła ta gwarantuje, że dana strona nie zostanie zapisana na dysku, jeżeli wcześniej nie zostanie zapisany na dysku odpowiedni rekord logu

BD – wykład 11 (29)

Reguła modyfikacji LRU jest następująca. Dana strona w buforze może zostać zapisana na dysk (zgodnie z LRU), wtedy i tylko wtedy gdy jej $LSN_PGMAX < LSN_BUFFMIN$. Reguła ta gwarantuje, że dana strona nie zostanie zapisana na dysku, jeżeli wcześniej nie zostanie zapisany na dysku odpowiedni rekord logu



Punkty kontrolne (1)

- Do jakiego stanu należy się cofnąć wykonując procedurę ROLLBACK?
 - do momentu inicjacji (startup) systemu
 - do stanu określonego przez użytkownika (administratora)
- Problem:
 - procedura ROLLBACK - większość transakcji to krótkie transakcje aktualizacji - stosunkowo efektywna
 - procedura ROLL FORWARD - konieczność przetworzenia całego pliku logu - niska efektywność

BD – wykład 11 (30)

Na zakończenie wykładu, krótko wspomnijmy o tak zwanych punktach kontrolnych. Do jakiego stanu należy się cofnąć w pliku logu wykonując procedurę ROLLBACK? do momentu inicjacji (startup) systemu, czy do stanu określonego przez użytkownika (administratora)? Problemem może być rozmiar pliku logu. Jak wygląda efektywność procedury ROLLBACK i ROLL FORWARD? Procedura ROLLBACK - większość transakcji to krótkie transakcje aktualizacji - jest stosunkowo efektywna. Procedura ROLL FORWARD - konieczność przetworzenia całego pliku logu - ma niską efektywność.



Punkty kontrolne (2)

- Trzy podejścia do tworzenia punktów kontrolnych w systemie bazy danych:

punkt kontrolny akceptacyjnie spójny
(ang. *commit-consistent checkpointing*)

punkt kontrolny spójny z pamięcią podręczną
(ang. *cache-consistent checkpointing*)

punkt kontrolny rozmyty
(ang. *Fuzzy checkpointing*)

BD – wykład 11 (31)

Zauważmy, że w trakcie wystąpienia awarii, tylko część transakcji była aktywna. Stąd, wycofanie zmian wprowadzonych przez te transakcje może być stosunkowo efektywnie zaimplementowane. W przypadku procedury ROLL FORWARD – zachodzi konieczność przetworzenia całego pliku logu! Musimy, de facto, wykonać wszystkie zaakceptowane transakcje raz jeszcze – operacja po operacji! Może to być bardzo kosztowne. Rozwiążaniem są tak zwane punkty kontrolne. Punkt kontrolny (ang. checkpoint) - punkt na osi czasu, od którego i do którego są realizowane procedury ROLLBACK i ROLL FORWARD. Punkty kontrolne są tworzone w trakcie działania systemu bazy danych i ograniczają konieczność przetwarzania całego pliku logu w procedurze odtwarzania. Po pierwsze, plik logu narasta bardzo szybko i możemy nie posiadać możliwości, aby przechowywać cały plik logu. Po drugie, przetwarzanie całego pliku logu byłoby niezmiernie kosztowne czasowo.

Trzy ogólne podejścia do tworzenia punktów kontrolnych w systemie bazy danych to:

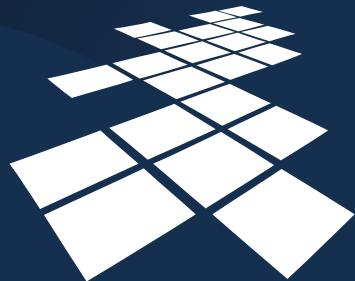
- punkt kontrolny akceptacyjnie spójny (ang. *commit-consistent checkpointing*),
- punkt kontrolny spójny z pamięcią podręczną (ang. *cache-consistent checkpointing*),
- punkt kontrolny rozmyty (ang. *fuzzy checkpointing*).

Opis wymienionych punktów kontrolnych i sposobów ich tworzenia można znaleźć w podanej literaturze.

Optymalizacja zapytań

część I

Wykład przygotował:
Tadeusz Morzy



**UCZELNIA
ONLINE**

BD – wykład 12

Wykład jest poświęcony problemom wykonywania i optymalizacji zapytań w systemach baz danych. Rozpoczniemy od krótkiego wprowadzenia do wykonywania zapytań. Przedstawimy poszczególne fazy przetwarzania zapytań, takie jak analiza zapytań, normalizacja zapytań, analiza semantyczna zapytań upraszczanie zapytań oraz restrukturyzacja zapytań. Następnie, przejdziemy do omówienia problemów związanych z optymalizacją zapytań. W kolejnej części wykładu, przedstawimy i omówimy algebraiczne reguły transformacji zapytań. Na zakończenie wykładu powiemy o technice przepisywania zapytań jako ważnym mechanizmie optymalizacji wykonywania zapytań.



BD – wykład 12 (2)

Proces wykonywania zapytania składa się, najogólniej mówiąc, z kilku faz: fazy dekompozycji, fazy optymalizacji zapytania, fazy generacji kodu, oraz fazy wykonania. Zapytanie wyrażone w języku SQL, w fazie dekompozycji, jest transformowane do postaci wyrażenia algebra relacji, umożliwiającej dalsze przetwarzania zapytania. Transformacja zapytania do postaci wyrażenia algebraicznego wymaga dostępu do katalogu bazy danych. W kolejnej fazie, wyrażenie reprezentujące zapytanie jest optymalizowane za pomocą dwóch mechanizmów. Pierwszym jest mechanizm reguł transformacji, który wykorzystując pewien zbiór reguł stara się uprościć zapytanie. Drugim, jest mechanizm optymalizacji kosztowej, który, dla danego wyrażenia, stara się określić optymalny plan wykonania zapytania (m.in. stara się określić optymalne drzewo połączeń). Moduł optymalizatora wykorzystuje w procesie optymalizacji zapytania statystyki, przechowywane w systemie bazy danych, dotyczące relacji, atrybutów i indeksów (rozmiary relacji, liczba stron relacji, liczba różnych wartości atrybutu, itp.). W kolejnym kroku, dla znalezionej planu wykonania zapytania, generowany jest kod zapytania, który jest, następnie, uruchamiany przez tak zwany silnik zapytań (ang. query engine).



Dynamiczna vs statyczna optymalizacja zapytań

Dynamiczna optymalizacja zapytań

Statyczna optymalizacja zapytań

Optymalizacja pojedynczego zapytania

Jednoczesna optymalizacja zbioru zapytań

BD – wykład 12 (3)

Zanim przejdziemy do przedstawienia i omówienia poszczególnych faz przetwarzania zapytania, wprowadzimy i omówimy krótko klasyfikację metod optymalizacji zapytań. Istnieje wiele klasyfikacji metod optymalizacji zapytań. Pierwsza z podanych klasyfikacji wyróżnia optymalizację statyczną i optymalizację dynamiczną. Optymalizacja statyczna polega na znalezieniu „najlepszego” planu wykonania zapytania, przed rozpoczęciem wykonywania zapytania. W trakcie realizacji zapytania plan wykonania zapytania nie ulega już zmianie – stąd nazwa optymalizacja statyczna. Optymalizacja dynamiczna polega na znalezieniu „najlepszego” planu wykonania zapytania, przed rozpoczęciem wykonywania zapytania, ale później, w trakcie wykonywania zapytania jego plan wykonania może ulegać zmianie. Aktualnie, komercyjne systemy baz danych zapewniają jedynie optymalizację statyczną, choć efektywność takiej optymalizacji jest najczęściej niższa aniżeli efektywność optymalizacji dynamicznej. Optymalizacja dynamiczna jest jednak znacznie bardziej kosztowna.

Druga z podanych klasyfikacji wyróżnia optymalizację pojedynczego zapytania oraz jednoczesną optymalizację wielu zapytań. W przypadku optymalizacji pojedynczego zapytania, optymalizacji podlega tylko jedno zapytanie. W przypadku jednoczesnej optymalizacji wielu zapytań, częściowe wyniki wykonania jednego zapytania mogą być wykorzystane przez wiele innych zapytań, co prowadzi do minimalizacji czasu wykonania zbioru zapytań. W chwili obecnej systemy komercyjnych baz danych zapewniają jedynie optymalizację pojedynczego zapytania.



Proces optymalizacji zapytań

- Transformacja zapytania SQL do postaci drzewa wyrażenia logicznego:
 - Identyfikacja bloków zapytania (odpowiadających zagnieżdżonym zapytaniom lub perspektywom)
- Faza przepisywania zapytania:
 - Zastosowania **transformacji algebraicznych** w celu uzyskania tańszego planu wykonania zapytania
- Optymalizacja bloku: zdefiniowania porządku wykonywania operacji połączenia
- Zakończenie optymalizacji: wybór uszeregowania

BD – wykład 12 (4)

Jak już wspomnieliśmy, pierwszą fazą przetwarzania zapytań jest faza transformacji zapytania SQL do postaci drzewa wyrażenia logicznego: Celem tej fazy jest również identyfikacja bloków zapytania (odpowiadających zagnieżdżonym zapytaniom lub perspektywom). W kolejnym kroku realizowana jest faza przepisywania zapytania za pomocą transformacji algebraicznych w celu uzyskania tańszego planu wykonania zapytania. W konsekwencji uzyskujemy zbiór najlepszych planów wykonania pojedynczych bloków zapytania. Pozostaje jeszcze problem połączenia bloków, w szczególności, problem zdefiniowania porządku wykonywania operacji połączenia. Wybór kolejności wykonywania operacji połączenia, tzn. wybór uszeregowania operacji połączenia, kończy proces optymalizacji zapytania. Obecnie, przejdziemy do przedstawienia poszczególnych faz przetwarzania zapytań.



Dekompozycja zapytania

- **Dekompozycja zapytania:** celem procesu dekompozycji zapytania jest transformacja zapytania wyrażonego w języku wysokiego poziomu na wyrażenie algebry relacji i weryfikacja syntaktycznej i semantycznej poprawności zapytania
- Etapy procesu dekompozycji zapytania:
 - Analiza zapytań
 - normalizacja zapytań
 - analiza semantyczna zapytań
 - upraszczanie zapytań
 - restrukturyzacja zapytania

BD – wykład 12 (5)

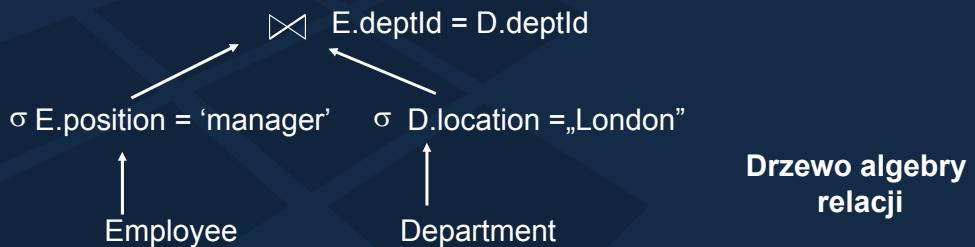
Pierwszą fazą przetwarzania zapytania jest faza dekompozycji zapytania. Celem procesu dekompozycji zapytania jest transformacja zapytania wyrażonego w języku wysokiego poziomu na wyrażenie algebry relacji i weryfikacja syntaktycznej i semantycznej poprawności zapytania. Proces dekompozycji składa się z następujących etapów:

–analiza zapytania,
–normalizacja zapytania,
–analiza semantyczna zapytania,
–upraszczanie zapytania,
–restrukturyzacja zapytania.



Analiza zapytania

- Analiza syntaktyczna poprawności zapytania
- Weryfikacja poprawności atrybutów i relacji
- Transformacja zapytania do postaci reprezentacji wewnętrznej, bardziej adekwatnej do procesu dalszego przetwarzania zapytania



BD – wykład 12 (6)

Celem etapu analizy jest analiza syntaktyczna poprawności zapytania. W skład tej analizy wchodzi weryfikacja poprawności atrybutów i relacji (czy w bazie danych występują wyspecyfikowane w zapytaniu relacje i atrybuty, czy zapytanie poprawnie specyfikuje typy danych). Następnie, zapytanie wyrażone w języku SQL jest transformowane do postaci reprezentacji wewnętrznej (wyrażenia algebry relacji), bardziej adekwatnej do procesu dalszego przetwarzania zapytania. Przedstawione na slajdzie drzewo algebry relacji, reprezentujące postać wewnętrzną zapytania, reprezentuje zapytanie, którego postać w języku SQL jest następująca:

Select *

From Employee E, Department D

Where E.deptId = D.DeptId

And E.position = 'manger" and D.location = 'London";



Normalizacja zapytania

- Celem tego etapu jest przekształcenia wewnętrznej reprezentacji zapytania do znormalizowanej postaci koniunkcyjnej lub dysjunkcyjnej postaci
- Dowolny predykat (w SQL) można przekształcić do jednej z dwóch postaci:
 - Normalnej postaci koniunkcyjnej
 $(position=„manager” or salary > 1000) and deptId =100$
 - Normalnej postaci dysjunkcyjnej
 $(position=„manager” and salary > 1000) or deptId =100$

BD – wykład 12 (7)

Kolejnym etapem fazy dekompozycji jest normalizacja zapytania. Celem etapu normalizacji zapytania jest przekształcenie wewnętrznej reprezentacji zapytania do znormalizowanej postaci koniunkcyjnej lub dysjunkcyjnej. W fazie tej sekwencja predykatów selekcji jest przekształcana do normalnej postaci koniunkcyjnej lub normalnej postaci dysjunkcyjnej. Postać dysjunkcyjna jest, najczęściej, mniej efektywna, gdyż wymaga niezależnego wartościowania poszczególnych składowych wyrażenia. Przykłady postaci koniunkcyjnej i dysjunkcyjnej wyrażenia zapytania przedstawiono na slajdzie.



Analiza semantyczna zapytania (1)

- Celem analizy jest odrzucenie niepoprawnie sformułowanych lub sprzecznych zapytań
- Zapytanie jest niepoprawnie sformułowane jeżeli jego elementy składowe nie prowadzą do generacji wyniku
- Zapytanie jest sprzeczne jeżeli jego predykaty nie mogą być spełnione przez żadną krotkę

position = 'manager' and position = 'assistant'
(position = 'manager' and position = 'assistant')
or salary > 1000 - można uprościć do salary > 1000;

wartość sprzecznej klauzuli interpretujemy jako wartość FALSE

BD – wykład 12 (8)

Kolejnym, ważnym, etapem dekompozycji zapytania jest etap analizy semantycznej zapytania. Celem analizy semantycznej zapytania jest odrzucenie niepoprawnie sformułowanych lub sprzecznych zapytań. Zapytanie jest niepoprawnie sformułowane, jeżeli jego elementy składowe nie prowadzą do generacji wyniku. Zapytanie jest sprzeczne, jeżeli jego predykaty nie mogą być spełnione przez żadną krotkę w bazie danych. Przykładem klauzuli, która jest sprzeczna jest wyrażenie: position = 'manager' and position = 'assistant'.

Zakładając, że baza danych jest w 1NF, nie istnieje w bazie danych żadna krotka, któreby jednocześnie spełniała oba predykaty. Wartość sprzecznej klauzuli interpretujemy jako wartość FALSE. W związku z tym, wyrażenie zawierające sprzeczną klauzulę można uprościć. Przykładowo, wyrażenie

(position = 'manager' and position = 'assistant') or salary > 1000;

ze względu na sprzeczność klauzuli : „position = 'manager' and position = 'assistant'” można uprościć do postaci „salary > 1000”



Analiza semantyczna zapytania (2)

- Algorytmy oceny poprawności semantycznej zapytań istnieją tylko dla pewnej klasy zapytań nie zawierających dysjunkcji i negacji
- Rozwiązywanie problemu zapytań niepoprawnie sformułowanych:
- Skonstruuj **graf połączenia relacji** (*relation connection graph*) – (wierzchołki odpowiadają relacjom, łuki odpowiadają operacjom połączenia) – jeżeli graf nie jest spójny, to zapytanie jest niepoprawnie sformułowane

Rozwiązywanie problemu zapytań sprzecznych:
Skonstruuj graf połączeń atrybutów
(*normalized attribute connection graph*)

BD – wykład 12 (9)

Niestety, algorytmy oceny poprawności semantycznej zapytań istnieją tylko dla pewnej klasy zapytań, nie zawierających dysjunkcji i negacji. W jaki sposób rozwiązywany jest problem zapytań niepoprawnie sformułowanych oraz zapytań sprzecznych? Rozwiązywanie problemu zapytań niepoprawnie sformułowanych opiera się na konstrukcji tak zwanego grafu połączenia relacji. W grafie tym, wierzchołki odpowiadają relacjom, natomiast łuki odpowiadają operacjom połączenia wyspecyfikowanych w zapytaniu. Dodatkowo, graf połączenia relacji zawiera wierzchołek reprezentujący wynik zapytania. Jeżeli graf połączenia relacji nie jest spójny, to zapytanie jest niepoprawnie sformułowane.



Analiza semantyczna zapytania (3)

- **Graf połączeń atrybutów** – dla każdej referencji do atrybutu utwórz w grafie wierzchołek atrybutu lub wierzchołek 0. Utwórz łuk skierowany pomiędzy wierzchołkami reprezentującymi operację połączenia, oraz łuk skierowany pomiędzy wierzchołkiem atrybutu a wierzchołkiem 0 reprezentującym warunek selekcji
- Następnie, nadajemy wagę łukom:
 - łuk: $a \rightarrow b$ -- waga c , jeżeli łuk reprezentuje warunek nierównościowy ($a \leq b+c$)
 - łuk: $0 \rightarrow a$ -- waga c , jeżeli łuk reprezentuje warunek nierównościowy ($a \geq c$)
 - łuki reprezentujące połączenia -- waga 0

BD – wykład 12 (10)

Rozwiązywanie problemu zapytań niepoprawnie sformułowanych opiera się na konstrukcji tak zwanego grafu połączeń atrybutów. Graf połączeń atrybutów konstruujemy następująco. Dla każdej referencji do atrybutu tworzymy w grafie wierzchołek atrybutu lub wierzchołek 0. Następnie, tworzymy łuk skierowany pomiędzy wierzchołkami reprezentującymi operację połączenia, oraz łuk skierowany pomiędzy wierzchołkiem atrybutu a wierzchołkiem 0, reprezentującym warunek selekcji. W kolejnym kroku nadajemy wagę łukom:

łuk: $a \rightarrow b$ posiada wagę c , jeżeli łuk reprezentuje warunek nierównościowy ($a \leq b+c$),

łuk: $0 \rightarrow a$ posiada wagę $-c$, jeżeli łuk reprezentuje warunek nierównościowy ($a \geq c$).

Łuki reprezentujące połączenia posiadają wagę 0.

Jeżeli graf połączeń atrybutów zawiera cykl, którego suma wag jest ujemna, zapytanie jest sprzeczne.



Analiza semantyczna zapytania (4)

- Jeżeli graf zawiera cykl, którego suma wag jest ujemna, zapytanie jest sprzeczne
- Zapytanie:**

```
Select p.propertyID, p.street  
From client c, viewing v, PropertyForRent p  
Where c.clientNo = v.clientNo and c.maxrent >= 500  
and c.prefType = 'Flat' and p.ownerNo = „CO93”
```



Zapytanie jest źle sformułowane – brakuje warunku połączeniowego $v.propertyNo = p.propertyNo$

BD – wykład 12 (11)

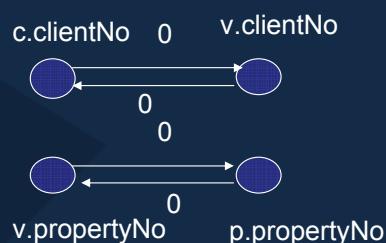
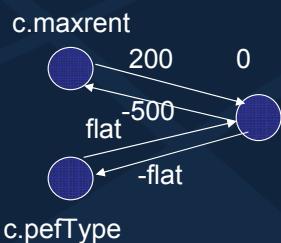
Dla ilustracji problemu zapytań niepoprawnie sformułowanych rozważmy przykładowe zapytanie przedstawione na slajdzie. Skonstruujmy dla podanego zapytania graf połączenia relacji. W grafie tym, przypomnijmy, wierzchołki odpowiadają relacjom, natomiast luki odpowiadają operacjom połączenia wyspecyfikowanym w zapytaniu. Zapytanie zawiera trzy relacje. Zatem, graf połączenia relacji będzie zawierał 4 wierzchołki. Po uzupełnieniu grafu lukami reprezentującymi operacje połączenia wyspecyfikowane w zapytaniu, otrzymujemy graf przedstawiony na slajdzie. Łatwo zauważyc, że podane zapytanie jest źle sformułowane, gdyż graf nie jest spójny. W zapytaniu brakuje warunku połączeniowego $v.propertyNo = p.propertyNo$.



Analiza semantyczna zapytania (5)

Zapytanie:

```
Select p.propertyID, p.street
From client c, viewing v, PropertyForRent p
Where c.clientNo = v.clientNo
and v.propertNo = p.propertyNo
and c.maxrent >= 500 and c.prefType = 'Flat'
and c.maxrent < 200;
```



BD – wykład 12 (12)

Dla ilustracji problemu zapytań sprzecznych rozważmy przykładowe zapytanie przedstawione na slajdzie. Skonstrujmy dla podanego zapytania graf połączeń atrybutów. Przypomnijmy, że graf połączeń atrybutów konstrujemy następująco. Dla każdej referencji do atrybutu tworzymy w grafie wierzchołek atrybutu lub wierzchołek 0. Następnie, tworzymy luk skierowany pomiędzy wierzchołkami reprezentującymi operację połączenia, oraz luk skierowany pomiędzy wierzchołkiem atrybutu a wierzchołkiem 0, reprezentujący warunek selekcji. W kolejnym kroku nadajemy wagę lukom. Zauważmy, że zapytanie zawiera dwa warunki połączniowe:

(1) $c.\text{clientNo} = v.\text{clientNo}$

(2) $v.\text{propertNo} = p.\text{propertyNo}$

reprezentowane przez wierzchołki atrybutów: $c.\text{clientNo}$, $v.\text{clientNo}$, $v.\text{propertNo}$, $p.\text{propertyNo}$. Wierzchołki te połączone są wzajemnie lukami o wagach 0. Podane zapytanie zawiera 3 predykaty selekcji (referencje do atrybutów):

$c.\text{maxrent} \geq 500$

$c.\text{prefType} = \text{'Flat'}$

$c.\text{maxrent} < 200$

reprezentowane przez wierzchołki atrybutów: $c.\text{maxrent}$, $c.\text{prefType}$, i $c.\text{maxrent}$.

Wierzchołki te połączone są wzajemnie lukami o wagach, odpowiednio, 200, -500, flat oraz -flat. Jak łatwo zauważyc, przedstawiony graf połączeń atrybutów zawiera cykl, którego suma wag jest ujemna (-300). Zatem, podane zapytanie jest sprzeczne.



Upraszczanie zapytania

- Celem jest identyfikacja wyrażeń redundantnych, eliminacja wspólnych podwyrażeń, i transformacja zapytania do równoważnej postaci ułatwiającej dalsze przekształcanie zapytania
- Początkowa optymalizacja polega na zastosowaniu znanych reguł algebry relacji:

– $p \wedge (p) = p$	$p \vee (p) = p$
– $p \wedge (\text{false}) = \text{false}$	$p \vee \text{false} = p$
– $p \wedge (\text{true}) = p$	$p \vee \text{true} = \text{true}$
– $p \wedge (\neg p) = \text{false}$	$p \vee (\neg p) = \text{true}$
– $p \wedge (p \vee q) = p$	$p \vee (p \wedge q) = p$

BD – wykład 12 (13)

Kolejnym etapem fazy dekompozycji jest upraszczanie zapytań. Celem tego etapu jest identyfikacja wyrażeń redundantnych, eliminacja wspólnych podwyrażeń, i transformacja zapytania do równoważnej postaci, ułatwiającej dalsze przekształcanie zapytania. Transformacja zapytania do postaci równoważnej polega na zastosowaniu znanych reguł algebry relacji podanych na slajdzie.



Budowa bloków

- Transformacje algebraiczne (wiele i bardzo różnych)
- Model kosztowy: estymacja kosztów i rozmiarów częściowych wyników zapytania
- Znajdowanie najlepszego drzewa operacji połączenia:
 - Podejście Bottom-up
(czasami nazywane podejściem w stylu systemu R)
(programowanie dynamiczne)

BD – wykład 12 (14)

Kolejnym etapem fazy dekompozycji jest etap restrukturyzacji, czy też transformacji zapytania. Zanim jednak przejdziemy do przedstawienia podstawowych reguł transformacji, wróćmy na chwilę do problemu konstrukcji podstawowych bloków zapytania. Tradycyjne podejście do konstrukcji bloków zapytania opera się na zastosowaniu transformacji algebraicznych, które zostaną przedstawione w dalszej części wykładu. jednakże, zbiór stosowanych transformacji różni się zasadniczo dla różnych systemów komercyjnych. Co więcej, nie wszystkie transformacje gwarantują minimalizację czasu wykonania danego bloku. W ostatnim czasie, coraz częściej, konstrukcja bloków opiera się na optymalizacji kosztowej, w której, dla każdego bloku, konstruujemy możliwe plany wykonania danego bloku i szacujemy koszt i rozmiar wykonania każdego planu. Ostatecznie wybierany jest plan wykonania o najniższym szacowanym koszcie. Do zakończenia procesu optymalizacji pozostaje jeszcze znalezienie najlepszego drzewa operacji połączenia, łączącego wyniki wykonania bloków zapytania. Tradycyjne podejście do problemu znajdowania najlepszego drzewa operacji połączenia (nazywane często podejściem w stylu systemu R) polega na zastosowaniu algorytmu programowania dynamicznego.



Problemy z optymalizacją

- Istnieje bardzo wiele podejść
- Konieczność uwzględniania bardzo wielu czynników
 - Klasyczny problem optymalizacyjny
 - **Optymalizacja zapytań nie została tak naprawdę jeszcze rozwiązana**
- Możliwe kierunki poprawy efektywności:
 - Nowe reguły algebraicznej transformacji złożonych zapytań
 - Nowe metody znajdowanie kolejności wykonywania operacji połączenia:
 - Nowe metody szacowania kosztów i rozmiarów wyników pośrednich zapytania

BD – wykład 12 (15)

Generalnie, problem optymalizacji zapytań jest problemem bardzo trudnym. Istnieje bardzo wiele podejść do optymalizacji. De facto, istnieje tyle podejść ile jest na rynku systemów zarządzania bazami danych. Złożoność problemu optymalizacji wynika z konieczności uwzględniania w procesie optymalizacji bardzo wielu czynników, które, dodatkowo, w trakcie wykonywania zapytania mogą ulegać modyfikacjom i zmianom (np. charakterystyka relacji i atrybutów, obciążenie stanowisk w systemie, fluktuacja obciążenia sieci, itp.). Należy stwierdzić, że problem optymalizacji zapytań nie został tak naprawdę jeszcze rozwiązany. Jest to szczególnie widoczne w przypadku dużych zapytań i aplikacji występujących w systemach wspomagania podejmowania decyzji. Ciągle trwają prace badawcze nad dalszą poprawą efektywności metod optymalizacji wykonywania zapytań, szczególnie, dla nowych typów danych (zbiory, sekwencje, grafy, dokumenty XML). Możliwe kierunki poprawy efektywności obejmują: opracowanie nowych reguł algebraicznej transformacji złożonych zapytań, opracowanie nowych metod znajdowanie kolejności wykonywania operacji binarnych (w tym szczególnie, operacji połączenia), oraz opracowanie nowych metod szacowania kosztów i rozmiarów wyników pośrednich zapytania. Po tym krótkim przedstawieniu problemów związanych z optymalizacją zapytań, wróćmy do reguł transformacji wyrażeń reprezentujących zapytania.



Operacje

- Operacja skanowania:
 - Skanowanie indeksu lub relacji
- Selekcja (filtrowanie)
- Projekcja (czy zawsze wymaga dostępu do danych?)
- Wszystkie operacje unarne staramy się przesunąć w dół drzewa zapytania
- Połączenie: nested loop (indeksowane), sort-merge, hash-join
- Grupowanie i agregacja (najczęściej wykonywane na końcu)

BD – wykład 12 (16)

Każdy plan wykonania zapytania jest częściowo uporządkowanym zbiorem operacji. W skład tego zbioru operacji wchodzą: operacja skanowania, selekcji, projekcji, połączenia, produktu kartezjańskiego, operacje grupowania i agregacji. Problem znalezienia najlepszego planu wykonania zapytania obejmuje, z jednej strony, określenie kolejności wykonania operacji wchodzących w skład zapytania, z drugiej, określenia metody wykonania poszczególnych operacji. Przykładowo, mamy dwie metody dostępu do relacji: bezpośrednie skanowanie (odczyt) relacji lub dostęp do relacji poprzez skanowanie indeksu założonego na relacji. Podstawowa reguła optymalizacji mówi, że wszystkie operacje unarne (projekcja i selekcja) należy przesunąć w dół drzewa zapytania, tzn. wykonywać w pierwszej kolejności. Operacje te charakteryzują się silną własnością redukcji (filtrowania) przetwarzanych danych. Redukując rozmiar przetwarzanych danych, operacje unarne prowadzą do poprawy efektywności wykonywania operacji binarnych. Dlatego, operacje binarne (połączenie, produkt kartezjański) należy przesunąć w kierunku korzenia drzewa zapytania. Dla operacji binarnych, np. połączenia, poza określeniem kolejności ich wykonywania, należy wybrać również metodę ich wykonania (dla połączenia - nested loop, sort-merge, hash-join). Najczęściej, na końcu planu wykonania zapytania znajdują się operacje grupowania i agregacji.



Prawa algebra relacji (1)

Reguły przemienności i łączności operacji

$$R \cup S = S \cup R, R \cup (S \cup T) = (R \cup S) \cup T$$

$$R \cap S = S \cap R, R \cap (S \cap T) = (R \cap S) \cap T$$

$$R \bowtie S = S \bowtie R, R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$

Reguły dystrybutywności

$$R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$$

BD – wykład 12 (17)

Podstawowe reguły transformacji wyrażeń reprezentujących zapytania wynikają z praw algebra relacji i obejmują, m. in. reguły przemienności i łączności operacji oraz reguły dystrybutywności. Przykładowe reguły przemienności i łączności operacji oraz reguły dystrybutywności operacji przedstawiono na slajdzie. Zgodnie z przedstawioną regułą dystrybutywności, wyrażenie będące połączeniem relacji R oraz relacji, będącej sumą relacji S i T, transformujemy do sumy relacji będących połączeniem relacji R z S i R z T. Reguła ta pozwala przesunąć operator sumy za operator połączenia. Ponieważ koszt operacji połączenia silnie zależy od rozmiarów łączonych relacji, zastosowanie transformacji, opartej o regułę dystrybutywności, transformuje wyjściowe wyrażenie do sumy relacji będących połączeniem mniejszych relacji, co powinno skutkować niższym kosztem wykonania całego wyrażenia. Reguła łączności operacji połączenia pozwala zmienić kolejność łączonych relacji. Łącząc, w pierwszej kolejności, mniejsze relacje, zmniejszamy rozmiary częściowych wyników operacji połączenia, co również powinno prowadzić do zmniejszenia kosztu wykonania całego wyrażenia.



Prawa algebra relacji (2)

Reguły dotyczące operacji selekcji:

Kaskada selekcji

$$1 \quad \sigma_C \text{ AND } C'(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R)$$

$$\sigma_C \text{ OR } C'(R) = \sigma_C(R) \cup \sigma_{C'}(R)$$

$$3 \quad \sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$$

– jeżeli C zawiera tylko atrybuty relacji R

$$4 \quad \sigma_C(R \bowtie S) = \sigma_C(R) \bowtie \sigma_C(S)$$

$$\sigma_C(R - S) = \sigma_C(R) - S$$

$$\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$$

$$\sigma_C(R \cap S) = \sigma_C(R) \cap S$$

BD – wykład 12 (18)

Kolejny slajd przedstawia reguły transformacji dla operacji selekcji. Szczególne znaczenie ma pierwsza reguła dotycząca koniunkcji predykatów selekcji:

selekcja C AND C'(R) = selekcja C(selekcja C'(R))

Reguła ta wynika z komutatywności operatora selekcji i pozwala zmienić kolejność wykonywania operacji selekcji. W pierwszej kolejności zawsze wykonywana jest operacja selekcji o większym współczynniku selektywności, bardziej redukująca rozmiar relacji R. Zwróćmy uwagę na regułę nr 3:

selekcja C (R połączenie S) = selekcja C (R) połączenie S (jeżeli C zawiera tylko atrybuty relacji R)

oraz regułę nr 4:

selekcja C (R połączenie S) = (selekcja C (R)) połączenie (selekcja C (S)) (jeżeli C zawiera atrybuty obu relacji R i S)

Transformacja oparta o te reguły pozwala przesunąć operator selekcji przed operator połączenia, redukując rozmiary łączonych relacji, co prowadzi do zmniejszenia kosztu wykonania zapytania. Podobna uwaga dotyczy ostatnich trzech przedstawionych reguł, które pozwalają przesunąć operator selekcji przed operatory binarne sumy, iloczynu i różnicę.



Prawa algebra relacji (3)

Komutatywność selekcji i projekcji

$$\Pi_M(\sigma_C(R)) = \sigma_C(\Pi_M(R))$$

- Przykład: $R(A, B, C, D), S(E, F, G)$

$$-\ \sigma_{F=3} (R \bowtie_{D=E} S) = ?$$

$$-\ \sigma_{A=5 \text{ AND } G=9} (R \bowtie_{D=E} S) = ?$$

BD – wykład 12 (19)

Kolejny slajd przedstawia podstawową regułę transformacji wykorzystującą własność komutatywności operacji selekcji i projekcji. Przesuwając operator projekcji przed operator selekcji zmniejszamy rozmiar argumentu operatora selekcji.

Rozważmy dwa przykłady ilustrujące działanie przedstawionych dotychczas reguł transformacji. Dane są relacje $R(A, B, C, D)$ i $S(E, F, G)$.

Jaką regułę transformacji należy zastosować w odniesieniu do pierwszego wyrażenia?

Odpowiedź – reguła: selekcja z relacji R przed połączeniem z relacją S, tj. „selekcja C (R połączenie S) = selekcja C (R) połączenie S” (jeżeli C zawiera tylko atrybuty relacji R). Po zastosowaniu powyższej reguły, przedstawione wyrażenie zostanie przetransformowane do postaci: „selekcja F=3 (S) połączenie R”.

Rozważmy drugie z podanych wyrażeń. Jaką regułę transformacji należy zastosować w odniesieniu do tego wyrażenia? Odpowiedź – złożenie dwóch reguł: reguły dotyczącej kaskady selekcji oraz reguły dotyczącej dystrybutywności selekcji i połączenia. Po zastosowaniu powyższych reguł, przedstawione wyrażenie zostanie przetransformowane do postaci: „(selekcja G=9 (S)) połączenie (selekcja A=5 (R))”.



Prawa algebra relacji (4)

Reguły dotyczące projekcji

$$\Pi_M(R \bowtie S) = \Pi_N(\Pi_P(R) \bowtie \Pi_Q(S))$$

gdzie N, P, Q są odpowiednimi podzbiorami atrybutów zbioru M

$$\Pi_M(\Pi_N(R)) = \Pi_{M,N}(R)$$

- Przykład: R(A,B,C,D), S(E, F, G)
 - $\Pi_{A,B,G}(R \underset{D=E}{\bowtie} S) = \Pi_? (\Pi_?(R) \underset{D=E}{\bowtie} \Pi_?(S))$

BD – wykład 12 (20)

Kolejny slajd przedstawia podstawowe reguły transformacji wyrażeń zawierających projekcje. Pierwsza reguła transformacji wynika z reguły dystrybutywności operacji połączenia względem projekcji. Przesuwając operator projekcji przed operator połączenia zmniejszamy rozmiar argumentu operatora połączenia. Druga reguła transformacji dotyczy kaskady projekcji, które mogą być łączone w jedną operację projekcji. Rozważmy przykład ilustrujący działanie przedstawionych reguł transformacji. Dane są relacje R(A, B, C, D) i S(E, F, G). W jaki sposób można uzyskać przedstawioną na slajdzie transformację? Jakie będą predykaty projekcji w wynikowym wyrażeniu? Z przedstawionych powyżej reguł wynika, że predykaty projekcji w wynikowym wyrażeniu mają następującą postać:

projekcjaA,B,G(R połączenie S) = projekcja A, B, G ((projekcja A,B,D (R)) połączenie D=E (projekcja E,G(S)))



Przepisywanie zapytań: podzapytania (1)

```
Select Emp.Name  
From Emp, Dept  
Where   Emp.Age < 30  
      AND Emp.Dept# IN (Select Dept.Dept#  
                           From Dept  
                           Where Dept.Loc = "Seattle"  
                           AND Emp.Emp#=Dept.Mgr)
```

BD – wykład 12 (21)

Przejdziemy obecnie do przedstawienia bardziej specyficznych reguł transformacji zapytań. Dane jest zapytanie przedstawione na slajdzie. Jak łatwo zauważyc, przedstawione zapytanie zawiera skorelowane podzapytanie zagnieżdżone. Zapytania zawierające skorelowane podzapytania zagnieżdżone są kosztowne w realizacji, gdyż wymagają sprawdzenia, dla każdej krotki zapytania zewnętrznego, czy spełniony jest dla tej krotki warunek podzapytania skorelowanego. Klasyczna metoda transformacji takich zapytań polega na przepisaniu zapytania w taki sposób, aby usunąć zagnieżdżenie (ang. unnesting). Usunięcie zagnieżdżenia polega na zastąpieniu zagnieżdżenia operacją połączenia.



Przepisywanie zapytań: podzapytania (2)

```
Select Emp.Name  
From Emp, Dept  
Where   Emp.Age < 30  
        AND Emp.Dept#=Dept.Dept#  
        AND Dept.Loc = "Seattle"  
        AND Emp.Emp#=Dept.Mgr
```

BD – wykład 12 (22)

Kolejny slajd przedstawia przepisanie zapytania ze slajdu nr 21, w którym zastąpiono podzapytanie zagnieżdżone operacją połączenia. Pozostałe warunki selekcji podzapytania zagnieżdżonego zostały przeniesione o zapytania zewnętrznego. Usunięcie zagnieżdżenia zdecydowanie poprawia czas realizacji zapytania.



Transformacja zapytań zagnieżdżonych (1)

```
Select distinct x.name, x.maker  
From product x  
Where x.color= "blue"  
    AND x.price >= ALL (Select y.price  
                        From product y  
                        Where x.maker = y.maker  
                        AND y.color="blue")
```

BD – wykład 12 (23)

Rozważmy inny przykład zapytania zawierającego skorelowane podzapytanie zagnieżdżone przedstawiony na slajdzie. Przykład ten ilustruje, że nie zawsze jest możliwe przetransformowanie zapytania do postaci bez zagnieżdżenia, i że, de facto, dla każdego typu zapytania, zawierającego skorelowane podzapytanie zagnieżdżone, należałoby zdefiniować specyficzne reguły transformacji. Celem zapytania przedstawionego na slajdzie jest znalezienie nazwy najdroższego produktu w kolorze niebieskim, i nazwy jego producenta.



Transformacja zapytań zagnieżdżonych (2)

Obliczamy uzupełnienie:

```
Select distinct x.name, x.maker  
From product x  
Where x.color= "blue"  
    AND x.price < SOME (Select y.price  
        From product y  
        Where x.maker = y.maker)  
            AND y.color=„blue”
```

BD – wykład 12 (24)

Na kolejnych slajdach przedstawiono transformację zapytania ze slajdu nr 23 do postaci nie zawierającej skorelowanego podzapytania zagnieżdżonego. Pierwszym krokiem transformacji jest znalezienie uzupełnienia zapytania ze slajdu 23. Przedstawione na slajdzie zapytanie znajduje nazwy niebieskich produktów, i nazwy ich producentów, dla których istnieją produkty niebieskie o wyższej cenie. Zauważmy, że zapytanie cały czas jest zapytaniem zawierającym skorelowane podzapytanie zagnieżdżone. Transformujemy podane zapytanie do postaci, w której podzapytanie zastąpiono operacją połączenia.



Transformacja zapytań zagnieżdżonych (3)

Dokonujemy transformacji zapytania:

```
Select distinct x.name, x.maker  
From product x, product y  
Where x.color= "blue" AND x.maker = y.maker  
AND y.color="blue" AND x.price < y.price
```

Zapytanie zwraca dokładnie to czego nie szukamy

BD – wykład 12 (25)

Transformujemy podane zapytanie do postaci, w której podzapytanie zastąpiono operacją połączenia. Jak już wspominaliśmy, operacja usunięcia podzapytania prowadzi do istotnej minimalizacji czasu wykonania zapytania. Przedstawione na slajdzie zapytanie zwraca w wyniku dokładnie to czego nie szukamy.



Transformacja zapytań zagnieżdżonych (4)

```
Select x.name, x.maker  
From product x  
Where x.color = "blue")
```

EXCEPT (MINUS)

```
(Select x.name, x.maker  
From product x, product y  
Where x.color= "blue" AND x.maker = y.maker  
AND y.color="blue" AND x.price < y.price)
```

BD – wykład 12 (26)

Ostatnim krokiem transformacji jest odjęcie od zbioru wszystkich niebieskich produktów tych niebieskich produktów, dla których istnieją produkty niebieskie o wyższej cenie. W wyniku wykonania operacji EXCEPT znajdujemy poszukiwany zbiór najdroższych niebieskich produktów. Zauważmy, że wynikowe zapytanie nie zawiera skorelowanego podzapytania i jego koszt będzie niższy niż oryginalnego zapytania ze slajdu 23. Zauważmy, jednakże, że przedstawiona transformacja nie jest trywialna.



Redukcja rozmiarów relacji

- Nie zawsze jest możliwe przetransformowanie zapytań w taki sposób, aby nie zawierało podzapytań (szczególnie dla podzapytań skorelowanych)
- Możliwość użycia sekwencji operacji **półpołączenia** w celu redukcji rozmiarów relacji uczestniczących w podzapytaniu

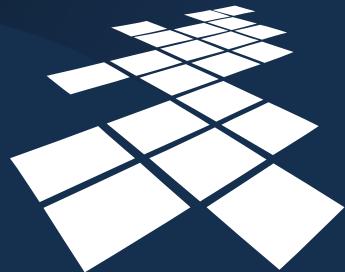
BD – wykład 12 (27)

Jak już wspomnieliśmy, zagadnienie optymalizacji jest zagadnieniem trudnym i istnieje bardzo wiele, specyficznych, reguł transformacji dla różnych typów zapytań. Co więcej, nie zawsze jest możliwe przetransformowanie zapytań w taki sposób, aby nie zawierało podzapytań (szczególnie dla podzapytań skorelowanych). W szczególnych przypadkach, gdy czas realizacji zapytania jest nieakceptowany, można zastosować technikę redukcji rozmiarów relacji uczestniczących w zapytaniu opartą o sekwencję operacji półpołączenia. Technika ta jest wykorzystywana do optymalizacji zapytań rozproszonych w systemach rozproszonych baz danych.

Optymalizacja zapytań

część II

Wykład przygotował:
Tadeusz Morzy



UCZELNIA
ONLINE

BD – wykład 13

Niniejszy wykład jest kontynuacją wykładu poświęconego problemom wykonywania i optymalizacji zapytań w systemach baz danych. Rozpoczniemy od dokończenia prezentacji technik przepisywania złożonych zapytań. Następnie, przejdziemy do przedstawienia zagadnień związanych z optymalizacją kosztową zapytań. Skoncentrujemy się na zagadnieniu szacowania rozmiarów wyników pośrednich wykonania podstawowych operacji w systemie bazy danych, takich jak: selekcja, projekcja, połączenie, agregacja. Przedstawimy, następnie, koncepcję histogramów i pokażemy w jaki sposób histogramy pozwalają dokładniej oszacować wynik wykonania operacji. Na zakończenie wykładu powiemy o typach drzew zapytań i zagadnieniu określania porządku wykonywania operacji połączenia.

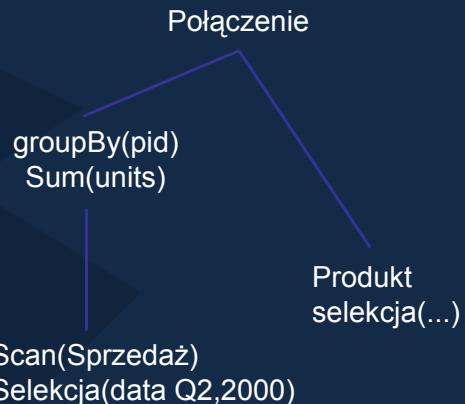


Przepisywanie zapytań: agregacja i połączenie

- **Schemat:**

- Produkt (**p_id**, cena,...)
- Sprzedaż (**s_id**, data, sklep, **p_id**, ilość)

- **Drzewo zapytania:**



BD – wykład 13 (2)

Prezentowany slajd przedstawia podstawową regułę transformacji wyrażeń zawierających operację połączenia i operację agregacji danych. Dane jest drzewo zapytania, przedstawione na slajdzie, które, dla każdego produktu, znajduje łączną liczbę sprzedanych produktów. Zapytanie może zawierać pewne dodatkowe predykaty selekcji. Podstawowa reguła transformacji takich drzew polega na przesunięciu operatora agregacji przed operator połączenia. W konsekwencji, jeden z argumentów operatora połączenia posiada znacznie mniejszy rozmiar, co prowadzi do minimalizacji czasu wykonywania zapytania.



Przykładowy schemat

Wykładowca (*pid, nazwisko, stanowisko, wiek*)

Wykład (*pid, wid, dzień, nazwa*)

- **Wykład:**

- Atrybuty: pid – id_wykładowcy, wid – id_wykładu,
- dzień – dzień, kiedy odbywa się wykład, nazwa – nazwa wykładu
- Rozmiar krotki - 40, bfr = 100 krotek/strona, 1000 stron (4000 krotek)

- **Wykładowca :**

- Atrybuty: pid – id_wykładowcy, nazwisko, stanowisko, wiek
- Rozmiar krotki - 50, bfr = 80 krotek/strona, 500 stron (40000 krotek)

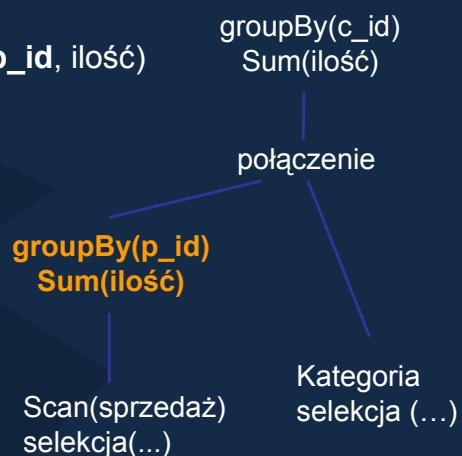
BD – wykład 13 (3)

Zanim przejdziemy do prezentacji dalszych reguł transformacji, wprowadzimy obecnie przykładową bazę danych, do której będziemy się odwoływać w dalszej części wykładu. Baza danych składa się z dwóch relacji: Wykładowca i Wykład. Kluczami relacji są atrybuty podkreślone linią ciągłą. Na slajdzie podane są rozmiary obu relacji.



Przepisywanie zapytań: wstawianie operacji

- **Schemat:** (p_id określa jednoznacznie c_id)
 - Kategoria (p_id , c_id , opis)
 - Sprzedaż (s_id , data, sklep, p_id , ilość)
- **Drzewo zapytania:**



BD – wykład 13 (4)

Kolejny slajd przedstawia zmodyfikowaną regułę transformacji wyrażeń zawierających operację połączenia i operację agregacji danych. Dane jest drzewo zapytania, przedstawione na slajdzie, które, dla każdej kategorii produktów, znajduje łączną liczbę sprzedanych produktów danej kategorii. Zauważmy, że przedstawiona na slajdzie transformacja, polegająca na przesunięciu operatora agregacji przed operator połączenia, nie może być bezpośrednio zastosowana, gdyż operator agregacji jest zdefiniowany na atrybutie, który nie występuje w relacji Sprzedaż. Przedstawiona na slajdzie modyfikacja reguły ze slajdu 2 polega na wprowadzeniu do drzewa zapytania dodatkowej operacji – w tym wypadku operacji agregacji. Wprowadzając operator agregacji GroupBy(p_id) zmniejszamy w istotny sposób rozmiar argumentu operacji połączenia. Operator GroupBy(c_id) pozostaje na swoim miejscu.



Przepisywanie zapytań: przesuwanie predykatów (1)



Przesuwamy predykaty selekcji w kierunku liści drzewa – mniej krotek weźmie udział w operacji połączenia. Wada?

BD – wykład 13 (5)

Kolejny slajd przedstawia regułę transformacji wykorzystującą własność dystrybutywności operacji selekcji i połączenia (transformacja przesuwania predykatów). Przesuwając operatory selekcji przed operator połączenia zmniejszamy rozmiary argumentów operatora połączenia. Oszacowanie opłacalności tej reguły jest trudne. Po pierwsze, jest to trudne ze względu na konieczność materializacji wyników wykonania operacji selekcji. Po drugie, automatycznie tracimy możliwość wykorzystania indeksów do wykonania operacji połączenia. Łączymy relacje o mniejszej liczbie krotek, ale musimy zastosować nieefektywne metody wykonania operacji połączenia.

Do kwestii opłacalności mechanizmu przepisywania zapytań (transformacji zapytań) wróćmy jeszcze w dalszej części wykładu.



Przepisywanie zapytań: przesuwanie predykatów (2)

```
Select wid, Max(wiek)
From wyklad w, wykadowca wk
Where w.pid=wk.pid
GroupBy wid
Having Max(wiek) > 40
```

```
Select wid, Max(wiek)
From wyklad w, wykadowca wk
Where w.pid=wk.pid and
wk.wiek > 40
GroupBy wid
```

Zapytanie: Dla każdego wykładu, znajdź wiek najstarszego wykładowcy, który prowadzi ten wykład

- Zaleta: minimalizacja wyniku wykonania operacji połączenia
- Wymaga zastosowania reguł specyficznych dla operatorów grupowania i agregacji

BD – wykład 13 (6)

Prezentowany slajd również przedstawia regułę transformacji wykorzystującą technikę przesuwania predykatów selekcji przed operator połączenia. Rozważmy zapytanie przedstawione na slajdzie, którego celem jest znalezienie, dla każdego wykładu (wid), wieku najstarszego wykładowcy, który prowadzi ten wykład. Zapytanie przedstawione po lewej stronie zawiera klauzulę „HAVING max(wiek) > 40”. W oryginalnym drzewie zapytania, ten predykat jest weryfikowany po wykonaniu operacji połączenia relacji Wykładowca i relacji Wykład. Przesuwając predykat selekcji, wyspecyfikowany przez tę klauzulę, przed operator połączenia, minimalizujemy rozmiar jednego z argumentów operacji połączenia. W tym przypadku, minimalizujemy, prawdopodobnie, również rozmiar wyniku wykonania operacji połączenia. Niestety, przedstawiona reguła jest specyficzna i pozwala przesunąć predykat selekcji, zdefiniowany klauzulą HAVING, ze względu na typ operatora agregacji. Czy można zastosować tą regułę w przypadku, gdy operatorem agregacji jest operator MIN?



Przepisywanie zapytań: podsumowanie

- Można opracować bardzo wiele poprawnych semantycznie reguł przepisywania zapytań
- Celem reguł jest minimalizacja kosztu wykonania zapytania – czy każda reguła minimalizuje koszt?
- Przepisywanie zapytań szczególnie ważne dla dużych złożonych zapytań
- Problem wyboru reguł transformacji
- Przepisywanie zapytań musi być uzupełnione o moduł szacowania kosztu wykonania planu danego zapytania

BD – wykład 13 (7)

Podsumowując przedstawione reguły transformacji i przepisywania zapytań, można stwierdzić, że istnieje bardzo wiele poprawnych semantycznie reguł przepisywania zapytań. Co więcej, można opracować bardzo wiele, bardzo specyficznych reguł przepisywania zapytań. Pytanie, na które, często jest trudno odpowiedzieć brzmi - czy każda reguła transformacji minimalizuje koszt wykonania zapytania? Przykładem tutaj może być nasza wcześniejsza dyskusja na temat przesuwania predykatów selekcji przed operator połączenia.

Niemniej, technika transformacji i przepisywania zapytań jest ważnym elementem procesu optymalizacji zapytań. Jest ona szczególnie istotna dla dużych złożonych zapytań, dla których optymalizacja kosztowa, o której powiemy za chwilę, jest zbyt „kosztowna” i, najczęściej, mało efektywna.

W poprzednich latach, proces optymalizacji zapytań ograniczał się do optymalizacji regułowej, nazywanej czasami syntaktyczną. Optymalizacja regułowa okazała się mało skuteczna. W późniejszym okresie, optymalizacja regułowa została uzupełniona o mechanizm optymalizacji kosztowej. Należy jeszcze wspomnieć, w odniesieniu do optymalizacji regułowej, o problemie wyboru reguł. Dane jest zapytanie i zbiór reguł transformacji, który można zastosować w stosunku do zapytania. Jakie reguły zastosować?



Szacowanie kosztu wykonania planu zapytania

- Dla każdego planu wykonania zapytania optymalizator musi oszacować:
 - **Szacunkowy koszt** wykonania każdego operatora w planie zapytania (zależny od rozmiarów argumentów wejściowych operatora)
 - **Szacunkowy rozmiar wyniku** wykonania każdego operatora w planie zapytania (dane znajdują się w katalogu bazy danych, dla operacji selekcji i projekcji zakładamy niezależność atrybutów)

BD – wykład 13 (8)

Jak już wspomnieliśmy, zdecydowana większość komercyjnych systemów zarządzania bazami danych stosuje zarówno optymalizację regułową jak i optymalizację kosztową. Optymalizacja kosztowa polega, najogólniej mówiąc, na wygenerowaniu wszystkich możliwych planów wykonania zapytania, oszacowaniu kosztu wykonania tych planów i wyborze planu o „najmniejszym” koszcie. Najczęściej, wygenerowanie wszystkich możliwych planów wykonania jest niemożliwe, stąd, optymalizator ogranicza się do pewnego podzbioru planów wykonania zapytania.

Kluczowe znacznie ma w przypadku optymalizacji kosztowej oszacowanie kosztu wykonania danego planu. Dla każdego planu wykonania zapytania optymalizator musi oszacować:

- Szacunkowy koszt wykonania każdego operatora w planie zapytania (zależny od rozmiarów argumentów wejściowych operatora)
- Szacunkowy rozmiar wyniku wykonania każdego operatora w planie zapytania (dane znajdują się w katalogu bazy danych, dla operacji selekcji i projekcji zakładamy niezależność atrybutów).



Katalog bazy danych

- Katalog bazy danych zawiera informacje o:
 - # krotek relacji R ($\text{card}(R)$) i # stron ($\text{Pcard}(R)$) dla każdej relacji, czasami # różnych wartości atrybutu A relacji R ($\text{val}(A[R])$)
 - # różnych wartości atrybutu (NKeys) i # stron dla każdego indeksu
 - Wysokość indeksu, minimalna i maksymalna wartość klucza indeksu (Low/High)
- Katalog odświeżany periodycznie (bieżąca aktualizacja zbyt kosztowna)
- Niektóre systemy przechowują histogramy wartości niektórych atrybutów

BD – wykład 13 (9)

Podstawowym źródłem informacji o relacjach, ich rozmiarach, itp. jest katalog bazy danych.

Katalog bazy danych zawiera informacje o:

– liczbie krotek każdej relacji R ($\text{card}(R)$) i liczbie stron ($\text{Pcard}(R)$) dla każdej relacji, czasami katalog przechowuje również informacje o liczbie różnych wartości każdego atrybutu relacji ($\text{val}(A[R])$),
– liczbie różnych wartości atrybutu (NKeys) i liczbie stron dla każdego indeksu,
– wysokości indeksu, minimalnej i maksymalnej wartości klucza indeksu (Low/High).

Katalog jest odświeżany periodycznie. Bieżąca, ciągła, aktualizacja katalogu, w przypadku każdej zmiany parametrów systemu, jest zbyt kosztowna. W ostatnim czasie, coraz częściej, systemy zarządzania bazami danych przechowują histogramy dla atrybutów relacji.



Szacowanie rozmiarów i współczynnik selektywności (1)

Select lista atrybutów
From lista relacji
Where **term₁** and **term₂**

- Rozważmy zapytanie:
- Maksymalny rozmiar wyniku = iloczyn rozmiarów relacji w klauzuli **FROM**
- **Współczynnik selektywności (sf)** związany z każdym predykatem term odzwierciedla wpływ predykatu term na redukcję rozmiaru wyniku

BD – wykład 13 (10)

Proces szacowania kosztu wykonania danego planu zapytania opiera się o oszacowanie rozmiarów wyników wykonania operatorów wchodzących w skład danego planu. Przedstawimy obecnie, na kolejnych slajdach, metody szacowania rozmiarów wyników wykonania poszczególnych operatorów. Rozpoczniemy od wprowadzenia współczynnika selektywności. Rozważmy ogólną postać zapytania, wyrażonego w języku SQL, przedstawioną na slajdzie. Górnne oszacowanie rozmiaru wyniku zapytania będzie równe iloczynowi rozmiarów relacji wyspecyfikowanych w klauzuli FROM. Oczywiście, to oszacowanie jest oszacowaniem maksymalnym, które, najczęściej, znacznie odbiega od rzeczywistego rozmiaru wykonania zapytania. Szacowanie wyniku wykonania zapytania bazuje głównie na wykorzystaniu w procesie szacowania tak zwanego współczynnika selektywności, który jest związany z każdym predykatem term i odzwierciedla wpływ predykatu na redukcję rozmiaru wyniku.



Szacowanie rozmiarów i współczynnik selektywności (2)

Rozmiar wyniku = max # krotek * iloczyn wszystkich sf

- Założenie: predykaty niezależne
- Predykat $atrybut=wartość$, wsp. selektywności $sf = 1/val(atr)$, gdzie atr oznacza atrybut (patrz selekcja)
- Predykat $atrybut=wartość$, wsp. selektywności $sf = 1/NKeys(l)$, jeżeli dany jest indeks l na atrybutie
- Predykat $atr1=atr2$, wsp. selektywności $sf = 1/MAX(NKeys(l1), NKeys(l2))$, dane indeksy l1 i l2
- Predykat $atr > wartość$ wsp. selektywności $sf = (High(l)-value)/(High(l)-Low(l))$

BD – wykład 13 (11)

Oszacowanie wyniku wykonania zapytania, przedstawionego na poprzednim slajdzie, z którym związany jest zbiór predykatów selekcji (term1, term2, ..., term k), jest zdefiniowane następującym wzorem:

rozmiar wyniku jest równy maksymalnej liczbie krotek * iloczyn wszystkich współczynników selektywności związanych z poszczególnymi predykatami selekcji. Podane powyżej oszacowanie wyniku zapytania zakłada, że predykaty są niezależne, co najczęściej nie jest spełnione w praktyce, ale istotnie ułatwia oszacowanie wyniku.

W jaki sposób obliczana jest wartość współczynnika selektywności dla danego predykatu?

Wartości współczynnika selektywności, dla poszczególnych typów predykatów przedstawiono na slajdzie. Przykładowo, jeżeli predykat term posiada postać „*atrybut = wartość*”, wówczas wartość współczynnika selektywności wynosi $1/val(atrybut)$, gdzie $val(atrybut)$ oznacza liczbę różnych wartości danego atrybutu. Założmy, że predykat posiada postać „*miasto = 'Konin'*” i $val(miasto) = 5$ (atrybut miasto przyjmuje 5 różnych wartości). Wówczas, szacunkowa liczba krotek spełniających dany predykat wynosi $1/5$ liczby krotek. Przykładowo, jeżeli dana relacja Pracownicy posiada 1000 krotek i zapytanie do relacji jest sformułowane następująco:

Select nazwisko

From Pracownicy

Where miasto='Konin'

Wówczas optymalizator szacuje, że w wyniku zapytania otrzymamy $1/5 * 1000 = 200$ krotek.

Oczywiście, ze względu na rozkład wartości atrybutu *miasto*, podane oszacowanie może się istotnie różnić od rzeczywistego rozmiaru wyniku.



Szacowanie rozmiarów selection (1)

Niech S oznacza wynik wykonania operacji selekcji na relacji R

- **Rozmiar relacji** ($card$) – z każdą selekcją związany jest współczynnik selektywności określający procent krotek spełniających predykat selekcji, dla prostego predykatu selekcji (Atrybut = wartość), sf definiujemy następująco:

1

$$sf = 1/val(A[R])$$

przy założeniu równomiernego rozkładu wartości krotek
Stąd:

2

$$card(S) = sf * card(R)$$

BD – wykład 13 (12)

Obecnie przejdziemy do bardziej szczegółowego przedstawienia szacowania rozmiarów wyników wykonania poszczególnych operatorów. Rozpoczniemy od operatora selekcji.

Niech S oznacza wynik wykonania operacji selekcji na relacji R. Niech card oznacza rozmiar relacji. Wówczas, dla prostego predykatu selekcji (Atrybut = wartość), sf definiujemy następująco:

$$sf = 1/val(A[R])$$

Przy założeniu równomiernego rozkładu wartości krotek:

$$card(S) = sf * card(R)$$

(patrz przykład relacji *Pracownicy* przedstawiony na poprzednim slajdzie).

Zauważmy, że selekcja nie wpływa na szerokość wynikowej relacji, stąd

$$size(S) = size(R)$$

(gdzie size(R) oznacza szerokość relacji R).



Szacowanie rozmiarów Selection (2)

- **Szerokość (size):** selekcja nie wpływa na szerokość wynikowej relacji

$$\text{size}(S) = \text{size}(R)$$

- **Liczba różnych wartości (val) :** zależy od predykatu selekcji

Rozważmy atrybut B, który nie uczestniczy w warunku selekcji. Określenie wartości $\text{val}(B[S])$ jest następujące: Dane $n=\text{card}(R)$ – dane równomiernie rozłożone pomiędzy $m = \text{val}(B[R])$ kolorów. Ile różnych kolorów $c = \text{val}(B[S])$ wybierzemy, jeżeli losowo wybierzemy r danych?

BD – wykład 13 (13)

Liczba różnych wartości (val) wyniku zależy, oczywiście, od predykatu selekcji. Jeżeli szacujemy liczbę różnych wartości atrybutu miasto w wyniku operacji selekcji, w przytaczanym przykładzie „miasto='Konin”, to oczywiście, wynik selekcji będzie zawierał krotki odnoszące się tylko i wyłącznie do jednego miasta– stąd, $\text{val}(\text{miasto}[S]) = 1$.

Rozważmy atrybut B, który nie uczestniczy w warunku selekcji. Określenie wartości $\text{val}(B[S])$ jest następujące: dane $n=\text{card}(R)$ (zakładamy, że krotki relacji R są równomiernie rozłożone pomiędzy $m = \text{val}(B[R])$ kolorów). Ile różnych kolorów $c = \text{val}(B[S])$ wybierzemy, jeżeli losowo wybierzemy r krotek z relacji R?



Szacowanie rozmiarów Selection (3)

- Aproksymacja Yao :**

$$c(n, m, r) = \begin{cases} r, & \text{dla } r < m/2 \\ (r+m)/3 & \text{dla } m/2 \leq r < 2m \\ m, & \text{dla } r \geq 2m \end{cases}$$

BD – wykład 13 (14)

Do oszacowania rozmiaru liczby różnych wartości atrybutu B w wyniku selekcji S ($c = \text{val}(B[S])$) stosuje się tak zwaną aproksymację Yao przedstawioną na slajdzie. Dla ilustracji rozważmy ponownie rozważany wcześniej przykład relacji Pracownicy. Niech $\text{card}(\text{Pracownicy}) = 1000$. Relacja Pracownicy zawiera, m. in. atrybuty *miasto* i *stanowisko*. Założymy, że liczba różnych wartości atrybutu miasto w relacji Pracownicy wynosi 5 ($\text{val}(\text{miasto}[Pracownicy]) = 5$) i $\text{val}(\text{stanowisko}[Pracownicy]) = 10$. Ile różnych wartości atrybutu stanowisko znajdzie się w wyniku zapytania S:

Select nazwisko, stanowisko

From Pracownicy

Where miasto='Konin'

Dane: $n=1000$ (krotek relacji Pracownicy), $m=10$ (różnych stanowisk), $r = 200 = \text{card}(S)$ (rozmiar wyniku zapytania). Stąd, zgodnie z aproksymacją Yao, $\text{val}(\text{stanowisko}[S]) = m = 10$.



Szacowanie rozmiarów Projection (1)

- Niech S oznacza wynik wykonania operacji projekcji na relacji R
- **Rozmiar relacji** (*card*) – projekcja wpływa na rozmiar relacji w przypadku usuwania duplikatów. Efekt trudny do oszacowania. Stosuje się trzy reguły do oszacowania:
 - Projekcja na pojedynczym atrybucie A
$$\text{card}(S) = \text{val}(A[R])$$
 - Jeżeli iloczyn $\prod_{A_i \in \text{Attr}(S)} \text{val}(A_i[R])$ mniejszy niż $\text{card}(R)$, gdzie $\text{Attr}(S)$ to zbiór atrybutów należący do wyniku projekcji, wówczas
 - $\text{card}(S) = \prod_{A_i \in \text{Attr}(S)} \text{val}(A_i[R])$

BD – wykład 13 (15)

Przejdziemy obecnie do przedstawienia oszacowania wyniku rozmiaru wykonania operacji projekcji. Podobnie jak poprzednio, niech S oznacza wynik wykonania operacji projekcji na relacji R. Jeżeli założymy, że operator projekcji usuwa duplikaty z wynikowej relacji, to oszacowanie rozmiaru relacji S jest trudne. Stosuje się trzy reguły oszacowania:

- Projekcja na pojedynczym atrybucie A: rozmiar wyniku zapytania S ($\text{card}(S) = \text{val}(A[R])$) jest równy liczbie różnych wartości atrybutu A w relacji R.
- Jeżeli iloczyn różnych wartości atrybutów wyspecyfikowanych w wyniku projekcji jest mniejszy niż $\text{card}(R)$, wówczas $\text{card}(S) = \text{iloczyn różnych wartości atrybutów wyspecyfikowanych w wyniku projekcji}$.
- Jeżeli projekcja zawiera klucz relacji R, wówczas $\text{card}(S) = \text{card}(R)$.



Szacowanie rozmiarów Projection (2)

- Jeżeli projekcja zawiera klucz relacji R, wówczas:

$$\text{card}(S) = \text{card}(R)$$

- Jeżeli projekcja nie eliminuje duplikatów, wówczas:

$$\text{card}(S) = \text{card}(R)$$

- Szerokość (Size):** szerokość wyniku projekcji jest równa sumie rozmiarów atrybutów wyspecyfikowanych w projekcji
- Liczba różnych wartości:** liczba różnych wartości atrybutów należących do wyniku projekcji S identyczna z liczbą różnych wartości tych samych atrybutów w relacji R

BD – wykład 13 (16)

Jeżeli założymy, że operator projekcji nie usuwa duplikatów z wynikowej relacji, to oszacowanie rozmiaru relacji S wynosi:

$$\text{card}(S) = \text{card}(R)$$

Szerokość wyniku projekcji jest równa sumie rozmiarów atrybutów wyspecyfikowanych w projekcji. Liczba różnych wartości atrybutów należących do wyniku projekcji S jest identyczna z liczbą różnych wartości tych samych atrybutów w relacji R.



Szacowanie rozmiarów GROUP BY

- Niech G oznacza zbiór atrybutów, na którym wykonywana jest operacja grupowania
- **Rozmiar relacji** – górne ograniczenie rozmiaru S:

$$\text{card}(S) \leq \prod_{A_i \in G} \text{val}(A_i[R])$$

- **Szerokość:** dla wszystkich atrybutów A należących do G

$$\text{size}(R.A) = \text{size}(S.A)$$

- **Liczba różnych wartości:** dla wszystkich atrybutów A należących do G

$$\text{val}(A[S]) = \text{val}(A[R])$$

BD – wykład 13 (17)

Przejdziemy obecnie do przedstawienia oszacowania wyniku rozmiaru wykonania operacji agregacji GROUP BY. Niech G oznacza zbiór atrybutów, na którym wykonywana jest operacja grupowania. Oszacowanie rozmiaru wynikowej relacji jest trudne. Stąd, system szacuje górne ograniczenia rozmiaru relacji S, które wynosi: $\text{card}(S) \leq \text{iloczyn różnych wartości atrybutów należących do zbioru } G$.

Dla wszystkich atrybutów A należących do G: $\text{size}(R.A) = \text{size}(S.A)$.

Podobnie, dla wszystkich atrybutów A należących do G: $\text{val}(A[S]) = \text{val}(A[R])$.



Szacowanie rozmiarów UNION

- Niech T oznacza wynik wykonania operacji sumy na relacjach R i S
- **Rozmiar relacji:**

$$\text{card}(T) \leq \text{card}(R) + \text{card}(S)$$

- Równość zachodzi w przypadku, gdy nie eliminujemy duplikatów
- **Szerokość:**

$$\text{size}(T) = \text{size}(R) = \text{size}(S)$$

- **Liczba różnych wartości :** górną granicą wynosi

$$\text{val}(A[T]) \leq \text{val}(A[R]) + \text{val}(A[S])$$

BD – wykład 13 (18)

Oszacowanie rozmiaru wykonania operacji sumy. Niech T oznacza wynik wykonania operacji sumy na relacjach R i S.

Górne ograniczenie rozmiaru relacji T wynosi: $\text{card}(T) \leq \text{card}(R) + \text{card}(S)$. Równość zachodzi w przypadku, gdy nie eliminujemy duplikatów. Szerokość relacji T wynosi: $\text{size}(T) = \text{size}(R) = \text{size}(S)$.

Górną granicą liczby różnych wartości atrybutu A w relacji T wynosi: $\text{val}(A[T]) \leq \text{val}(A[R]) + \text{val}(A[S])$.



Szacowanie rozmiarów JOIN (1)

- **Rozmiar relacji:** oszacowanie rozmiaru relacji T będącej wynikiem połączenia relacji R i S jest bardzo trudne, górne ograniczenie rozmiaru:

1

$$\text{card}(T) \leq \text{card}(R) \times \text{card}(S)$$

ale jest to bardzo duże przybliżenie. Najczęściej podaje się następujące przybliżenie rozmiaru relacji T:

2

$$\text{card}(T) = (\text{card}(R) \times \text{card}(S)) / \max\{\text{val}(A[R]), \text{val}(A[S])\}$$

gdzie A oznacza atrybut połączeniowy relacji R i S

BD – wykład 13 (19)

Przejdziemy obecnie do przedstawienia oszacowania wyniku rozmiaru wykonania operacji połączenia. Podobnie jak poprzednio, niech T oznacza wynik wykonania operacji połączenia relacji R i S.

Oszacowanie rozmiaru relacji T, będącej wynikiem połączenia relacji R i S, jest bardzo trudne. Można łatwo podać górne ograniczenie rozmiaru relacji T, które wynosi: $\text{card}(T) \leq \text{card}(R) * \text{card}(S)$, ale jest to bardzo duże przybliżenie. Najczęściej podaje się następujące przybliżenie rozmiaru relacji T:

$\text{card}(T) = (\text{card}(R) * \text{card}(S)) / \max\{\text{val}(A[R]), \text{val}(A[S])\}$, gdzie A oznacza atrybut połączeniowy relacji R i S.



Szacowanie rozmiarów JOIN (2)

- **Szerokość:**

1

$$\text{size}(T) = \text{size}(R) = \text{size}(S)$$

W przypadku połączenia naturalnego od szerokości wyniku odejmujemy szerokość atrybutu połączeniowego

- **Liczba różnych wartości :** jeżeli A jest atrybutem połączeniowym, górne ograniczenie wynosi:

2

$$\text{val}(A[T]) \leq \min(\text{val}(A[R]), \text{val}(B[S]))$$

jeżeli A nie jest atrybutem połączeniowym, górne ograniczenie wynosi:

3

$$\text{val}(A[T]) \leq \text{val}(A[R]) + \text{val}(B[S]))$$

BD – wykład 13 (20)

Szerokość relacji T jest równa sumie szerokości relacji R i S: $\text{size}(T) = \text{size}(R) + \text{size}(S)$.

W przypadku połączenia naturalnego, od szerokości wyniku odejmujemy szerokość atrybutu połączeniowego.

Niech A oznacza atrybut połączeniowy, wówczas górne ograniczenie liczby różnych wartości atrybutu A wynosi: $\text{val}(A[T]) \leq \min(\text{val}(A[R]), \text{val}(B[S]))$.

Jeżeli A nie jest atrybutem połączeniowym, wówczas liczba różnych wartości atrybutu A wynosi: $\text{val}(A[T]) \leq \text{val}(A[R]) + \text{val}(B[S])$.



Histogramy (1)

- Umożliwiają uzyskanie znacznie dokładniejszych oszacowań rozmiarów wyników wykonania operacji.
- Różne wersje:
 - **Equi-depth** (równa liczba wartości dla pojedynczego interwału)
 - **Equi-width** (równa szerokość każdego interwału)
- Histogramy są szczególnie przydatne do oszacowania wyniku operacji połączenia

BD – wykład 13 (21)

Jak już wspominaliśmy, szacowanie wyników wykonania operacji składających się na dany plan zapytania w oparciu o współczynniki selektywności charakteryzuje się dużą niedokładnością. Zaletą tego podejścia jest prostota i łatwość obliczania współczynników selektywności. W ostatnim czasie, coraz częściej, wykorzystuje się do szacowania rozmiarów wyników histogramy.

Histogramy umożliwiają uzyskanie znacznie dokładniejszych oszacowań rozmiarów wyników wykonania operacji. Histogramy występują w różnych wersjach:

- Histogram typu equi-depth (o równej głębokości) oznacza histogram, w którym dla każdego pojedynczego interwału histogramu występuje równa liczba wartości atrybutu;
- Histogram typu equi-width (o równej szerokości) oznacza histogram, w którym wszystkie interwały posiadają równą szerokość.

Ze względu na znaczenie operacji połączenia w systemach relacyjnych baz danych i trudności w oszacowaniu rozmiaru tej operacji, histogramy są szczególnie przydatne do oszacowania wyniku operacji połączenia.



Histogramy (2)

- Wykładowca(pid, nazwisko, zarobki, wiek)
- Przykładowy histogram na atrybucie *zarobki*:

Zarobki:	0..1k	1k..2k	2k..3k	3k..4k	4k..5k	> 5k
Krotki	10	30	25	25	9	4

Rozmiar relacji Wykładowca = 103, ale znamy rozkład wartości atrybutu

BD – wykład 13 (22)

Niniejszy slajd przedstawia przykładowy histogram dla atrybutu *zarobki* przedstawionej relacji *Wykładowca*. Przedstawiony histogram jest histogramem typu equi-with (o równej szerokości). Jak łatwo zauważać, histogram można interpretować jako aproksymację rozkładu wartości atrybutu *zarobki*. Przykładowo, z histogramu można odczytać, że 30 krotek relacji *Wykładowca* posiada wartość atrybutu zarobki z przedziału wartości (1000, 2000), natomiast 4 krotki posiadają wartość atrybutu *zarobki* powyżej 5000.



Histogramy (3)

- Etaty(etat, zarobki)
- Oszacujmy wynik połączenia relacji
Wykładowca \bowtie Etaty
zarobki

Wykładowca	0..1k	1k..2k	2k..3k	3k..4k	4k..5k	> 5k
	10	30	25	25	9	4

Etaty	0..1k	1k..2k	2k..3k	3k..4k	4k..5k	> 5k
	8	20	20	20	10	2

BD – wykład 13 (23)

Jak już powiedzieliśmy wcześniej, histogramy są szczególnie przydatne do oszacowania wyniku operacji połączenia. Rozważmy przykład przedstawiony na slajdzie. Dana jest relacja *Wykładowca*, przedstawiona na poprzednim slajdzie, oraz relacja *Etaty*. Oszacujmy wynik połączenia obu relacji, zakładając, że dane są histogramy dla atrybutu połączyciowego *zarobki*, przedstawione na slajdzie.



Operacje połączenia

- Dane relacje:
 - R: $\text{card}(R)$, $\text{val}(A[R])$
 - S: $\text{card}(S)$, $\text{val}(A[S])$
- Oszacowanie wyniku operacji połączenia $R \bowtie_A S$

$$\frac{\text{card}(R) * \text{card}(S)}{\max\{\text{val}(A[R]), \text{val}(A[S])\}}$$

- Dla przykładu Wykładowca $\bowtie_{\text{zarobki}} \text{Etaty}$
 Niech: $\text{val}(\text{zarobki}[\text{Etaty}]) = 20$
- $\text{val}(\text{zarobki}[\text{Wykładowca}]) = 25$
 oszacowanie wyniku operacji połączenia: $103 * 80 / 25 = 330$

BD – wykład 13 (24)

Oszacowanie rozmiaru wyniku wykonania operacji połączenia relacji *Wykładowca* i *Etaty*, w oparciu o oszacowanie selektywności przedstawione na slajdzie 19, daje 330 krotek, przy założeniu, że liczba różnych wartości atrybutu *zarobki* relacji *Wykładowca* wynosi 25, natomiast liczba różnych wartości atrybutu *zarobki* relacji *Etaty* wynosi 20. Oszacowanie to wynika z przedstawionego na slajdzie wzoru. Rozmiar relacji *Wykładowca* wynosi 103, rozmiar relacji *Etaty* wynosi 80, maksymalna liczba różnych wartości atrybutu *zarobki* w relacji *Etaty* i *Wykładowca* wynosi 25 ($\max\{\text{val}(\text{zarobki}[\text{Etaty}]), \text{val}(\text{zarobki}[\text{Wykładowca}])\}$). Stąd, rozmiar wyniku operacji połączenia wynosi: $103 * 80 / 25 = 330$.



Operacje połączenia: (histogramy)

- Niech: $\text{val}(\text{zarobki}[\text{Etaty}]) = 20$
 $\text{val}(\text{zarobki}[\text{Wykładowca}]) = 25$
- Then $\text{card}(\text{Wykładowca} \bowtie_{\text{zarobki}} \text{Etaty})$
- $= \sum_{i=1,6} \text{card}_i(\text{Wykładowca}) * \text{card}_i(\text{Etaty}) / 25$
 $= (10 \times 8 + 30 \times 20 + 25 \times 20 + 25 \times 20 + 9 \times 10 + 4 \times 2) / 250$
 $= 72$

Błąd oszacowania: ponad 400% !

BD – wykład 13 (25)

Oszacowanie rozmiaru wyniku wykonania operacji połączenia relacji *Wykładowca* i *Etaty*, z wykorzystaniem przedstawionych wcześniej histogramów, daje 72 krotki. Zauważmy, że błąd oszacowania poprzednią metodą jest ponad 400 razy większy! Ten przykład ilustruje przydatność histogramów do szacowania wyniku wykonania operacji połączenia.



Plany wykonania zapytań

- **Zadanie:** skonstruuj plan wykonania zapytania dla każdego bloku typu select-projection-group-by
- **Idea:** przeanalizuj wszystkie możliwe ścieżki dostępu (ang. **access path**) do krotek relacji. Wybierz „najtańszą” ścieżkę dostępu
- Operacje leżące na pojedynczej ścieżce planu wykonania zapytania są wykonywane łącznie - przetwarzanie potokowe (np., jeżeli wykorzystujemy indeks w celu selekcji krotek relacji, operacja projekcji jest wykonywana dla każdej wybranej krotki, która następnie przesyłana jest do operatora agregacji)

BD – wykład 13 (26)

Jak już wspominaliśmy wcześniej, optymalizator zapytań konstruuje plan wykonania zapytania dla każdego bloku typu select-projection-group-by. Następnie, wyniki poszczególnych bloków są łączone operatorami binarnymi (sumy, iloczynu, połączenia lub iloczynu kartezjańskiego). W przypadku pojedynczego bloku, optymalizator poszukuje najlepszego planu wykonania tego bloku. Punktem wyjścia jest znalezienie najlepszych ścieżek dostępu do relacji wyspecyfikowanych w zapytaniu. Optymalizator analizuje wszystkie możliwe ścieżki dostępu do krotek relacji i wybiera „najtańszą” ścieżkę dostępu.

W większości komercyjnych systemów zarządzania bazami danych, stosuje się tak zwany model operatorowy, w którym operacje leżące na pojedynczej ścieżce planu wykonania zapytania są wykonywane łącznie. Ma to na celu umożliwienie przetwarzania potokowego krotek (np., jeżeli wykorzystujemy indeks w celu selekcji krotek relacji, operacja projekcji jest wykonywana dla każdej wybranej krotki, która następnie przesyłana jest do operatora agregacji). W konsekwencji nie zachodzi potrzeba materializowania (tj. zapisywania na dysk do osobnego pliku temporalnego) częściowych wyników wykonania operacji danego planu.



Przykład

```
SELECT stanowisko
FROM wykładowca w
WHERE w.stanowisko='adiunkt';
```

- Jeżeli mamy indeks (I) na atrybutie stanowisko:
 - $(1/N\text{Keys}(I)) * \text{card}(R) = (1/10) * 40000$ otrzymanych krotek
 - **Indeks zgrupowany (clustered index):** $(1/N\text{Keys}(I)) * (N\text{Pages}(I)+N\text{Pages}(R)) = (1/10) * (50+500)$ otrzymanych stron (**= 55**)
 - **Indeks niezgrupowany (unclustered index):**
 $(1/N\text{Keys}(I)) * (N\text{Pages}(I)+N\text{Tuples}(R)) = (1/10) * (50+40000)$ otrzymanych stron
- Jeżeli mamy indeks na sid:
 - musimy odczytać wszystkie krotki/strony indeksu i pliku danych. Z indeksem zgrupowanym koszt operacji wyniesie 50+500
- Jeżeli zastosujemy operacje skanowania relacji (file scan), to koszt odczytu wyniesie 500 stron

BD – wykład 13 (27)

Prezentowany slajd przedstawia przykładowe zapytanie, składające się z pojedynczego bloku.

Optymalizator rozpoczyna analizę możliwych ścieżek dostępu do relacji *Wykładowca*. Jeżeli w systemie został zdefiniowany indeks (I) na atrybutie *stanowisko* relacji *Wykładowca*, to wówczas jedną z możliwych ścieżek dostępu do tej relacji jest dostęp poprzez zdefiniowany indeks. Oszacowanie kosztu dostępu do relacji za pomocą indeksu zależy od typu indeksu. Jeżeli jest to indeks gęsty, wówczas, zgodnie ze wzorem przedstawionym na slajdzie 11, wynikowy rozmiar selekcji wynosi $1/10 * 40000$ (liczba krotek relacji *Wykładowca*) = 4000 krotek. Liczba operacji odczytu stron bazy danych zależy od typu indeksu – czy jest to indeks zgrupowany czy też indeks nie zgrupowany. Oszacowanie liczby odczytywanych stron, dla tych dwóch typów indeksów, przedstawiono na slajdzie. W przypadku, gdy w systemie został zdefiniowany indeks (I) na kluczu relacji *Wykładowca* (sid), ale brak indeksu na atrybutie *stanowisko*, wówczas należy odczytać wszystkie strony indeksu i relacji. Liczba odczytanych stron wynosi wówczas 550 stron. W przypadku, gdy nie dysponujemy indeksem na relacji *Wykładowca*, pozostaje operacja skanowania relacji, której koszt wynosi 500 stron. Po przeanalizowaniu wszystkich możliwych ścieżek dostępu, optymalizator wybiera ścieżkę o „najmniejszym” koszcie.



Określenie porządku operacji połączenia

- $R1 \bowtie R2 \bowtie \dots \bowtie Rn$
- Drzewo połączeń:



Drzewo połączeń reprezentuje plan wykonania zapytania,
którego wierzchołkami są wyniki
wykonania bloków zapytania

BD – wykład 13 (28)

Po zakończeniu procesu optymalizacji bloków zapytania, optymalizator rozpoczyna poszukiwanie najlepszego planu połączenia wyników wykonania bloków. Problem ten można zdefiniować jako problem znalezienia najlepszej kolejności wykonania operacji połączenia (w ogólnym przypadku). Problemu znalezienia najlepszej kolejności wykonania operacji sumy lub produktu kartezjańskiego, jest znacznie łatwiejszy – wystarczy łączyć relacje w kolejności rosnących rozmiarów łączonych relacji (zaczynamy od relacji o najmniejszym rozmiarze i łączymy, kolejno, z relacjami o coraz większych rozmiarach). Kolejność porządku wykonania operacji połączenia można przedstawić w postaci tak zwanego drzewa połączeń (ang. join tree). Drzewo połączeń reprezentuje plan wykonania zapytania, którego wierzchołkami są wyniki wykonania bloków zapytania (relacje).

- Drzewa lewostronne zagnieżdżone:

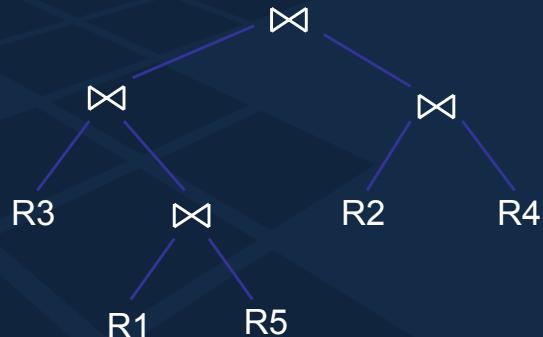


BD – wykład 13 (29)

Istnieje wiele struktur drzew połączeń. Większość komercyjnych systemów baz danych dopuszcza tylko niektóre z nich. Krótko przedstawimy możliwe i stosowane drzewa połączeń. Podstawowym typem drzew połączeń, stosowanym przez większość optymalizatorów zapytań, są tak zwane drzewa lewostronne zagnieżdżone (ang. left deep join tree). Cechą charakterystyczną tych drzew jest to, że sekwencja operacji połączenia jest wykonywana w sposób potokowy. Relacje (lub wyniki wykonania bloków zapytania) są zmaterializowane (relacje R3, R1, R5, R2, R4). Po połączeniu relacji R3 z R1, krotki tego połączenia mogą być przetwarzane potokowo – są one łączone z krotkami relacji, będącymi prawymi argumentami operacji połączenia. Zauważmy, że krotki otrzymywane w wyniku połączenia nie muszą być materializowane! Są one natychmiast konsumowane przez następny operator połączenia. Co więcej, w procesie łączenia relacji można wykorzystać indeksy na relacjach będących prawymi argumentami operacji połączenia. Większość systemów zarządzania bazami danych opuszcza tylko i wyłącznie drzewa tego typu. Dlaczego? Liczba możliwych drzew połączeń dla n relacji może być bardzo duża, natomiast liczba drzew lewostronne zagnieżdżonych jest znacznie mniejsza. W przypadku n relacji istnieje tylko jeden kształt drzewa zagnieżdżonego, do którego można dopasować relacje na n silnia sposobów.



- Drzewa krzaczaste:



BD – wykład 13 (30)

Kolejnym typem drzew połączzeń są tak zwane drzewa krzaczaste (ang. bushy tree). Plany zapytań reprezentowane w postaci drzew krzaczastych, z reguły, charakteryzują się krótszym czasem wykonania zapytania. Niestety, ich zasadniczą wadą jest konieczność materializacji wyników wykonania operacji połączenia. Materializacja wiąże się, z jednej strony, z koniecznością rezerwacji pamięci dyskowej, z drugiej, zwiększa narzuć czasowy na wykonanie planu zapytania. Łącząc relacje R1 i R5, wynik połączenia materializujemy na dysku, a następnie, ponownie odczytujemy ten wynik po to, aby wykonać połączenie z relacją R3. W przypadku systemów skoncentrowanych baz danych, większość optymalizatorów nie analizuje i nie generuje drzew krzaczastych, ze względu na materializację wyników częściowych, jak również, ze względu na fakt, że liczba możliwych drzew krzaczastych dla n łączonych relacji jest znacznie większa, aniżeli drzew zagnieżdżonych.



- Drzewa prawostronne zagnieżdżone:



BD – wykład 13 (31)

Kolejnym typem drzew połączeń są tak zwane drzewa prawostronne zagnieżdżone. Drzewa prawostronne zagnieżdżone są drzewami binarnymi, w których wynik wykonania poprzedniego połączenia jest prawym argumentem kolejnej operacji połączenia. Drzewa prawostronne zagnieżdżone charakteryzują się, najczęściej, niższą efektywnością, co wynika z implementacji algorytmów wykonywania operacji połączenia, stosowanych w systemach komercyjnych baz danych (dotyczy to głównie metody nested loop).



Problem znajdowania optymalnego drzewa połączeń

- Dane zapytanie: $R1 \bowtie R2 \bowtie \dots \bowtie Rn$
- Dana funkcja kosztu $\text{cost}()$ określająca koszt wykonania każdego drzewa połączeń

Znajdź najlepszy plan – drzewo połączeń o najniższym koszcie

BD – wykład 13 (32)

Wróćmy do problemu ustalenia optymalnej kolejności wykonywania operacji połączenia, lub inaczej mówiąc, do problemu znalezienia „najlepszego” drzewa połączenia. Założymy, że dana jest funkcja kosztu $\text{cost}()$, określająca koszt wykonania każdego drzewa połączeń. Zadaniem optymalizatora jest znalezienie drzewa połączeń o najniższym koszcie.



Problem znajdowania optymalnego drzewa połączeń (2)

- Problem znajdowania optymalnego drzewa połączeń można traktować jako problem optymalizacyjny - **problem poszukiwania** optymalnego stanu w przestrzeni stanów zawierającej wszystkie możliwe drzewa połączeń dla danego zapytania
Stan optymalny jest to stan o najmniejszej wartości funkcji kosztu
- Optymalizator zapytań jest charakteryzowany przez:
 - Reguły transformacji
 - Algorytm przeszukiwania przestrzeni stanów
 - Funkcja kosztu

BD – wykład 13 (33)

Zauważmy, że, de facto, problem znajdowania optymalnego drzewa połączeń można traktować jako problem optymalizacyjny - problem poszukiwania optymalnego stanu w przestrzeni stanów zawierającej wszystkie możliwe drzewa połączeń dla danego zapytania. Stan optymalny jest to stan o najmniejszej wartości funkcji kosztu.

Z tego punktu widzenia, można rozpatrywać działanie optymalizatora zapytań w trzech aspektach:

- Modelu planu wykonywania zapytania (struktura drzewa połączeń),
- Algorytmu przeszukiwania przestrzeni stanów,
- Funkcji kosztów.

Jak już wspominaliśmy, większość komercyjnych systemów zarządzania bazami danych ogranicza się do analizy drzew lewostronnie zagnieżdżonych. Mimo przyjęcia określonego kształtu drzewa połączeń, nadal liczba możliwych drzew, które należy przeanalizować, i których koszt należy oszacować, może być bardzo duża. Pozostaje zatem problem wyboru strategii przeszukiwania przestrzeni możliwych drzew połączeń. Funkcja kosztu, stosowana przez optymalizatory, opiera się na oszacowaniach rozmiarów wyników operatorów, które przedstawiliśmy ma poprzednich slajdach.



Algorytmy przeszukiwania

1. Algorytmy dokładne

- pełne przeszukiwanie (*exhaustive search*)
- programowanie dynamiczne

2. Algorytmy przybliżone

- heurystyki
- optymalizacja składniowa

3. Metaheurystyki - algorytmy kombinatoryczne

- *Iterative Improvement, Simulated Annealing, Tabu Search, algorytmy genetyczne*

BD – wykład 13 (34)

Jeżeli chodzi o algorytmy przeszukiwania przestrzeni możliwych drzew połączeń, to stosuje się trzy podstawowe podejścia:

- algorytmy dokładne,
- algorytmy przybliżone,
- metaheurystyki - algorytmy kombinatoryczne.

Algorytmy dokładne, stosowane, w praktyce, analizują wszystkie możliwe drzewa połączeń (tzw. pełne przeszukiwanie) lub ograniczają się do przejrzenia tylko niektórych podzbiorów. Najpopularniejszym algorytmem, stosowanym przez większość systemów komercyjnych, jest algorytm programowania dynamicznego. Koncepcja programowania dynamicznego polega na wypełnieniu tabeli kosztów wykonania połączeń tak, aby przechowywać tylko minimum danych potrzebnych do analizy. Tabela ta jest konstruowana, kolejno, dla pojedynczych relacji, par relacji, trojek relacji, itd. Wadą tych algorytmów jest zależność czasu optymalizacji od liczby operacji połączenia.

Algorytmy przybliżone, do znalezienia „najlepszego” planu stosują bądź heurystyki lub ograniczają się do optymalizacji regułowej, o której już mówiliśmy. Zasadniczą wadą tych algorytmów jest to, że, najczęściej, plan znaleziony przez algorytm przybliżony znacznie odbiega od „najlepszego” możliwego planu wykonania zapytania.

Trzecie z możliwych podejść do znajdowania najlepszego drzewa połączeń polega na zastosowaniu algorytmów kombinatorycznych takich jak: Iterative Improvement, Simulated Annealing, Tabu Search, algorytmy genetyczne. Algorytmy te pozwalają na znaczną penetrację przestrzeni możliwych drzew połączeń, a jednocześnie, czas optymalizacji nie zależy od liczby operacji połączenia. Jak pokazują badania, algorytmy, mimo, że nie znajdują optymalnych drzew połączeń, znajdują jednak plany znacznie efektywniejsze niż w przypadku stosowania prostych heurystyk. Jednak ich zasadniczą zaletą, w stosunku do algorytmów dokładnych, jest to, że pozwalają one optymalizować nawet bardzo duże zapytania (o liczbie połączeń dochodzących do 100 operacji).

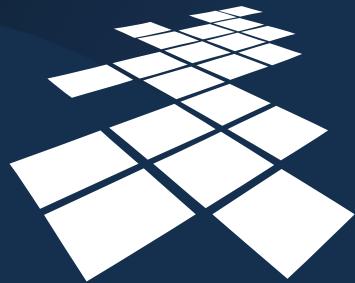
Problem optymalizacji zapytań jest ciągle obszarem aktywnych badań naukowych. Wiąże się to, z jednej strony, z problemem coraz bardziej złożonych zapytań i nowych typów danych, z drugiej, z nowymi potrzebami wynikającymi z popularności Internetu.

Systemy rozproszonych baz danych – 1

Problematyka rozproszonych baz danych

Wykład przygotował:

Robert Wrembel



UCZELNIA
ONLINE

ZSBD – wykład 1 (1)



Plan wykładu

- Wprowadzenie do problematyki
- Definicja rozproszonej bazy danych
- 12 reguł Date
- Podstawowa architektura systemu rozproszonej bazy danych

ZSBD – wykład 1 (2)

Celem pierwszego wykładu jest wprowadzenie do problematyki rozproszonych baz danych. Zostaną tu omówione:

- definicja rozproszonej bazy danych,
- dwanaście uznanych reguł zdefiniowanych przez C.J.Date, które powinien spełniać system rozproszonej bazy danych,
- podstawową architekturę sfederowanej bazy danych, jak przykład architektury implementacyjnej.



Wprowadzenie

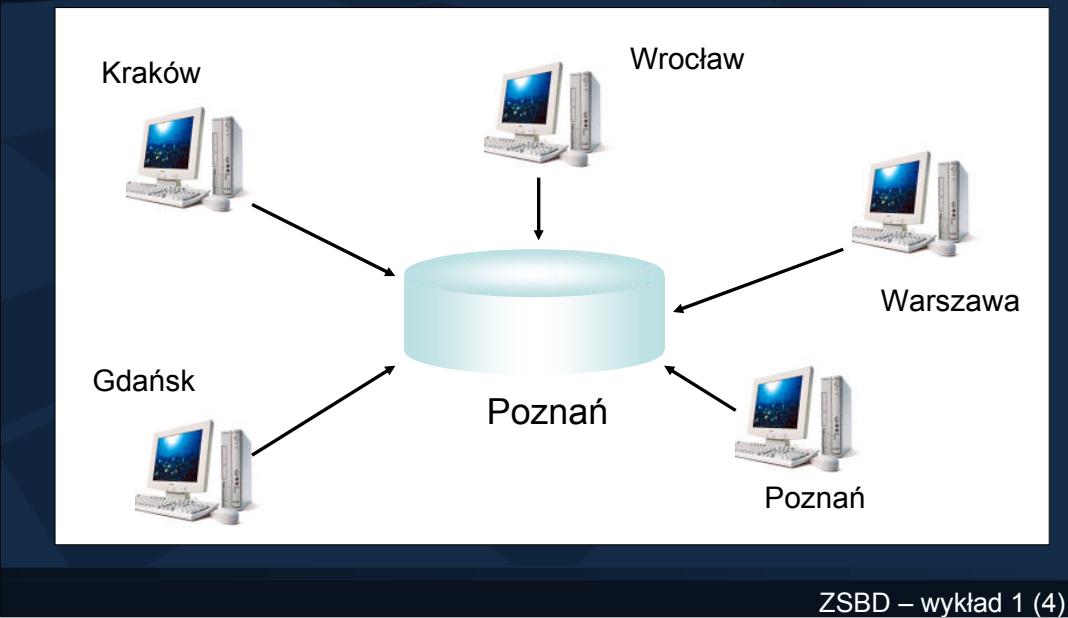
- Typowa architektura systemu informatycznego
 - scentralizowana baza danych
 - aplikacje klient-serwer lub w architekturze 3-warstwowej
- Scentralizowane bazy danych nie zawsze oferują wymaganą funkcjonalność i zadowalającą efektywność
⇒ rozproszone bazy danych (RBD)

ZSBD – wykład 1 (3)

W typowych zastosowaniach systemów baz danych wykorzystuje się architekturę scentralizowaną, w której system zarządzania bazą danych (SZBD) i wszystkie dane znajdują się w tym samym węźle sieci informatycznej. Dostęp do takiej bazy danych jest realizowany albo za pomocą aplikacji pracujących w architekturze klient-serwer albo pracujących w architekturze 3-warstwowej. Istnieje jednak wiele zastosowań, w których scentralizowane bazy danych nie zapewniają wymaganej funkcjonalności i efektywności pracy. W takich przypadkach, stosuje się tzw. rozproszone bazy danych.



Przykład - scentralizowana BD

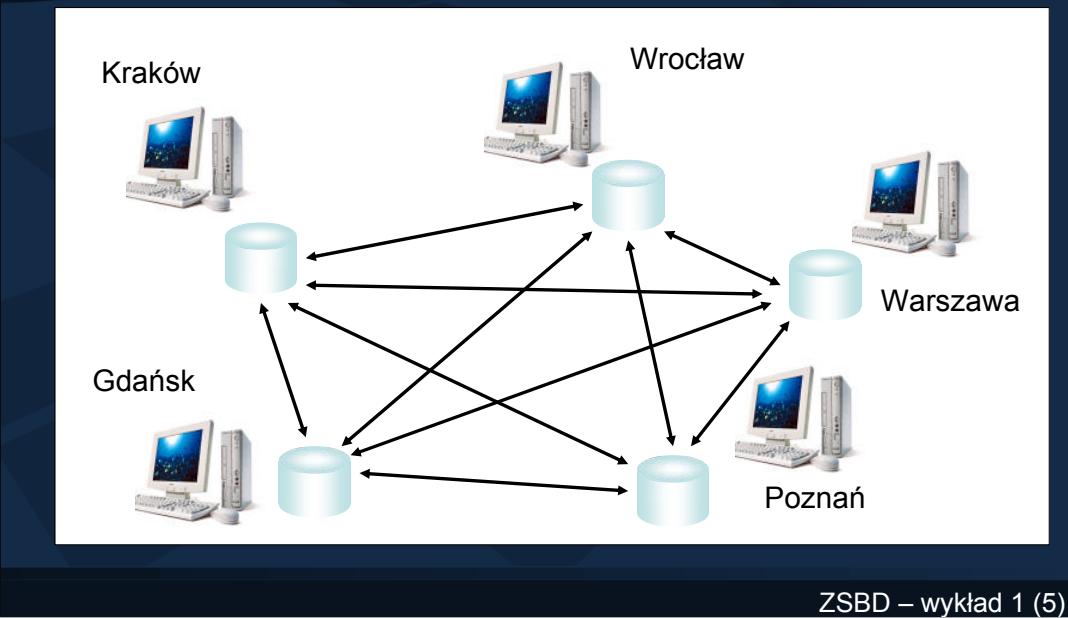


ZSBD – wykład 1 (4)

Przykładowo, rozważmy sieć dużych warsztatów samochodowych w Polsce, z kilkoma oddziałami w każdym dużym mieście. Gdyby zbudować system informatyczny dla tej sieci oparty o scentralizowaną bazę danych umieszczoną np. w Poznaniu, wówczas każde odwołanie do tej bazy z innego miasta wymagałoby transmisji sieciowej. Przy sieci o niskiej przepustowości i dużej częstotliwości odwołań, poprawne wykorzystywanie systemu stałoby się niemożliwe. Dodatkowo, taki system byłby znacznie bardziej podatny na awarie niż system rozproszony. Awaria serwera scentralizowanej bazy danych powodowałaby niemożliwość korzystania z systemu we wszystkich oddziałach firmy!



Przykład - rozproszona BD



ZSBD – wykład 1 (5)

Alternatywnym rozwiążaniem do przedstawionego poprzednio jest zastosowanie wielu lokalnych baz danych, np. po jednej w każdym dużym mieście, czyli tzw. systemu rozproszonych baz danych. Każda z tych baz lokalnych przechowywałaby informacje o klientach z danego regionu.



Rozproszona baza danych (1)

- Zbiór współpracujących z sobą baz danych (lokalne bd)
- Każda z baz lokalnych znajduje się na innym serwerze
- Z punktu widzenia użytkownika bazy lokalne logicznie stanowią jedną bd

ZSBD – wykład 1 (6)

Rozproszona baza danych jest zbiorem współpracujących z sobą baz danych, z których każda znajduje się na innym serwerze. Bazy te dalej będziemy nazywali lokalnymi. Z punktu widzenia użytkownika wszystkie te bazy logicznie stanowią jedną rozproszoną bazę danych.



Rozproszona baza danych (2)

- Zalety

- dane umieszczone "blisko" użytkownika \Rightarrow skrócenie czasu transmisji sieciowej
- mniejsze ryzyko utraty wszystkich danych na skutek awarii systemu
- wzrost niezawodności całego systemu

ZSBD – wykład 1 (7)

Dzięki umieszczeniu danych „blisko” ich użytkowników, skraca się opóźnienia transmisji sieciowej ponieważ dane specyficzne dla węzła są składowane i przetwarzane lokalnie. Dodatkowo, zmniejsza się ryzyko utraty wszystkich danych na skutek awarii systemu i wzrasta niezawodność całego systemu, ponieważ awaria jednej bazy danych np. w Krakowie nie ma wpływu na bazy danych w pozostałych miastach tak długo, dopóki żądania nie są kierowane do bazy w Krakowie.



Rozproszoną bazę danych (3)

- Wady
 - utrudniony dostęp do pełnego (zintegrowanego) zbioru danych
 - konieczność utrzymywania kopii danych (replik) i ich odświeżania

ZSBD – wykład 1 (8)

Architektura rozproszonych baz danych ma dwie podstawowe wady. Po pierwsze, rozproszenie danych utrudnia dostęp do pełnego-zintegrowanego zbioru danych pochodzących z różnych baz i ich analizę. Przykładowo, zarząd sieci warsztatów samochodowych będzie zainteresowany zestawieniami ilości sprzedaży i usług zrealizowanych w poszczególnych warsztatach. Uzyskanie takich informacji wymaga zintegrowania danych pochodzących ze wszystkich baz danych firmy.

Po drugie, wszystkie warsztaty korzystają z pewnego wspólnego zbioru informacji, tzw. słowników. Przykładem takiego słownika jest wykaz części znajdujących się w sprzedaży wraz z ich aktualnymi cenami. Gdyby dane słownikowe były przechowywane centralnie, tj. w jednej bazie danych, wówczas powstawałyby omówione wcześniej problemy architektury scentralizowanej. W związku z tym, informacje słownikowe są najczęściej powielane w każdej bazie danych firmy. Są to tzw. repliki.

W przypadku replik, występuje problem utrzymywania ich aktualnej zawartości, w przypadku, gdy oryginalne dane słownikowe ulegają modyfikacjom. Przykładowo, zmiana ceny opony w centrali firmy musi być propagowana do wszystkich oddziałów.



Komponenty architektury

- Sprzętowe komponenty rozproszonej bazy danych (RBD)
 - węzły - komputery, na których działa lokalna bd
 - sieć komputerowa
- Programowe komponenty RBD
 - protokoły sieciowe, np. TCP/IP, IPX/SP, LU6.2, DEC Net
 - dedykowane oprogramowanie realizujące dostęp z jednej bazy danych do drugiej

ZSBD – wykład 1 (9)

W skład systemu rozproszonej bazy danych wchodzą komponenty sprzętowe i programowe. Do pierwszej grupy zalicza się:

- tzw. węzły, czyli komputery na których działają lokalne bazy danych
- sieć komputerowa, dzięki której poszczególne węzły mogą się z sobą komunikować, a użytkownik z dowolnego węzła może sięgnąć do dowolnych innych węzłów systemu.

Do grupy programowych komponentów zalicza się:

- protokoły sieciowe np. TCP/IP, IPX/SPX, LU6.2, DEC Net,
- dedykowane oprogramowanie umożliwiające dostęp z jednej bazy danych do innej i przetwarzanie danych z innej bazy tak, jakby dane te były przechowywane lokalnie.



Reguły Date (1)

1. Lokalna autonomia
2. Uniezależnienie od centralnego miejsca
3. Działanie ciągłe
4. Niezależność lokalizacji
5. Niezależność fragmentacji
6. Replikacja
7. Niezależność sprzętowa

ZSBD – wykład 1 (10)

C.J.Date zaproponował 12 reguł jakie powinien spełniać system rozproszonej bazy danych. W dalszej części wykładu zostaną omówione te reguły.



Reguły Date (2)

8. Niezależność od systemu operacyjnego
9. Niezależność od systemu zarządzania bazą danych
10. Niezależność od sieci
11. Rozproszone zarządzanie transakcjami
12. Rozproszone przetwarzanie zapytań



1. Lokalna autonomia

- Każdy węzeł RBD jest zarządzany niezależnie od pozostałych węzłów systemu
- Wszystkie operacje na danych w węźle są kontrolowane przez ten węzeł
- Działanie węzła X nie powinno zależeć od działania węzła Y
- Na każdym węźle działa niezależny system zarządzania bazą danych

ZSBD – wykład 1 (12)

Lokalna autonomia oznacza, że:

- każdy węzeł należący do rozproszonej bazy danych jest zarządzany (administrowany) niezależnie od pozostałych węzłów,
- wszystkie operacje na danych w węźle są kontrolowane przez ten węzeł,
- działanie węzła X nie powinno zależeć od działania lub niedziałania innych węzłów,
- na każdym węźle działa niezależny system zarządzania bazą danych.



2. Uniezależnienie od centralnego węzła

- Wszystkie węzły są traktowane jednakowo
- Nie ma wyróżnionego centralnego węzła usługi (np. przetwarzanie zapytań)
 - mógłby stanowić wąskie gardło całego systemu

ZSBD – wykład 1 (13)

Uniezależnienie od centralnego węzła oznacza, że: wszystkie węzły systemu rozproszonej bazy danych są traktowane jednakowo i nie ma wyróżnionego węzła oferującego usługi dla pozostałych węzłów, np. przetwarzania zapytań. Taki wyróżniony węzeł mógłby stanowić tzw. "wąskie gardło" całego systemu.



3. Działanie ciągłe

- System RBD jest bardziej odporny na awarie
 - awaria jednego węzła nie wpływa na pracę innych (autonomia)
 - dzięki replikacji danych inny węzeł może udostępniać dane węzła uszkodzonego

ZSBD – wykład 1 (14)

Działanie ciągłe oznacza, że:

- awaria jednego węzła nie wpływa na pracę innych węzłów (jest to zagwarantowane przez autonomię węzłów),
- dzięki zastosowaniu mechanizmu replikowania danych do wielu węzłów, inny węzeł może udostępnić replikę oryginalnych danych w przypadku awarii węzła przechowującego dane oryginalne.

Uwaga: mechanizm replikacji zostanie omówiony w wykładzie drugim.



4. Niezależność lokalizacji

- Sposób dostępu do danych powinien być jednakowy
 - niezależny od fizycznego umiejscowienia sposobu składowania danych
- Użytkownik nie powinien być świadomym fizycznego umiejscowienia danych (przezroczystość lokalizacji)
 - powinien mieć wrażenie, że dane są przechowywane lokalnie

ZSBD – wykład 1 (15)

Niezależność lokalizacji oznacza, że sposób dostępu do danych przechowywanych w węzłach systemu RDB powinien być jednakowy, niezależny od fizycznego umiejscowienia danych i niezależny od ich fizycznego sposobu składowania. Ponadto, system powinien zapewniać tzw. przezroczystość lokalizacji (ang. location transparency), czyli ukrywać przed użytkownikiem fizyczne miejsce składowania danych. Innymi słowy, przezroczystość lokalizacji gwarantuje, że użytkownik nie musi znać fizycznego umiejscowienia danych a dostęp do danych jest realizowany w taki sposób, jak gdyby dane były przechowywane lokalnie.



5. Niezależność fragmentacji

- Dane można dzielić na fragmenty
- Każdy fragment można umieścić w dowolnym węźle
- Użytkownik nie powinien być świadomym istnienia fragmentów i ich lokalizacji
- Dostęp do fragmentu jest jednakowy i nie zależy od lokalizacji

ZSBD – wykład 1 (16)

Niezależność fragmentacji oznacza, że:

- dane, np. tabelę lub indeks, można dzielić na fragmenty,
- każdy fragment można niezależnie umieścić w innym węźle systemu RBD,
- użytkownik nie powinien być świadomym fizycznej lokalizacji fragmentów, często nie powinien też być świadomym istnienia fragmentów,
- dostęp do fragmentów powinien być jednakowy, niezależny od lokalizacji.

Uwaga: mechanizm fragmentacji zostanie omówiony w wykładzie drugim.



6. Replikacja

- Mechanizm tworzenia kopii danych pochodzących z jednego węzła w innym węźle
- Użytkownik może operować zarówno na danych oryginalnych, jak i na ich kopii w taki sam sposób (bez ograniczeń)

ZSBD – wykład 1 (17)

Replikacja jest mechanizmem polegającym na tworzeniu kopii danych pochodzących z jednego węzła w innym węźle. Użytkownik może operować w taki sam sposób na danych oryginalnych (źródłowych), jak i na kopii danych. Podstawowym obiektem bazy danych, który się replikuje jest tabela.



7. Niezależność sprzętowa

- Możliwość korzystania z tego samego SZBD na różnych platformach sprzętowych i ich współpraca w jednym systemie rozproszonym

ZSBD – wykład 1 (18)

Niezależność sprzętowa oznacza, że ten sam system zarządzania bazą danych (pochodzący od jednego producenta, w jednej wersji) może zostać zainstalowany na różnych platformach sprzętowych, z których każda stanowi osobny węzeł. SZBD działający na różnych platformach sprzętowych może wejść w skład tego samego systemu RBD.



8. Niezależność od systemu operacyjnego

- Możliwość korzystania z tego samego SZBD w różnych systemach operacyjnych
- Przykład
 - Oracle10g Release 2
 - z/Linux
 - z/OS
 - Solaris (x86-64)
 - HP-UX Itanium
 - Linux Itanium
 - Microsoft Windows
 - AIX5L

ZSBD – wykład 1 (19)

Niezależność od systemu operacyjnego oznacza, że ten sam system zarządzania bazą danych (pochodzący od jednego producenta, w jednej wersji) może zostać zainstalowany w różnych systemach operacyjnych i może wejść w skład tego samego systemu RBD.

Przykładowo, SZBD Oracle10g Release 2 może zostać zainstalowany m.in. w systemie operacyjnym z/Linux, z/OS, Solaris (x86-64), HP-UX Itanium, Linux Itanium, Microsoft Windows, AIX5L. Każda z tych instalacji może wchodzić w skład tego samego systemu RBD.



9. Niezależność od SZBD

- W skład systemu RBD mogą wchodzić bazy danych zarządzane przez różne SZBD
 - np. Oracle10g, IBM DB2, MS SQLServer2005, Sybase Adaptive Server Enterprise
- Dostęp do tych baz danych powinien być jednolity
 - jednolity/ustandardyzowany interfejs dostępu

ZSBD – wykład 1 (20)

Niezależność od systemu zarządzania bazą danych oznacza, że po pierwsze, w skład systemu RBD mogą wchodzić bazy danych zarządzane przez różne systemy zarządzania bazami danych, np. Oracle10g, IBM DB2, MS SQLServer2005, Sybase Adaptive Server Enterprise. Po drugie, sposób dostępu do tych baz danych powinien być jednolity. Oznacza to, że każdy z SZBD powinien dostarczać jednolity i ustandardyzowany interfejs dostępu do bazy danych.



10. Niezależność od sieci

- System RDB powinien pracować w różnych architekturach sieciowych i z różnymi protokołami sieciowymi
- Dostęp do poszczególnych węzłów powinien być jednolity, niezależnie od architektury i protokołów sieciowych

ZSBD – wykład 1 (21)

Niezależność od sieci oznacza, że system RDB powinien pracować również w różnych architekturach sieciowych i z różnymi protokołami sieciowymi. W takim przypadku, dostęp do poszczególnych węzłów powinien być jednolity i niezależny od architektury sieciowej i niezależny od wykorzystywanych protokołów sieciowych.



11. Rozproszone zarządzanie transakcjami

- W systemie RBD można realizować transakcję, która odwołuje się do wielu węzłów \Leftrightarrow transakcję rozproszoną
- Należy zagwarantować trwałość, spójność, atomowość i izolację transakcji rozproszonych

ZSBD – wykład 1 (22)

Rozproszone zarządzanie transakcjami oznacza, że w systemie RBD można realizować transakcje rozproszone. Transakcja rozproszona odwołuje się do wielu węzłów systemu. W przypadku tego typu transakcji system RBD powinien zagwarantować cztery cechy transakcji rozproszonej, tzn. jej trwałość, spójność, atomowość i izolację, podobnie jak w przypadku standardowych transakcji scentralizowanych.

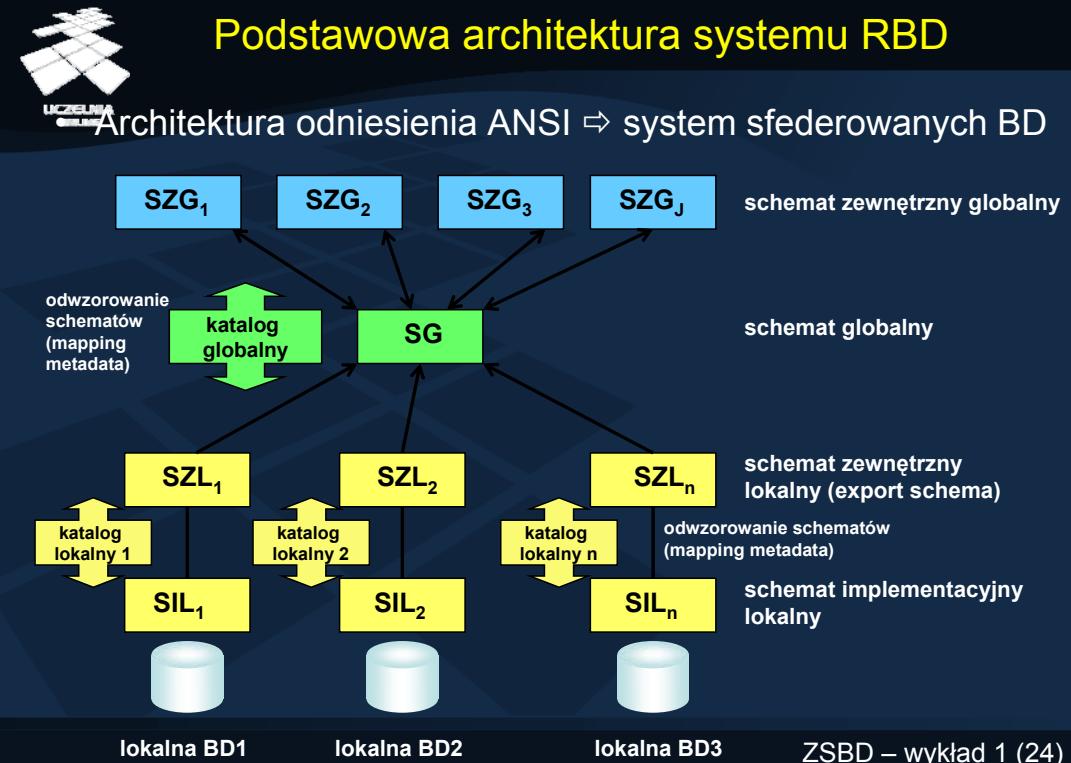


12. Rozproszone przetwarzanie zapytań

- Jedno zapytanie użytkownika może adresować wiele węzłów systemu RBD
- Należy zagwarantować optymalny sposób wykonania takiego zapytania

ZSBD – wykład 1 (23)

Rozproszone przetwarzanie zapytań gwarantuje możliwość wykonania zapytania, które adresuje jednocześnie wiele węzłów systemu RBD. W takim przypadku, system powinien zagwarantować optymalny lub suboptymalny sposób wykonania takiego zapytania, zgodnie z przyjętym kryterium kosztu wykonania.



Jedną z podstawowych architektur implementacyjnych systemu RBD jest tzw. system sfederowanych baz danych. Jego ogólną architekturę przedstawia slajd.

Każda z lokalnych baz danych BD1, BD2 i BD3 posiada swój schemat implementacyjny lokalny (SIL1, SIL2, SIL3). Schemat implementacyjny określa w jakim implementacyjnym modelu danych są reprezentowane dane.

Przykładami takich modeli są: model relacyjny, obiektowy, obiektowo-relacyjny, semistrukturalny.

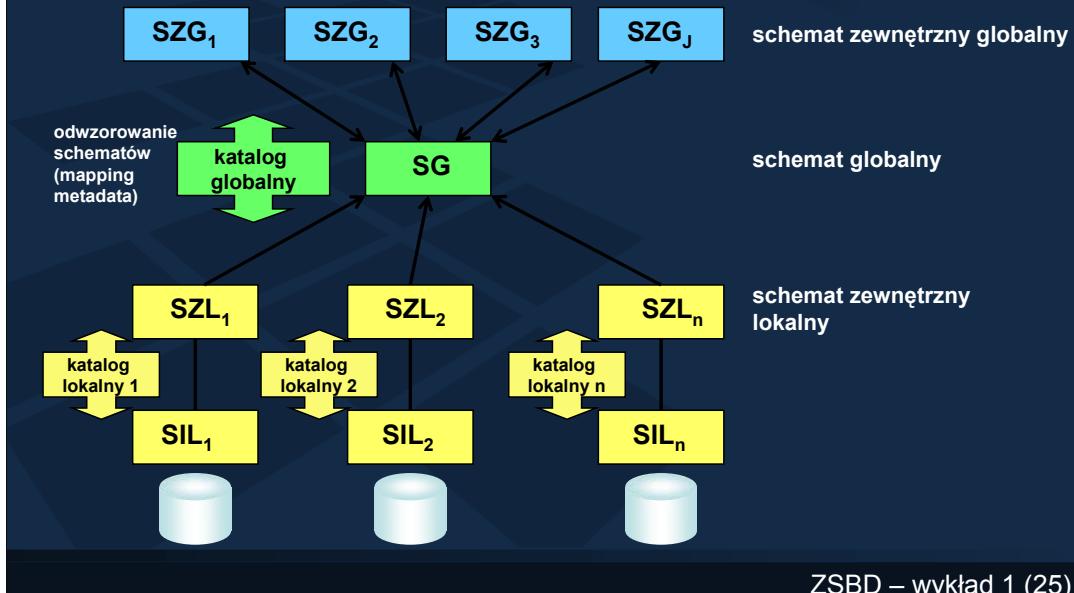
Schemat implementacyjny lokalny jest niedostępny na zewnątrz węzła. Z tego względu, każdy węzeł udostępnia tzw. schemat zewnętrzny lokalny (SZL1, SZL2, SZL3). Schemat ten, pełni dwie zasadnicze funkcje. Po pierwsze, umożliwia on dokonanie konwersji schematu implementacyjnego lokalnego do wspólnego modelu danych wykorzystywanego w systemie sfederowanych BD. Najczęściej jest to model relacyjny. Po drugie, umożliwia udostępnienie nie całego schematu implementacyjnego lokalnego, ale jego fragmentu.

Schemat implementacyjny lokalny jest odwzorowywany w schemat zewnętrzny lokalny z wykorzystaniem katalogu lokalnego. Przechowuje on m.in. odwzorowania nazw obiektów, informacje o prawach dostępu, procedury konwersji.



Podstawowa architektura systemu RBD cd.

Architektura odniesienia ANSI \Rightarrow system sfederowanych BD



ZSBD – wykład 1 (25)

Schematy zewnętrzne lokalne są integrowane w jeden schemat globalny (SG). Schemat ten zapewnia, że wszystkie lokalne bazy danych są widziane jako jedna, spójna baza. Schematy zewnętrzne lokalne są odwzorowywane w SG z wykorzystaniem katalogu globalnego. Podobnie, jak katalog lokalny, katalog globalny przechowuje m.in. odwzorowania nazw obiektów, informacje o prawach dostępu, procedury konwersji.

Na podstawie schematu globalnego, użytkownicy systemu sfederowanych BD tworzą własne schematy, tzw. schematy zewnętrzne globalne. Umożliwiają one zawężenie danych udostępnianych przez SG do wycinka interesującego użytkowników. Użytkownicy pracują z systemem poprzez swoje schematy zewnętrzne globalne.



System sfederowanych DB

- Dwa niezależne systemy BD + mechanizm konsolidujący
- Każdy system BD jest autonomiczny i ma swoich użytkowników

ZSBD – wykład 1 (26)

System sfederowanych baz danych to system składający się z co najmniej dwóch niezależnych, różnych systemów baz danych oraz odpowiedniego mechanizmu konsolidującego wszystkie ich komponenty. Ponadto, każdy system BD jest niezależnym i autonomicznym decentralizowanym SZBD, który ma swoich własnych lokalnych użytkowników.



System sfederowanych BD - przykład

- System kontroli opłat abonamentowych TVP
- Komponenty - autonomiczne bazy danych:
 - Urzędu Miasta
 - dane meldunkowe obywateli
 - sieci MediaMarkt
 - dane o sprzedaży odbiorników RTV
 - Urzędu Radiofonii i TV
 - dane o płaconych abonamentach

ZSBD – wykład 1 (27)

Jako przykład systemu sfederowanych BD rozważmy system kontroli opłat abonamentowych TVP. Założymy, że w jego skład wchodzą trzy autonomiczne bazy danych należące do: Urzędu Miasta, sieci sklepów MediaMarkt, Urzędu Radiofonii i TV.

Pierwsza z nich udostępnia dane meldunkowe obywateli, a jej zawartość jest niezbędna do wysyłania kar i wezwań do uiszczenia opłat abonamentowych. Druga baza udostępnia dane o sprzedaży odbiorników RTV, tj. rachunki lub faktyry wystawiane kupującym sprzęt RTV. Jej zawartość jest niezbędna do zidentyfikowania kupujących sprzęt RTV. Trzecia baza udostępnia dane abonentów, którzy zarejestrowali odbiorniki RTV.

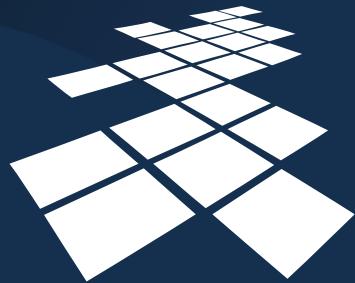
W takim systemie sfederowanych baz danych, odpowiednio uprzywilejowany użytkownik mógłby wydać zapytanie o dane meldunkowe obywateli (baza Urzędu Miasta), którzy w ostatnim roku zakupili odbiorniki RTV (baza sieci MediaMarkt), ale którzy nie płacą abonamentu, tj. nie zostali zarejestrowani w bazie danych Urzędu Radiofonii i TV.

Systemy rozproszonych baz danych – 2

Fragmentacja, replikacja, zarządzanie transakcjami

Wykład przygotował:

Robert Wrembel



ZSBD – wykład 2 (1)



Plan wykładu

- Cel i techniki fragmentacji danych
- Cel i techniki replikacji danych
- Problematyka zarządzania transakcjami rozproszonymi

ZSBD – wykład 2 (2)

Celem wykładu jest omówienie podstawowych zagadnień związanych z implementacją systemu rozproszonej bazy danych. Omówione zostaną następujące zagadnienia:

- cel i techniki fragmentacji danych (tzw. partycjonowanie),
- cel i techniki replikacji danych,
- problematyka zarządzania transakcjami rozproszonymi.

Problematyka optymalizacji zapytań rozproszonych zostanie omówiona w ramach laboratorium nr 3, wraz z praktyczną ilustracją konkretnych technik w systemie Oracle9i/10g.



Wprowadzenie

- Problematyka projektowania RBD, optymalizacja zapytań, zarządzanie współbieżnością transakcji ⇔ znacznie trudniejsze niż w klasycznych BD
- Systemy komercyjne
 - Oracle9i/10g
 - Sybase Adaptive Server Enterprise, Adaptive Server Anywhere,
 - IBM DB2
 - MS SQLServer2000, SQLServer2005

ZSBD – wykład 2 (3)

Wiele problemów związanych z projektowaniem i zarządzaniem scentralizowanymi bazami danych, m.in. projektowanie struktury bazy danych, przetwarzanie i optymalizacja zapytań, zarządzanie współbieżnością transakcji, staje się znacznie trudniejsze w przypadku baz rozproszonych.

Dostępne na rynku komercyjne SZBD (Oracle9i/10g, Sybase Adaptive Server Enterprise i Sybase Adaptive Server Anywhere, IBM DB2, MS SQLServer2000 i SQLServer2005) oferują mniej lub bardziej zaawansowaną funkcjonalność wymaganą przy budowie systemów RBD.



Fragmentacja/partycjonowanie (1)

- Podział tabeli na części, zwane fragmentami lub partycjami
- Techniki fragmentacji
 - pozioma
 - pionowa
 - mieszana

ZSBD – wykład 2 (4)

Jednym z podstawowych zagadnień projektowania RDB jest określenie sposobu podziału danych pomiędzy węzły, tak aby efektywność całego systemu była zadowalająca. Jedną z technik stosowanych w tym zakresie jest fragmentacja i alokacja.

Fragmentacja polega na podziale obiektu przechowującego dane, najczęściej tabeli, na mniejsze części, zwane fragmentami lub partycjami. W praktyce wyróżnia się trzy techniki fragmentacji, tj. fragmentację poziomą, pionową i mieszana.



Fragmentacja/partycjonowanie (2)

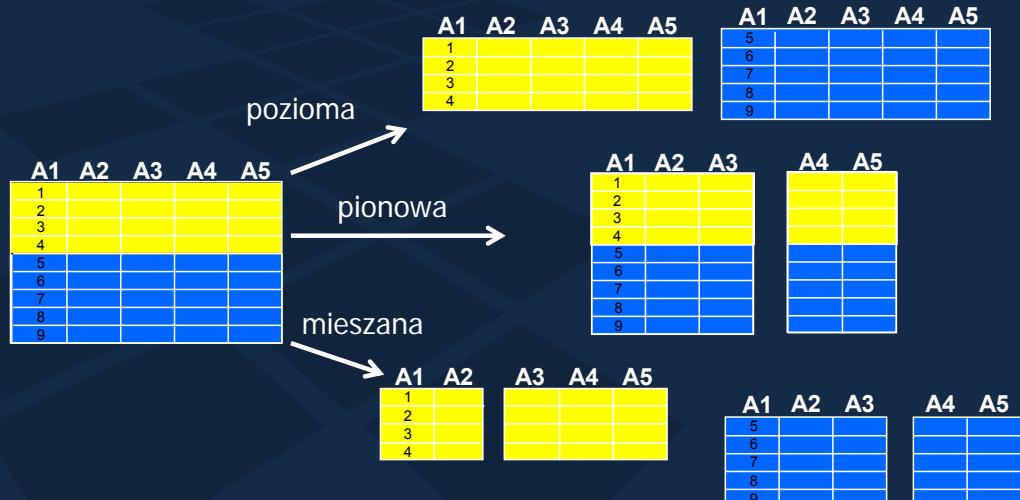
- Cel: zwiększenie efektywności dostępu do danych
 - zmniejszenie rozmiaru danych, które należy przeszukać
 - równoleglenie operacji dostępu do dysków, na których umieszczono fragmenty
 - alokowanie fragmentów "blisko" miejsca ich wykorzystania - redukcja kosztów transmisji sieciowej

ZSBD – wykład 2 (5)

Celem fragmentacji jest zwiększenie efektywności dostępu do danych. Dzięki właściwemu doborowi kryterium podziału obiektów na fragmenty można zapewnić, że aplikacje użytkowników będą adresowały tylko wybrane interesujące je partie, w ten sposób ograniczając wolumeny przeszukiwanych danych. Ponadto, każdy z fragmentów może być umieszczony na innym dysku, co zapewni równoległy dostęp do wielu fragmentów równocześnie. Jeżeli fragmenty zostaną umieszczone "blisko" miejsca ich wykorzystania, wówczas można zredukować koszty transmisji sieciowej w sieciach o niskiej przepustowości.



Rodzaje fragmentacji



ZSBD – wykład 2 (6)

Na slajdzie w sposób schematyczny przedstawiono wspomniane wcześniej techniki fragmentacji.



Fragmentacja pozioma

- Podział zbioru rekordów na podzbioru
- Każdy podzbiór opisany identyczną liczbą atrybutów
- Wybór fragmentu (partycji) do którego trafi rekord realizowany na podstawie wartości jednego lub kilku wybranych atrybutów tabeli – tzw. atrybutów fragmentujących (partycjonujących)

ZSBD – wykład 2 (7)

Fragmentacja pozioma (ang. horizontal fragmentation) umożliwia podział zbioru rekordów tabeli na mniejsze podzbiory, z których każdy jest opisany identyczną liczbą atrybutów.

Wybór fragmentu (partycji) do którego trafi rekord jest realizowany na podstawie wartości jednego lub kilku wybranych atrybutów tabeli – tzw. atrybutów fragmentujących (partycjonujących).



Fragmentacja pionowa

- Rozbicie tabeli na fragmenty złożone z podzbioru atrybutów
- Każdy fragment zawiera identyczną liczbę rekordów
- Atrybut niekluczowy X może się znaleźć tylko w jednym fragmencie
- Atrybut kluczowy znajduje się w każdym fragmencie
 - służy do łączenia fragmentów

ZSBD – wykład 2 (8)

Fragmentacja pionowa (ang. vertical fragmentation) umożliwia podział tabeli w pionie, tj. jej rozbicie na fragmenty złożone z podzbiorów atrybutów tabeli pierwotnej. Każdy fragment zawiera identyczną liczbę rekordów. Dany atrybut może się znaleźć tylko w jednym fragmencie. Nie dotyczy to atrybutów kluczowych, które są umieszczane w każdym fragmencie i służą one do łączenia fragmentów.



Fragmentacja mieszana

- Połączenie fragmentacji poziomej i pionowej
- Warianty
 - Fragmentacja pozioma + pionowa
 - Fragmentacja pionowa + pozioma

ZSBD – wykład 2 (9)

Fragmentacja mieszana (ang. hybrid fragmentation) stanowi połączenie fragmentacji poziomej i pionowej. Występuje w dwóch wariantach. W pierwszym tabela jest najpierw dzielona poziomo, a następnie wszystkie lub wybrane jej fragmenty są dalej dzielone pionowo. W wariantie drugim, tabela jest najpierw dzielona pionowo, a następnie wszystkie lub wybrane jej fragmenty są dalej dzielone poziomo.



Kryteria poprawności fragmentacji (1)

- Kompletność (ang. completeness)
 - jeżeli tabela T została podzielona na partycje TP_1, TP_2, \dots, TP_n , to każdy rekord z T lub jego fragment musi się znaleźć w jednej z partycji TP_1, TP_2, \dots, TP_n
 - żadne dane nie są tracone

ZSBD – wykład 2 (10)

Poprawna fragmentacja musi spełniać trzy następujące kryteria: kompletność, rozłączność i rekonstrukcję.

Kompletność (ang. *completeness*) oznacza, że jeżeli tabela T została podzielona na partycje TP_1, TP_2, \dots, TP_n , to każdy rekord z T lub jego fragment musi się znaleźć w jednej z partycji TP_1, TP_2, \dots, TP_n . Kryterium to gwarantuje, że na skutek partycjonowania żadne dane z tabeli pierwotnej T nie zostaną utracone.



Kryteria poprawności fragmentacji (2)

- Rozłączność (ang. disjointness)
 - jeżeli tabela T została podzielona na partycje TP_1, TP_2, \dots, TP_n , to każdy rekord z T lub jego fragment nie może się znaleźć w więcej niż jednej partycji
 - nie pojawią się dane nadmiarowe
 - wyjątek: partycjonowanie pionowe
 - klucze podstawowe występują w każdej partycji

ZSBD – wykład 2 (11)

Rozłączność (ang. *disjointness*) oznacza, że jeżeli tabela T została podzielona na partycje TP_1, TP_2, \dots, TP_n , to każdy rekord z T lub jego fragment nie może się znaleźć w więcej niż jednej partycji. Kryterium to gwarantuje, że na skutek partycjonowania w bazie danych nie pojawią się dane nadmiarowe. Wyjątkiem od tej reguły jest partycjonowanie pionowe, w którym wartości atrybutów stanowiących klucz podstawowy tabeli występują w każdej partycji.



Kryteria poprawności fragmentacji (3)

- Rekonstrukcja (ang. reconstruction)
 - możliwość zrekonstruowania pierwotnej tabeli T ze wszystkich jej partycji
 - nie prowadzi do utraty danych
 - nie prowadzi do powstania danych nadmiarowych
- Rekonstrukcja z fragmentów poziomych
 - operator sumy zbiorów
- Rekonstrukcja z fragmentów pionowych
 - operator połączenia

ZSBD – wykład 2 (12)

Rekonstrukcja (ang. *reconstruction*) oznacza, że musi istnieć możliwość zrekonstruowania pierwotnej tabeli T ze wszystkich jej partycji. Rekonstrukcja ta nie może doprowadzić ani do utraty żadnych danych, ani do powstania danych nadmiarowych.

W przypadku partycjonowania poziomego, rekonstrukcji pierwotnej tabeli dokonuje się z wykorzystaniem operatora sumy zbiorów rekordów znajdujących się we wszystkich partycjach. W przypadku partycjonowania pionowego, rekonstrukcję realizuje się za pomocą łączenia partycji wykorzystując do tego klucz podstawowy tabeli pierwotnej.



Algorytmy fragmentacji poziomej

- Round-robin
 - cykliczne rozmieszczenie danych w węzłach systemu
- Bazujący na wartości
 - rozmieszczenie danych w węzłach zależy od wartości danych
 - zakresowy
 - haszowy

ZSBD – wykład 2 (13)

Fragmentację poziomą implementuje się z wykorzystaniem dwóch podstawowych algorytmów, tj. round-robin i bazującego na wartości. Algorytm round-robin rozmieszczza rekordy cyklicznie we wszystkich węzłach systemu. W przypadku algorytmów bazujących na wartości, tj. zakresowego i hash'owego, rozmieszczenie danych w sieci zależy od wartości samych danych.



Algorytm round-robin

- Równomierne rozpraszanie danych w węzłach sieci
 - równoważenie obciążenia
- Dane rozpraszane w sposób przypadkowy
 - odnalezienie żądanego informacji wymaga przeszukania wszystkich węzłów

ZSBD – wykład 2 (14)

Algorytm round-robin umożliwia równomierne rozpraszanie danych w węzłach sieci. Przykładowo, jeśli w sieci znajdują się trzy węzły, to pierwszy rekord tabeli zostanie umieszczony w węźle pierwszym, drugi — w węźle drugim, trzeci rekord — w węźle trzecim, czwarty — znów w węźle pierwszym itp. Ponieważ dane są rozpraszane w sposób przypadkowy, więc odnalezienie żądanego rekordu wymaga przeszukania wszystkich węzłów, co jest wadą tego rozwiązania.



Algorytm zakresowy

- Każdy węzeł składa dane o wartościach atrybutu fragmentującego z zadanego zakresu lub o zadanej wartości



ZSBD – wykład 2 (15)

W algorytmie zakresowym, każdy węzeł przechowuje dane o wartościach atrybutu fragmentującego z zadanego zakresu lub o zadanej wartości.

Przykładowo, na slajdzie przedstawiono trzy węzły BD1, BD2, BD3. Pierwszy z nich zawiera fragment tabeli faktury, przechowujący faktury z lat od 2000 do 2002. Drugi węzeł zawiera fragment tabeli przechowujący faktury z lat 2003-2005, a trzeci - zawiera fragment tabeli przechowujący faktury z roku 2006.



Algorytm haszowy

- Dane umieszczane w węzłach na podstawie wartości funkcji haszowej
 - argument wywołania \Rightarrow wartość atrybutu fragmentującego
 - wynik \Rightarrow adres węzła
- Umieszczanie w jednym węźle rekordów z różnych powiązanych tabel
 - efektywność operacji łączenia tabel

ZSBD – wykład 2 (16)

W przypadku algorytmu fragmentacji haszowej, dane są umieszczane w węzłach zgodnie z wartością systemowej funkcji haszowej. Argumentem wejściowym tej funkcji jest wartość atrybutu, a jej wynikiem — adres węzła, w którym zostanie umieszczony rekord. W celu odnalezienia żądanego rekordu SZBD wykorzystuję tę samą funkcję haszową, która została wykorzystana do fragmentacji danych. Zaletą tej metody jest możliwość automatycznego umieszczania w tym samym węźle rekordów pochodzących z różnych, powiązanych z sobą tabel. W ten sposób zwiększa się efektywność wykonywania operacji łączenia tabel, gdyż łączone z sobą rekordy znajdują się w tym samym węźle.



Problem alokacji (1)

- Określenie miejsca fizycznego składowania danych/fragmentów
 - umieszczenie danych we właściwych węzłach sieci - "blisko" miejsca ich wykorzystania
 - kryterium alokacji: maksymalizacja efektywności systemu RBD

ZSBD – wykład 2 (17)

Ponieważ dane w systemie RBD mogą być składowane na dowolnym jego węźle, więc zachodzi konieczność takiego rozmieszczenia danych (np. tabel, fragmentów, indeksów), który zapewni największą efektywność całego systemu. Jest to tzw. problem alokacji.



Problem alokacji (2)

- Problem alokacji
 - dla danego zbioru fragmentów
 $F=\{F_1, F_2, \dots, F_n\}$
i zbioru węzłów systemu
 $W=\{W_1, W_2, \dots, W_m\}$
w których działają aplikacje
 $A=\{A_1, A_2, \dots, A_j\}$
 - znaleźć optymalny przydział F_i ($i=1..n$) do W_j ($j=1..m$)

ZSBD – wykład 2 (18)

Problem alokacji można zdefiniować następująco: dla danego zbioru fragmentów $F=\{F_1, F_2, \dots, F_n\}$ i danego zbioru węzłów systemu RBD $W=\{W_1, W_2, \dots, W_n\}$ w których działa zbiór aplikacji użytkowników $A=\{A_1, A_2, \dots, A_j\}$ należy znaleźć optymalny przydział fragmentów do węzłów.



Problem alokacji (3)

- Optimum jest definiowane w kontekście
 - łącznego kosztu
 - przechowywania F_i w węźle W_j
 - odczytu F_i z węzła W_j
 - zmodyfikowania F_i w W_j
 - komunikacji z W_j
 - efektywności systemu
 - mierzona jego przepustowością w każdym z węzłów

ZSBD – wykład 2 (19)

Optimum przydziału jest definiowane w kontekście łącznego kosztu i efektywności systemu. Łączny koszt uwzględnia:

- koszt przechowywania fragmentu F_i w węźle W_j ,
- koszt odczytu fragmentu F_i z węzła W_j ,
- koszt zmodyfikowania fragmentu F_i w węźle W_j ,
- koszt komunikacji z węzłem W_j .

Efektywność systemu jest natomiast mierzona przepustowością w każdym z jego węzłów.

W literaturze zaproponowano wiele algorytmów alokacji. Ich omówienie wykracza jednak poza zakres tego wykładu.



Replikacja danych

- Tworzenie kopii danych pochodzących z jednego węzła i przechowywanie ich w innych węzłach
- Cele
 - przyspieszenie dostępu do danych wykorzystywanych często
 - słownik miast, województw, kodów adresowych
 - katalog produktów
 - równoważenie obciążenia węzłów
 - równoległe wykonywanie zapytań
 - zwiększenie efektywności dostępu do danych przez wybór repliki do której dostęp jest najszybszy

ZSBD – wykład 2 (20)

Drugim podstawowym zagadnieniem projektowania RBD jest określenie sposobu duplikowania danych w wielu węzłach systemu RBD. Duplikowanie takie będziemy dalej nazywali replikacją danych.

Replikacja polega na tworzeniu kopii danych pochodzących z jednego węzła i przechowywaniu tych kopii w innych węzłach.

Cele replikacji są następujące:

- przyspieszenie dostępu do danych poprzez ich umieszczenie w węźle, który z nich intensywnie korzysta; przykładami mogą być tabele słownikowe miast, województw i kodów adresowych w systemach ewidencji, katalogi produktów w systemach sprzedaży;
- równoważenie obciążenia węzłów poprzez powielanie intensywnie wykorzystywanych (odczytywanych) danych do innych węzłów;
- równoległe wykonywanie zapytań adresujących te same dane; w takim przypadku 2 zapytania odwołujące się do tej samej tabeli mogą zostać wykonane na dwóch różnych replikach tej tabeli;
- zwiększenie efektywności dostępu do danych przez wybór repliki, do której dostęp jest najszybszy.



Problematyka replikacji

- Określenie jednostki replikacji
- Określenie ilości replikowanych danych
- Określenie momentu odświeżania
- Określenie sposobu odświeżania

ZSBD – wykład 2 (21)



Określenie jednostki replikacji

- Wskazanie tabeli, pionowego lub poziomego fragmentu zbioru tabel, tabeli w momencie definiowania repliki
 - za pomocą zapytania SQL
 - za pomocą narzędzi dostarczanych przez producenta konkretnego systemu

ZSBD – wykład 2 (22)

Określenie jednostki replikacji polega na wskazaniu jak duża jest zawartość pojedynczej repliki. Replika może być zbudowana na pojedynczej tabeli, pionowym lub poziomym jej fragmencie, bądź na zbiorze tabel. Jednostkę replikacji określa się w momencie definiowania repliki albo za pomocą zapytania SQL albo za pomocą narzędzi programistycznych dostarczanych przez producenta konkretnego systemu.



Określenie ilości replikowanych danych (1)

- Bazy w pełni replikowane
 - każdy węzeł zawiera wszystkie dane z pozostałych węzłów
 - cechy
 - wysoka efektywność wykonywania zapytań adresujących dane z innych węzłów
 - duży narzut czasowy na odświeżanie replik

ZSBD – wykład 2 (23)

W zakresie określenia ilości replikowanych danych wyróżnia się bazy danych w pełni replikowane (ang. fully replicated databases) i bazy danych częściowo replikowane (ang. partially replicated databases).

W bazach w pełni replikowanych każdy węzeł zawiera wszystkie dane z pozostałych węzłów. Bazy takie cechują się wysoką efektywnością wykonywania zapytań, ponieważ każde zapytanie może zostać wykonane lokalnie. Wadą tego typu baz danych jest duży narzut czasowy na odświeżanie replik, ponieważ jest ich zbyt dużo. W praktyce tego typu bazy danych stosuje się niezwykle rzadko.



Określenie ilości replikowanych danych (2)

- Bazy częściowo replikowane
 - węzeł może zawierać wybrane dane z innych węzłów
 - cechy
 - mniejsza efektywność wykonywania zapytań adresujących dane z innych węzłów
 - mniejszy narzut czasowy na odświeżanie replik

ZSBD – wykład 2 (24)

W bazach danych częściowo replikowanych każdy węzeł może zawierać wybrane dane z innych węzłów. Tego typu bazy danych cechuje mniejsza efektywność od poprzednich ponieważ część zapytań będzie wymagała sięgnięcia do zdalnych węzłów. Mniejszy jest też narzut czasowy na odświeżanie replik, ponieważ replik tych jest mniej niż w bazie danych w pełni replikowanej.



Określenie momentu odświeżania

- Synchronicznie
 - natychmiast po zmianie danych w źródle
 - po zatwierdzeniu transakcji w źródle
- Asynchronicznie
 - na żądanie
 - automatycznie z zadanyem okresem

ZSBD – wykład 2 (25)

Wałąną cechą odświeżania repliki jest wybór momentu odświeżania. W praktyce stosuje się dwie techniki, tj. odświeżanie synchroniczne i odświeżanie asynchroniczne.

Pierwsza technika polega na propagowaniu danych ze źródła do repliki albo natychmiast po zmianie zawartości źródła albo po zatwierdzeniu transakcji w źródle. Drugie rozwiązanie jest stosowane w praktyce.

Odświeżanie asynchroniczne polega na propagowaniu danych do repliki po jakimś czasie od wprowadzenia zmian w źródle. Możliwe są tu dwa rozwiązania, albo odświeżanie na żądanie repliki, albo odświeżanie automatyczne z zadanyem okresem. W praktyce stosuje się oba rozwiązania.



Określenie sposobu odświeżania

- Pełne
 - ze źródła do repliki przesyłane wszystkie dane
 - cechy
 - łatwe implementacyjnie
 - kosztowne
- Przyrostowe
 - ze źródła do repliki przesyłane dane zmienione
 - cechy
 - trudniejsze implementacyjnie
 - mniej kosztowne

ZSBD – wykład 2 (26)

Kolejną ważną cechą odświeżania repliki jest określenie sposobu jej odświeżania. W praktyce stosuje się dwie techniki, tj. odświeżanie pełne i odświeżanie przyrostowe.

Odświeżanie pełne polega na każdorazowym przesyłaniu ze źródła do repliki wszystkich danych, które replika udostępnia. Ta technika odświeżania charakteryzuje się łatwością implementacji i dużym kosztem odświeżania ze względu na dużą ilość przesyłanych danych.

Odświeżanie przyrostowe polega na przesyłaniu do repliki wyłącznie zmian dokonanych w źródle od czasu ostatniego odświeżenia. Ta technika cechuje się trudniejszą implementacją i niższymi kosztami odświeżania.



Implementacja replikacji

- Replika - migawka - perspektywa zmaterializowana
 - obiekt bazy danych, którego definicja zawiera
 - moment odświeżania
 - sposób odświeżania
 - zakres replikowanych danych z tabel źródłowych (najczęściej specyfikacja za pomocą zapytania SQL)
 - systemowy proces odświeżający

ZSBD – wykład 2 (27)

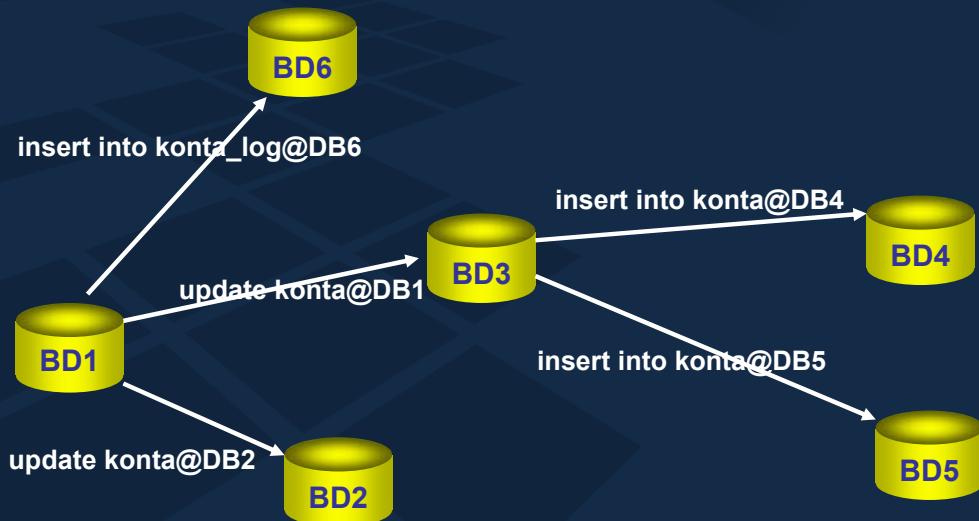
W praktyce replika jest implementowana jako pewien obiekt bazy danych, którego definicja zawiera:

- moment odświeżania,
- sposób odświeżania,
- zakres replikowanych danych z tabel źródłowych; zakres ten najczęściej specyfikuje się za pomocą zapytania SQL.

Z repliką jest związany systemowy proces odpowiedzialny za jej odświeżanie.



Zarządzanie transakcjami w RBD



ZSBD – wykład 2 (28)

W systemie RBD użytkownik może wykonać transakcję, która adresuje wiele węzłów. Przykład takiej transakcji przedstawiono na slajdzie. W skład systemu wchodzi 6 baz danych BD1-BD6. Z bazy BD1 użytkownik wykonuje transakcję która wstawia dane do tabeli *konta_log* w bazie BD6, modyfikuje zawartość tabeli *konta* w bazie BD2 i BD1. Ta ostatnia modyfikacja uruchamia polecenia *insert* do tabel w bazach DB4 i BD5. Transakcję taką będziemy nazywali rozproszoną (ang. distributed transaction) lub globalną (ang. global transaction).



Transakcja rozproszona (1)

- Odwołuje się przynajmniej do jednego zdalnego węzła systemu RBD
- Transakcja rozproszona jest reprezentowana przez zbiór transakcji lokalnych
 - w każdej z baz danych, do której odwołuje się transakcja rozproszona tworzona jest jedna transakcja lokalna
- Cechy transakcji rozproszonej
 - trwałość
 - spójność
 - izolacja
 - atomowość

ZSBD – wykład 2 (29)

Transakcja rozproszona odwołuje się przynajmniej do jednego zdalnego węzła systemu RBD. Transakcja rozproszona jest reprezentowana przez zbiór **transakcji lokalnych**. W każdej z baz danych, do której odwołuje się transakcja rozproszona tworzona jest jedna transakcja lokalna. Zarówno każda z transakcji lokalnych jak i rozproszona mają cechy trwałości, spójności, izolacji i atomowości.



Transakcja rozprosiona (2)

- Atomowość
 - wszystkie transakcje lokalne zatwierdzone lub wszystkie wycofane
- Problemy
 - węzeł BD3 ulega awarii w trakcie realizowania polecenia INSERT
 - połączenie sieciowe z węzłem BD6 zostaje przerwane po wykonaniu polecenia INSERT
 - węzeł BD2 ulega awarii w trakcie zatwierdzania transakcji rozproszonej

ZSBD – wykład 2 (30)

Cecha atomowości w odniesieniu do transakcji rozproszonej oznacza, że wszystkie transakcje lokalne wchodzące w skład transakcji rozproszonej muszą zostać zatwierdzone. Jeśli choć jedna z transakcji lokalnych nie może zostać zatwierdzona, wówczas całą transakcję rozproszoną należy wycofać.

W czasie realizowania transakcji rozproszonej system RDB może ulec częściowej awarii. Przykładowo:

- węzeł BD3 (ze slajdu 28) może ulec awarii w trakcie realizowania polecenia INSERT,
- połączenie sieciowe z węzłem BD6 może zostać przerwane po wykonaniu polecenia INSERT,
- węzeł BD2 może ulec awarii w trakcie zatwierdzania transakcji rozproszonej.

W tych i wielu innych przypadkach system musi zagwarantować zakończenie rozproszonej transakcji, w taki sposób aby zagwarantować cztery jej omówione wcześniej cechy.



Transakcja rozproszona (3)

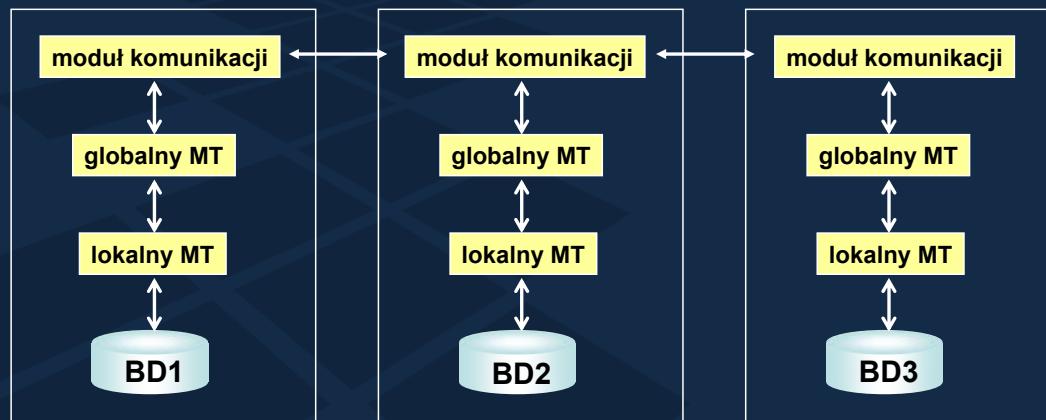
- Standardowy sposób zatwierdzania transakcji nie gwarantuje atomowości transakcji rozproszonej
- Potrzebny zaawansowany mechanizm zatwierdzania \Rightarrow protokół 2PC

ZSBD – wykład 2 (31)

Głównym problemem jest zagwarantowanie atomowości transakcji rozproszonej a standardowy mechanizm zatwierdzania transakcji nie gwarantuje jej atomowości. W związku z tym, zatwierdzanie lub wycofywanie transakcji rozproszonej, gwarantujące atomowość jest realizowane za pomocą specjalizowanego mechanizmu, tzw. **protokołu zatwierdzania dwu-fazowego — 2PC** (ang. *two-phase commit*).



Architektura zarządzania transakcjami rozproszonymi



ZSBD – wykład 2 (32)

Podstawową architekturę zarządzania transakcjami rozproszonymi przedstawiono na slajdzie. Każda z trzech baz danych BD1, BD2, BD3 posiada swój własny moduł *lokalnego menadżera transakcji* (lokalny MT), identycznie jak w standardowej scentralizowanej bazie danych. Ponadto, do zarządzania transakcjami rozproszonymi jest niezbędny moduł *globalnego menadżera transakcji* (globalny MT). Jego zadaniem jest koordynowanie wykonania zarówno lokalnych jak i rozproszonych transakcji zainicjowanych w jego węźle. Poszczególne węzły realizujące transakcję rozproszoną komunikują się za pośrednictwem *modułu komunikacji*, istniejącego w każdym węźle.



Protokół 2PC (1)

- Fazy realizacji
 - przygotowanie/głosowanie (ang. voting phase)
 - decyzja (ang. decision phase)
- Warianty
 - scentralizowany 2PC
 - zdecentralizowany 2PC
 - liniowy 2PC

ZSBD – wykład 2 (33)

Protokół zatwierdzania 2-fazowego składa się z 2 faz: przygotowania, zwanej również głosowaniem (ang voting phase) i decyzji (ang. decision phase).

Protokół ten może być implementowany w jednym z trzech wariantów:

- scentralizowanego 2PC,
- zdecentralizowanego 2PC,
- liniowego 2PC.



Protokół 2PC (2)

- Węzły
- Koordynator globalny (KG)
 - węzeł inicjujący transakcję rozproszoną
 - koordynuje proces jej zatwierdzania/wycofywania
- Uczestnik (U)
 - węzeł realizujący transakcję lokalną będącą częścią rozproszonej

ZSBD – wykład 2 (34)

Z punktu widzenia protokołu 2PC w transakcji rozproszonej biorą udział dwa rodzaje węzłów, tj. koordynator globalny i uczestnik. Koordynator globalny jest węzłem inicjującym transakcję rozproszoną i koordynującym jej zatwierdzanie lub wycofywanie. Uczestnik jest węzłem realizującym transakcję lokalną, która to transakcja jest częścią transakcji rozproszonej.



Scentralizowany 2PC (1)

- Koncepcja
 - KG pyta uczestników, czy są gotowi do zatwierdzenia
 - jeśli przynajmniej jeden uczestnik odpowiada ABORT, lub nie odpowie w zadany czasie \Rightarrow KG wysyła do wszystkich uczestników komunikat wycofania transakcji
 - jeśli wszyscy uczestnicy gotowi \Rightarrow KG wysyła do nich komunikat zatwierdzenia transakcji

ZSBD – wykład 2 (35)

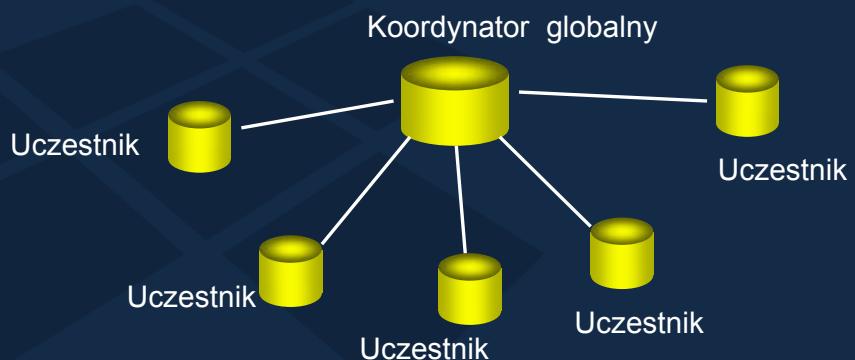
Koncepcja pracy protokołu 2PC jest następująca.

Koordynator globalny, pyta uczestników czy są gotowi do zatwierdzenia swoich transakcji lokalnych. Jeśli przynajmniej jeden uczestnik odpowiada komunikatem ABORT (nie jest gotowy), lub nie odpowie w zadany czasie, wówczas koordynator globalny wysyła do wszystkich uczestników komunikat wycofania transakcji. Jeśli natomiast wszyscy uczestnicy zgłoszą gotowość do zatwierdzenia swoich transakcji lokalnych, wówczas koordynator globalny wysyła do nich komunikat żądający zatwierdzenia transakcji.



Scentralizowany 2PC (1)

- Topologia komunikacyjna



ZSBD – wykład 2 (36)

Jak wspomniano, protokół 2PC może być implementowany w jednym z trzech wariantów:

- scentralizowanego 2PC,
- zdecentralizowanego 2PC,
- liniowego 2PC.

Topologia komunikacyjna scentralizowanego protokołu 2PC została przedstawiona na slajdzie. Jak widać, wszystkie węzły komunikują się z centralnym węzłem, którym jest koordynator globalny.



Scentralizowany 2PC - przygotowanie (1)

- Koordynator globalny
 - zapisanie w lokalnym logu (na dysku) informacji o rozpoczęciu zatwierdzania
 - wysłanie do uczestników komunikatu PREPARE (przygotowanie do zatwierdzania)
 - przejście do stanu oczekiwania na komunikaty od uczestników

ZSBD – wykład 2 (37)

Na kolejnych slajdach zostaną szczegółowo omówione poszczególne fazy protokołu 2PC w topologii scentralizowanej. W praktyce ta topologia jest wykorzystywana najczęściej.

W fazie przygotowania, koordynator globalny wykonuje następujące czynności:

- zapisuje na dysku w swoim lokalnym pliku logu rekord mówiący o rozpoczęciu zatwierdzania transakcji;
- następnie wysyła do uczestników żądanie przygotowania do zatwierdzania; jest to komunikat PREPARE;
- po wysłaniu tego komunikatu KG przechodzi w stan oczekiwania na komunikaty od uczestników.



Scentralizowany 2PC - przygotowanie (2)

- Uczestnik
 - odebranie komunikatu PREPARE od KG
 - a) przygotowanie do zatwierdzenia
 - zapisanie do logu stanu transakcji i rekordu o gotowości do zatwierdzania
 - wysłanie do KG komunikatu READY_COMMIT
 - b) przygotowanie do wycofania
 - zapisanie do logu stanu transakcji i rekordu o wycofywaniu
 - wysłanie do KG komunikatu ABORT
 - przejście do stanu oczekiwania na kolejny komunikat od KG

ZSBD – wykład 2 (38)

W fazie przygotowania, uczestnik odbiera komunikat PREPARE od koordynatora globalnego i przygotowuje się do zatwierdzenia swojej transakcji lokalnej. Przygotowanie to polega na zapisaniu w swoim lokalnym pliku logu stanu transakcji i rekordu mówiącego o gotowości do zatwierdzania. Następnie uczestnik wysyła do KG komunikat READY_COMMIT, potwierdzający przygotowanie.

W przypadkach, gdy uczestnik nie może się przygotować do zatwierdzenia lub odbierze od KG komunikat ABORT (przygotowanie do wycofania), wówczas zapisuje w swoim logu stan transakcji i rekord mówiący o wycofywaniu transakcji. Następnie uczestnik wysyła komunikat ABORT do KG i przechodzi do stanu oczekiwania na kolejny komunikat.



Scentralizowany 2PC - decyzja (1)

- Koordynator globalny
 - a) jeżeli wszyscy uczestnicy odpowiedzieli READY_COMMIT
 - zapis do logu informacji o zatwierdzeniu
 - wysłanie komunikatu GLOBAL_COMMIT do uczestników
 - b) jeżeli przynajmniej jeden U odpowiedział ABORT
 - zapis do logu informacji o wycofaniu
 - wysłanie GLOBAL_ABORT do uczestników
 - oczekiwanie na potwierdzenia od uczestników

ZSBD – wykład 2 (39)

W fazie decyzji, koordynator globalny wykonuje następujące czynności.

Jeżeli wszystkie odebrane komunikaty to READY_COMMIT, wówczas KG zapisuje do swojego logu rekord mówiący o zatwierdzeniu transakcji rozproszonej. Następnie wysyła komunikat GLOBAL_COMMIT do uczestników.

Jeżeli przynajmniej jeden uczestnik odpowiedział komunikatem ABORT, wówczas KG zapisuje do swojego logu rekord mówiący o wycofaniu transakcji rozproszonej. Następnie wysyła komunikat GLOBAL_ABORT do uczestników.

Po wysłaniu komunikatu, KG przechodzi w stan oczekiwania na potwierdzenia uczestników.



Scentralizowany 2PC - decyzja (2)

- Uczestnik
- jeśli odebrany GLOBAL_COMMIT
 - zapis do logu informacji o zatwierdzeniu
 - zatwierdzenie transakcji lokalnej
 - zwolnienie blokad i zasobów systemowych
 - wysłanie potwierdzenia do KG
- jeśli odebrany GLOBAL_ABORT
 - zapis do logu informacji o wycofaniu
 - wycofanie transakcji lokalnej
 - zwolnienie blokad i zasobów systemowych
 - wysłanie potwierdzenia do KG

ZSBD – wykład 2 (40)

W fazie decyzji uczestnik wykonuje następujące czynności.

Jeżeli odebrał komunikat GLOBAL_COMMIT, wówczas:

- zapisuje w swoim logu rekord informujący o zatwierdzeniu swojej transakcji lokalnej,
- zatwierdza transakcję lokalną,
- zwalnia blokady założone przez transakcję lokalną,
- zwalnia zasoby systemowe przeznaczone do obsługi transakcji lokalnej,
- wysyła potwierdzenie zatwierdzenia transakcji do KG.

Jeżeli uczestnik odebrał komunikat GLOBAL_ABORT, wówczas:

- zapisuje w swoim logu rekord informujący o wycofaniu swojej transakcji lokalnej,
- wycofuje transakcję lokalną,
- zwalnia blokady i zasoby systemowe,
- wysyła potwierdzenie wycofania transakcji do KG.



Scentralizowany 2PC - decyzja (3)

- Koordynator globalny
 - odbiór wszystkich potwierdzeń od uczestników
 - zapis do logu informacji o zakończeniu transakcji

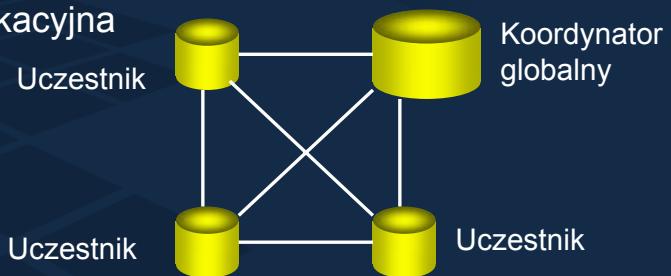
ZSBD – wykład 2 (41)

W ostatnim kroku fazy decyzji, KG odbiera potwierdzenia o zatwierdzeniu od wszystkich uczestników i zapisuje do logu rekord zakończenia transakcji.



Zdecentralizowany 2PC (1)

- Topologia komunikacyjna



- Koncepcja

- KG wysyła komunikat GLOBA_COMMIT (GLOBAL_ABORT) do wszystkich uczestników
- każdy uczestnik wysyła odpowiedz do wszystkich pozostałych węzłów

ZSBD – wykład 2 (42)

W architekturze zdecentralizowanej nie ma centralnego węzła - każdy węzeł komunikuje się z każdym.

Koncepcja działania protokołu 2PC w architekturze zdecentralizowanej jest następująca.

KG wysyła komunikat GLOBA_COMMIT lub GLOBAL_ABORT do wszystkich uczestników. Uczestnicy przygotowują się w sposób identyczny do omówionego wcześniej, ale komunikat READY_COMMIT bądź ABORT wysyłają do wszystkich węzłów. W ten sposób, każdy węzeł ma pełen obraz stanu transakcji rozproszonej.



Zdecentralizowany 2PC (2)

- Cechy
 - większy ruch sieciowy (większa liczba przesyłanych komunikatów)
 - każdy węzeł zna decyzje pozostałych węzłów i na tej podstawie zatwierdza/wycofuje sam
 - faza decyzji nie jest konieczna

ZSBD – wykład 2 (43)

Zdecentralizowany protokół 2PC cechuje się:

- większym ruchem sieciowym ze względu na większą liczbę przesyłanych komunikatów,
- możliwością wyeliminowania fazy decyzji ponieważ każdy uczestnik zna odpowiedź pozostałych uczestników i na jej podstawie może podjąć decyzję o zatwierdzeniu lub wycofaniu swojej transakcji lokalnej.



Liniowy 2PC (1)

- Topologia komunikacyjna

Koordynator globalny



- Koncepcja

- węzły są uporządkowane liniowo
- każdy węzeł otrzymuje numer od 1 (KG) do n (U)
- węzeł o numerze i komunikuje się tylko z węzłami sąsiednimi, tj. o numerach $i-1, i+1$

ZSBD – wykład 2 (44)

W architekturze liniowego 2PC węzły są uporządkowane liniowo i każdy z nich otrzymuje numer. Węzeł KG otrzymuje numer 1. Uczestnicy otrzymują kolejne numery, 2, 3, do n. W czasie wymiany komunikatów, każdy węzeł komunikuje się tylko z węzłami sąsiednimi, tj. węzłem bezpośrednio go poprzedzającym i węzłem bezpośrednio po nim następującym.



Liniowy 2PC (2)

- Faza przygotowania
 - komunikaty wysyłane od węzła 1 do n
 - podejmuje decyzję, wysyła swoją decyzję z komunikatem do węzła o numerze **i+1**
 - węzeł o numerze **i+1** podejmuje swoją decyzję i załącza ją w komunikacie do węzła **i+2**, itd.
 - ostatni węzeł w łańcuchu podejmuje decyzję o zatwierdzeniu/wycofaniu na podstawie zawartości komunikatu od węzła **n-1**
 - komunikat ten zawiera decyzje wszystkich wcześniejszych węzłów

ZSBD – wykład 2 (45)

W fazie przygotowania, komunikaty są wysyłane przez KG do węzła o numerze 2. Węzeł ten po przygotowaniu się dokłada do komunikatu otrzymanego od KD swoją decyzję. Taki rozszerzony komunikat jest następnie wysyłany do węzła 3. Węzeł 3 po przygotowaniu się dokłada swoją decyzję do komunikatu otrzymanego od węzła 2 i wysyła tak rozszerzony komunikat do węzła 4. Każdy kolejny węzeł rozszerza otrzymany komunikat o swoją decyzję dopóki komunikat nie dotrze do węzła ostatniego. Po otrzymaniu komunikatu, węzeł ostatni podejmuje decyzję o zatwierdzeniu lub wycofaniu swojej transakcji. Decyzja ta jest podejmowana na podstawie zawartości komunikatu, który otrzymał. Zawiera on bowiem informacje o przygotowani lub nieprzygotowaniu się wszystkich wcześniejszych węzłów.



Liniowy 2PC (3)

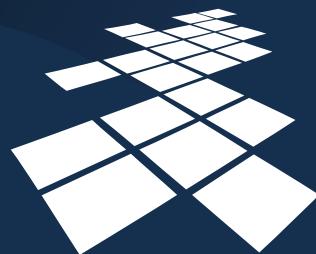
- Faza decyzji
 - komunikaty wysyłane od węzła n do 1
 - węzeł **i** podejmuje decyzję o zatwierdzeniu/wycofaniu i załączą ją w komunikacie do węzła **i-1**
 - komunikat trafiający do KG zawiera decyzje wszystkich węzłów

ZSBD – wykład 2 (46)

Po podjęciu decyzji o zatwierdzeniu (wycofaniu) ostatni węzeł umieszcza tę decyzję w komunikacie wysyłanym do węzła poprzedniego (o numerze n-1). W fazie decyzji komunikat wędruje od węzła ostatniego do KG. Każdy węzeł po drodze dopisuje do otrzymanego komunikatu swoją decyzję. W ten sposób, komunikat docierający do KG zawiera decyzje wszystkich węzłów. Na tej podstawie KG podejmuje decyzję o ostatecznym zatwierdzeniu lub wycofaniu transakcji. Decyzja ta jest wysyłana komunikatem do węzła nr 2 i propaguje się do węzła ostatniego.

Aktywne bazy danych

Wykład prowadzi:
Tomasz Koszlajda



UCZELNIA
ONLINE

Aktywne bazy danych

Niniejszy wykład jest poświęcony aktywnym bazom danych. W przeciwieństwie do klasycznych – biernych baz danych, systemy zarządzania aktywnymi bazami danych mogą podejmować autonomiczne działania związane z procesami informacyjnymi użytkowników bazy danych. Mechanizmy aktywności baz danych pozwalają w prosty sposób rozszerzać funkcjonalność systemów zarządzania bazami danych.



Plan wykładu

- Model ECA
- Dziedziny zastosowania
- Schematy aktywności
- Zdarzenia elementarne
- Definiowanie aktywnych reguł
- Zdarzenia złożone
- Metodyki projektowania

Aktywne bazy danych (2)

Celem wykładu jest poznanie mechanizmów aktywności oferowanych przez współczesne systemy zarządzania bazami danych.

W ramach wykładu zostanie przedstawiony powszechnie akceptowany model aktywności ECA. Pokazane zostaną typowe dziedziny zastosowań dla aktywnych baz danych. Następnie poznamy podstawowe właściwości aktywnych baz danych: dostępne schematy aktywności, typy zdarzeń elementarnych oraz sposób definiowania zdarzeń złożonych. Definiowanie aktywnych reguł zostanie zilustrowane przykładami z różnych komercyjnych aktywnych systemów baz danych. Na koniec przedstawimy podstawowe metodyki projektowania zbiorów aktywnych reguł.



Funkcjonalność aktywnych bazy danych

Nowa funkcjonalność:

- monitorowanie zdarzeń,
- ewaluacja warunków logicznych,
- autonomiczne uruchamianie akcji,

umożliwia podejmowanie przez system bazy danych autonomicznej aktywności w obszarze zastrzeżonym dotąd dla aplikacji bazy danych.



Aktywne bazy danych (3)

W klasycznych, nieaktywnych bazach danych wszelkie działania wykonywane przez system bazy danych, a związane z realizacją procesów informacyjnych użytkownika są uaktywniane przez aplikacje bazy danych. Autonomiczne w stosunku do aplikacji użytkownika są jedynie procesy realizujące działania systemowe, na przykład takie jak obsługa logów, wykrywanie zakleszczeń, przydział i zwalnianie zasobów, itp.

Aktywne systemy baz danych potrafią same uruchamiać zadania związane z realizacją procesów informacyjnych użytkownika, w sposób niezależny od aplikacji bazy danych. Wymaga to rozszerzenia funkcjonalności klasycznych systemów baz danych o trzy dodatkowe funkcje: monitorowania przez system zarządzania bazą danych zdarzeń zachodzących w bazie danych, ewaluacji warunków przypisanych tym zdarzeniom oraz autonomicznego „*odpalania*” akcji, czyli uruchamiania kodu specjalnych procedur składowanych w bazie danych.

Na slajdzie zilustrowano ideę działania aktywnych baz danych. Pojawiające się w historii życia bazy danych i zdefiniowane przez jej użytkowników zdarzenia są przyczyną autonomicznego *odpalania* akcji. Odpalone akcje mogą generować zdarzenia, które będą przyczyną odpalenia kolejnych akcji.

Większość współczesnych systemów baz danych posiada funkcjonalność aktywnej bazy danych.



Model ECA

Model: Event (i) - Condition (i, ii) - Action (ii)

Definiowanie aktywnych reguł przez trzy elementy:

- wystąpienie zdarzenia
- weryfikacja warunku
- *odpalenie* akcji

Aktywne reguły są wykonywane w dwóch fazach:

- (i) - faza wystąpienie zdarzenia
- (ii) - faza odpalenia akcji

Aktywne bazy danych (4)

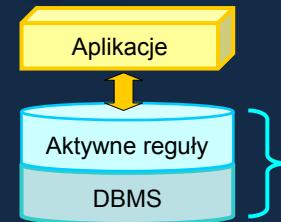
Aktywna baza danych pozwala użytkownikom wskazywać zdarzenia, które mają być monitorowane przez system i kojarzyć z wystąpieniem tych zdarzeń określone akcje do uruchomienia. Definicje obejmujące typy zdarzeń, dodatkowe warunki logiczne i akcje są nazywane aktywnymi regułami, wyzwalaczami lub triggerami. Ogólnie przyjętym modelem definiowania aktywnych reguł jest tak zwany model ECA, który to akronim pochodzi od angielskiego Event, Condition, Action, czyli po polsku zdarzenie, warunek i akcja. Dodatkowo model ECA zakłada, że te trzy elementy mogą być wykonywane w dwóch fazach. Pierwsza faza jest związana z momentem wystąpienia zdarzenia, a druga z momentem odpalenia akcji. Weryfikacja warunków logicznych może być wykonywana w pierwszej lub drugiej fazie.



Dziedziny zastosowania

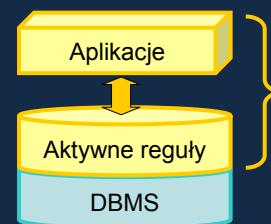
Systemowe – nowe funkcje DBMS

- Weryfikacja złożonych więzów integralności
- Utrzymanie danych wywiedzionych
- Zarządzanie rozproszonymi bazami danych
- Zarządzanie przepływami pracy



Aplikacyjne – logika biznesowa w bazie danych

- Zarządzanie procesami przemysłowymi
- Systemy giełdowe
- Bazy wiedzy



Aktywne bazy danych (5)

Można wyróżnić dwie podstawowe klasy zastosowań aktywnych baz danych.

Pierwszą klasą zastosowań są rozwiązania systemowe rozszerzające uniwersalną funkcjonalność systemu zarządzania bazą danych. Zastosowanie aktywnych reguł pozwala wzbogacić i dopasować funkcjonalność systemu bazy danych do potrzeb danego wdrożenia. Przykładem takich zastosowań jest możliwość zdefiniowania nowych rodzajów więzów integralności o bardziej złożonej semantycze niż predefiniowane rozwiązania systemowe. Innym przykładem jest automatyczne utrzymywanie przez system spójności danych wywiedzionych. Spektakularnym polem dla zastosowania takiego mechanizmu są systemy klasy OLAP ze zmaterializowanymi agregatami danych zdefiniowanymi na bazie wielu danych elementarnych. Kolejnym przykładem są rozproszone bazy danych z mechanizmami fragmentacji i replikacji danych, które rzadko są w pełni zaimplementowane przez komercyjnie dostępne systemy zarządzania bazami danych. Ostatnim przykładem niech będą systemy zarządzania przepływami pracy, w których niezbędny jest automatyzm w przekazywaniu informacji i przepływu sterowania między rozproszonymi aplikacjami realizującymi elementarne zadania procesów informacyjnych.

Druga klasa zastosowań polega na przeniesieniu wyspecjalizowanej funkcjonalności systemów informatycznych z aplikacji bazy danych do systemu bazy danych. Naturalnym przykładem takiego zastosowania aktywnych reguł są systemy zarządzania procesami przemysłowymi. Aktywne reguły pozwalały zautomatyzować żmudne monitorowanie przez użytkowników wielu krytycznych parametrów zarządzanych procesów przemysłowych. Innym przykładem są systemy do obsługi giełdy, które są odpowiedzialne za monitorowanie zmian cen akcji i szybkie podejmowanie decyzji o ich sprzedaży lub zakupie. Ostatnim przykładem niech będą bazy wiedzy będące zbiorami złożonych reguł wnioskowania, które w pewnych sytuacjach muszą być przeszacowywane.

Od kilku lat znacznym zainteresowaniem informatyków cieszą się tak zwane strumieniowe bazy danych. Są to wyspecjalizowane bazy danych, które mają służyć do sterowania procesami przemysłowymi. Jednym z podstawowych założeń strumieniowych baz danych jest wydajna obsługa tysięcy aktywnych reguł monitorujących strumienie informacji pochodzących z procesów i urządzeń przemysłowych.



Właściwości aktywnych baz danych

- **Modele aktywności** – zależności czasowe i przyczynowo-skutkowe między zdarzeniami i akcjami
- **Zdarzenia elementarne** – zbiór typów zdarzeń, które mogą być podstawą definiowania aktywnych reguł
- **Operatory zdarzeniowe** – zbiór operatorów umożliwiających specyfikację złożonych wyrażeń zdarzeniowych
- **Kontekst definicji aktywnych reguł** – pojedyncza dana, zbiór danych, baza danych

Aktywne bazy danych (6)

Ze względu na bardzo szeroką klasę zastosowań aktywnych baz danych niezbędna jest możliwość elastycznej konstrukcji aktywnych reguł dla ich dostosowania do zastosowań o różnej charakterystyce. Wyróżnia się cztery podstawowe właściwości aktywnych baz danych, które decydują o sile modelu poszczególnych systemów.

Pierwszą taką właściwością są dostępne w aktywnej bazie danych schematy aktywności określające zależności zachodzące między zdarzeniem, a wyzwalaną przez nie akcją. Duża liczba dostępnych schematów aktywności przyczynia się do uniwersalności modelu danej aktywnej bazy danych. Drugą właściwością jest zbiór typów zdarzeń elementarnych, które mogą być użyte do definicji aktywnych reguł. Większy zbiór typów zwiększa zakres zastosowań aktywnej bazy danych. Kolejną właściwością jest zbiór dostępnych operatorów zdarzeniowych za pomocą których definiuje się zdarzenia złożone, na które może składać się wiele zdarzeń elementarnych. Ostatnią właściwością jest kontekst definiowania aktywnych reguł. Kontekstem definicji może być pojedyncza dana, zbiór danych lub cała baza danych.



Schematy aktywności

Można wyróżnić trzy relacje zachodzące między fazami wystąpienia zdarzenia i uruchomienia akcji:

- Czasowa
- Transakcyjna
- Zależności



Aktywne bazy danych (7)

Można wyróżnić trzy relacje zachodzące między wystąpieniem zdarzenia zdefiniowanego w aktywnej regule, a odpalaną w związku z tym akcją. Są to relacje czasowa, transakcyjna i zależności.

Relacja czasowa określa, czy moment odpalenia akcji reguły pokrywa się z momentem wystąpienia zdarzenia, czy też akcja jest odroczona w czasie do zakończenia transakcji która uruchomiła zdarzenie odpalające akcję.

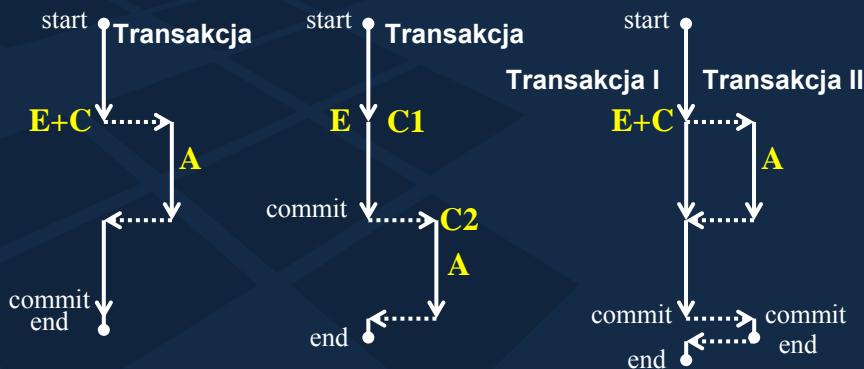
Relacja transakcyjna określa, czy procedura akcji jest uruchamiana jako część transakcji, w ramach której wystąpiło zdarzenie, czy jest osobną transakcją.

Relacja zależności określa, czy zatwierdzenie zakończenia akcji jest zależne od pomyślnego zakończenia transakcji w ramach której wystąpiło zdarzenie.

Te trzy relacje definiują trójwymiarową przestrzeń dostępnych schematów aktywności. Większość systemów komercyjnych wspiera jedynie najprostszy schemat aktywności obejmujący natychmiastowe odpalenie akcji, w ramach tej samej transakcji, która aktywowała zdarzenie. Jednak za pomocą dodatkowych mechanizmów systemowych można nieco większym nakładem pracy budować reguły o innych, bardziej złożonych modelach aktywności.

Przykłady schematów aktywności

- a) Natychmiastowe, zależne i łączne
- b) Odroczone, zależne i łączne
- c) Natychmiastowe, zależne, osobne



Aktywne bazy danych (8)

Na slajdzie zobrazowano trzy przykładowe schematy aktywności.

W pierwszym przykładzie faza zdarzenia i faza akcji są natychmiastowe, zależne i łączne. Źródłem zdarzenia jest transakcja, która wykonała operację stanowiącą zdarzenie uaktywniające reguły. W momencie, w którym wystąpiło zdarzenie transakcja ta jest przerywana i odpalana jest akcja reguły. Akcja wykonuje się jako część transakcji, która była przyczyną jej uruchomienia. Efektem jest zależność wykonania akcji od tej transakcji, bowiem wycofanie transakcji wycofa również wszystkie odwracalne działania, które wykonała akcja. Po zakończeniu wszystkich działań składających się na akcję, sterowanie wraca do głównego wątku transakcji, która jest kontynuowana. Taki model aktywności może być wykorzystany na przykład do utrzymywania danych wywiedzionych lub weryfikacji więzów integralności.

Drugi przykład pokazuje model, w którym faza zdarzenia i faza akcji są odroczone, zależne i łączne. W tym wypadku, w momencie wystąpienia zdarzenia może być weryfikowana część warunków logicznych zdefiniowanych w aktywnej regule. Jednak wątek transakcji nie jest przerywany. Odpalenie akcji jest odroczone do osiągnięcia przez transakcję fazy akceptacji. W tym momencie, przed odpaleniem akcji dodatkowo może być weryfikowana druga część warunku logicznego. Po zakończeniu akcji transakcja jest zatwierdzana. Ten model aktywności jest właściwy do zarządzania procesami przemysłowymi. W tym wypadku bowiem pewne działania akcji wykonywane na zarządzanym procesie przemysłowym mogą być nieodwracalne, i w związku z tym powinny być przesunięte na koniec transakcji.

W trzecim przykładzie fazy zdarzenia i akcji są natychmiastowe, zależne i osobne. W momencie, w którym wystąpiło zdarzenie jest tworzona osobna transakcja, która jest przetwarzana równolegle do transakcji powodującej wystąpienie zdarzenia. Przedstawiany model jest zależny, bowiem zatwierdzenie akcji jest zależne od zatwierdzenia właściwej transakcji. Przykładem zastosowania tego modelu aktywności są sytuacje, w których działania akcji muszą być wykonywane na konto innego użytkownika bazy danych niż użytkownik głównej transakcji.



Zdarzenia elementarne

- **Zmiany stanu bazy danych**
Wstawienie, usunięcie, modyfikacja, odczyt, dostęp
- **Zmiany schematu bazy danych**
Tworzenie, modyfikacja, usunięcie
- **Zdarzenia w systemie bazy danych**
LOGIN, LOGOFF, STARTUP, SHUTDOWN, ERROR
- **Zdarzenia związane ze zmianami stanu transakcji**
Rozpoczęcie, punkt akceptacji, zatwierdzenie, wycofanie
- **Zdarzenia temporalne**
punkt czasu, upływ czasu, periodycznie
- **Zdarzenia zgłaszane przez użytkownika**

Aktywne bazy danych (9)

Funkcjonalność aktywnych reguł zależy od zbioru dostępnych typów zdarzeń elementarnych, które mogą odpalać akcję reguły. Najbardziej typową grupą zdarzeń są operacje wykonywane przez transakcje na stanie bazy danych. W większości komercyjnych systemów baz danych zbiór ten jest ograniczony do operacji modyfikacji: wstawienia nowych danych do bazy danych, zmiany wartości danych i usunięcia danych. Niektóre systemy prototypowe umożliwiają zastosowanie do definicji aktywnych reguł również operację odczytu danych.

Drugą grupą zdarzeń są operacje modyfikacji schematu bazy danych obejmujące tworzenie nowych struktur danych, ich modyfikacja lub usuwanie, nadawanie lub odbieranie uprawnień użytkownikom bazy danych, przyłączanie i odłączanie do zbiorów danych ich statystyk wykorzystywanych przy optymalizacji zapytań, itp. Przykładem systemu, w którym definicje reguł mogą korzystać z takiej kategorii zdarzeń jest system Oracle.

Kolejną grupą zdarzeń są operacje w systemie zarządzania bazą danych obejmujące przyłączanie się i odłączanie użytkowników od systemu bazy danych, pojawiające się w trakcie pracy błędów systemowych oraz operacje uruchamiania i zatrzymywania systemu. Zdarzenia takie mogą być wykorzystane do diagnostyki systemu bazy danych.

Następna grupa zdarzeń jest związana z przetwarzaniem transakcji i obejmuje operacje rozpoczęcia, osiągania punktu akceptacji, zatwierdzania i wycofywania transakcji.

Przedostatnia grupa dotyczy zdarzeń temporalnych. Obejmuje ona zdarzenia polegające na wystąpieniu określonych punktów czasu, na przykład piątki godzina 17:15. Innym typem zdarzeń temporalnych jest wzajemny upływ czasu od danego momentu, na przykład upływ pięciu minut i piętnastu sekund. Ostatnim typem zdarzenia należącego do tej grupy jest periodyczny upływ czasu, na przykład co dwie godziny.

Ostatnia grupa obejmuje pojedynczy typ zdarzenia, którym jest jawne zgłoszenie zdarzenia przez użytkownika, które może być przesłane z poziomu aplikacji bazy danych. Ostatnie trzy wymienione kategorie zdarzeń występują jedynie w systemach prototypowych.



Definiowanie aktywnych reguł Oracle

```
create trigger Zmodyfikuj_stan_osobowy
  after insert on Pracownicy
  for each row
begin
  update Zespoły
  set stanOsob = stanOsob + 1
  where idZesp = :new.idZesp;
end;
```

idPrac	Nazwisko	idZesp
100	Tarzan	10
110	Kowalski	10

↑ zdarzenie: insert

120	Nowak	10
-----	-------	----

idZesp	stanOsob
10	2



idZesp	stanOsob
10	3

Aktywne bazy danych (10)

Na slajdzie pokazano przykład aktywnej reguły zdefiniowanej w systemie bazy danych Oracle. Zadaniem reguły jest utrzymywanie wywiedzonego atrybutu „stanOsobowy” relacji „Zespoły” w wypadku dodawania do relacji „Pracownicy” krotek reprezentujących nowo przyjmowanych pracowników. Kompletny przykład powinien jeszcze obejmować przypadki zwalniania pracowników i przenoszenia ich między zespołami.

Zdarzeniem uaktywniającym regułę jest wystąpienie operacji INSERT na relacji „Pracownicy”. Słowo kluczowe AFTER użyte w drugiej linii definicji oznacza, że akcja reguły zostanie odpalona bezpośrednio po wykonaniu pojedynczej operacji wstawienia krotki. Fraza "for each row" występująca w trzeciej linii określa, że kontekstem akcji reguły jest pojedyncza krotka. Dla pojedynczego wywołania instrukcji INSERT, która wstawi 100 nowych krotek do relacji, akcja reguły zostanie odpalona stukrotnie, raz dla każdej wstawionej krotki. Akcja reguły jest zdefiniowana między słowami kluczowymi „begin” i „end”. Prefiks :New poprzedzający nazwę atrybutu „IdZesp” w ciele akcji reguły jest odwołaniem do wartości tego atrybutu we wstawianej krotce.

Na rysunku poniżej definicji przedstawiono przykładowy scenariusz odpalenia akcji zdefiniowanej reguły „Zmodyfikuj_Stan_Osobowy”. Do relacji „Pracownicy” zawierającej dwie krotek reprezentujące pracowników Tarzana i Kowalskiego dodano trzecią krotkę reprezentującą nowego pracownika Nowaka. Bezpośrednio po wstawieniu nowej krotki jest odpalana akcja reguły. W wyniku wykonania akcji stan osobowy zespołu dziesiętego jest zwiększany o jeden.



Definiowanie aktywnych reguł Oracle

```
create trigger Kontrola_płac
  after update of pensja on Pracownicy
  for each row
  when (new.pensja < 1000)
begin
  raise_application_error(
    -20000, 'Poniżej płacy minimalnej');
end;
```

$(0.9 * 1800 < 1000) \rightarrow \text{False}$

idPrac	Nazwisko	pensja
100	Tarzan	1800
110	Kowalski	2500

X
Nie

idPrac	Nazwisko	pensja
100	Tarzan	1620
110	Kowalski	2500

↑ zdarzenie:
`update Pracownicy
 set pensja = 0.9 * pensja
 where nazwisko = 'Tarzan'`

Aktywne bazy danych (11)

Na slajdzie przedstawiono kolejny przykład definicji aktywnej reguły w systemie Oracle, który zawiera warunek logiczny determinujący odpalenie akcji reguły. Zadaniem pokazanej reguły jest uniemożliwienie obniżenia płacy pracowników poniżej kwoty 1000 zł. Zdarzeniem uaktywniającym regułę jest modyfikacja atrybutu „*pensja*” relacji „*Pracownicy*”. Definicja reguły zawiera dodatkową klauzulę WHEN, w której zdefiniowano warunek konieczny dla odpalenia akcji reguły. Prefiks „*new*” użyty w warunku logicznym służy do odwołania się do wartości atrybutów modyfikowanych krotek. Zdefiniowana w ciele reguły akcja zawiera wywołanie funkcji zgłaszającej błąd i wycofującej modyfikację.

Poniżej definicji przedstawiono przykładowy scenariusz, w którym jest modyfikowana pensja pracownika o nazwisku Tarzan. W tym wypadku weryfikacja warunku skojarzonego z regułą jest negatywna. Pensja po modyfikacji pozostaje na poprawnym poziomie powyżej 1000 zł. W związku z tym, mimo wystąpienia zdarzenia odpalającego akcję, w tym wypadku akcja reguły nie zostanie odpalone.



Definiowanie aktywnych reguł SQLServer

```
create trigger liczba_prac ON PRACOWNICY
after delete as
    update zespoly
    set liczba_prac = liczba_prac - (
        select count(*) from deleted
        where deleted.id_zesp = zespoly.id_zesp)
    where id_zesp in (
        select distinct id_zesp from deleted)
```

idPrac	Nazwisko	idZesp
100	Tarzan	10
110	Kowalski	10

idZesp	stanOsob
10	2



idZesp	stanOsob
10	1

↑ zdarzenie:
`delete from Pracownicy
where nazwisko='Tarzan'`

Aktywne bazy danych (12)

Kolejny przykład jest ilustracją sposobu definiowania aktywnych reguł w systemie SQLServer. Zadaniem aktywnej reguły przedstawionej na slajdzie jest utrzymywanie poprawnej wartości atrybutu „StanOsobowy” ze względu na operację usuwania krotek z relacji Pracownicy. W systemie SQLServer jedynym dostępnym kontekstem definiowania akcji reguł jest relacja. W porównaniu z systemem Oracle nie jest dostępny kontekst pojedynczych krotek. Dostęp do wartości modyfikowanych danych jest możliwy przez dwie robocze relacje: „*insertem*” i „*deleted*”, które są widoczne tylko i wyłącznie wewnątrz akcji aktywnych reguł. Schemat tych relacji jest taki sam jak schemat relacji, na której ma miejsce zdarzenie modyfikacji uaktywniające daną regułę. Relacja „*insertem*” przechowuje nowo wstawione krotki lub nowe wartości modyfikowanych krotek. Relacja „*deleted*” przechowuje usunięte krotki i stare wartości modyfikowanych krotek.

W pokazanym przykładzie akcja reguły odejmuje od starej wartości atrybutu „StanOsobowy” liczbę usuniętych krotek pracowników dla danego zespołu. Scenariusz obejmuje operację usunięcia z relacji „*Pracownicy*” krotki pracownika z zespołu 10. Uaktywniona przez to zdarzenie aktywna reguła zmniejsza składowaną liczbę pracowników tego zespołu.



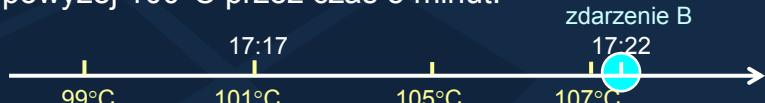
Zdarzenia złożone

Aktywne reguły wyzwalane zdarzeniami złożonymi:

- Zdarzenie A: trzy kolejne obniżki cen akcji giełdowych w czasie trzech dni



- Zdarzenie B: utrzymywanie się temperatury reaktora powyżej 100°C przez czas 5 minut.



Aktywne bazy danych (13)

Wszystkie pokazane dotąd aktywne reguły były wyzwalane pojedynczymi zdarzeniami elementarnymi. Tymczasem jest wiele zastosowań aktywnych baz danych, które wymagają definiowania aktywnych reguł na bazie zdarzeń złożonych. Na slajdzie pokazano dwa przykłady zdarzeń złożonych.

Pierwszy przykład dotyczy dziedziny notowań giełdowych. Aktywna reguła, której celem jest automatyczny zakup akcji po trzykrotnym spadku jej wartości mającym miejsce w czasie nie dłuższym niż trzy dni. Na dodatek spadki cen akcji nie mogą być przedzielone wzrostem cen. Zdarzenie to zostało zilustrowane na pierwszym wykresie. Punktem czasowym wystąpienia tego zdarzenia złożonego jest trzeci spadek ceny z dnia 3 kwietnia.

Drugi przykład odwołuje się do dziedziny zarządzania procesami przemysłowymi. Wyobraźmy sobie aktywną regułę, której celem jest awaryjne wyłączenie reaktora jeżeli jego temperatura przez czas dłuższy niż 5 minut będzie przekraczać 100°C . Przykład takiego zdarzenia pokazano na drugim wykresie. Odczyt temperatury reaktora z godziny 17:17 pokazał temperaturę przekraczającą wartość progową 100°C . Kolejne odczyty również pokazywały temperatury wyższe niż 100°C . Po pięciu minutach o godzinie 17:22 wystąpiło określone zdarzenie złożone.

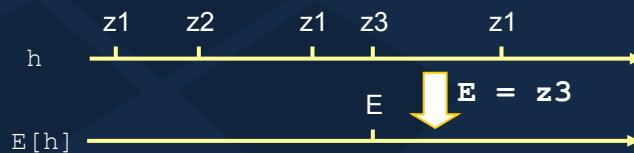


Wyrażenia zdarzeniowe

Ewaluacja zdarzeń złożonych wymaga utrzymywania i analizy historii zdarzeń.

Historia zdarzeń jest skończonym zbiorem zdarzeń, w którym żadne dwa zdarzenia nie mogą mieć tego samego znacznika czasowego.

Wyrażenie zdarzeniowe E określone w historii zdarzeń h wyznacza podzbiór tych zdarzeń historii h , w których zdarzenie E jest spełnione.



Aktywne bazy danych (14)

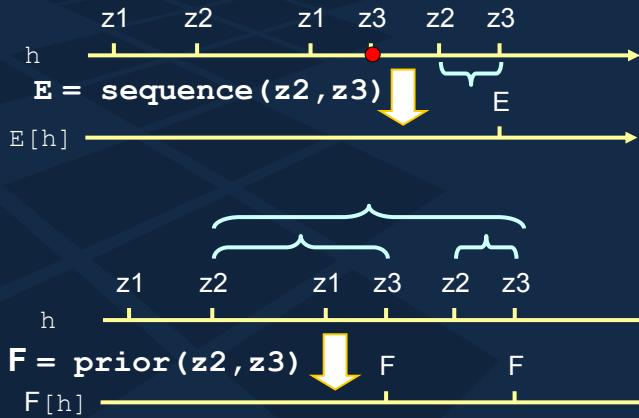
Jak pokazano na przykładach na pojedyncze zdarzenie złożone może się składać wiele wystąpień zdarzeń elementarnych. Wartościowanie zdefiniowanych zdarzeń złożonych wymaga utrzymywania i analizy całych historii zdarzeń elementarnych. Historia zdarzeń jest skończonym zbiorem zdarzeń, w którym każdemu zdarzeniu jest przypisany unikalny znacznik czasowy. Znaczniki czasowe jednoznacznie porządkują zbiór zdarzeń tworzących daną historię. W aktywnej bazie danych z każdą aktywną regułą jest skojarzona osobna historia pamiętająca zdarzenia elementarne użyte w definicji zdarzenia złożonego.

Zdarzenia złożone są definiowane przez wyrażenia zdarzeniowe konstruowane za pomocą zdarzeń elementarnych i operatorów zdarzeń złożonych. Wyrażenie zdarzeniowe wartościowane w ramach danej historii wyznacza podzbiór zdarzeń historii, w których wyrażenie to jest spełnione. Ilustruje to rysunek na dole slajdu. Na historię „ h ” składa się pięć wystąpień trzech różnych typów zdarzeń: „ $z1$ ”, „ $z2$ ” i „ $z3$ ”. Zdefiniowane wyrażenie zdarzeniowe składa się z nazwy typu zdarzenia elementarnego „ $z3$ ”. Zdarzenia elementarne są szczególnym przypadkiem zdarzeń złożonych. Tak zdefiniowane zdarzenie występuje w historii „ h ” tylko raz w momencie wystąpienia zdarzenia „ $z3$ ”.



Operatory zdarzeniowe

Operatory następstwa: **sequence** i **prior**



Aktywne bazy danych (15)

Na kolejnych slajdach zostaną pokazane wybrane operatory zdarzeniowe. Propozycja przedstawionego zbioru operatorów pochodzi z prototypowego systemu aktywnej bazy danych ODE (Object Database and Environment).

Dwa pokazane na slajdzie operatory umożliwiają opis własności następstwa dwóch zdarzeń w historii. Operator sekwencji – „sequence” opisuje następstwo polegające na bezpośrednim wystąpieniu jednego zdarzenia po drugim, w taki sposób, że nie są one przedzielone wystąpieniem żadnego innego zdarzenia. Wyrażenie zdarzeniowe pokazane na rysunku opisuje sekwencję wystąpień dwóch zdarzeń elementarnych typu „z2” i „z3”. Zdarzenie złożone E zdefiniowane za pomocą tego wyrażenia nie zajdzie dla pierwszego wystąpienia zdarzenia „z3” – co symbolizuje czerwona kropka, ponieważ wystąpienie to jest oddzielone od wystąpienia zdarzenia „z2”, zdarzeniem „z1”. Dopiero drugie wystąpienie zdarzenia „z3” jest wystąpieniem zdarzenia złożonego E.

Drugi operator następstwa – „prior” opisuje następstwo dwóch zdarzeń, które mogą być przedzielone dowolną liczbą innych zdarzeń. Zdefiniowane w drugim przykładzie wyrażenie zdarzeniowe opisuje następstwo typu „prior” dla zdarzeń typu „z2” i „z3”. Zdarzenie złożone F opisane tym wyrażeniem wystąpi dwukrotnie w danej historii h, w punktach wystąpienia zdarzenia „z3”. Zwróćmy uwagę, że drugie wystąpienie zdarzenia F miałoby miejsce również w sytuacji, gdyby usunąć z historii „h” drugie wystąpienie zdarzenia „z2”. Zdarzenie „z3” występuje po pierwszym wystąpieniu zdarzenia „z2” dwukrotnie. Każde z tych wystąpień jest wystąpieniem zdarzenia złożonego F. Symbolizuje to duża błękitna klamra nad rysunkiem.



Operatory zdarzeniowe

Operatory logiczne: or, and i not



Aktywne bazy danych (16)

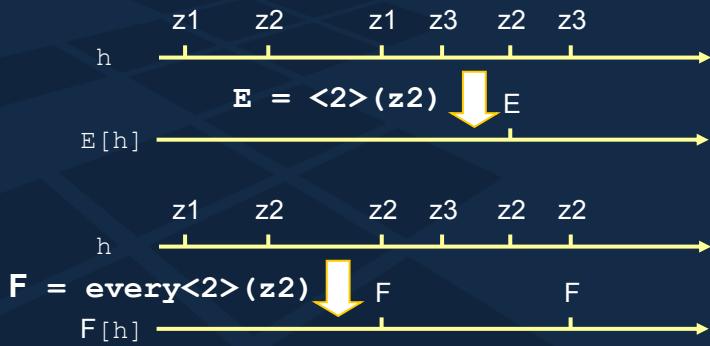
Kolejną grupą operatorów zdarzeniowych są operatory logiczne: sumy, iloczynu i negacji zdarzeń. Suma logiczna dwóch typów zdarzeń obejmuje wszystkie wystąpienia zdarzenia pierwszego lub drugiego typu. Iloczyn logiczny dwóch typów zdarzeń występuje w punktach czasowych, w których jednocześnie występują zdarzenia obydwu typów. Z definicji historii zdarzeń elementarnych wynika, że iloczyn logiczny dwóch różnych typów zdarzeń elementarnych jest dla dowolnej historii pusty. W użytecznych zastosowaniach argumentami operatora iloczynu zdarzeń muszą być zdarzenia złożone, których nie dotyczy ograniczenie unikalności znaczników czasowych. Z kolei negacją zdarzenia danego typu są wystąpienia wszystkich innych typów zdarzeń.

Na slajdzie pokazano przykład wyrażenia, które zwiera operatory sumy i negacji. Zdarzenie E będzie występowało w punktach czasowych, w których występuje zdarzenie „ $z3$ ” lub dowolne inne zdarzenie różne od zdarzenia „ $z1$ ”. W pokazanym konkretnym przypadku wartościami tego wyrażenia będą wystąpienia zdarzeń „ $z2$ ” i „ $z3$ ”.



Operatory zdarzeniowe

Operatory wyliczeniowe: $\langle n \rangle E$, $\text{every}\langle n \rangle E$



Aktywne bazy danych (17)

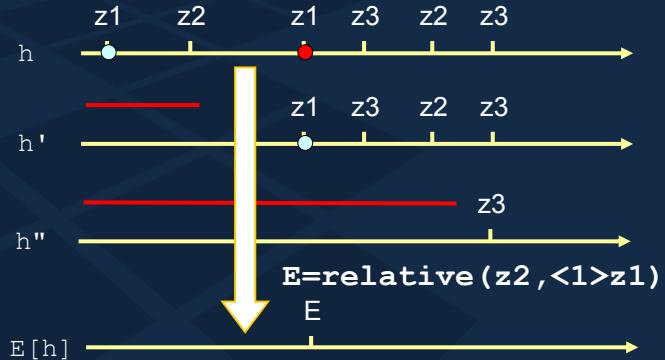
Następna grupa operatorów to operatory wyliczeniowe. Pierwszy z operatorów wyliczeniowych wyznacza dla danego typu zdarzenia „ t ” i danej liczby naturalnej „ n ”, „ n ”-te wystąpienie zdarzenia typu „ t ”. Działanie tego operatora ilustruje pierwszy rysunek. Zdarzenie złożone E jest opisane przez wyrażenie wyznaczające drugie wystąpienie zdarzenia typu „ $z2$ ”. W dowolnie długiej historii może wystąpić co najwyżej jedno takie zdarzenie.

Drugi operator jest cyklicznym operatorem wyliczeniowym, który dla danego typu zdarzenia „ t ” i danej liczby naturalnej „ n ” wyznacza każde „ n ”-te wystąpienie zdarzenia typu „ t ”. W drugim przykładzie jest zdefiniowane zdarzenie złożone F , jako co drugie wystąpienie zdarzenia typu „ $z2$ ”.



Operatory zdarzeniowe

Operator następstwa: **relative**



Aktywne bazy danych (18)

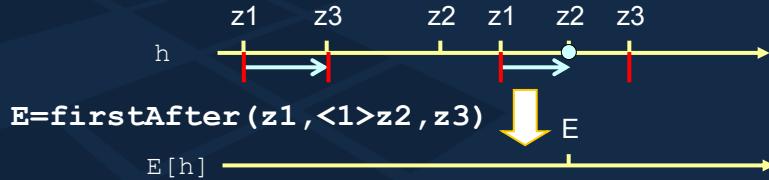
Następnym rozpatrywanym operatorem jest operatorem względnego następstwa zdarzeń, czyli operator „*relative*”. Operator ten ma odmienną semantykę od poznanych wcześniej operatorów następstwa „sequence” i „prior”. Operator „*relative*” jest takim następstwem dwóch zdarzeń „z1” i „z2”, w którym wystąpienie zdarzenia typu „z2” musi wystąpić w historii zdarzeń h' uzyskanej ze źródłowej historii „h”, przez usunięcie z niej wystąpienia zdarzenia „z1” i wszystkich zdarzeń je poprzedzających.

Na rysunku pokazano przykład definicji zdarzenia złożonego E, które jest pierwszym wystąpieniem zdarzenia „z1” w historii po wystąpieniu zdarzenia „z2”. W pierwszej linii na niebiesko zaznaczono pierwsze wystąpienie zdarzenia „z1” w całej historii zdarzeń „h”. Druga linia pokazuje historię h' uzyskaną przez usunięcie z historii „h” pierwszego wystąpienia zdarzenia typu „z2” i wszystkich zdarzeń je poprzedzających. Tym razem, niebieska kropka pokazuje pierwsze wystąpienia zdarzenia „z1” w tej pod-historii. Punkt ten jest jedynym wystąpieniem zdarzenia złożonego E w historii „h”, bowiem w historii h' utworzonej przez usunięcie z niej drugiego wystąpienia zdarzenia „z2” i wszystkich zdarzeń je poprzedzających, zdarzenie „z1” i w konsekwencji zdarzenie E już nie występują.



Operatory zdarzeniowe

Operator okna: **firstAfter**



Aktywne bazy danych (19)

Ostatnim prezentowanym przykładem operatora zdarzeniowego jest operator okna „`firstAfter`”. Argumentami tego operatora są trzy typy zdarzeń. Pierwszy i trzeci argument definiują okno, w którym oczekiwane jest zdarzenie będące drugim argumentem operatora. Na slajdzie za pomocą operatora „`firstAfter`” jest zdefiniowane zdarzenie złożone E . Okno określają wystąpienia zdarzeń „ z_1 ” i „ z_3 ”. Zdarzenie E będzie zachodzić w pierwszym wystąpieniu zdarzenia „ z_2 ”, które mieści się w takim oknie. W podanym scenariuszu, pierwsze okno ograniczone przez zdarzenia „ z_1 ” i „ z_3 ” jest puste. W drugim oknie znajduje się wystąpienie zdarzenia „ z_2 ”. Moment jego wystąpienia jest wystąpieniem zdarzenia E .

Przedstawione operatory zdarzeniowe nie są jedynymi zdefiniowanymi w aktywnej bazie danych ODE. System ten zawiera jeszcze operatory rekurencyjne, potokowe i inne odmiany operatora okien. W komercyjnych systemach baz danych zbiór dostępnych operatorów zdarzeniowych jest ograniczony do operatora sumy logicznej.



Aktywna reguła ze zdarzeniem złożonym

Przełącz układ sieci jeżeli przepustowość przez 5 minut jest mniejsza od wartości progowej równej 10

```
#define PoniżejProgu AFTER UPDATE && przepustowość<10
#define PowyżejProgu AFTER UPDATE && przepustowość>10
class Łącze {
private:
    float przepustowość;
public:
    boolean Przełącz();
trigger:
    SpadekPrzepustowości():separate dependent
    firstAfter( relative(
        ZmianaPowyżejProgu,
        <1>(PoniżejProgu),
        AFTER TIME (M=5), PowyżejProgu)
    ==> Przełącz());}
```

Aktywne bazy danych (20)

Na slajdzie pokazano przykład kompletnej definicji aktywnej reguły w systemie ODE zawierającej specyfikację zdarzenia złożonego. Definicja wykorzystuje poznane na wykładzie operatory zdarzeniowe. Zadaniem aktywnej reguły jest przełączenie łącznika komputerowej w wypadku gdy przepustowość łącznika przez czas pięciu minut będzie niższa niż minimalna wartość progowa równa 10.

System bazy danych ODE ma model obiektowy. Definicja aktywnych reguł jest częścią definicji klas. Przykład zawiera definicję klasy „Łącze”. Stan atrybutu przepustowość odzwierciedla aktualną przepustowość danego łącznika. Nie pokazana na przykładzie metoda klasy „Łącze” monitoruje faktyczny stan łącznika i odpowiednio modyfikuje wartości atrybutu „przepustowość”. Dwie makrodefinicje opisują dwa zdarzenia elementarne modyfikacji stanu obiektów klasy „Łącze”. Pierwsze zdarzenie dotyczy modyfikacji, po której stan atrybutu „przepustowość” jest mniejszy niż 10, a drugie modyfikacji po której wartość tego atrybutu jest większa niż 10. Operator && służy do definicji warunku reguły. Publiczna metoda „Przełącz” zawiera kod akcji reguły.

Schemat aktywności zdefiniowanej reguły o nazwie „SpadekPrzepustowości” jest określony przez słowa kluczowe „separate” i „dependent”. Słowo „separate” oznacza, że akcja reguły jest odpalana jako osobna transakcja. Natomiast słowo kluczowe „dependent” oznacza, że zatwierdzenie działania akcji jest zależne od zatwierdzenia transakcji modyfikującej atrybut „przepustowość”. Do definicji zdarzenia złożonego wykorzystano operator okna „firstAfter”. Okno jest ograniczone przez zdarzenia spadku przepustowości poniżej wartości progowej i następującego po nim powrotu powyżej wartości progowej. Zdarzeniem wyszukiwanym w oknie jest upływ 5 minut od otwarcia okna, który odpowiada celowi działania aktywnej reguły.



Metodyka projektowania aktywnych reguł

Projekt dużego zbioru potencjalnie interferujących ze sobą reguł wymaga spełnienia pewnych wymagań:

- **własność stopu** – przetwarzanie akcji reguł uaktywnionych przez pojedyncze zdarzenie zostanie zakończone w skończonym czasie
- **determinizm stanu** – kolejność wykonania akcji reguł uaktywnionych tym samym zdarzeniem nie ma wpływu na końcowy stan bazy danych

Aktywne bazy danych (21)

Aktywne reguły są autonomicznymi jednostkami programowymi. Poprawne działanie pojedynczej reguły, nie gwarantuje poprawności działania całego zbioru niezależnie zaprojektowanych i zbudowanych reguł. Aktywne reguły mogą wchodzić we wzajemne interakcje, na przykład gdy dwie różne reguły są uaktywniane tym samym zdarzeniem lub gdy operacje wykonywane w ramach akcji jednej z reguł są zdarzeniami uaktywniającymi inne reguły. Popularną metodą służącą do zaprojektowania dużego zbioru reguł jest modularyzacja polegająca na grupowaniu reguł w autonomiczne warstwy, na przykład tematyczne. Reguły znajdujące się w różnych warstwach nie powinny wchodzić we wzajemne interakcje. Natomiast problem potencjalnych interakcji reguł znajdujących się w tej samej warstwie powinien być rozwiązyany przez wspólny projekt wszystkich reguł z danej warstwy.

Zbiór potencjalnie powiązanych reguł powinien spełniać dwa podstawowe wymagania. Po pierwsze taki zbiór powinien posiadać własność stopu, która wymaga by przetwarzanie akcji reguł wyzwolonych przez zdarzenie zostało zakończone w skończonym czasie. Spełnienie tego warunku dla zbioru autonomicznie projektowanych aktywnych reguł potencjalnie nie powiązanych tematycznie jest trudniejsze niż dla zintegrowanych programów. Drugą wymaganą własnością, którą musi spełniać zbiór aktywnych reguł jest determinizm stanu. Własność ta polega na tym, by kolejność odpalania akcji reguł uaktywnianych tym samym zdarzeniem nie miała wpływu na końcowy stan bazy danych. Trudność w spełnieniu tego wymagania wynika z faktu, że w komercyjnych systemach aktywnych baz danych nie ma mechanizmu do programowego określania kolejności uruchamiania reguł uaktywnionych tym samym zdarzeniem. Oznacza to, że system zarządzania bazą danych odpala akcje takich reguł w arbitralnie ustalonej kolejności.

Własność stopu zbioru reguł

```

create trigger zwiększa_rangę
after update of wysokość on Premie
for each row
when new.wysokość - old.wysokość > 400
begin
    update Pracownicy set ranga = ranga+1
    where nr = :new.nr_prac;
end

create trigger zwiększa_premię
after update of ranga on Pracownicy
for each row
begin
    update Premie set wysokość=wysokość+500*:new.ranga
    where nr_prac = :new.nr;
end

```

Aktywne bazy danych (22)

Na slajdzie pokazano przykład zbioru dwóch reguł, który nie posiada własności stopu. Przedstawione reguły obsługują bazę danych pracowników. Pozycja każdego pracownika jest określona przez jego rangę. Im większa ranga danego pracownika tym większe premie otrzymuje. Awans o jedną rangę oznacza podwyżkę premii o 500 zł. Ta zależność jest utrzymywana automatycznie przez regułę „*zwiększa_premię*”. Premie mogą być podnoszone również w wyniku innych zasług pracownika. Podniesienie premii o ponad 400 zł skutkuje podwyższeniem rangi pracownika. Ta zależność jest utrzymywana przez regułę o nazwie „*zwiększa_rangę*”. Pojawienie się z zewnątrz akcji reguł zdarzenia zwiększenia rangi pracownika pociąga za sobą cykliczne wywołania dwóch przedstawionych reguł. Podniesie rangi pociąga za sobą podniesienie premii, które z kolei jest przyczyną podniesienia rangi, itd. Podany zbiór reguł nie posiada więc własności stopu.

Własność stopu

```

CREATE TRIGGER kotwica_budżetowa
  AFTER INSERT OR UPDATE OF budżet ON Zespoły
  DECLARE
    suma float;
  BEGIN
    SELECT sum(budżet) INTO suma FROM Zespoły;
    IF suma > 100 THEN
      UPDATE Zespoły
      SET budżet = 0.9 * budżet;
    END IF;
  END;
  
```

idZesp	budżet
10	50
20	50

insert

idZesp	budżet
10	45
20	45
30	18

fire

idZesp	budżet
10	40,5
20	40,5
30	16,2

fire

stop

Aktywne bazy danych (23)

Wzajemne wywoływanie się reguł nie musi wiązać się z brakiem własności stopu. Po kilku iteracjach taki cykl może zostać zatrzymany. Przypadek takiego pokazano na przykładzie pojedynczej reguły, której akcja obejmuje operacje, które są zdarzeniem rekurencyjnie odpalającym akcję reguły.

Zadaniem przedstawionej reguły jest uniemożliwienie przekroczenia zadanego górnego pułpu budżetu firmy ustalonego na 100 jednostek. Modyfikacje budżetu poszczególnych zespołów firmy są weryfikowane przez regułę „kotwica_budżetowa”. W wypadku przekroczenia zadanego progu budżetu całej firmy, budżety poszczególnych zespołów są redukowane do 90% poprzedniej wartości.

Pokazany na slajdzie scenariusz, w którym do dwóch istniejących zespołów dodano trzeci o budżecie w wysokości 20 jednostek wyzwolił akcję reguły. Ponieważ sumaryczny budżet wynosi teraz 120 jednostek, akcja reguły redukuje budżety zespołów. Modyfikacja budżetów cyklicznie uruchamia akcję reguły. Ograniczony budżet do wysokości 108 jednostek w dalszym ciągu przekracza dopuszczalną wartość. Akcja rekurencyjnie wywołanej reguły dokonuje kolejnej redukcji budżetu, tym razem do akceptowalnego poziomu. W tym momencie, cykl wywoływania reguły zastaje zatrzymany.



Statyczna walidacja własności stopu

Graf wyzwalania (GW)

Węzły grafu reprezentują zbiór aktywnych reguł. Dwa węzły grafu GW, reprezentujące reguły R1 i R2 są połączone krawędzią skierowaną od węzła R1 do R2, jeżeli kod akcji reguły R1 zawiera instrukcje manipulacji danymi na relacji, której modyfikacja jest zdarzeniem uaktywniającym regułę R2.

Brak cyklu w grafie wyzwalania oznacza, że analizowany zbiór reguł posiada własność stopu. Występowanie cyklu w grafie wyzwalania oznacza, że zbiór reguł może nie posiadać własności stopu.



Aktywne bazy danych (24)

Jak wiadomo z teorii algorytmów formalna analiza własności stopu jest w ogólności problemem nieroziwiążalnym. W praktyce dla weryfikacji własności stopu stosuje się składniową analizę zbioru reguł. Analiza składniowa jest zachowawcza, co oznacza, że istnieją zbiory reguł posiadające własność stopu, które zostaną uznane w wyniku analizy składniowej jako niepoprawne.

Analiza składniowa opiera się na badaniu własności grafu wyzwalania. Graf ten jest konstruowany w następujący sposób. Węzłami grafu są aktywne reguły z analizowanego zbioru reguł. Krawędzie w grafie reprezentują składniowe zależności między regułami. Między węzłami, które reprezentują reguły R1 i R2 jest wstawiana krawędź skierowana od R1 do R2, jeżeli kod akcji reguły R1 zawiera operacje, które są zdarzeniem wywołującym regułę R2. Na poprzednich dwóch slajdach takie zależności zostały pokazane za pomocą strzałek. Brak cyklu w grafie wyzwalania oznacza, że badany zbiór reguł posiada własność stopu. Występowanie cyklu w grafie oznacza, że dany zbiór reguł może nie posiadać własności stopu.

Na rysunku pokazano grafy wyzwalania dla zbioru reguł „zwiększ_range” i „zwiększ_premię” oraz dla reguły „kotwica_budżetowa”. Ze względu na zachowawczość analizy składniowej mimo, że obydwa grafy są cykliczne, jednak w praktyce tylko pierwsza para reguł nie posiada własności stopu.

Komercyjne aktywne systemy baz danych zazwyczaj dokonują składniowej analizy własności stopu tylko podczas komplikacji pojedynczych reguł. W przypadku zbioru reguł brak własności stopu jest diagnozowany przez system dopiero podczas aktywowania reguł. Dla bardziej złożonych przypadków błęd taki może być wykryty dopiero po długim okresie eksploatacji systemu.

```

CREATE TRIGGER zwiększ_premię_1
    AFTER UPDATE OF wysokość ON Sprzedaż
    FOR EACH ROW
    WHEN new.wysokość - old.wysokość > 100
    BEGIN
        UPDATE Pracownicy
        SET premia = premia + 1000
        WHERE nr = new.nr_prac;
    END

CREATE TRIGGER zwiększ_premię_2
    AFTER UPDATE OF wysokość ON Sprzedaż
    FOR EACH ROW
    WHEN new.wysokość - old.wysokość > 500
    BEGIN
        UPDATE Pracownicy
        SET premia = 1.5 * premia
        WHERE nr = new.nr_prac;
    END

```

Aktywne bazy danych (25)

Następną wymaganą właściwością zbioru aktywnych reguł jest determinizm stanu. Problem determinizmu stanu jest konsekwencją występowania w zbiorze, reguł uaktywnianych tym samym zdarzeniem i modyfikujących ten sam fragment bazy danych. Kolejność odpalania akcji dwóch reguł uaktywnianych tym samym zdarzeniem jest w komercyjnych bazach danych ustalana arbitralnie przez system zarządzania bazą danych. Jeżeli akcje takich reguł nie są komutatywne, to końcowy stan bazy danych jest zależny od arbitralnie ustalonej przez system kolejności ich uruchomienia.

Przykład taki pokazano na slajdzie. W bazie danych pracowników, którzy są akwizytorami zdefiniowano dwie aktywne reguły automatycznie ustalające wysokość premii na podstawie wyników sprzedaży danego pracownika. Pierwsza reguła pracownikowi, który zwiększył sprzedaż o 100 jednostek podnosi premię o 1000 zł. Druga reguła w wypadku zwiększenia sprzedaży przez pracownika o ponad 500 jednostek podnosi mu pensję o 50%. W sytuacji gdy wzrost sprzedaży przekroczy 500 jednostek odpalone zostaną akcje obydwu reguł. Ponieważ akcje te nie są komutatywne, końcowy stan bazy danych zależy od kolejności ich odpalenia. Wypadek ten ilustrują scenariusze z prawej strony slajdu. Dla początkowego stanu premii pracownika wynoszącego 5000 zł odpalenie akcji w kolejności najpierw reguła pierwsza, a potem reguła druga ustala ostatecznie premię w wysokości 9000 zł. Odwrotna kolejność odpalenia reguł daje w efekcie inną kwotę premii wynoszącą w tym wypadku 8500 zł.

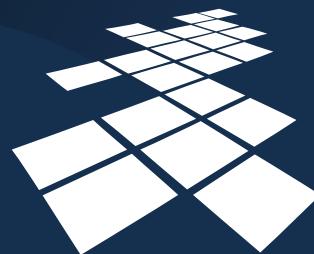
Silniejszym wymaganiem niż determinizm stanu bazy danych jest determinizm zachowania systemu informatycznego, obejmujący poza stanem bazy danych dodatkowo informacje wychodzące na zewnątrz systemu w postaci zawartości ekranów komputerów, wydruków czy innych wysyłanych na zewnątrz systemu komunikatów.

Podobnie jak w wypadku własności stopu również determinizm stanu bazy może być weryfikowany przez zachowawczą analizę składniową weryfikującą komutatywność akcji reguł.

Obiektowe bazy danych

Obiektowy model danych

Wykład prowadzi:
Tomasz Koszlajda



UCZELNIA
ONLINE

Obiektowe bazy danych – Obiektowy model danych

Tematyka obiektowych baz danych obejmuje trzy jednostki wykładowe. Pierwszy wykład dotyczy obiektowego modelu danych. Na drugim wykładzie zostaną przedstawione dwa alternatywne rozwiązania: obiektowe i obiektowo-relacyjne bazy danych. Trzeci wykład jest poświęcony implementacji obiektowych systemów baz danych.

Niniejszy wykład będzie wprowadzeniem do tematyki obiektowych baz danych. Obiektowe bazy danych są propozycją nowego, uniwersalnego i rozszerzalnego modelu danych dla systemów baz danych. W momencie ich pojawienia się miały zastąpić relacyjny model danych, jako lepiej przystosowane do nowych dziedzin zastosowań systemów baz danych.



Plan wykładu

- Przesłanki dla nowej generacji systemów baz danych
- Podstawowe elementy obiektowego modelu danych
- Konstruktory złożonych typów danych
- Abstrakcyjne typy danych
- Dziedziczenie
- Związki między danymi
- Hierarchie kolekcji obiektów
- Polimorfizm i późne wiązanie
- Tożsamość danych
- Trwałość danych

Obiektowe bazy danych – Obiektowy model danych (2)

Celem wykładu jest poznanie własności nowego modelu danych zastosowanego w obiektowych bazach danych. Najpierw zostaną przedstawione przesłanki pojawienia się nowej generacji baz danych. Poznamy nowe dziedziny zastosowań systemów baz danych wymagające silniejszego i bardziej elastycznego modelu danych niż model relacyjnych baz danych. W trakcie wykładu przedstawione zostaną podstawowe koncepcje obiektowego modelu danych: możliwość modelowania abstrakcyjnych typów danych, mechanizm dziedziczenia typów danych, konstruktory złożonych typów danych, jawne związki między danymi, hierarchiczne zależności między kolekcjami obiektów oraz mechanizmy polimorfizmu i późnego wiązania. Na koniec zostaną przedstawione rozwiązania służące do zapewnienia obiektom przechowywanym w bazie danych systemowej tożsamości i trwałości.



Nowe dziedziny zastosowań baz danych

- Systemy wspomagania projektowania
- Systemy informacji przestrzennej
- Multimedialne bazy danych

Charakterystyka nowych dziedzin zastosowań

- Złożone struktury danych
- Behawioralne własności danych
- Nowe modele przetwarzania

Nowe technologie budowy aplikacji

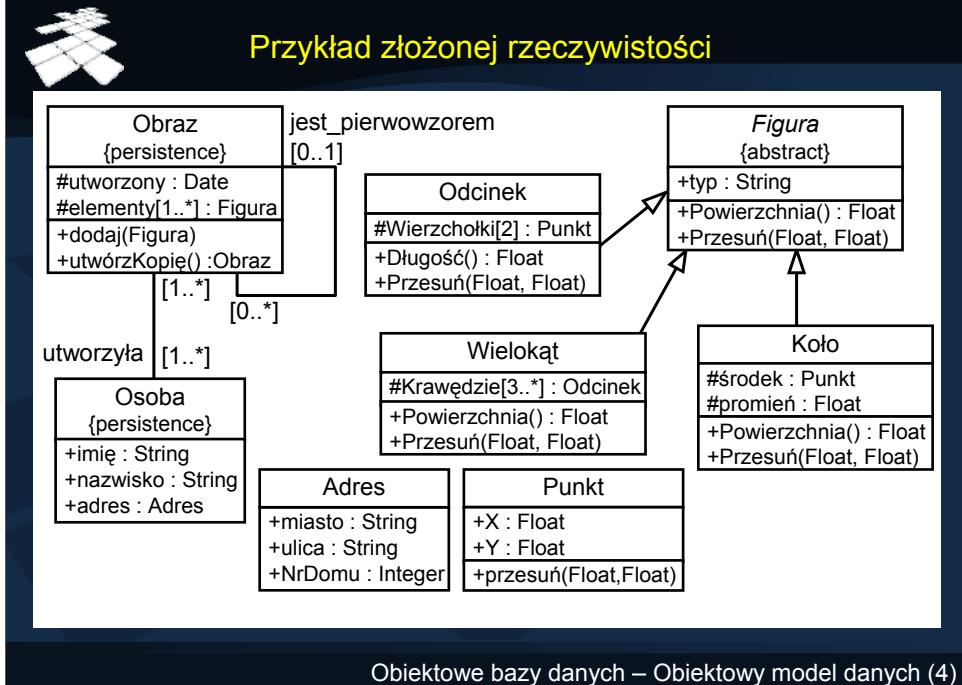
- Języki obiektowe

Obiektowe bazy danych – Obiektowy model danych (3)

W czasie powstawania relacyjnego modelu danych typowymi dziedzinami zastosowań systemów baz danych były bankowość, ubezpieczenia, finanse, gospodarka magazynowa, itp. Wspólna charakterystyka systemów tej klasy obejmuje proste struktury danych i prosty model ich przetwarzania. Typowymi wartościami atrybutów danych są teksy, liczby i daty.

Na początku lat osiemdziesiątych, rozpoczęły się próby stosowania systemów baz danych w nowych dziedzinach, takich jak, systemy wspomagania projektowania, systemy informacji przestrzennej lub systemy multimedialne. Charakterystyka tej klasy zastosowań jest diametralnie odmienna. Przetwarzane i składowane dane są złożone strukturalnie. Typowe są hierarchicznie złożone struktury danych oraz liczne i intensywnie przetwarzane powiązania między danymi. Powiązanie te mają złożoną semantykę: referencji, agregacji lub kompozycji. Również semantyka danych jest bardziej złożona. Informacje, które są przetwarzane w nowych dziedzinach zastosowań to długie dokumenty tekstowe, obrazy, animacje, dane wielowymiarowe, itp.

Lata osiemdziesiąte to również okres, kiedy rozpowszechniły się języki obiektowe. Chętnie i powszechnie stosowanymi narzędziami programowymi stosowanymi do budowy aplikacji stały się języki, takie jak: C++, Delphi i Java. Integracja aplikacji baz danych pisanych za pomocą języków obiektowych z relacyjnymi bazami danych była trudna i nienaturalna ponieważ system typów bazy danych i system typów aplikacji są całkowicie odmienne.



Obiektowe bazy danych – Obiektowy model danych (4)

Na rysunku przedstawiono model fragmentu świata rzeczywistego o charakterystyce reprezentatywnej dla wymienionych nowych dziedzin zastosowań. Przykład został zamodelowany za pomocą diagramu klas języka UML.

Pierwszą charakterystyczną cechą zamodelowanej rzeczywistości są złożone struktury danych. Przykładem jest klasa „*Obraz*”, która jest nieograniczoną kolekcją elementów składowych, którymi są „*Figury*”. Poszczególne typy figur również mają złożoną konstrukcję. Na przykład „*Wielokąty*” są kolekcjami krawędzi, które z kolei są parami „*Wierzchołków*”. Na koniec typami danych „*Wierzchołków*” jest klasa „*Punkt*”.

Kolejną cechą przykładu jest zdefiniowana przez użytkownika semantyka operacji dostępnych dla zdefiniowanych klas, wykraczająca poza operacje predefiniowanych typów danych. Przykładem są operacje dostępne dla klasy „*Figura*”: wyznaczanie powierzchni i przesuwanie figur na płaszczyźnie.

Demonstrowany przykład obejmuje również hierarchię klas, której korzeniem jest abstrakcyjna klasa „*Figura*”, a liściemi klasy reprezentujące różne typy figur: „*Odcinki*”, „*Koła*” i „*Wielokąty*”. Dzięki temu atrybut „*Elementy*” klasy „*Figura*” na charakter polimorficzny. Wartościami kolekcji „*Elementy*” są obiekty o różnej strukturze i semantyczne operacji.

Ostatnią specyfczną właściwością ilustrowaną przez przykład są jawne związki między klasami. Są to: binarny związek między klasą „*Obraz*” i klasą „*Osoba*” reprezentujący autorstwo poszczególnych „*Obrazów*” oraz unarny związek między „*Obrazami*” reprezentujący zapożyczenia między „*Obrazami*”.



Ograniczenia relacyjnego modelu danych

- Płaskie jednowymiarowe struktury danych
- Potrzeba sztucznych kluczy podstawowych
- Semantyka niestandardowych operacji musi być implementowana poza bazą danych
- Brak pojęcia związków
- Brak hierarchii typów

```
Figury(id_f PK, typ, powierzchnia)
Odcinki(id_odc PK FK(Figury), typ, x1, y1, x2, y2)
Wielokąty(id_w PK FK(Figury), typ)
Krawędzie(id_k PK, x1, y1, x2, y2, id_w FK(Wielokąty))
Koła(id_k PK FK(Figury), typ, x, y, promień)
Obrazy(id_ob PK, utworzony)
Osoby(id_os PK, imię, nazwisko, miasto, ulica, numer_domu)
Autorstwo(id_ob FK(Obrazy), id_os FK(Osoby))
Modyfikacje(wzorzec FK(Obrazy), modyfikacja FK(Obrazy))
```

Obiektowe bazy danych – Obiektowy model danych (5)

Reprezentacja przedstawionego fragmentu rzeczywistości za pomocą relacyjnego modelu danych nie jest ani prosta, ani naturalna.

W przykładzie występują złożone struktury danych, a jedyną strukturą dostępną w relacyjnym modelu danych jest krotka, czyli płaska lista prostych wartości. Relacyjny model danych nie umożliwia zagnieżdżania konstruktorów typów danych. W związku z tym, relacyjna reprezentacja przykładu będzie rozproszonym zbiorem niepowiązanych i jednowymiarowych struktur danych. Przedstawiony na slajdzie schemat relacyjnej bazy danych nie potrafi odzwierciedlić hierarchicznych zależności między danymi. Metodyki poprawnego projektowania schematów relacyjnych baz danych wykluczają również składowanie w bazie danych złożonych wartości atrybutów w sposób transparentny dla modelu danych. Ograniczenie to jest zdefiniowane jako tak zwana pierwsza postać normalna.

Potrzeba jednoznacznej identyfikacji pojedynczych danych przechowywanych w bazie danych wymaga rozszerzania schematów relacji o sztuczne klucze podstawowe. Dotyczy to przypadków, gdy zdefiniowane atrybuty nie gwarantują unikalności wartości poszczególnych danych. Ponieważ klasy Figura, Koło, Wielokąt, Odcinek nie posiadają atrybutów jednoznacznie identyfikujących ich wystąpienia transformacja tych klas do schematu relacyjnej bazy danych wymaga dodania sztucznych kluczy podstawowych do reprezentujących je relacji.

Relacyjny model danych pozwala na korzystanie jedynie z ograniczonego zbioru prostych predefiniowanych typów danych. Niestandardowa semantyka przetwarzania danych w rozwiązaniach relacyjnych musi być w całości zaimplementowana poza systemem bazy danych, w aplikacjach bazy danych.

Relacyjny model danych nie obejmuje pojęcia związków między danymi. Związki między danymi nie mogą, więc być składowane w bazie danych. Zależności klucz obcy – klucz podstawowy służą jedynie do weryfikacji poprawności danych i nie mogą być wykorzystane do nawigacji między danymi. Związki między danymi są kreowane dynamicznie przez operacje połączenia. Systemy relacyjne realizując operacje połączenia dopiero „w locie” ustalają powiązania między danymi.

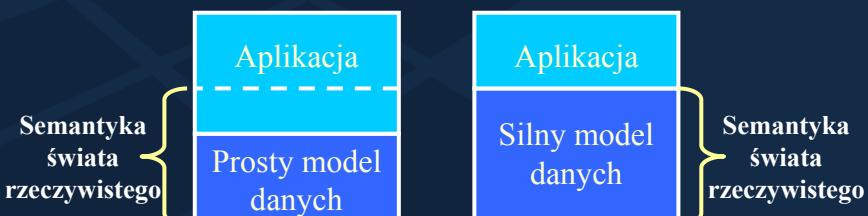
Kolejnym ograniczeniem modelu relacyjnego jest brak możliwości modelowania hierarchicznych zależności między kolekcjami danych, między którymi zachodzi relacja podzbioru. Integracja danych reprezentujących różne podzbiory Figur będzie wymagać wykonywania operacji połączenia lub sumy.

Podsumowując relacyjna implementacja bardziej złożonej rzeczywistości wymaga przeniesienia części jej semantyki do aplikacji bazy danych. Taka semantyka jest trudniej utrzymywana, bo informacji o niej nie ma w bazie danych, lecz trzeba ją utrzymywać w źródłowym kodzie aplikacji. Ponadto, odpowiedzialność za wydajne przetwarzanie semantycznych danych, musi być w tym wypadku przeniesiona z systemu zarządzania bazą danych na programistów tworzących aplikacje. W związku z tym będą to rozwiązania mniej skalowalne.



Przesłanki nowej generacji baz danych

- Potrzeba bogatszego modelu danych
- Rozszerzalny model danych umożliwiający ścisłe dopasowanie dla dowolnych dziedzin zastosowań
- Ścisłejsza integracja z obiektowymi aplikacjami bazy danych



Obiektowe bazy danych – Obiektowy model danych (7)

Reprezentacja świata rzeczywistego o złożonej semantyce wymaga odpowiednio silnego modelu danych. Takiego modelu danych, który pozwoli bezpośrednio wyrazić całą semantykę wybranego fragmentu rzeczywistości. Dzięki temu, aplikacje będą odpowiedzialne „tylko” za realizację logiki procesów biznesowych oraz interfejs z użytkownikami.

Idealny model danych powinien również być uniwersalny, po to by można go było dopasować do charakterystyki dowolnej dziedziny zastosowań. Pomyśl ten prowadzi do idei „*rozszerzalnego modelu danych*”. To jest modelu, w którym użytkownicy mogą rozszerzać predefiniowany system typów danych, o dowolnie złożone własne typy danych. Pozwoli to w każdej sytuacji zastosować dokładnie taką siłę modelu, jaka będzie potrzebna dla konkretnego przypadku.

Dodatkowo poszukiwany model danych systemu bazy danych powinien być łatwo integrowalny z nowoczesnymi aplikacjami baz danych budowanymi za pomocą języków obiektowych. Idealna sytuacja polegałaby na przepływie danych między bazą danych a aplikacją bez konwersji niezbędnej dla niedopasowanych systemów typów danych.



Podstawowe elementy obiektowego modelu danych

- Obiekt: stan i funkcjonalność
- Cechy obiektów: atrybuty i związki
- Funkcjonalność obiektu: metody
- Tożsamość obiektu
- Hermetyczność obiektów
- Klasa: typ danych i moduł programowy
- Dziedziczenie: współdzielenie implementacji i relacja podtypu
- Przeciążanie i dynamiczne wiązanie funkcjonalności obiektów

Obiektowe bazy danych – Obiektowy model danych (8)

Modelem danych, który spełnia większość wymienionych wymagań jest model wypracowany dla obiektowych języków programowania. Model ten został zaadoptowany do potrzeb systemów baz danych.

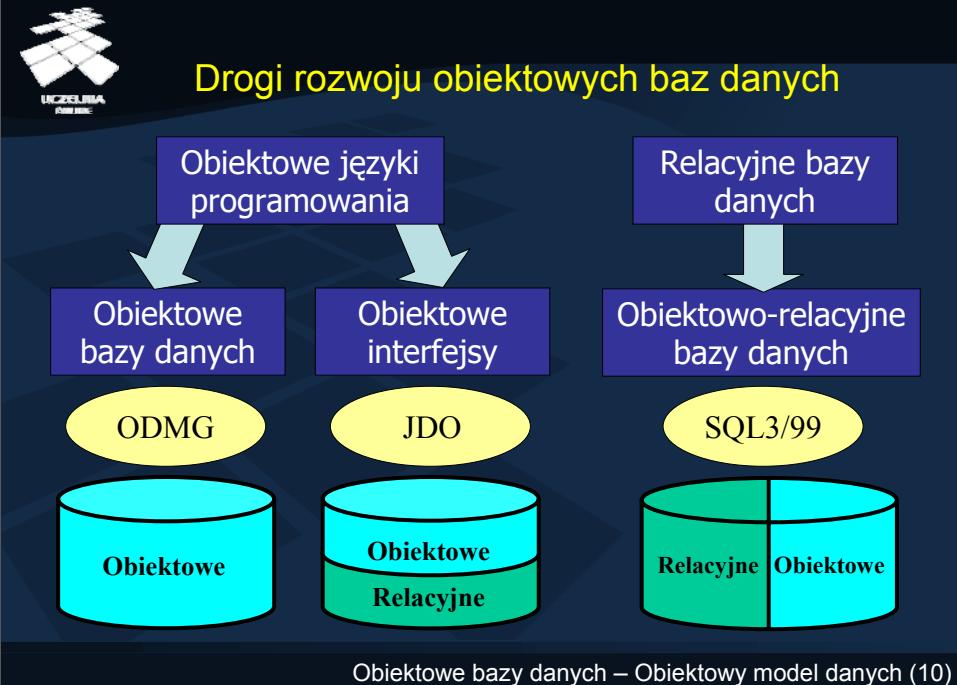
Podstawowym pojęciem tego modelu jest pojęcie obiektu, który umożliwia reprezentowanie cech strukturalnych i behawioralnych obiektów świata rzeczywistego. Struktura obiektu jest opisana przez zbiór atrybutów i związków nazywanych łącznie cechami obiektu. Wartościami atrybutów mogą być wystąpienia prostych typów danych lub obiekty składowe. Wartościami związków są referencje na inne, zewnętrzne obiekty. Zbiór wartości wszystkich atrybutów i związków tworzy stan obiektu. Z kolei własności behawioralne obiektu są reprezentowane przez zbiór dedykowanych procedur zwanych metodami.

Obiekty są jednoznacznie identyfikowane za pomocą systemowego atrybutu nazywanego identyfikatorem obiektu lub w skrócie - OID. Wartości tego atrybutu są unikalne i niezmienne.

Wewnętrzna struktura obiektu oraz implementacja metod są ukryte przed użytkownikami obiektu. Mówiąc, że są to prywatne własności obiektów, niedostępne z zewnątrz. Dostęp do obiektów umożliwia ich publiczny interfejs, czyli wywołania metod obiektu. Metody obiektu są wywoływane przez wysyłanie do obiektu odpowiednich komunikatów.

Obiekty o tej samej strukturze i metodach należą do tej samej klasy obiektów. Klasy posiadają dualną naturę. Z jednej strony są odpowiednikami typów danych. Są definicjami własności obiektów. Z drugiej strony klasy są modułami programowymi, które zawierają implementację funkcjonalności typów danych.

Klasy mogą być definiowane jako specjalizacje innych klas. Klasa wyspecjalizowana jest nazywana podklassą i dziedziczy ona cechy i metody swojej nadklasy. Dziedziczenie umożliwia współdzielenie implementacji klas. Dodatkowo klasa wyspecjalizowana jest podtypem swojej nadklasy. Umożliwia to polimorficzne przetwarzanie kolekcji klas i dynamiczne wiązanie przesyłanych komunikatów z metodami klasy właściwej dla danego obiektu.



Istnieją dwie podstawowe strategie budowy systemów baz danych o obiektowym modelu danych: rewolucyjna, w której nowe systemy baz danych są budowane w całości od podstaw oraz ewolucyjna, w której obiektowe bazy danych powstają przez przyrostowe modyfikacje istniejących systemów baz danych.

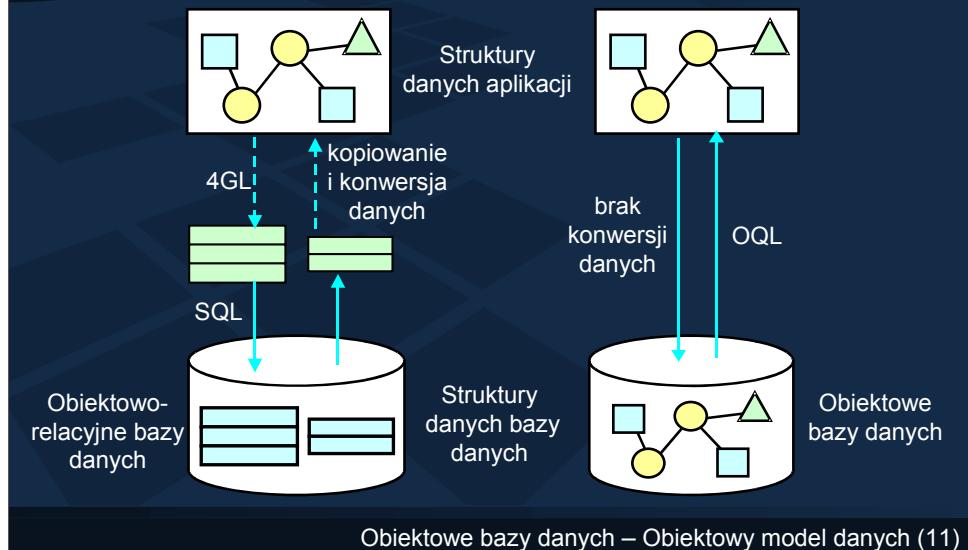
Historycznie pierwszym rozwiązaniem była budowa systemów baz danych od nowa na bazie obiektowych języków programowania. Rozwiązanie to polega na rozszerzeniu funkcjonalności obiektów tworzonych i przetwarzanych przez języki obiektowe o własności typowe dla danych przechowywanych w systemach baz danych, to jest: trwałości, współdzielenia, synchronizacji dostępu, odtwarzania spójnego stanu po awarii, wydajnego przetwarzania dużych zbiorów obiektów, itp. Wynikiem tej strategii są "czysto" obiektowe bazy danych posiadające jednorodny obiektowy model danych. Standard dla modelu danych tej klasy systemów został opracowany przez organizację „Object Database Management Group” i od nazwy tej grupy jest nazwany „ODMG 3.0”.

Kontynuatorem tej strategii jest rozwiązanie, w którym funkcjonalność obiektowego modelu danych jest implementowana przez dodatkową warstwę programową budowaną na systemie bazy danych o dowolnym modelu danych. Standardem związanym z tą strategią jest „Java Data Objects” - JDO.

Całkowicie odmiennym rozwiązaniem jest ewolucyjna modyfikacja relacyjnych systemów baz danych polegająca na rozszerzeniu relacyjnego modelu danych o własności modelu obiektowego: hermetycznych obiektów zintegrowanych z metodami, dziedziczenia, polimorfizmu, dynamicznego wiązania, itp. Wynikiem jest obiektowo-relacyjny model danych, który jest konglomeratem cech relacyjnych i obiektowych. Właściwości modelu obiektowo-relacyjnego są opisane w standardzie SQL3 (SQL99).



Architektury obiektowych i obiektowo-relacyjnych systemów baz danych



Obiektowo-relacyjne systemy bazy danych dziedziczą po systemach relacyjnych cechę niedopasowania systemów typów danych bazy danych i aplikacji bazy danych, co zostało zilustrowane na rysunku po lewej stronie slajdu. Dane przechowywane w bazie danych mają inną strukturę i semantykę niż dane przetwarzane przez aplikacje. Ta sama informacja w ciągu swojego cyklu życia obejmującego przenoszenie jej między pamięcią dyskową, a pamięcią operacyjną zmienia swoją fizyczną reprezentację. W bazie danych jest krotką o określonej reprezentacji fizycznej poszczególnych typów danych, tekstowych, numerycznych i dat. Z kolei podczas wizualizowania na ekranie komputera przez aplikacje ta sama informacja jest kolekcją danych o odmiennych reprezentacjach fizycznych.

Na przykład, nazwisko studenta w bazie danych jest przechowywane w polu o typie VARCHAR(20), którego fizyczna reprezentacja jest dwuelementową tablicą, której pierwszy element pamięta faktyczną długość tekstu, a drugi jest łańcuchem kodów ASCII. Tymczasem aplikacja napisana w języku C++ przechowuje nazwisko studenta w zmiennej wskaźnikowej na tekst, który jest ciągiem kodów ASCII zakończonym wyróżnioną wartością zera binarnego. Przenoszenie informacji między bazą danych, a aplikacją wymaga więc bezustannej transformacji fizycznej reprezentacji tej informacji. Niedopasowanie systemów typów danych ogranicza wydajność działania oraz komplikuje tworzenie aplikacji.

Czysto obiektowe bazy danych nie posiadają tej wady. Obiekty w bazie danych i obiekty przetwarzane przez aplikacje mają dokładnie taką samą strukturę i semantykę. Przenoszenie danych między pamięcią operacyjną i dyskową nie wymaga żadnego przekształcania danych. Ilustruje to rysunek po prawej stronie slajdu.



Baza danych jako zbiór kolekcji obiektów

- Stan obiektowej bazy danych jest zbiorem kolekcji trwałych i rozróżnialnych obiektów.
- Schemat obiektowej bazy danych jest zbiorem klas, które definiują strukturę i funkcjonalność obiektów.
- W obiektowych bazach danych rozróżnia się pojęcie klasy jako definicji własności obiektów od pojęcia rozszerzenia klasy będącego zbiorem trwałych obiektów.

```
class Figura { // nazwa klasy jako typu
    extent Figury} // nazwa rozszerzenia klasy jako
                    // zbioru wystąpień
```

Obiektowe bazy danych – Obiektowy model danych (12)

Obiekt jest podstawową jednostką danych składowanych w obiektowej bazie danych. W przeciwieństwie do obiektów przetwarzanych w zwykłych programach obiektowych, obiekty utworzone przez aplikacje obiektowych baz danych są autonomiczne w stosunku do tych aplikacji i trwałe. Autonomia obiektów polega na możliwości istnienia obiektu poza programem, który go utworzył. Natomiast trwałość oznacza, że czas życia obiektu jest dłuższy niż czas życia zmiennej, której był przypisany w momencie utworzenia. Aplikacja, która utworzyła obiekt może zostać wyłączona, a utworzone przez nią trwałe obiekty będą pamiętane w bazie danych.

Struktura i funkcjonalność obiektów bazy danych jest zdefiniowana za pomocą klas. Zbiór definicji klas tworzy schemat bazy danych. W przeciwieństwie do systemów relacyjnych, gdzie relacja pełniła jednocześnie rolę definicji struktury danych oraz zbioru wszystkich danych o tej strukturze, w obiektowych bazach danych te role są rozdzielone. Zbiór trwałych wystąpień danej klasy tworzy tak zwane rozszerzenie klasy. Ilustruje to przykład na dole slajdu. „*Figura*” jest nazwą klasy definiującej własności obiektów, a „*Figury*” są nazwą rozszerzenia, które jest zbiorem wszystkich trwałych wystąpień klasy „*Figura*”.



Abstrakcyjne typy danych

Możliwość definiowania nowych typów danych o dowolnej złożoności i funkcjonalności. Typy danych użytkownika mogą być podstawą definicji pojedynczych atrybutów klas jak również całych klas.

```
class Punkt {
    Float X, Y;
    void przesuń (in Float x,
                  in Float y);};

class Odcinek {
    Punkt W1, W2;
    void przesuń (in Float a,
                  in Float b) {
        W1.przesuń(a, b);
        W2.przesuń(a, b); }};


```

Punkt
+X : Float
+Y : Float
+przesuń(Float , Float)

Odcinek
{persistence}
#Wierzchołki[2]:Punkt
+przesuń(Float , Float)

Obiektowe bazy danych – Obiektowy model danych (13)

Relacyjne bazy danych oferują projektantowi schematu bazy danych ograniczony zbiór prostych, predefiniowanych typów danych dla atrybutów relacji. Przykładem mogą być systemowe typy tekstowe: varchar i char, numeryczne: int i float oraz temporalne: date. Typy danych niedostępne w relacyjnym modelu danych muszą być implementowane poza systemem bazy danych. Kod obsługujący ich semantykę musi być implementowany i powielany we wszystkich aplikacjach bazy danych.

Obiektowe bazy danych umożliwiają projektantom definiowanie nowych typów danych. Definiowanie własnego typu danych obejmuje definicję struktury i funkcjonalności typu. Definicje typów są składowane w systemie bazy danych. Sposób korzystania z typów danych zdefiniowanych przez użytkownika nie różni się niczym od korzystania z typów systemowych. Dzięki temu obiektowy model danych jest rozszerzalny. Można go elastycznie dopasowywać do dowolnej dziedziny zastosowania wymagającej unikalnych, specyficznych typów danych.

Rolę typów danych definiowanych przez użytkownika pełnią w obiektowych bazach danych klasy i interfejsy. Definicja interfejsu jest opisem funkcjonalności nowego typu danych, czyli opisem operacji dostępnych dla danego typu. Definicja klasy obejmuje dodatkowo implementację typu danych: to jest wewnętrzną strukturę danych służącą do przechowywania stanu wystąpień typu danych oraz kod operacji zdefiniowanych dla danego typu danych.

W zaprezentowanym przykładzie zdefiniowano dwa typy danych użytkownika: klasy Punkt i Odcinek. Część strukturalna klasy Punkt to dwa atrybuty X i Y reprezentujące lokalizację punktu na płaszczyźnie. Część funkcjonalna tej klasy jest ograniczona do metody Przesuń służącej do zmiany lokalizacji punktów. Klasa Punkt została wykorzystana jako typ danych dla atrybutów W1 i W2 klasy Odcinek. Wystąpienia klasy Punkt będą wartościami tych atrybutów. Klasa Odcinek służy do definiowania typu obiektów składowanych w bazie danych. Metoda przesuń jest zaimplementowana jako sekwencja wywołań operacji przesunięcia dla dwóch końców W1 i W2 danego odcinka.

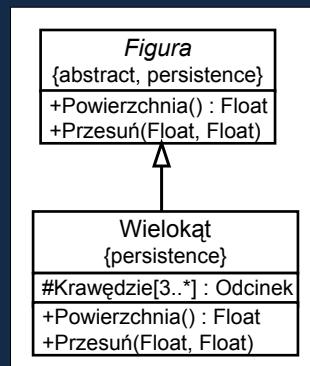


Dziedziczenie

Klasy mogą być specjalizowane przez mechanizm dziedziczenia. Klasa pochodna dziedziczy funkcjonalność i implementację klasy bazowej. W klasie pochodnej można dodać nową funkcjonalność lub redefiniować funkcjonalność odziedziczoną.

```
class Wielokąt extends Figura
// dziedziczenie
{ ... };
class Wielokąt : Figura
// relacja podtypu
{ ... };
```

Relacja podtypu \subseteq Dziedziczenie



Obiektowe bazy danych – Obiektowy model danych (15)

W obiektowych bazach danych można definiować nowe klasy i interfejsy wywodząc je ze zdefiniowanych wcześniej klas lub interfejsów. Model ODMG rozróżnia dwa typy powiązań między klasami lub interfejsami: związek relacji podtypu i związek dziedziczenia. W językach obiektowych te dwa związki są traktowane jako tożsame.

Związek relacji podtypu może dotyczyć klas i interfejsów. Oznacza on dziedziczenie przez typ pochodny funkcjonalności typu bazowego. Klasy i interfejsy połączone związkiem podtypu tworzą sieć powiązań o topologii grafu acyklicznego skierowanego. Oznacza to, że pojedyncza klasa lub interfejs może dziedziczyć funkcjonalność po wielu klasach lub interfejsach.

Związek dziedziczenia łączy jedynie klasy. Oznacza on dziedziczenie zarówno funkcjonalności jak i implementacji. Związek dziedziczenia obejmuje semantykę relacji podtypu. Klasy połączone związkiem dziedziczenia tworzą sieć powiązań o topologii hierarchii. Oznacza to, że pojedyncza podklasa może dziedziczyć zarówno funkcjonalność jak i implementację po dokładnie jednej nadklasie.

Dziedziczona funkcjonalność może być rozszerzona w stosunku do typu bazowego. Dziedziczona implementacja może być rozszerzana lub przesłaniana. Przez przesłaniane rozumie się definicje nowego kodu dla odziedziczonych metod, który zastąpi stary kod.

Przedstawiony na slajdzie przykład pokazuje dwa alternatywne rozwiązania. W pierwszym klasa Wielokąt dziedziczy po klasie Figura funkcjonalność i implementację, w drugim klasa Wielokąt przez relację podtypu dziedziczy funkcjonalność bez implementacji.

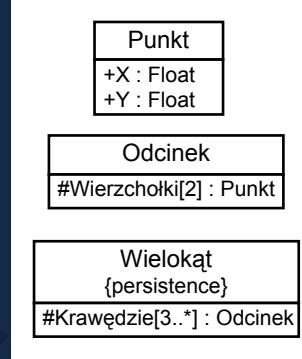


Złożone struktury danych

Możliwość naturalnego modelowania atrybutów **złożonych** i **wielowartościowych**. Przykład zastosowania konstruktorów typów złożonych w języku ODL.

```
class Wielokąt {
    struct Punkt {
        Float X, Y; }
    struct Odcinek {
        Punkt Wierzchołek_1;
        Punkt Wierzchołek_2; }

    attribute set<Odcinek> krawędzie; };
```



Obiektowe bazy danych – Obiektowy model danych (16)

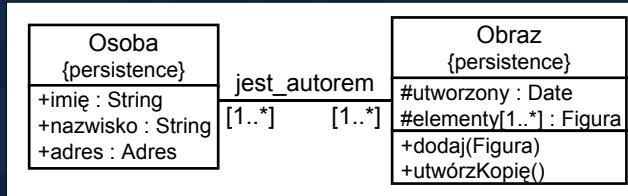
Kolejnym ograniczeniem przełamany przez obiektowy model danych jest możliwość przechowywania w bazie danych, obiektów o złożonej strukturze. Elementem standardów ODMG i SQL3 jest bogaty zbiór konstruktorów złożonych typów danych. Kompletna lista obejmuje konstruktory: krotki, zbioru, wielozbioru, listy, tablicy i słownika. Konstruktory złożonych typów danych mogą być wielopoziomowo zagnieżdżane.

Przykład na slajdzie pokazuje transformację do schematu obiektowej bazy danych klasy Wielokąt o atrybucie wielowartościowym i złożonym. Atrybutem klasy Wielokąt jest zbiór Odcinków, z których każdy jest strukturą pary Punktów, które z kolei są strukturą pary współrzędnych. W przykładzie zastosowano dwa typy konstruktorów: konstruktora krotki – „struct” i konstruktora zbioru – „set”.



Związki między danymi

Możliwość definiowania i składowania w bazie danych związków między danymi.



```

class Osoba {
    relationship set<Obraz> jest_autorem
    inverse Obraz::jest_utworzony_przez;
    ...
}
  
```

Obiektowe bazy danych – Obiektowy model danych (17)

Obiektowy model danych pozwala na jawną reprezentację związków między danymi. W przeciwieństwie do relacyjnego modelu danych, gdzie ziązki są ustalane w sposób dynamiczny w momencie wykonywania zapytań, a dokładniej operacji połączenia (ang. join), w obiektowym modelu danych związki są pamiętane w bazie danych. Podczas przetwarzania powiązanych danych dostępna jest operacja nawigacji wzdułż tych powiązań. W modelu ODMG przyjęto dodatkowo, że wszystkie ziązki są dwukierunkowe, co oznacza, że dla powiązanych obiektów A i B możliwa jest nawigacja od obiektu A do obiektu, jak i na odwrót, od obiektu B do obiektu A. Model obiektowy nie stawia ograniczeń na krotność związku. Dozwolone są powiązania jednokrotne i wielokrotne.

Na slajdzie pokazano przykład związku łączącego klasę *Obraz* z klasą *Osoba*, które je utworzyły. Obiekty obydwu tych klas przechowują informacje o obiektach, z którymi są powiązane. Po stronie osób związek ten nosi nazwę: „*jest_autorem*”, a po stronie obrazów: „*jest_utworzony_przez*”. Po obydwu stronach jest to powiązanie wielokrotne. Pojedyncza osoba może być autorem wielu obrazów, a pojedynczy obraz może mieć wielu autorów. Wartością związku „*jest_autorem*” jest zbiór identyfikatorów obiektów, które reprezentują obrazy utworzone przez daną osobę. Analogicznie wartością związku „*jest_utworzony_przez*” jest zbiór identyfikatorów obiektów, które reprezentują osoby będące autorami danego obrazu.

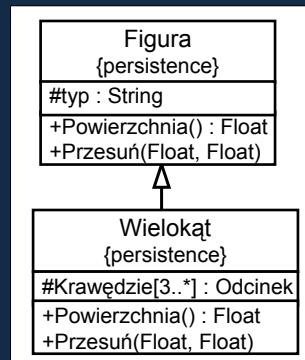


Hierarchia kolekcji obiektów

Związek dziedziczenia między klasami, których rozszerzenia są składowane w bazie danych implementuje związek zawierania się podzbiorów obiektów. Rozszerzenie klasy pochodnej jest podzieleniem rozszerzenia klasy bazowej.

```
class Figura {
    (extent Figury)
    ...
}
class Wielokąt extends Figura{
    (extent Wielokąty)
    ...
}
```

Wielokąty ⊆ Figury



Obiektowe bazy danych – Obiektowy model danych (18)

Związek dziedziczenia lub czysta relacja podtypu łączące klasy definiuje relację podzbiorów między ich rozszerzeniami. Rozszerzenie klasy pochodnej jest podzieleniem rozszerzenia klasy bazowej. Rozszerzenie klasy bazowej obejmuje rekurencyjnie obiekty należące do wszystkich rozszerzeń bezpośrednich i pośrednich klas pochodnych. Relacja ta pozwala na przetwarzanie heterogenicznych zbiorów obiektów. Operacje wykonywane na rozszerzeniu klasy bazowej są automatycznie propagowane na rozszerzenia wszystkich klas pochodnych.

Na slajdzie jest to zilustrowane przykładem relacji podzbioru łączącej rozszerzenie „*Figury*” klasy „*Figura*” z rozszerzeniem „*Wielokąty*” klasy „*Wielokąt*”. Operacje wykonywane na zbiorze wystąpień klasy „*Figura*” będą wykonywane również na wystąpieniach klasy „*Wielokąt*”.



Polimorfizm i późne wiązanie

Relacja podtypu łącząca klasy i interfejsy umożliwia podstawienia polimorficzne polegające na podstawieniu pod zmienną typu klasy bazowej obiektu, który jest wystąpieniem klasy pochodnej.

Podstawienia polimorficzne umożliwiają dynamiczne wiązanie nazw metod.

```

zmienna polimorficzna           podstawienie polimorficzne
Figura f = new Koło(10, 6, 5);   ← Koło::powierzchnia()
Float p = f.powierzchnia();       ← wiązanie dynamiczne
f = new Wielokąt(p1, p2, p3);
p = f.powierzchnia();             ← Wielokąt::powierzchnia()

```

Obiektowe bazy danych – Obiektowy model danych (19)

O obiektach składowanych w rozszerzeniach klas, które mają klasy pochodne mówi się, że są polimorficzne, bo mają różne cechy i metody. Na przykład wystąpienia klasy Figura mają określony atrybut „*Typ*” służący do przechowywania informacji o typie figury oraz metodę „*Powierzchnia*” służącą do wyznaczania powierzchni figur. Z kolei wystąpienia klasy pochodnej „*Wielokąt*” oprócz odziedziczonego atrybutu „*Typ*” mają dodatkowo wielowartościowy atrybut „*Krawędzie*”. Odziedziczona po klasie „*Figura*” metoda „*Powierzchnia*” jest w klasie „*Wielokąt*” redefiniowana ponieważ sposób wyznaczania powierzchni wielokątów jest inny niż uniwersalnych figur. Rozszerzenie klasy „*Figura*” obejmuje zarówno obiekty klasy „*Figura*”, jak również różniące się od nich obiekty, które są wystąpieniami klasy „*Wielokąt*”.

Na poziomie składni języków obiektowych polimorfizm obiektów wyraża się w występowaniu zmiennych polimorficznych i podstawień polimorficznych. Przez zmienną polimorficzną rozumie się taką zmienną, pod którą są podstawiane obiekty, które są wystąpieniami różnych klas, ale które występują w relacji podtypu z typem zmiennej polimorficznej. W podanym przykładzie, zmienną polimorficzną jest zmienna „*f*”. Mimo, że typem zmiennej „*f*” jest klasa „*Figura*”, to przypisywane są jej obiekty innych klas. Najpierw pod zmienną „*f*” jest podstawiony obiekt typu „*Koło*”, a później obiekt typu „*Wielokąt*”. Następnie przez zmienną „*f*” do przypisanych jej obiektów są przesyłane komunikaty o nazwie „*Powierzchnia*”. Właściwy kod metody wyznaczania powierzchni jest ustalany dynamicznie na podstawie typu obiektu przypisanego w danym momencie zmiennej „*f*”. W pokazanym przykładzie najpierw jest to kod metody „*Powierzchnia*” zdefiniowanej w klasie „*Koło*”, następnie kod metody „*Powierzchnia*” klasy „*Wielokąt*”.

Szczególnym przypadkiem zmiennych polimorficznych są nazwy rozszerzeń klas, które mają klasy pochodne. Nazwa rozszerzenia klasy bazowej pozwala przetwarzać obiekty należące do rozszerzeń wszystkich klas pochodnych.



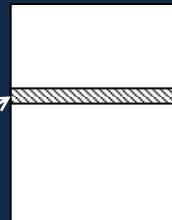
Tożsamość danych

Obiektowe języki programowania: identyfikacja danych przez symboliczną nazwę i fizyczny adres w pamięci operacyjnej.

```
class Pracownik { . . . };  
Pracownik Nowak("Jan", "Nowak");  
  
Nowak.nazwisko = "Kowalski";
```

PAO

0c78:89ad



Obiektowe bazy danych – Obiektowy model danych (20)

Dostęp do danych w bazie danych z poziomu aplikacji bazy danych wymaga mechanizmu identyfikacji danych. Programista musi dysponować jakimiś środkami wyrazu, które pozwolą mu wskazać te dane, które chce przetwarzać. Dostęp do danych lokalnych programów jest realizowany poprzez zmienne przypisane przetwarzanym danym. Nazwy zmiennych są środkiem jednoznacznej identyfikacji danych. W danej części programu nazwy zmiennych muszą być unikalne. W przedstawionym przykładzie zmienna o nazwie Nowak jednoznacznie określa podstawiony pod nią obiekt, który jest wystąpieniem klasy Osoba. Przesyłanie komunikatów do tego właśnie obiektu wymaga użycia nazwy zmiennej.

W wykonywalnej postaci programów nazwy zmiennych są transformowane do adresów w pamięci operacyjnej, pod którymi obiekty zostaną ulokowane. W naszym przykładzie symboliczna nazwa Nowak zostaje zamieniona na adres pamięci operacyjnej 0c78:89ad.



Tożsamość danych

Model relacyjny: identyfikacja danych przez ich wartości

```
SELECT * FROM płatnicy
WHERE Nazwisko = 'NOWAK' ;
```

JAN	NOWAK
TADEUSZ	KOWALSKI
MACIEJ	NOWAK
JAN	RZEPKA
KUBA	TARZAN
JÓZEF	MALINIAK
JAN	NOWAK

Jednoznaczna identyfikacja danych wymaga zdefiniowania dla każdej relacji – klucza podstawowego

Obiektowe bazy danych – Obiektowy model danych (21)

Cechą relacyjnego modelu danych jest identyfikacja danych poprzez wartość. W przeciwieństwie do danych przetwarzanych w proceduralnych językach programowania, pojedyncze krotki w bazie danych nie mają przypisanych żadnych symbolicznych nazw. Przez nazwę identyfikowalne są jedynie całe relacje. Również fizyczna lokalizacja krotek w pamięci dyskowej nie jest podstawą do ustalenia ich tożsamości. Fizyczna reorganizacja danych może się bowiem wiązać z realokacją krotek w pamięci dyskowej.

Jedyną cechą danych do której można się odwołać wyszukując ich w bazie danych są ich wartości. Zapytania w relacyjnych bazach danych wyrażane w języku SQL opierają się na wyrażeniach logicznych odwołujących się do wartości wybranych atrybutów krotek. Pokazuje to przykład na slajdzie, w którym w zbiorze Płatników są wyszukiwane osoby, dla których wartość atrybutu Nazwisko jest równa stałej tekstowej „Nowak”. W sytuacji gdy atrybuty, do których odwołuje się zapytanie nie mają cechy unikalności identyfikacja poprzez wartość nie jest jednoznaczna. W podanym przykładzie, aż trzy osoby noszą nazwisko Nowak. Dla jednoznacznej identyfikacji danych schematy relacji muszą posiadać atrybuty o własności klucza podstawowego. Dopiero odwołanie się w zapytaniu do wartości klucza podstawowego gwarantuje jednoznaczność identyfikacji i pozwala na dostęp do pojedynczych krotek.



Tożsamość danych w obiektowej bazie danych

- Identyfikacja na podstawie wartości identyfikatora obiektu nazywanego OID
- Identyfikacja przez OID jest niezależna od wartości atrybutów obiektu

```
Osoba *kowalski, *kowalska;  
kowalski = new Osoba("Jan", "Kowalski");  
kowalska = new Osoba("Janina", "Kowalska");  
Kowalska->małżonek = kowalski; //podstawienie oid  
Kowalski->nazwisko = "Nowak";  
Kowalski->pesel = 68040102766;  
Kowalski->płeć = "K";  
Kowalska->małżonek->dochody->pokaż();
```

Obiektowe bazy danych – Obiektowy model danych (22)

W obiektowych bazach danych wprowadzono nowy mechanizm identyfikacji danych. Obiekty są identyfikowane za pomocą systemowego identyfikatora OID. Jest to dodatkowy atrybut każdego obiektu, którego wartości są generowane i utrzymywane w sposób transparentny dla użytkowników przez system zarządzania bazą danych. Wartość tego atrybutu jest generowana w momencie tworzenia każdego obiektu i niemodyfikowalna w ciągu całego życia obiektu. Identyfikatory obiektów są wykorzystywane do implementacji związków między obiektami. Powiązane obiekty pamiętają nawzajem swoje identyfikatory.

Na przykładzie zilustrowano własności identyfikatorów obiektów. Identyfikator obiektu reprezentującego osobę Jana Kowalskiego został zapamiętany w obiekcie reprezentującym żonę Kowalskiego Janinę Kowalską. Mimo zmiany wartości kluczowych atrybutów obiektu, takich jak nazwisko, numer PESEL i płeć, obiekt reprezentujący byłego Kowalskiego nie zmienił swojej tożsamości określonej za pomocą jego identyfikatora.



Trwałość danych

- Obiekty mogą być tworzone w trwałym lub ulotnym obszarze składowania
- Trwałość powinna być własnością poszczególnych obiektów, a nie klas obiektów
- Trwałość powinna być własnością obiektów dowolnej klasy
- Sposób operowania na obiektach trwałych i ulotnych powinien być taki sam
- Obiekty w ciągu cyklu życia mogą być przenoszone między trwałym i nietrwałym obszarem składowania

Obiektowe bazy danych – Obiektowy model danych (23)

W obiektowych bazach danych powstały jako rozwinięcie języków obiektów jest potrzebny dodatkowy mechanizm utrwalania w bazie danych wybranych obiektów tworzonych przez obiektowe aplikacje bazy danych. W świecie systemów relacyjnych programiści tworzący aplikacje bazy danych muszą korzystać z dwóch języków o diametralnie różnej filozofii. Po pierwsze z języka dostępu do trwałych danych składowanych w bazie danych, na przykład języka SQL i po drugie z języka służącego do budowy funkcjonalności i interfejsu aplikacji. Jednym z celów budowy obiektowych baz danych była jednorodność języka służącego do pisania logiki przetwarzania aplikacji i dostępu do bazy danych. Aby ten cel osiągnąć poszukiwane rozwiązania muszą spełniać pięć podstawowych wymagań.

Pierwszym wymaganiem jest by aplikacje baz danych mogły równolegle tworzyć i przetwarzać zarówno zwykłe nietrwałe lokalne obiekty aplikacji, jak i obiekty trwałe, składowane w bazie danych.

Drugie wymaganie postuluje by własność trwałości dotyczyła pojedynczych obiektów, a nie całych klas obiektów. Oznacza, to że wystąpienia tej samej klasy mogą być zarówno trwałe jak i nietrwałe. Wymaganie to istotnie różni modele obiektowy i relacyjny. W modelu relacyjnym równolegle funkcjonują dwa systemy typów danych to jest trwałe relacje i lokalne dane aplikacji.

Trzecie wymaganie ma pozwolić by można było tworzyć trwałe wystąpienia dowolnych klas definiowanych przez użytkowników, a nie tylko wyróżnionych klas systemowych. W praktyce, klasy dla których chcemy tworzyć trwałe wystąpienia, muszą być wywiedzione z wyróżnionych klas systemowych o dodatkowej wymaganej funkcjonalności.

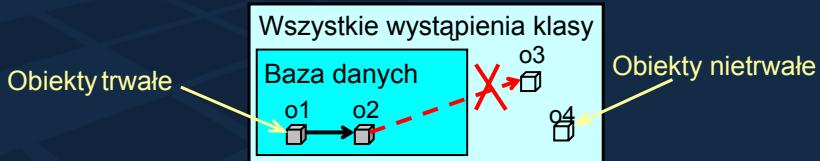
Czwarłe wymaganie postuluje by konstrukcje języka obiektowego zastosowanego do budowy aplikacji bazy danych nie rozróżniały obiektów trwałych i nietrwałych. To znaczy, że sposób przetwarzania trwałych i nietrwałych wystąpień tej samej klasy ma być taki sam.

Ostatnim wymaganiem jest by obiekty w ciągu swojego życia mogły wielokrotnie przechodzić między trwałym i nietrwałym obszarem składowania.



Trwałość danych

Obiekty stają się trwałe przez jawne utrwalenie lub przez bycie osiągalnymi przez inne obiekty trwałe.



```

pm = pmf.getPersistenceManager();
Punkt p = new Punkt(10.5, 7.1); // obiekt nietrwały
Koło k = new Koło(p, 4); // obiekt nietrwały
transaction = pm.currentTransaction();
pm.makePersistent(k); // utrwalenie obiektów p i k w bazie danych
transaction.commit();

```

Obiektowe bazy danych – Obiektowy model danych (25)

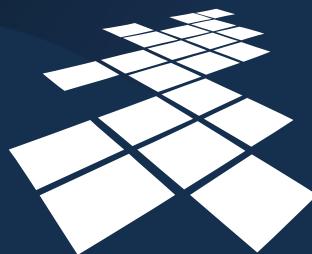
W rozwiązaaniach wypracowanych przez standardy ODMG i JDO wyróżnia się dwie klasy obiektów trwałych. Do pierwszej klasy należą obiekty, które stają się trwałe poprzez bezpośrednie wywołanie wyspecjalizowanej metody systemowej przesuwającej dany obiekt lub zbiór obiektów z nietrwałego do trwałego obszaru składowania. Na rysunku przykładem takiego obiektu jest obiekt „o1”. Do drugiej klasy należą obiekty, które są utrwalane w sposób pośredni, przez powiązanie ich z obiektami trwałymi. Obiekty należące do tej klasy są tak długo trwałe, jak długo są osiągalne z obiektów trwałych. Przykładem obiektu należącego do tej klasy trwałości jest obiekt „o2”. Tymczasem obiekt „o3”, który przez jakiś czas był trwały, w wyniku usunięcia powiązania z obiektem „o2” zostaje przesunięty do nietrwałego obszaru składowania. Jeżeli dodatkowo obiekt ten przestanie być osiągalny przez tymczasowe zmienne aplikacji bazy danych, zostanie z czasem usunięty przez systemowe procesy odśmiecania bazy danych.

Przedstawiony poniżej rysunku przykład wykorzystuje rozwiązania wypracowane w standardzie JDO. Obiekty „p” i „k” są w momencie utworzenia nietrwałe. Metoda systemowa „MakePersistent” przenosi obiekt „k” do bazy danych. Obiekt „k” należy do pierwszej klasy trwałości. Obiekt „p” zostanie również utrwalony przez związek łączący go z obiektem „k”. Obiekt „p” należy do drugiej klasy trwałości.

Obiektowe bazy danych

Obiektowe
i obiektowo-relacyjne
bazy danych

Wykład prowadzi:
Tomasz Koszlajda



UCZELNIA
ONLINE

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych

W ramach niniejszego wykładu zostaną przedstawione dwa alternatywne podejścia do budowy systemów obiektowych. Pierwsze z nich, polega na budowie nowego systemu bazy danych całkowicie od podstaw jako systemu obiektowego. Dalej takie systemy będziemy nazywali obiektowymi bazami danych. Drugie podejście, polega na ewolucji relacyjnych baz danych w kierunku modelu o własnościach obiektowych. Takie systemy będziemy nazywali obiektowo-relacyjnymi bazami danych.



Plan wykładu

- Model ODMG
- Język ODL
- Język OQL
- Obiektowo-relacyjne bazy danych
- Obiektywne rozszerzenia relacyjnych struktur danych
- Obiektywne rozszerzenia języków zapytań i modyfikacji danych

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (2)

Celem wykładu jest poznanie własności modelu danych obiektowych i obiektowo-relacyjnych baz danych. W ramach wykładu poznamy konstrukcje językowe umożliwiające tworzenie schematów baz danych oraz przetwarzanie danych.

Jako wzorzec obiektowych systemów baz danych został wykorzystany standard ODMG 3.0. Przedstawione zostaną dwie podstawowe części tego standardu: język definicji schematu bazy danych ODL oraz wzorowany na języku SQL język zapytań OQL. Z kolei jako przykład klasy obiektowo-relacyjnych systemów baz danych wybrano komercyjny systemem Oracle wzorowany na standardzie SQL3. Przedstawione zostaną konstrukcje językowe służące do tworzenia obiektowych struktur danych oraz do wyszukiwania i przetwarzania obiektów w bazach danych.



Model ODMG

Standard ODMG składa się z czterech podstawowych części:

- Opis modelu obiektowego
- **ODL** - język definicji schematu obiektowej bazy danych
- **OQL** - obiektowy język zapytań wzorowany na SQL
- **OML** – rozszerzenia obiektowych języków programowania: C++, Smalltalk i Java, do przetwarzania trwałych obiektów w obiektowych bazach danych

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (3)

W drugiej połowie lat osiemdziesiątych powstało wiele prototypów i komercyjnych produktów obiektowych systemów baz danych tworzonych na bazie języków obiektowych. Modele obiektowe tych systemów różniły się znacznie między sobą. Uważano wtedy, że jest to podstawową przyczyną braku komercyjnych sukcesów obiektowych baz danych. W ramach wysiłków ujednolicenia obiektowego modelu danych powstała inicjatywa standaryzacyjna ukonstytuowana w grupie o nazwie ODMG (ang. Object Database Management Group). W trakcie swojego działania w latach 1993 do 2001 grupa ta tworzyła kolejne wersje standardu. Ostatnim był standard ODMG w wersji 3.0. Obiektowy model zdefiniowany w standardzie ODMG jest wzorcem dla czysto obiektowych baz danych.

Standard ODMG składa się z czterech podstawowych części. Pierwszą częścią jest słowny opis własności i pojęć modelu obiektowej bazy danych. W stosunku do języków obiektowych model ten zawiera istotne rozszerzenia o własności charakterystyczne dla systemów baz danych. Rozszerzenia te dotyczą pojęć związków między obiektemi, tożsamości obiektów, trwałosci obiektów, rozszerzeń klas, struktur masowych, przetwarzania transakcyjnego, struktur metadanych, itp.

Druga część standardu zwiera definicję składni i semantyki języka ODL (ang. Object Definition Language) służącego do definiowania klas i interfejsów w schemacie obiektowej bazy danych. Język ODL jest wzorowany na języku IDL wypracowanym w ramach wcześniejszego standardu OMG (ang. Object Management Group) dedykowanego obiektowej integracji systemów informatycznych.

Trzecią częścią standardu jest definicja języka zapytań OQL (ang. Object Query Language). Jest to język wzorowany na języku SQL, ale wzbogacony o wszystkie koncepcje obiektowego modelu danych. W przeciwieństwie do języka SQL język OQL jest ograniczony do instrukcji zapytań. Z założenia nie obejmuje on instrukcji modyfikacji danych, które znajdują się w czwartej części standardu.

Ostatnia, czwarta część standardu obejmuje propozycje rozszerzeń służących do przetwarzania trwałych obiektów w obiektowych bazach danych. Rozszerzenia te są aplikowane do trzech języków obiektowych: C++, SmallTalk i Java.

Prace nad powiązaniem aplikacji baz danych pisanych za pomocą języka Java są kontynuowane w ramach standardu JDO (ang. Java Data Object). W roku 2006 powstała nowa grupa robocza Object Database Technology Working Group (ODBT WG), która zamierza opracować standard modelu danych nowej generacji obiektowych baz danych.



Język ODL

```

class Obraz {
    extent Obrazy
    attribute Date utworzony;
    attribute set<Figura> Elementy;
    relationship set<Osoba> autorzy
        inverse Osoba::utworzył;
    relationship set<Obraz> jestPierwowzorem
        inverse Obraz::jestModyfikacją;
    relationship set<Obraz> jestModyfikacją
        inverse Obraz::jestPierwowzorem;
    void dodaj (in Figura fig)
        raises (NiewłaściwaFigura);
    Obraz utwórzKopię()
        raises (ZaDużoKopii);    };

```

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (5)

Język ODL nie jest językiem programowania. Służy on do definiowania klas w schemacie obiektowej bazy danych. Definicja funkcjonalności obejmuje specyfikację atrybutów, związków i metod.

Definicja atrybutu klasy obejmuje nazwę i typ. Typem atrybutu mogą być proste predefiniowane typy danych lub typy złożone. Złożony typ danej może być skonstruowany na jeden z dwóch sposobów. Pierwszy sposób polega na modelowaniu złożonej struktury atrybutu za pomocą konstruktorów typów złożonych: krotki – „*struct*”, zbioru – „*set*”, wielozbioru – „*multiset*”, listy – „*list*”, tablicy - „*array*” i słownika – „*dictionary*”. Drugi sposób do definicji typu atrybutu wykorzystuje klasy zdefiniowane w schemacie bazy danych. Wartościami takich atrybutów będą obiekty tych klas. Atrybuty zdefiniowane w oparciu o klasy pozwalają modelować nie tylko złożoną strukturę atrybutów, ale również zdefiniowaną przez użytkownika nową semantykę.

Przykład zawiera definicję dwóch atrybutów: „*utworzony*” i „*Elementy*”. Pierwszy atrybut jest zdefiniowany na prostym typie danych „*Date*”. Drugi atrybut został zdefiniowany za pomocą konstruktora zbioru – „*set*” i klasy „*Figura*”. Wartościami tego atrybutu będą zbiory obiektów wystąpień klasy „*Figura*”, które nie będą niezależnie przechowywane w bazie danych, ale będą częścią składową obiektów wystąpień klasy „*Obraz*”.

Definicja związku obejmuje nazwę związku, typ związku i odwołanie do związku zwrotnego. Definicja typu związku obejmuje nazwę klasy, z którą obiekty danej klasy będą powiązane oraz krotność związku. Związki wielokrotne są modelowane za pomocą konstruktorów typów, tych samych co atrybuty złożone. Odwołanie do związku zwrotnego zawiera nazwę drugiego końca związku.

języków programowania.

Przedstawiony na slajdzie przykład zawiera dwie metody. Metoda „*dodaj*” nie zwraca wartości, posiada jeden parametr wejściowy „*fig*” typu „*Figura*” i może zgłosić jeden wyjątek o nazwie „*NiewłaściwaFigura*”. Metoda utwórz „*utwórzKopię*” jest bezparametrową, natomiast zwraca wartości typu „*Obraz*”. Metoda ta może zgłosić wyjątek o nazwie „*ZaDużoKopii*”.



Język OQL

- Deklaratywny język zapytań wzorowany na standardzie SQL92
- Zgodny z modelem obiektów ODMG
- Występuje w wersji języka ad-hoc lub zagnieżdzanego w językach obiektowych
- Obejmuje tylko instrukcje zapytań

```
select struct(liczba: count(*))
from f in Figury ←———— Rozszerzenie klasy
where f.powierzchnia() > 100
```

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (7)

Język OQL jest deklaratywnym językiem zapytań wzorowanym na standardzie SQL92. Język ten powstał by umożliwić wykonywanie w obiektowych bazach danych zapytań ad-hoc. W pierwszych obiektowych systemach baz danych zapytania trzeba było implementować za pomocą języków obiektowych nieprzystosowanych do przetwarzania dużych zbiorów danych.

Język OQL istotnie różni się od klasycznego języka SQL możliwością wyszukiwania złożonych strukturalnie i semantycznie oraz powiązanych danych. Jest on w pełni zgodny z modelem obiektów standardu ODMG. Język ten może być zastosowany na dwa sposoby: jako język ad-hoc interpretujący interakcyjnie przekazywane zapytania i wizualizujący ich wyniki lub jako język zanurzony w środowisku obiektowych języków programowania dla ułatwienia tworzenia aplikacji bazy danych.

Język OQL jest ograniczony do języka zapytań. W przeciwieństwie do języka SQL nie zawiera operacji modyfikacji danych.

Na slajdzie przedstawiono przykładowe zapytanie w języku OQL. Konstrukcja klauzuli SELECT umożliwia ustalenie na podstawie składni zapytania - struktury wyniku zapytania. Argumentem klauzuli FROM jest nazwa rozszerzenia klasy. Warunek logiczny zawarty w klauzuli WHERE zwiera wywołanie metody „powierzchnia” dla obiektów należących do rozszerzenia „Figury”.



Struktura wyników zapytań

Wynikiem zapytań w języku OQL mogą być kolekcje dowolnie złożonych struktur danych

```
select struct(t: w.typ, zw: (
    select struct (wx: wr.X, wy: wr.Y)
    from wr in wierzchołki)
from w in Wielokąty
```

Wynikiem zapytania jest wielozbiór danych o strukturze:

```
struct<p:String,multiset<wx:Float,wy:Float>>
```

↑
Typ wielokąta

← →
Współrzędne
wierzchołków wielokąta

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (8)

Wynikiem zapytań wyrażonych w języku OQL są kolekcje obiektów lub wartości. Struktura i funkcjonalność zwracanych obiektów jest zdefiniowana w schemacie bazy danych. Wynikiem mogą być obiekty odczytane z bazy danych lub obiekty dynamicznie utworzone w trakcie wykonywania zapytania za pomocą konstruktora klasy. Struktury wartości wynikowych są dynamicznie definiowane w zapytaniu. Struktura wartości, które są wynikiem zapytania może być dowolnie złożona. Wartościami składowymi złożonej wartości są wartości lub obiekty.

W podanym przykładzie struktura danych wynikowych jest krotką, co wynika z zastosowanego konstruktora typu – „*struct*”. Typem pierwszego atrybutu o nazwie „*t*” jest typ tekstowy – „*String*”. Wynika to z definicji atrybutu o nazwie „*Typ*” zdefiniowanego w klasie „*Wielokąty*”, składowanej w schemacie bazy danych. Drugi atrybut krotki o nazwie „*zw*” jest złożony. Jego wartościami są wielozbiory par liczb zmiennoprzecinkowych. Wartości tego atrybutu są generowane przez zagnieżdżone pod-zapytanie, które zwraca współrzędne wszystkich wierzchołków danego wielokąta.

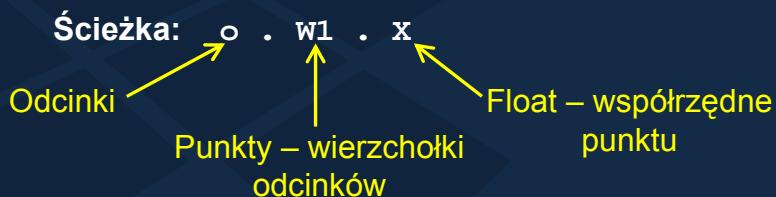
Zmienna „*w*” w zapytaniu reprezentuje obiekty klasy „*Wielokąt*”. Wynikiem zapytania, w którym argumentem klauzuli SELECT byłaby zmienna *w* będzie wielozbiór nie wartości, ale obiektów – wystąpień klasy „*Wielokąt*”.



Wyrażenia ścieżkowe

Wyrażenia ścieżkowe umożliwiają nawigację wzdłuż powiązań między obiektami, w głąb struktur atrybutów złożonych lub wyników metod.

```
select struct(x1: o.W1.X, x2: o.W2.X)
from o in Odcinki
```



Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (9)

Możliwość definiowania atrybutów złożonych i związków między obiektami spowodowała pojawienie się zupełnie nowych konstrukcji języka zapytań. Jedną z nowych możliwości są wyrażenia ścieżkowe reprezentujące wieloczlonową operację nawigacji. Wyrażenia ścieżkowe służą do specyfikacji operacji nawigacji w głąb obiektów złożonych, wzdłuż związków łączących obiekty lub do wyników zwróconych przez wywołania metod bezparametrowych. Pojedyncza, dowolnie dłużna ścieżka, może łączyć te trzy przypadki nawigacji.

Wartością wyrażenia ścieżkowego może być jedynie pojedynczy obiekt lub wartość o typie wynikającym z typu ostatniego elementu ścieżki. Wynika stąd ograniczenie stosowalności wyrażeń ścieżkowych do atrybutów jednowartościowych i związków jednokrotnych. Składniowym operatorem służącym do tworzenia ścieżek może być operator kropki: "." lub strzałki \rightarrow . Wyrażenia ścieżkowe mogą być używane w klauzulach: SELECT, FROM i WHERE.

Na slajdzie pokazano przykład wyrażenia ścieżkowego trójczłonowego reprezentującego nawigację w głąb atrybutów złożonych obiektów. Nawigacja przechodzi w głąb obiektów klasy „*Odcinek*” przez wierzchołki odcinków do współrzędnych wierzchołków na osi X. Wynikiem zapytania są pary liczb zmiennoprzecinkowych.



Operacje połączenia

Model ODMG obejmuje dwa rodzaje łączenia kolekcji obiektów

- Strukturalne - przez jawne związki między obiektemi

```
select struct(n: os.nazwisko, d: obr.utworzony)
from os in Osoby, obr in os.utworzyła
```

- Dynamiczne - przez zależności między wartościami atrybutów obiektów

```
select struct(n1: o1.nazwisko, n2: o2.nazwisko)
from o1 in Osoby, o2 in Osoby
where o1.adres.miasto = o2.adres.miasto
and o1 != o2
```

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (10)

Język OQL rozróżnia dwa rodzaje operacji łączenia (ang. join) kolekcji obiektów. Oprócz znanej z modelu relacyjnego operacji dynamicznego łączenia krotek, na podstawie zdefiniowanego warunku połączniowego, wprowadzono nową klasę łączenia strukturalnego, na podstawie związków zdefiniowanych w bazie danych. Składniowo połączenia strukturalne są definiowane jako wielowartościowe wyrażenia ścieżkowe umieszczone w klauzuli FROM.

W podanym przykładzie łączenie obiektów rozszerzania „Osoby” z obiektami rozszerzenia „Obrazy” jest wykonywane przez nawigację wzdłuż związku o nazwie „utworzyła”. System zarządzania obiektową bazą danych w tym wypadku nie dopasowuje dynamicznie par obiektów na podstawie warunku połączniowego, tylko nawiguje wzdłuż już ustalonych powiązań.

W języku OQL pozostawiono możliwość wykonywania relacyjnych operacji połączenia. Przykładem jest drugie zapytanie, w którym są łączone dwie kolekcje osób na podstawie dynamicznie weryfikowanego warunku zamieszkiwania w tym samym mieście. Dodatkowy warunek połączniowy „o1” != „o2” wyklucza łączenie tych samych osób.



Polimorfizm i dynamiczne wiązanie

- Język OQL pozwala wykonywać zapytania na polimorficznych kolekcjach obiektów.
- Dla tej klasy zapytań jest dostępne dynamiczne wiązanie nazw metod obiektów.

```
select f.powierzchnia()
from f in Figury
```

dynamiczne
wiązanie

**zmienna
polimorficzna**

wielokąt::powierzchnia()
koło::powierzchnia()

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (11)

Język OQL umożliwia wykonywanie zapytań na kolekcjach obiektów polimorficznych. Kolekcje obiektów polimorficznych powstają w konsekwencji specjalizacji klas przez mechanizm dziedziczenia. Jak już to było mówione rozszerzenia klas bazowych obejmują rozszerzenia wszystkich klas pochodnych. W związku z tym, wynikiem zapytań mogą być kolekcje obiektów różnych klas tworzących jedną hierarchię dziedziczenia. Wszystkie te obiekty współdzielą funkcjonalność zdefiniowaną w klasie, która jest korzeniem takiej hierarchii.

Dodatkowo język OQL realizuje mechanizm dynamicznego wiązania nazw komunikatów z różnymi metodami polimorficznej kolekcji obiektów. Umożliwia to pisanie zapytań, w których przypisanie występującym w zapytaniu komunikatom właściwych metod odbywa się dopiero w trakcie wykonywania tego zapytania. Ponadto, w ramach pojedynczego zapytania przypisanie to może być różne dla różnych polimorficznych obiektów.

W zapytaniu pokazanym na slajdzie jest przetwarzany heterogeniczny zbiór figur. Obiekty, które będą wynikiem zapytania mogą być wystąpieniami klas Wielokąt lub Koło, które różnią się strukturą i implementacją metod. Klasy te różni na przykład sposób wyznaczania powierzchni. Zmienna f zdefiniowana w zapytaniu jest więc zmienną polimorficzną, bo w trakcie wykonywania zapytania przypisywane jej będą zarówno wielokąty, jak i koła. Komunikat powierzchnia() przesyłany do zmiennej f będzie wiązany dynamicznie. Oznacza to, że metoda przypisywana do tego komunikatu będzie zależna od klasy obiektu, który aktualnie jest przypisany zmiennej f. Dla kół metodą tą będzie metoda zdefiniowana w klasie Koło, a dla wielokątów metoda z klasą Wielokąt.



Obiektowo-relacyjne bazy danych

Ewolucja rozszerzeń relacyjnego modelu danych:

- Składowanie kodu procedur w bazie danych
- Możliwość definiowania nowych typów danych
- Konstruktory złożonych typów danych
- Dziedziczenie typów danych
- Możliwość definiowania obiektowych typów danych
- Referencyjny typ danych
- Hierarchie zbiorów danych

Standard: SQL3/SQL99

Produkty komercyjne: Oracle, DB2

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (12)

Historia obiektowo-relacyjnych baz danych jest krótsza niż czysto obiektowych. Pierwsze prace na systemami tej klasy rozpoczęły się dopiero w połowie lat dziewięćdziesiątych. Równolegle trwały wysiłki standaryzacyjne, w ramach standardu SQL3 oraz prace nad systemami komercyjnymi, na przykład nad systemem Oracle lub DB2 oraz systemami klasy Open Source na przykład PostgreSQL.

Sposób konstrukcji systemów obiektowo-relacyjnych jest całkowicie odmienny od systemów czysto obiektowych. Punktem wyjścia jest tu zachowanie kompletnej funkcjonalności systemów relacyjnych i ewolucyjne modyfikacje wprowadzające dodatkowo oprócz własności relacyjnych - własności obiektowe. Pierwszym krokiem było wprowadzenie możliwości pamiętania w bazie danych oprócz danych, również procedur, początkowo słabo zintegrowanych z danymi. Następne kroki polegały na wprowadzeniu własności stricte obiektowych, takich jak możliwość definiowania nowych typów danych, dziedziczenia typów danych, definiowanie złożonych struktur danych, typów referencyjnych i hierarchii podzbiorów.

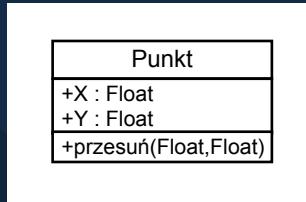


Definiowanie nowych typów danych

```

create type Punkt as object(
    x Float,
    y Float,
    member procedure przesun(px Float, py Float));
create type body Punkt as
    member procedure przesun(px Float, py Float) is
        begin
            self.x := self.x + px;
            self.y := self.y + py;
        end przesun;
    end;
create type koło as object(
    środek Punkt,
    Promień Float);

```



Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (13)

Obiektowo-relacyjny model danych umożliwia definiowanie użytkownikom bazy danych nowych typów danych, czyli odpowiednika klas w systemach czysto obiektowych. Definicja typu danych obejmuje definicję interfejsu typu oraz jego implementację. Interfejsem typu danych jest zbiór atrybutów oraz składnia operacji dostępnych dla danego typu. Implementacja typu danych jest implementacją operacji typu danych. Operacje są implementowane za pomocą proceduralnych języków bazy danych. Współczesne relacyjne bazy danych oferują proceduralne rozszerzenia dla języka SQL, stosowane do implementacji operacji typów danych oraz do implementacji niezależnych funkcji i procedur składowanych w bazie danych. W systemie Oracle proceduralnym językiem bazy danych jest język PL/SQL, w systemie DB2 jest to język o nazwie DB2 SQL.

Zdefiniowane przez użytkowników bazy danych nowe typy danych mogą być wykorzystywane do definicji atrybutów innych złożonych typów danych lub do definicji atrybutów schematów relacji. Ten typ danych jest nazywany typem atrybutowym. Wystąpieniami typów atrybutowych nie są pełnoprawne obiekty, gdyż nie posiadają one tożsamości i mogą być przechowywane w bazie danych tylko jako obiekty składowe.

Na slajdzie przedstawiono przykład definicji typu danych „*Punkt*” w systemie obiektowo-relacyjnej bazy danych Oracle. Pierwsza konstrukcja przedstawiona na slajdzie definiuje interfejs typu, który obejmuje atrybuty „x” i „y” reprezentujące współrzędne punktu oraz składnię operacji „*przesuń*” realizującej przesunięcie punktu na płaszczyźnie o dany wektor. Następna konstrukcja zawiera implementację procedury realizującej operację przesuwania punktów. Ostania, trzecia konstrukcja pokazuje zastosowanie zdefiniowanego typu „*Punkt*” do definicji innego typu „*Koło*”. Typ „*Punkt*” został wykorzystany do definicji środka „*Koła*”.



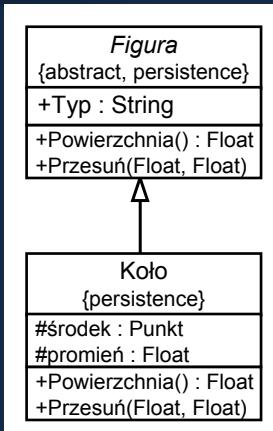
Dziedziczenie

```

create type figura as object (
    typ varchar(10),
    not instantiable member procedure
        przesuń(px Float, py Float),
    not instantiable member function
        powierzchnia return Float)
not instantiable not final;

-- klasa koło dziedziczy po klasie figura
create type koło under figura (
    środek Punkt,
    promień Float,
    overriding member procedure
        przesuń(px Float, py Float),
    overriding member function
        powierzchnia return Float);

```



Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (14)

Zdefiniowane przez użytkownika typy danych mogą być specjalizowane za pomocą mechanizmu dziedziczenia. W typach pochodnych w sposób przyrostowy definiuje się różnice między typem pochodnym i bazowym. W typie pochodnym do cech odziedziczonych z typu bazowego można dodać nowe atrybuty, nowe operacje, redefiniować kod odziedziczonych operacji lub zaimplementować kod odziedziczonej operacji abstrakcyjnej. Operacje abstrakcyjne mają zdefiniowaną składnię, ale nie posiadają implementacji. Ich zastosowaniem są typy danych, które służą jedynie jako typy bazowe dla innych docelowych typów danych.

Na slajdzie przedstawiono definicje dwóch typów: typu bazowego „*Figura*” i dziedziczącego po nim typu pochodnego „*Koło*”. Typ bazowy „*Figura*” jest typem abstrakcyjnym, ma dwie abstrakcyjne operacje: „*przesuń*” i „*powierzchnia*”. Abstrakcyjność tych operacji, jak i całego typu deklaruje się w sposób jawnego za pomocą słowa kluczowego: „*non instantiable*”. Z kolei słowo kluczowe „*not final*” oznacza, że typ ten może służyć jako typ bazowy. Typ *Koło* dziedziczy po typie „*Figura*” atrybut o nazwie „*typ*” i dwie wymienione metody abstrakcyjne. Definicja typu „*Koło*” zawiera dwa dodatkowe atrybuty: „*środek*” i „*promień*”, oraz implementację odziedziczonych operacji abstrakcyjnych. Za pomocą słowa kluczowego „*overriding*” definicja typu „*Koło*” jawnie informuje o redefinicyjnych odziedziczonych operacji. Na slajdzie nie przedstawiono wymaganej w tym wypadku implementacji operacji.



Tabele obiektów

W obiektowo-relacyjnych bazach danych tabele mogą przechowywać krotki albo obiekty.

```
create type Koło as object(
    środek Punkt,
    promień Float,
    member procedure
        przesuń(px Float, py Float),
    member function
        powierzchnia return Float);
```

Koło
{persistence}
#środek : Punkt
#promień : Float
+Powierzchnia() : Float
+Przesuń (Float, Float)

```
create table Koła of Koło;
insert into Koła values
    (new Koło(new Punkt(10.5, 7.2), 10.0));
```

Tabela obiektów

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (15)

Typy danych definiowane przez użytkownika bazy danych mogą być wykorzystywane do definiowania schematu całej tabeli. Dla rozróżnienia z typem atrybutowym, typ danych zastosowany do definicji schematu tabeli jest nazywany typem obiektowym lub krotkowym. Rozróżnienie między typami atrybutowymi i obiektowymi wprowadzone przez standard SQL3, nie dotyczy sposobu definiowania typu, ale jego zastosowania. Wartościami typów krotkowych są pełnoprawne obiekty posiadające systemowe identyfikatory obiektów OID, przechowywane jako niezależne dane w tabelach obiektów. Odwołując się do pojęć czysto obiektowych baz danych, tabela obiektów odpowiada rozszerzeniu klasy. Obiektowo-relacyjne bazy danych zawierają więc tabele krotek (relacje) i tabele obiektów.

Na slajdzie przedstawiono definicję typu obiektowego „Koło”, analogiczną do poprzednich przykładów. Jednak w tym wypadku, typ danych „Koło” został użyty do definicji schematu tabeli obiektowej „Koła”. Kolejna pokazana instrukcja wstawia do tej tabeli utworzony za pomocą konstruktora obiekt, który jest wystąpieniem typu danych „Koło”. Wywołanie konstruktora typu danych „Koło” zawiera zagnieżdżone wywołanie konstruktora obiektu składowego, który jest wystąpieniem typu atrybutowego „Punkt”.

*) Od tego miejsca zamiast terminu relacja, będzie używany równoważny termin tabela, bo literatura nie zna określenia relacja obiektów na nazwę zbioru trwałych obiektów w bazie danych.



Tożsamość danych

Obiekty przechowywane w tabelach mają przypisany systemowy identyfikator OID gwarantujący rozróżnialność obiektów niezależnie od przechowywanych w nich wartości.

```
create table Koła of Koło;
insert into Koła values
    (new Koło(new Punkt(10.5, 7.2), 10.0));

select ref(k), value(k)
from Koła k;
```

identyfikator obiektu **wartość obiektu**

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (16)

W odróżnieniu od krotek przechowywanych w relacjach, obiekty składowane w tabelach są identyfikowalne nie tylko poprzez wartości atrybutów, ale również przez systemowe identyfikatory obiektów OID. Wartość identyfikatora OID może być odczytana z obiektu za pomocą operatora referencji REF. Dostępna jest również operacja odwrotna, konwersji identyfikatora obiektu na wartość obiektu za pomocą operatora VALUE.

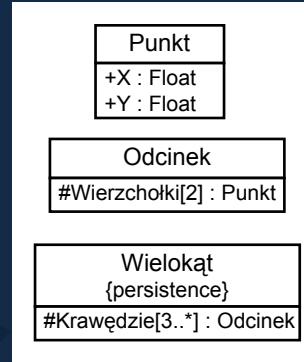
W przykładzie pokazano zastosowanie operatorów referencji i dereferencji. Do tablicy obiektów Koła wstawiono pojedynczy obiekt – koło o środku w punkcie o współrzędnych (10.5, 7.2) i promieniu o długości 10. Wynik zapytania wykonanego na tabeli Koła zawiera identyfikator obiektu i wartość obiektu.



Złożone struktury danych

Konstruktory typu umożliwiają modelowanie złożonych i wielowartościowych atrybutów danych.

```
create type Punkt as object(
    x Number,
    y Number);
create type TablicaWierzchołków
    as varray(2) of Punkt;
create type Odcinek as object(
    Wierzchołki TablicaWierzchołków);
create type ZbiórKrawędzi
    as table of Odcinek;
create type Wielokąt as object(
    Krawędzie ZbiórKrawędzi);
```



Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (17)

Obiekty składowane w tabelach mogą mieć złożoną strukturę. Standard SQL3 obejmuje zbiór konstruktorów typów analogiczny do tego ze standardu ODMG. W komercyjnych produktach lista konstruktorów typów danych jest trochę uboższa. W systemie Oracle lista ta obejmuje trzy konstruktory: konstruktor krotki dziedziczący po relacyjnym modelu danych, konstruktor zbioru i konstruktor tablicy. W definicji typu danych konstruktory te mogą być wzajemnie zagnieżdżane.

Na slajdzie pokazano zastosowanie konstruktorów typów na przykładzie złożonego typu danych Wielokąt. Najpierw do zdefiniowania typu danych Punkt zastosowano operator krotkowy. Następna definicja wykorzystuje operator tablicy VARRAY, do reprezentacji par punktów, zastosowany do definicji typu Odcinek. Kolejnym operatorem jest operator zbioru TABLE, zastosowany do reprezentacji zbioru odcinków. I na końcu definicja zbioru odcinków jest wykorzystana do modelowania zbioru krawędzi Wielokąta.



Heterogeniczne kolekcje danych

Tabele obiektów mogą przechowywać obiekty różnych typów danych o potencjalnie różnej strukturze.

```
create type figura as object (typ Varchar(10));
create type koło under figura (
    środek Punkt,
    promień Float);
create type wielokąt under figura (...);
create table Figury of figura;
insert into Figury values
    (new Koło(new Punkt(10.5, 7.2), 10.0));
insert into Figury values(new Wielokąt(...));
select value(f) from Figury f; -- wynikiem są koła i wielokąty
select value(f) from Figury f
    where value(f) is of (koło); -- wynikiem są koła
```

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (18)

Tabele obiektów, których schemat jest zdefiniowany na podstawie bazowego typu danych z którego są wywiedzione typy pochodne, mogą być heterogeniczne. To znaczy, że mogą przechowywać obiekty polimorficzne. Obiekty przechowywane w tabelach heterogenicznych mogą być wystąpieniami zarówno typu bazowego jak i typów pochodnych. W związku z tym, mogą miećną strukturę. Wystąpienia typów pochodnych mogą posiadać dodatkowe atrybuty nie występujące w typie bazowym.

Na slajdzie pokazano przykład tabeli obiektów która może zawierać dwa typy obiektów: koła i wielokąty. Dodatkowo, jeżeli typ figura nie byłby typem abstrakcyjnym, tabela obiektów mogłaby jeszcze zawierać generyczne figury, które nie są ani kołami, ani wielokątami. Wystąpienia typu Figura mają jeden atrybut, a wystąpienia typu Koło dwa dodatkowe atrybuty: środek i promień.

Do heterogenicznej tabeli Figury zdefiniowanej na typie bazowym Figura wstawiono dwa obiekty, pierwszy typu Koło i drugi typu Wielokąt. Następnie na tabeli Figury wykonano dwa zapytania. Wynikiem pierwszego zapytania jest dwuelementowa kolekcja obiektów polimorficznych: jedno koło i jeden wielokąt. Wynik drugiego zapytania jest ograniczony do obiektów typu koło w wyniku zastosowania operatora "is of". W naszym konkretnym przypadku wynik zapytania będzie zawierał dokładnie jeden obiekt.



Dynamiczne wiązanie metod

```

create type figura as object (
    member function powierzchnia return Float);
create type koło under figura (overriding member
    function powierzchnia return Float);
create type wielokąt under figura (overriding member
    function powierzchnia return Float);
create table Figury of figura;
insert into Figury values
    (new Koło(...));
insert into Figury values
    (new Wielokąt(...));

select f.powierzchnia( ) from Figury f;

```

dynamiczne wiązanie

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (19)

W zapytaniach wykonywanych na heterogenicznych tabelach obiektów można stosować dynamiczne wiązanie nazw operacji z implementującym je kodem, analogicznie jak wiązanie nazw komunikatów z kodem metod w czysto obiektowych bazach danych. Mechanizm ten zilustrowano na przykładzie, który jest rozszerzeniem przykładu z poprzedniego slajdu. Definicje typów Figura, Koło i Wielokąt rozszerzono o operację wyznaczania powierzchni figur. Kod tej operacji zdefiniowany dla bazowego typu Figura jest redefiniowany w typach pochodnych Koło i Wielokąt. Do heterogenicznej tabeli obiektów wstawiono dwa polimorficzne obiekty: koło i wielokąt. W zapytaniu wykonywanym na tabeli Figury wywołanie operacji powierzchnia() jest dynamicznie związane odpowiednio do typu obiektu, raz do kodu operacji powierzchnia() zdefiniowanej dla typu Koło i raz do kodu operacji powierzchnia() zdefiniowanej dla typu Wielokąt.

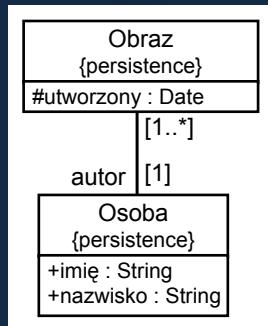


Referencyjny typ danych

Nowy systemowy typ danych REF służy do modelowania związków między obiektami.

```

create type Osoba as object (
    imię varchar(10),
    nazwisko varchar(20));
create type Obraz as object (
    utworzony Date,
    autor REF Osoba);
create table Obrazy of Obraz;
create table Osoby of Osoba;
insert into Osoby values ('Jan', 'Tarzan');
insert into Obrazy
    select '1-04-2006', ref(o)
    from Osoby o where nazwisko = 'Tarzan';
  
```



Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (20)

W obiektowo-relacyjnych bazach danych związki między obiektami są modelowane za pomocą nowego systemowego typu danych – typu referencyjnego. Wartościami referencyjnego typu danych są identyfikatory obiektów. Referencyjny typ danych służy do przechowywania identyfikatorów innych obiektów powiązanych z danym obiektem. Mechanizm referencyjnych typów danych w przeciwieństwie do związków w czysto obiektowym modelu danych reprezentuje jedynie związki jednokierunkowe. Modelowanie związków dwukierunkowych wymaga odrębnego użycia dwóch różnych referencji.

Na slajdzie pokazano przykład jednokierunkowego związku jednokrotnego łączącego obiekty typu Obraz z wystąpieniami typu Osoba. Definicja typu Obraz obejmuje atrybut autor zdefiniowany na referencyjnym typie danych. Wartościami tego atrybutu będą identyfikatory obiektów typu Osoba.

Na bazie typów Obraz i Osoba utworzono tabele obiektów: Obrazy i Osoby. Następnie do tabeli Osoby wstawiono obiekt reprezentujący Jana Tarzana, a do tabeli Obrazy wstawiono obiekt reprezentujący obraz utworzony w dniu 1-IV-2006 roku utworzony przez Jana Tarzana. Zapytanie zagnieżdżone w instrukcji INSERT wydobywa z tabeli osób identyfikator obiektu reprezentującego Jana Tarzana i umieszcza go w atrybucie referencyjnym.

Związki o krotności N

```

create type ZbiórAutorów as table of ref Osoba;
create type Obraz as object(
    utworzony Date,
    autorzy ZbiórAutorów);
create table Obrazy of Obraz
    nested table autorzy store as aut;
...
insert into Obrazy
    select '1-04-2006',
        TypAutorzy(ref(o)) from Osoby o
    where nazwisko = 'Tarzan';
insert into Table (select autorzy
    from Obrazy)
    (select ref(o) from Osoby o
    where nazwisko = 'Nowak')

```

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (21)

Zamodelowanie związków wielokrotnych wymaga połączenia koncepcji konstruktora zbioru i typu referencyjnego. Ilustruje to przykład pokazany na slajdzie, w którym obraz może mieć więcej niż jednego autora. Do przechowywania zbioru identyfikatorów skonstruowano typ „*ZbiórAutorów*”, którego struktura służy do przechowywania zbioru identyfikatorów Osób. Typ został następnie użyty do zdefiniowania atrybutu autorzy w ramach typu „*Obraz*”.

Definicja tabeli obiektów „*Obrazy*” obejmuje konstrukcję definiującą zagnieżdzoną tabelę do przechowywania wielokrotnych referencji na autorów dla każdego z obrazów. Pierwsza z operacji wstawienia dotyczy tabeli „*Obrazy*”. Wstawiany obiekt zawiera pojedynczą referencję na osobę Tarzana. Kolejna operacja wstawiania dotyczy wielowartościowego atrybutu referencyjnego. Do zbioru autorów obrazu oprócz Tarzana jest dodawany identyfikator kolejnego autora (lub autorów) o nazwisku Nowak. Druga operacją INSERT nie tworzy więc nowego obiektu w tabeli „*Obrazy*”, ale dodaje kolejne powiązanie do związku wielokrotnego.



Wyrażenia ścieżkowe

Możliwe jest tworzenie wieloczłonowych wyrażeń ścieżkowych nawigujących wzdłuż powiązań między obiekttami lub w głąb złożonych struktur obiektów.

ścieżka w głąb obiektu

```
select o.utworzony, a.nazwisko, o.elementy.typ  
from Obrazy o, table(o.autorzy) a
```



ścieżka wzdłuż związku

Obiektowe bazy danych – Obiektowe i obiektowo-relacyjne bazy danych (22)

Atrybuty złożone i typy referencyjne umożliwiają wykonywanie operacji nawigacji w głąb obiektu lub wzdłuż referencji między obiektami. Opisujące takie nawigacje wyrażenia ścieżkowe mogą być wieloczłonowe i mogą być użyte w klauzulach: SELECT, FROM i WHERE.

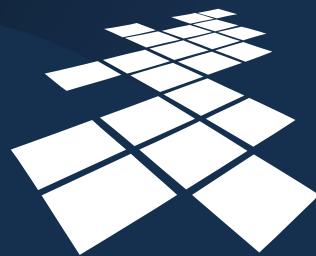
Na slajdzie pokazano dwa wyrażenia ścieżkowe. Pierwsze w klauzuli SELECT nawiguje w głąb atrybutu złożonego, od obiektu typu Obraz do figur, które są elementami składowymi obrazu, i dalej do atrybutu prostego typu figury.

Drugie wyrażenie ścieżkowe umieszczone w klauzuli FROM, służy do realizacji połączenia strukturalnego, analogicznie jak w modelu czysto obiektowym. Ścieżka nawigacji przechodzi w tym wypadku od obiektów z tabeli „Obrazy” wzdłuż wielokrotnych referencji związku autorzy do obiektów tabeli „Osoby”. Uzyskane w ten sposób połączenie łączy obrazy z ich autorami.

Obiektowe bazy danych

Implementacja
obiektowych baz danych

Wykład prowadzi:
Tomasz Koszlajda



UCZELNIA
ONLINE

Obiektowe bazy danych – Implementacja obiektowych baz danych

Niniejszy wykład jest poświecony problemom implementacji systemów obiektowych baz danych. Nowe własności modelu danych wymagają wydajnej implementacji uwzględniającej specyfikę obiektowego modelu danych i nowego modelu przetwarzania..



Plan wykładu

- Architektura obiektowych systemów baz danych uwzględniająca specyficzny model przetwarzania
- Fizyczne zarządzanie obiektemi
- Wydajny dostęp do kolekcji obiektów
- Obsługa atrybutów i związków wielowartościowych

Obiektowe bazy danych – Implementacja obiektowych baz danych (2)

Celem wykładu jest poznanie rozwiązań implementacyjnych dedykowanych dla obiektowych i obiektowo-relacyjnych systemów baz danych.

W ramach wykładu zostaną przedstawione nowe architektury systemów zarządzania bazami danych uwzględniające charakterystyki przetwarzania danych właściwe dla nowych dziedzin zastosowań. Następnie poznamy rozwiązania dotyczące fizycznego zarządzania obiektemi oraz nowe struktury danych przyśpieszające przeszukiwanie dużych powiązanych kolekcji obiektów. Na końcu zostaną przedstawione rozwiązania wydajnego przetwarzania atrybutów wielowartościowych i związków wielokrotnych.



Specyfika obiektowego modelu danych

- Identyfikatory obiektów
- Powiązania referencyjne między obiektami
- Atrybuty wielowartościowe
- Wyrażenia ścieżkowe
- Hierarchie rozszerzeń klas
- Duże obiekty
- Nowy model przetwarzania danych

Obiektowe bazy danych – Implementacja obiektowych baz danych (3)

Obiektowy model danych posiada wiele nowych unikalnych cech nieznanych w relacyjnym modelu danych. Należą do nich systemowe identyfikatory obiektów OID, referencyjne związki między obiektami, atrybuty wielowartościowe, wyrażenia ścieżkowe i hierarchie rozszerzeń klas. Te nowe elementy modelu wymagają nowych i wydajnych rozwiązań implementacyjnych.

Ponadto, dziedziny zastosowań obiektowych baz danych mają inne wymagania co właściwości systemów baz danych. Dotyczy to na przykład konieczności przechowywania w bazie danych bardzo dużych obiektów, do przechowywania na przykład projektów, animacji lub długich dokumentów. Kolejną cechą specyficzną nowych dziedzin zastosowań jest inny model przetwarzania danych, polegający na wielokrotnym i intensywnym przetwarzaniu dużych powiązanych kolekcji obiektów, na przykład w ramach systemy wspomagania projektowania. W systemach relacyjnych podstawowe problemy wydajności dotyczyły efektywnego wyszukiwania w bardzo dużych zbiorach danych niewielkich podzbiorów danych oraz efektywnej realizacji operacji łączenia relacji. W zastosowaniach obiektowych baz danych bardziej typowe są operacje nawigacji wzdłuż długich hierarchicznych ścieżek powiązań.



Architektura obiektowych baz danych

Klasyczna architektura klient-serwer – przesyłanie zapytań



Obiektowa architektura klient-serwer – przesyłanie danych



Obiektowe bazy danych – Implementacja obiektowych baz danych (4)

Model przetwarzania danych dla dziedzin zastosowań typowych dla okresu dominacji relacyjnego modelu danych polega na jednokrotnym wykonywaniu przez daną transakcję małej liczby operacji na bardzo niewielkim podzbiorze danych bardzo dużej bazy danych. Typowymi wykonywanymi operacjami w tym modelu są operacje selekcji i łączenia. W bankowym systemie informatycznym typowym przykładem jest transakcja wypłaty pieniędzy z konta obejmująca wyszukanie pojedynczej krotki reprezentującej dane konto w zbiorze setek tysięcy kont, następnie dodanie pojedynczej krotki do liczącego miliony krotek zbioru operacji bankowych i na koniec zmodyfikowanie salda w pojedynczej krotce reprezentującej konto.

Dla takiego wzorca przetwarzania właściwy model przepływu danych w architekturze klient-serwer minimalizujący wolumen transferu danych, polega na przesyłaniu w jedną stronę żądań wykonania operacji na bazie danych i w drugą stronę niewielkich przetwarzanych zbiorów danych. Przetwarzanie zapytań odbywa się w całości na serwerze bazy danych. Do klientów przez sieć komputerową docierają tylko wyniki tego przetwarzania, czyli najczęściej niewielkie podzbiory danych.

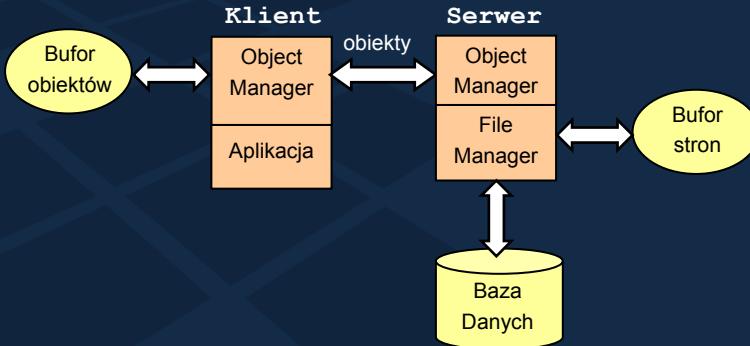
Model przetwarzania właściwy dla zastosowań typowych dla obiektowych baz danych polega intensywnym przetwarzaniem stosunkowo dużego zbioru powiązanych danych. W ramach pojedynczej sesji mają miejsce wielokrotne odwołania do tych samych obiektów. Typowymi operacjami są nawigacje między powiązanymi obiektami lub w głąb atrybutów złożonych. W systemie wspomagania projektowania przykładem są operacje projektowe polegające na cyklicznych modyfikacjach niewielkiego zbioru danych i ich weryfikacji w szerszym kontekście wszystkich powiązanych obiektów.

Dla takiego wzorca przetwarzania przedstawiony model przepływu danych nie jest optymalny. Dla pojedynczej sesji projektowej wiążałby się on z bardzo dużą liczbą żądań wykonania operacji przez serwer bazy i dodatkowo dużą część tych żądań powtarzałaby się. Bardziej wydajnym rozwiązaniem jest przeniesienie funkcjonalności przetwarzania zapytań na stronę klienta serwera bazy danych. W tej sytuacji serwer bazy danych będzie przesyłał do klienta całe strony dyskowe, które następnie będą utrzymywane w lokalnym buforze stron dyskowych. Teraz wielokrotnie powtarzające się odwołania do tych samych obiektów będą obsługiwane lokalnie.



Buforowanie obiektów

Buforowanie pojedynczych obiektów zamiast całych stron dyskowych.



Obiektowe bazy danych – Implementacja obiektowych baz danych (5)

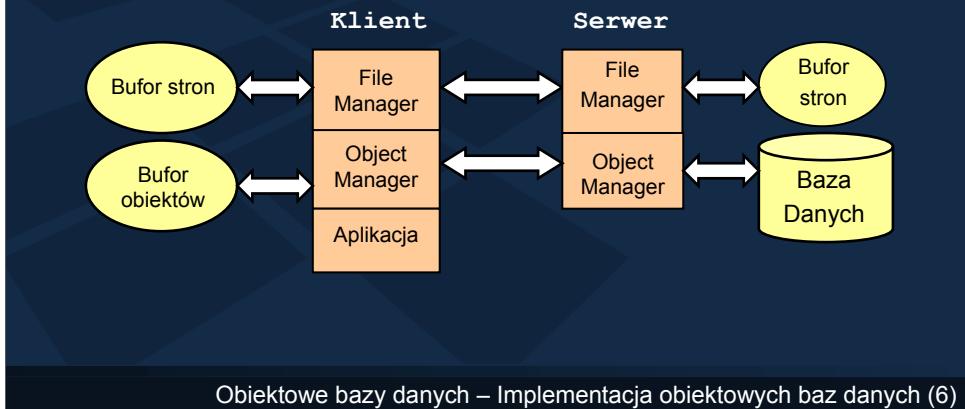
Dla słabo pogrupowanych danych wydajność bufora stron klientów serwera bazy danych będzie niska. Jedynie niewielki procent transferowanych i przechowywanych w buforze stron danych jest użyteczny. Reszta nie przetwarzanych w ramach danej sesji obiektów, a znajdująca się na tych samych stronach dyskowych stanowi tylko zbędny balast. Nawigacja do kolejnych obiektów przyypełnionym niepotrzebnymi danymi buforze stron będzie wymagała częściej i wielokrotniej wymiany stron w buforze.

Rozwiązaniem jest zastąpienie słabo wypełnionego użytkownikiem z punktu widzenia danej sesji danymi bufora stron lepiej upakowanym buforem obiektów. Serwer bazy danych będzie dla minimalizacji transferu zamiast całych stron przesyłać do Klienta pojedyncze obiekty, a bufor obiektów klienta będzie służył do przechowywania jedynie przetwarzanych w danym momencie obiektów. Dzięki lepszemu upakowaniu danych w buforze znacznie rzadziej potrzebna będzie czasochłonna wymiana zawartości bufora.



Dwupoziomowy bufor bazy danych

Zwiększa wydajność przetwarzania dla słabo pogrupowanych danych. Bufor obiektów gwarantuje lepsze upakowanie danych.



Obiektowe bazy danych – Implementacja obiektowych baz danych (6)

Rozwiązywanie polegające na zamianie bufora stron na bufor obiektów jest bardzo wydajne dla słabo zgrupowanych danych. Tymczasem jednym z rozwiązań stosowanych w systemach zarządzania obiektowymi bazami danych jest fizyczne grupowanie danych polegające na przechowywaniu powiązanych i często przetwarzanych w ramach jednej sesji danych w tych samych lokalizacjach pamięci dyskowej. Obsługa dobrze pogrupowanych danych jest w wypadku zastawania buforów obiektów mało wydajna. Pobranie z serwera bazy danych na przykład stu powiązanych ze sobą obiektów znajdujących się na jednej stronie dyskowej będzie wymagało stukrotnego wywołania żądania pobrania z serwera pojedynczego obiektu.

Żeby uczynić przedstawioną na poprzednim slajdzie architekturę systemów obiektowych baz danych bardziej elastyczną wprowadzono po stronie klienta dwupoziomowy bufor składający się z bufora stron i bufora obiektów. Między serwerem a klientem są przesyłane całe strony dyskowe. Dla dobrze pogrupowanych danych minimalizuje to komunikację między klientem a serwerem. Jednak przetwarzane przez aplikację bazy danych obiekty mogą być kopiowane z bufora stron do bufora obiektów. Optymalizuje to wykorzystanie buforów dla źle pogrupowanych danych. Usuwanie stron dyskowych z przepełnionego bufora stron nie oznacza braku dostępności znajdujących się na nich obiektów. Przed usunięciem strony dyskowej z bufora stron aktualnie przetwarzane obiekty będą skopiowane do bufora obiektów. Dzięki temu kolejne odwołania do tych obiektów nie będą wymagały ponownego załadunku zawierających je stron. Aplikacja będzie przetwarzała kopie tych obiektów w buforze obiektów.



Strategie utrzymywania bufora obiektów

Kopiowanie obiektów między buforem stron a buforem obiektów

- Kopiowanie obiektu z bufora stron do bufora obiektów
 - Natychmiastowe
 - Opóźnione
- Relokacja obiektu do bufora stron
 - Natychmiastowa
 - Opóźniona

Obiektowe bazy danych – Implementacja obiektowych baz danych (7)

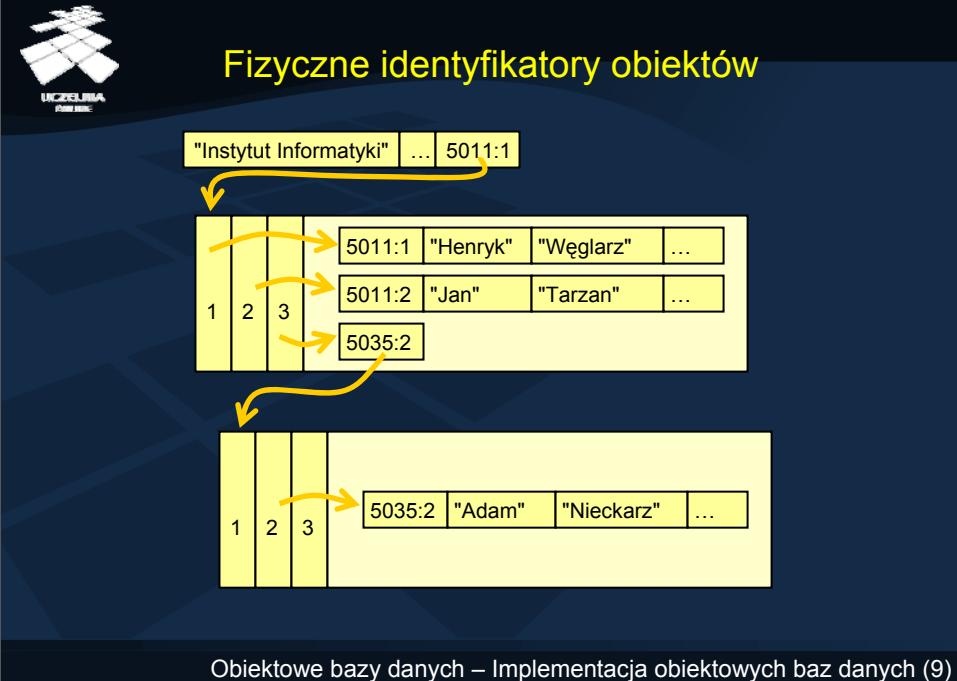
Proponowane są różne strategie utrzymywania buforów obiektów właściwe dla różnych modeli przetwarzania. Strategie te różnią się momentem kopiowania obiektu z bufora stron do bufora obiektów oraz momentem realokacji zmodyfikowanego obiektu z bufora obiektów z powrotem do bufora stron.

Można wyróżnić dwie opcje momentu kopiowania obiektu z bufora stron do bufora obiektów. Pierwsza opcja zakłada, że aplikacja może korzystać tylko i wyłącznie z obiektów znajdujących się w buforze obiektów. Wymaga to natychmiastowego kopiowania obiektów z bufora stron do bufora obiektów, przy pierwszym odwołaniu do danego obiektu. Bufor obiektów powinien być w tej sytuacji dużo większy niż bufor stron. Lepsze upakowanie obiektów zmniejsza znacznie liczbę błędów stron. Opcja ta będzie optymalna dla długiego i intensywnego przetwarzania większego zbioru obiektów.

Druga opcja zakłada, że aplikacja może odwoływać się do kopii obiektów ulokowanych w buforze obiektów, jak i do oryginałów obiektów znajdujących się buforze stron. Pozwala to na utrzymywanie tylko jednej kopii obiektu albo w buforze obiektów, albo w buforze stron. W tej sytuacji obiekty są kopiowane do bufora obiektów jedynie w momencie usuwania zawierającej je strony z bufora stron, po to by następnie odwołanie do danego obiektu nie spowodowało błędu strony, czyli konieczności jej powtórnego załadowania z serwera bazy danych. W tej sytuacji rozmiary buforów stron i buforów obiektów powinny być zrównoważone. Opcja ta będzie optymalna dla krótkiego przetwarzania dużych zbiorów obiektów.

Następne dwie opcje dotyczą momentu realokacji obiektu zmodyfikowanego w buforze obiektów z powrotem na właściwe miejsce na stronie w buforze stron. Pierwsza opcja zakłada, że zmodyfikowany obiekt jest umieszczany z powrotem na właściwej stronie w momencie powtórnego załadowania tej strony do bufora stron. Druga opcja polega na przesunięciu momentu realokacji, aż do czasu gdy obiekt przestanie być przetwarzany. W wypadku błędu strony, będzie to wymagać ściagnięcia tej strony z serwera bazy danych do bufora stron.

Kombinacje wymienionych opcji dają w sumie cztery potencjalne strategie utrzymania buforów obiektów. Wybór określonej strategii zależy od charakterystyki przetwarzania danej dziedziny zastosowania.



Obiektowe bazy danych – Implementacja obiektowych baz danych (9)

Stosowane są dwa rodzaje identyfikatorów obiektów: identyfikatory fizyczne i identyfikatory logiczne.

Jako identyfikator obiektu może służyć fizyczna lokalizacja obiektu w bazie danych. Bazy danych odwzorowują zajmowaną przez nie przestrzeń dyskową na uporządkowany i ponumerowany zbiór stron dyskowych, które są minimalnymi jednostkami transferu danych z pamięci dyskowej do pamięci operacyjnej. Na pojedynczej stronie dyskowej alokowanych jest wiele obiektów. Zazwyczaj na identyfikator obiektu składa się numer strony dyskowej, na której jest ulokowany obiekt oraz numer lokalizacji obiektu na tej stronie nazywany „*slotem*”.

Podstawową zaletą identyfikatorów fizycznych jest szybkość dostępu do obiektu. Znajomość identyfikatora umożliwia natychmiastowy dostęp do obiektu. Jest to bardzo istotne dla wydajności operacji nawigacji. Wadą jest przywiązywanie identyfikatorów do fizycznej lokalizacji, co utrudnia fizyczną reorganizację bazy danych. Na przykład w wypadku, gdy rozmiar obiektu zlokalizowanego na danej stronie w wyniku modyfikacji przekroczy rozmiar wolnego miejsca na stronie, obiekt musi być przesunięty na inną stronę. Żeby uniknąć zmiany identyfikatora przesuwanej obiektu obiekt zwalnia miejsce na stronie, ale nie zwalnia przydzielonego numeru slotu. Ilustruje to przykład. Identyfikator obiektu reprezentujący osobę Adama Niekarza, który przesunięto ze strony dyskowej o numerze 5011 na stronę o numerze 5035, pozostał niezmieniony. Na stronie 5011 w slacie 3 wstawiony został wskaźnik na nową lokalizację obiektu. Jednak pozostawienie tego samego identyfikatora wydłuży czas dostępu do tego obiektu. Zamiast pojedynczego dostępu do dysku teraz potrzebne będą dwa dostępy. Najpierw do oryginalnej strony, na którą wskazuje identyfikator obiektu, a potem do strony, na której faktycznie znajduje się przesunięty obiekt.



Logiczne identyfikatory obiektów

- B+-drzewo założone na OID



- tablica haszowa
- odwzorowanie bezpośrednie – logiczny identyfikator jest indeksem w tablicy odwzorowania

Obiektowe bazy danych – Implementacja obiektowych baz danych (10)

Alternatywą dla fizycznych identyfikatorów obiektów są identyfikatory logiczne. Identyfikatory logiczne są sztucznie generowanymi wartościami, całkowicie niezależnymi od fizycznej lokalizacji obiektu. Zaletą tego rozwiązania jest swoboda w reorganizacji fizycznej struktury bazy danych, nie wpływającej w żaden sposób na wartości identyfikatorów, czy pogorszenie warunków dostępu do obiektów. Wada logicznych identyfikatorów jest dłuższy czas dostępu do obiektów na podstawie znajomości identyfikatora obiektu.

Stosowane są trzy metody odwzorowania logicznego identyfikatora obiektu na jego fizyczny adres. Pierwszą jest założenie indeksu na logicznych identyfikatorach obiektu. Przykład takiego indeksu jest zilustrowany na slajdzie. Drugą metodą odwzorowania jest zastosowanie funkcji haszowej. Argumentem funkcji haszowej jest logiczny identyfikator obiektu. Trzecią metodą jest bezpośrednie odwzorowanie w tablicy, w której identyfikator służy jako indeks tablicy, a komórki tablicy zawierają fizyczne adresy obiektów.

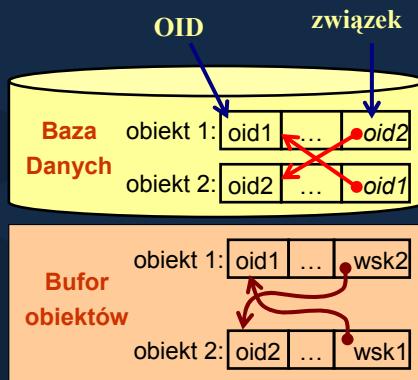


Transformacja identyfikatorów obiektów

Transformacja identyfikatorów obiektów to zamiana identyfikatora obiektu na adres obiektu w pamięci operacyjnej.

Alternatywy:

- Miejsce transformacji:
bufor stron ↔ bufor obiektów
- Czas transformacji:
natychmiastowe ↔ opóźnione
- Sposób transformacji:
bezpośrednia ↔ pośrednia



Obiektowe bazy danych – Implementacja obiektowych baz danych (11)

Transformacja identyfikatorów obiektów (ang. pointer swizzling) polega na zamianie w obiektach składowanych w pamięci operacyjnej identyfikatorów obiektów reprezentujących związki między obiekty na adresy tych obiektów w pamięci operacyjnej. Taka transformacja może mieć miejsce, jeżeli zarówno obiekt zawierający referencje, jak i obiekt przez nią wskazywany znajdują się jednocześnie w pamięci operacyjnej, to jest w buforze obiektów lub w buforze stron. Celem transformacji identyfikatorów jest przyśpieszenie operacji nawigacji. Nawigacja wykonywana przez adres obiektu w pamięci operacyjnej jest znacznie szybsza, niż nawigacja wymagająca przejścia przez skoncentrowane struktury dostępu, na przykład B+-drzewo założone na identyfikatorach obiektów, a następnie ustalenia lokalizacji obiektu w buforach. Zysk z zastosowania takiej transformacji jest wprost proporcjonalny do liczby powtórzeń operacji nawigacji. Jednak jak ustaliliśmy wielokrotne przetwarzanie tego samego zbioru obiektów jest typowe dla modelu przetwarzania obiektowych baz danych.

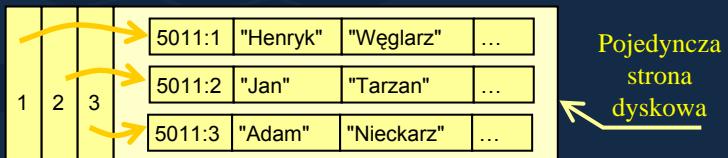
Idea transformacji identyfikatorów obiektów została przedstawiona na rysunku. Baza danych zawiera dwa obiekty o identyfikatorach oid1 i oid2. Obiekty te są wzajemnie powiązane. Obiekt o identyfikatorze oid1 zawiera referencję na obiekt o identyfikatorze oid2 i na odwrót, obiekt oid2 zawiera referencję na obiekt oid1. Te dwa obiekty załadowane do bufora w pamięci operacyjnej mają przetransformowane referencje z globalnych identyfikatorów obiektów na wskaźniki w pamięci operacyjnej.

Są trzy podstawowe alternatywy dla implementacji mechanizmu transformacji identyfikatorów. Pierwsza dotyczy miejsca wykonywania transformacji w dwupoziomowym systemie buforów. Polega ona na wyborze miejsca wykonywania transformacji: w buforze stron albo w buforze obiektów. Druga alternatywa dotyczy czasu transformacji. Możliwe opcje to: natychmiast po załadowaniu obiektów do pamięci operacyjnej albo dopiero przed pierwszą próbą dostępu przez aplikacje bazy danych do danego obiektu. Ostatnia alternatywa dotyczy sposobu transformacji i obejmuje dwie opcje. Pierwszą opcją jest transformacja referencji zawierających identyfikatory obiektów na adresy tych obiektów w pamięci operacyjnej. Druga opcja to transformacja na adres pośredniczącej struktury w pamięci operacyjnej. To drugie rozwiązanie umożliwia przesuwanie obiektów w buforach bez konieczności modyfikacji wartości przetransformowanych wskaźników.

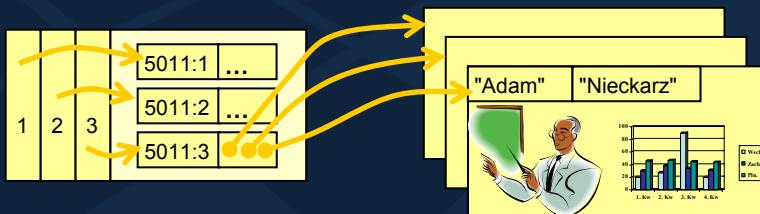


Składowanie obiektów

Małe obiekty mieszą się w całości na pojedynczej stronie dyskowej



Duże obiekty wymagają przydania wielu stron dyskowych



Obiektowe bazy danych – Implementacja obiektowych baz danych (13)

W obiektowych bazach danych ważna jest wydajna implementacja składowania dużych obiektów. Rozmiary obiektów zawierających długie animacje, duże rysunki lub długie dokumenty mogą osiągać wielkości mierzone w gigabajtach. Rozróżnienie między dużymi a małymi obiektami jest całkowicie jednoznaczne. Małe obiekty to takie, które mieszą się całości na pojedynczej stronie dyskowej. Podczas gdy rozmiar dużych obiektów przekracza rozmiar strony dyskowej.

Górny rysunek na slajdzie ilustruje sposób organizacji małych obiektów. Na pojedynczej stronie dyskowej znajdują się w całości trzy małe obiekty. Dolny rysunek ilustruje zmianę organizacji obiektu reprezentującego Adama Niekarza. Do obiektu dodano: animacje, wykresy i obrazy, w wyniku czego rozmiar obiektu przekroczył rozmiar całej strony dyskowej, a nie tylko rozmiar wolnego miejsca na stronie, którą współdzielił z trzema innymi obiektami. Do przechowywania wartości tego obiektu będą potrzebne trzy nowe strony dyskowe. Będą to prywatne strony dużego obiektu, nie będą one współdzielone z innymi obiektami. Nagłówek obiektu po jego transformacji do struktury dużego obiektu pozostanie na źródłowej stronie o numerze 5011.



Zarządzanie dużymi obiektami

Struktura dużych obiektów musi umożliwiać wydajny dostęp do mniejszych fragmentów obiektów.



Rozmiar dużych obiektów może być na tyle duży, że istotne staje się wydajne zarządzanie pojedynczymi dużymi obiektami. Dla dużych obiektów należy zagwarantować wydajną realizację operacji wyszukiwania, wstawiania, modyfikacji i usuwania sekwencji bajtów o określonej długości i lokalizacji w strukturze obiektu rozumianej jako ciąg bajtów.

Wydajna implementacja dużych obiektów opiera się na dwóch istotnych założeniach. Po pierwsze zapewnienia wydajnej metody wyszukiwania w ciągu bajtów rozrzuconych po wszystkich stronach dyskowych dużego obiektu bajtu o numerze określającym jego pozycję w ciągu bajtów. Po drugie zagwarantowania, że lokalne modyfikacje obiektu będą wymagały jedynie lokalnej rekonstrukcji jego struktury.

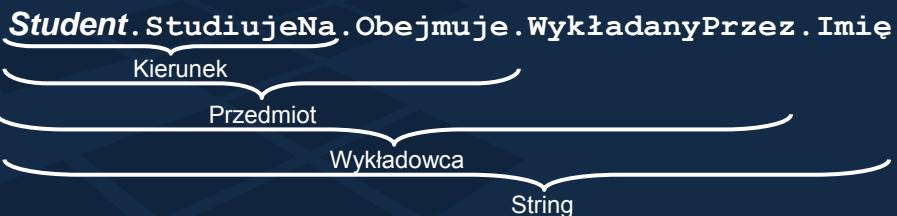
Jednym z rozwiązań wydajnej implementacji dużych obiektów jest ich organizacja w postaci rzadkiego indeksu o strukturze B-drzewa. Atrybutem na którym jest zakładany taki indeks jest numer pozycji danego bajtu w ciągu wszystkich bajtów składających się na duży obiekt. Liściami indeksu są strony dyskowe składające się na duży obiekt. Węzły wewnętrzne indeksu nie przechowują zawartości obiektu, lecz służą do wyszukiwania bajtów o określonej lokalizacji. Dla zagwarantowania wydajności operacji modyfikacji wypełnienie stron dyskowych nie jest maksymalizowane. Tak jak w zwykłych B-drzewach algorytmy utrzymania indeksu gwarantują wypełnienie liści w granicach od 50 do 100%. Dzięki temu operacje powiększania dużego obiektu przez wstawienie w określonej pozycji nowego podciągu bajtów lub usunięcie z obiektu podciągu o określonej pozycji i długości, wymagają jedynie lokalnych rekonstrukcji ograniczonej liczby stron dyskowych.

Opisywana struktura została zilustrowana na slajdzie. Przedstawiony duży obiekt składa się z sześciu stron dyskowych. Maksymalny rozmiar stron to 512 bajtów. Strony te zawierają kolejne fragmenty całego obiektu o długościach odpowiednio: 450, 500, 379, 400, 510 i 496 bajtów. Dwa wyższe poziomy indeksu składają się z trzech węzłów wewnętrznych. Na slajdzie pokazano operację wyszukiwania pojedynczego bajtu, który jest 1800 bajtem obiektu. W korzeniu indeksu pozycja ta jest przeliczona na pozycję w prawym poddrzewie indeksu. Lewe poddrzewo indeksuje bajty począwszy od 1 do 1329. Stąd względna pozycja szukanego bajtu w prawym poddrzewie jest różnicą bezwzględnej pozycji szukanego bajtu: $n=1800$ i liczby bajtów znajdujących się w lewym poddrzewie. Przeliczone względna pozycja w prawym poddrzewie jest równa $n'=461$. Z kolejnego przeliczenia w prawym węźle wewnętrznym drugiego poziomu wynika, że poszukiwany bajt jest 61 bajtem piątej strony dyskowej całego obiektu.



Indeksowanie wyrażeń ścieżkowych

Znajdź imiona wykładowców, wykładających przedmioty przypisane do kierunku, na którym studiuje dany student.



W celu przyśpieszenia nawigacji wzdłuż długich ścieżek:

- Materializacja kompletnych ścieżek – relacje ASR
- Hierarchia kombinacji ścieżek binarnych

Obiektowe bazy danych – Implementacja obiektowych baz danych (16)

Ścieżki są wieloczłonowymi wyrażeniami opisującymi nawigację wzdłuż powiązań między obiektami, w głąb atrybutów złożonych i do wyników metod bezparametrowych. Wartościami wyrażeń ścieżkowych są wartości proste lub obiekty, których typ jest określony przez ostatni element ścieżki. Na slajdzie pokazano przykład wyrażenia ścieżkowego, które dla uczelianej bazy danych pozwala znaleźć imiona wykładowców, którzy wykładają przedmioty na kierunku, na którym studiuje dany student. Kolejne człony zdefiniowanej ścieżki opisują przejścia między kolejnymi zbiorami obiektów: kierunki, na których studiuje student, przedmioty, które są wykładowane na tych kierunkach, wykładowcy, którzy wykładają te przedmioty i na końcu zbiór imion tych wykładowców.

Szybka nawigacja wzdłuż długich wyrażeń ścieżkowych wymaga wydajnej implementacji struktur dostępu do wartości wyrażeń ścieżkowych. Znane są dwa rozwiązania tego problemu. Pierwsze polegające na materializacji kompletnych ścieżek dostępu przez zapamiętanie wszystkich potencjalnych dróg nawigacji ze zbioru obiektów reprezentujących pierwszy człon ścieżki do zbioru obiektów osiągalnego za pomocą ścieżki i reprezentujących ostatni człon ścieżki.

Zmaterializowane ścieżki są składowane w tak zwanych relacjach ASR (ang. Access Support Relations). Drugie rozwiązanie polega na materializacji hierarchii wszystkich kombinacji ścieżek binarnych w wyrażeniu ścieżkowym pamiętających tylko początki i końce ścieżek, bez członów pośrednich. Obydwa rozwiązania dopuszczają by definiowane ścieżki obejmowały nawigacje wzdłuż związków wielokrotnych i atrybutów wielwartościowych.



Relacje ASR

Dana baza obiektów:

<id1,'Józef','Nowak',21,{id2,id3}> :	Student
<id2,'Informatyka',5,{id1},{id4,id5}> :	Kierunek
<id3,'Zarządzanie',5,{id1},{}> :	Kierunek
<id4,'Bazy danych',45,{id2},{id6}> :	Przedmiot
<id5,'Grafika',30,{id2},{id7}> :	Przedmiot
<id6,'Jan','Tarzan',{id4}> :	Wykładowca
<id7,'Henryk','Nowek',{id5}> :	Wykładowca
<id8,'Józef','Buła',{}> :	Wykładowca

Relacja ASR:

S ₀ :OID _{student}	S ₁ :OID _{kierunek}	S ₂ :OID _{przedmiot}	S ₃ :OID _{wykładowca}	S ₄ :string
id1	id2	id4	id6	'Jan'
id1	id2	id5	id7	'Henryk'

Obiektowe bazy danych – Implementacja obiektowych baz danych (17)

Na slajdzie pokazano przykładową bazę danych zawierającą cztery kolekcje obiektów w uczelnianej bazie danych: studentów, kierunki studiów, wykładane przedmioty i wykładowców. Z przedstawionego stanu bazy danych wynika, że student Józef Nowak studiuje równolegle na dwóch kierunkach studiów: Informatyce i Zarządzaniu. Na kierunku Informatyka są wykładowane dwa przedmioty: Bazy danych i Grafika. Wykładowca Jan Tarzan wykłada Bazy danych, a wykładowca Henryk Nowek – grafikę.

Na dole slajdu pokazano przykładową strukturę relacji ASR materializującą ścieżki nawigujące od Studentów, przez Kierunki studiów, wykładane przedmioty i wykładowców do imion wykładowców. Są tylko dwie kompletne ścieżki. Pierwsza prowadzi od studenta Józefa Nowaka przez kierunek studiów Informatyka, przedmiot Bazy danych, wykładowcę Jana Tarzana do imienia Jan. Początkiem drugiej ścieżki jest również student Józef Nowak. Prowadzi ona przez kierunek studiów Informatyka, przedmiot Grafiką, wykładowcę Henryka Nowka, do imienia wykładowcy: Henryk.

Technologia relacji ASR przewiduje możliwość pamiętania również niekompletnych ścieżek. W podanym przykładzie niekompletną ścieżką jest ścieżka prowadząca od studenta Józefa Nowaka i kończąca się na kierunku Zarządzania. Kierunek ten nie przypisanych żadnych przedmiotów, więc ścieżka kończy się przedwcześnie.

Relacje ASR pozwalają na wydajną nawigację wzdłuż wszystkich ścieżek, które są fragmentami maksymalnej ścieżki zdefiniowanej relacji ASR. Na przykład, relacja ASR może być zastosowana dla przyśpieszenia nawigacji wzdłuż ścieżki wyszukującej imion wykładowców pracujących na kierunku Informatyka, która ta ścieżka jest częścią maksymalnej ścieżki zdefiniowanej w relacji ASR.

Dla przyśpieszenia dostępu do relacji ASR na obydwu końcach relacji, to jest na pierwszym i ostatnim atrybutem relacji są zakładane dotykowe struktury fizyczne, na przykład indeksy typu B-drzewo.

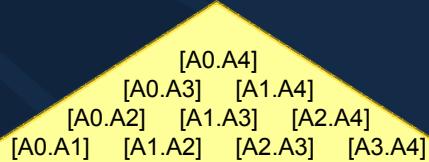


Hierarchia ścieżek binarnych

Składowanie pierwszego i ostatniego elementu całej ścieżki
oraz licznika połączeń

S ₀ :OID _{student}	S ₄ :string	Count
id1	'Jan'	1
id1	'Henryk'	1

Hierarchia wszystkich kombinacji ścieżek binarnych



Obiektowe bazy danych – Implementacja obiektowych baz danych (18)

Alternatywą dla relacji ASR są hierarchie kombinacji ścieżek binarnych. W rozwiązaniu tym, w binarnej relacji ścieżek jest materializowany tylko pierwszy i ostatni element ścieżki. Ponieważ może być wiele ścieżek prowadzących od tego samego źródła do tego samego celu ścieżki, dodatkowo jest pamiętana liczba tych ścieżek.

Na slajdzie pokazano przykład relacji binarnej dla bazy danych z poprzedniego slajdu. Relacja ta będzie zawierała dwie krotki reprezentujące ścieżki prowadzące od studenta Józefa Nowaka do imion jego wykładowców. Obydwie ścieżki są jednokrotne.

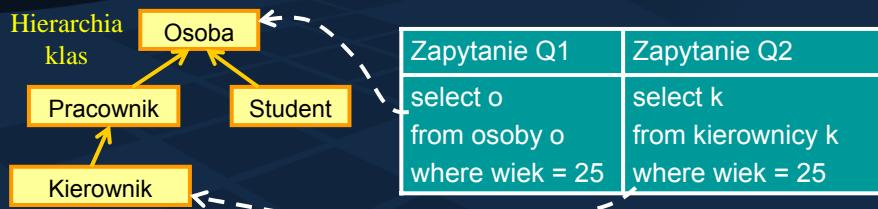
Zwróćmy uwagę, że pokazana relacja binarna pozwoli na przyspieszenie nawigacji jedynie wzduż kompletnych ścieżek. Przechowywane w niej dane będą nieprzydatne dla przyspieszenia nawigacji w zapytaniach zawierających jedynie fragment maksymalnej ścieżki. Dlatego kompletne rozwiązanie obejmuje całą hierarchię relacji binarnych dla fragmentów ścieżek. W ogólności, hierarchia ta nie musi być kompletna. Wystarczy, że będzie ona zawierać tylko i wyłącznie relacje binarne, które będą przydatne w konkretnym zastosowaniu, dla określonego podzbioru wszystkich potencjalnych ścieżek zapytań.

Przykład na slajdzie pokazuje pełną hierarchię relacji binarnych dla maksymalnej ścieżki składającej się z pięciu elementów: A0.A1.A2.A3.A4. Najwyższy poziom hierarchii zawiera relacje binarne dla wszystkich ścieżek długości dwóch elementów, które są fragmentem ścieżki maksymalnej. Następny poziom zawiera relacje binarne dla wszystkich ścieżek o długości trzech elementów. Najniższy poziom zawiera relację binarną dla maksymalnej ścieżki.

Tak jak dla relacji ASR również, w wypadku hierarchii relacji binarnych na każdej relacji zakłada się dwukierunkowe szybkie struktury dostępu.



Indeksowanie hierarchii rozszerzeń klas



Zastosowanie klasycznych indeksów:

- Jeden indeks dla wszystkich podzbiorów w hierarchii - wydajna realizacja zapytań klasy Q1, niewydajna zapytań klasy Q2.
- Osobne indeksy dla każdego podzbioru – wydajna realizacja zapytań klasy Q2, niewydajna zapytań klasy Q1.

Obiektowe bazy danych – Implementacja obiektowych baz danych (19)

W obiektowych bazach danych typowe są zapytania wykonywane na heterogenicznych kolekcjach obiektów. Przykład takich zapytań przedstawiono na slajdzie. Zapytania adresują heterogeniczny zbiór osób obejmujący: kierowników, pracowników, którzy nie są kierownikami, studentów i pozostałe osoby, które nie są ani pracownikami, ani studentami. Zapytanie Q1 wyszukuje w tej heterogenicznej kolekcji, osób w wieku poniżej 30 lat. Wynikiem zapytania mogą być kierownicy, pracownicy, którzy nie są kierownikami, studenci oraz osoby nie będące ani pracownikami, ani studentami. Z kolei zapytanie Q2 dotyczy wyspecjalizowanego podzbioru osób, które pracują na stanowisku kierowniczym.

Dostępne są dwie alternatywy zastosowania klasycznych indeksów dla przyśpieszenia wykonywania przedstawionych zapytań. Pierwsze rozwiązanie polega na utworzeniu jednego wspólnego indeksu dla wszystkich podzbiorów w hierarchii. Taki indeks będzie przydatny dla efektywnej realizacji zapytań klasy Q1, natomiast nie będzie przydatny dla zapytań klasy Q2. W tym drugim przypadku duża część wskazywanych przez indeks obiektów spełniających zadane kryterium wiekowe, nie będzie spełniała kryterium typu danych. Ponieważ indeks wskazuje również osoby, które nie są kierownikami.

Drugie potencjalne rozwiązanie polega na utworzeniu osobnych indeksów dla każdego podzbioru obiektów. Przeciwieństwo niż dla poprzedniego rozwiązania, zbiór indeksów będzie dobrze wspierać wydajną realizację zapytań klasy Q2, a znacznie gorzej będzie wspierać zapytania klasy Q1. Znalezienie wszystkich osób w wieku 25 lat będzie wymagało przejrzenia wszystkich czterech indeksów.

Z powyższej analizy wynika, że potrzebne są nowe rozwiązania, które będą dedykowane dla heterogenicznych kolekcji danych i, które będą przydatne dla dowolnej klasy selektywnych zapytań na takiej kolekcji.



Indeksowanie hierarchii rozszerzeń klas

Nowe wyspecjalizowane rodzaje indeksów:

- Indeks hierarchii klas

... klucz =25 Pracownicy:{p₁₁, p₁₂, ...} Studenci:{p₂₁, p₂₂, ...} ... klucz =26 ...

- Indeks h-drzewo



Obiektowe bazy danych – Implementacja obiektowych baz danych (20)

Na slajdzie przedstawiono dwie dedykowane metody wydajnego wyszukiwania obiektów w hierarchiach rozszerzeń klas. Obydwie polegają na modyfikacjach struktury indeksu typu B-drzewo.

Pierwsze rozwiązanie polega na rozszerzeniu struktury liści indeksu w celu przechowywania dodatkowej informacji o typie każdego indeksowanego obiektu. Ogólna struktura informacji przechowywanej w liściach indeksu jest następująca. Z każdym kluczem jest związana lista zbiorów wskaźników na obiekty o danej wartości klucza. Elementy listy odpowiadają poszczególnym podzbiorom danych. Na slajdzie pokazano kawałek liścia takiego indeksu założonego na atrybutie „wiek”. Z kluczem o wartości równej 25 lat jest związana lista zbiorów wskaźników „p”. Wskaźniki są pogrupowane w zależności od klasy obiektów których dotyczą. Dwie pierwsze grupy to wskaźniki na pracowników i studentów.

Drugim rozwiązaniem są tak zwiane h-drzewa. H-drzewo jest hierarchią B-drzew odpowiadającą hierarchii podzbiorów. Korzeniem tej hierarchii jest indeks odpowiadający klasie bazowej całej hierarchii klas, z którego są osiągalne wszystkie obiekty. Z indeksów ulokowanych niżej w hierarchii osiągalne są tylko obiekty należące do odpowiedniej pod-hierarchii klas. Indeksy ulokowane liściach hierarchii adresują obiekty będące wystąpieniami pojedynczych klas. Wynika stąd, że zakresy zasięgów indeksów z różnych poziomów hierarchii pokrywają się. Natomiast zakresy zasięgów indeksów znajdujących się na tym poziomie hierarchii są rozłączne. W pośrednich węzłach indeksów występują dwa rodzaje wskaźników: wskaźniki na węzły niższego poziomu w tym samym indeksie i wskaźniki na węzły w innym indeksie znajdującym się na niższym poziomie hierarchii indeksów.

Przykładowe H-drzewo zostało pokazane na slajdzie. Korzeniem hierarchii indeksów jest indeks odpowiadający klasie „Osoba”. Z tego indeksu są osiągalne zarówno wystąpienia klasy „Osoba”, jak i wystąpienia klasy „Student”. Indeksem niższego poziomu jest indeks odpowiadający klasie „Student”, która jest podklassą klasy „Osoba”. Z tego indeksu są osiągalne jedynie obiekty, które są wystąpieniami klasy „Student”. W indeksie odpowiadającym klasie „Osoba” znajduje się węzeł oznaczony przez „N”, który zawiera wskaźniki na inne węzły tego indeksu i dodatkowo zawiera wskaźnik na węzeł oznaczony przez „n”, który znajduje się wewnątrz indeksu odpowiadającego klasie „Student”. Na rysunku pokazano przykładowe ścieżki wyszukiwania. Dwie z nich zamkają się wewnątrz indeksów składowych H-drzewa. Środkowa ścieżka pokazuje nawigację między różnymi indeksami hierarchii.



Związki wielokrotne i atrybuty wielowartościowe

```
class Wielokąt {
    attribute set<Punkt> wierzchołki;
}
class Koło {
    attribute Punkt środek;
    attribute Float promień;
}
```

Znajdź pary kół i wielokątów, dla których środek koła pokrywa się z jednym z wierzchołków wielokąta.

```
select k, w
from k in Koła, w in Wielokąty
where k.środek <= w.wierzchołki
```

Operator przynależności
elementu do zbioru

Obiektowe bazy danych – Implementacja obiektowych baz danych (22)

Obiektowy model danych pozwala bezpośrednio modelować struktury zbiorowe, takie jak związki wielokrotne i atrybuty wielowartościowe. Ze strukturami zbiorowymi skojarzone są operacje logiczne na zbiorach, na przykład niepustego przecięcia, podzbioru lub przynależności elementu do zbioru.

Na slajdzie pokazano przykład operacji na zbiorach dotyczącej atrybutów wielowartościowych. Definicja klasy Wielokąt zawiera atrybut wielowartościowy będący zbiorem punktów, który reprezentuje zbiór wierzchołków wielokąta. Definicja klasy Koło obejmuje z kolei jednowartościowy atrybut typu Punkt, który reprezentuje środek koła.

Pokazane na dole slajdu zapytanie wyszukuje w bazie danych pary Koło i Wielokąt takie, że środek koła pokrywa się z jednym z wierzchołków wielokąta. Zapytanie to zawiera operator zbiorowy, testowania przynależności elementu do zbioru.

Fizyczna implementacja zapytań zawierających operację łączenia kolekcji obiektów, w której warunek połączniowy zawiera operacje porównania zbiorów jest bardzo czasochłonna z dwóch podstawowych powodów. Po pierwsze, realizacja takich operacji wymaga porównania wszystkich elementów dwóch łączonych kolekcji obiektów, i dodatkowo dla każdej pary porównywanych obiektów porównania wszystkich elementów zbiorów, to jest atrybutów wielowartościowych lub związków wielokrotnych. Dla dwóch kolekcji obiektów o rozmiarach M i N, i dla średniej liczebności atrybutów zbiorowych lub związków wielokrotnych równej O i P, złożoność kombinatoryczna operacji połączenia wynosi: $M \cdot N \cdot O \cdot P$. Drugą przyczyną czasochłonności takich operacji połączeń jest niemożność zastosowania w tym wypadku szybszych metod wykonywania połączeń, takich jak metody sort-merge lub hash-join. Metoda sort-merge nie może być zastosowana w tym wypadku ze względu na brak relacji porządkującej zbiór zbiorów wartości lub związków. Natomiast metoda hash-join nie może być zastosowana ze względu na jej ograniczenie do warunków połączenia zawierających operator równości.



Sygnatury zbiorów

- Każdy element zbioru jest odwzorowany przez unikalną sygnaturę
- Za pomocą sumowania logicznego sygnatury wszystkich elementów zbioru są składane w pojedynczą sygnaturę aproksymującą cały zbiór
- Dla operatora podzbioru sygnatury zbiorów muszą spełniać następującą zależność:

$$s \subseteq t \Rightarrow \text{sig}(s) \& \sim\text{sig}(t) = 0$$

$\text{Sig}(\text{Punkt A (4.35, 11.26)}) \rightarrow 00010001$

$\text{Sig}(\text{Punkt B (7.10, 12.05)}) \rightarrow 10001000$

$\text{Sig}(\{\text{Punkt A, Punkt B}\}) \rightarrow 10011001$

Obiektowe bazy danych – Implementacja obiektowych baz danych (23)

Metoda, która pozwala na wydajną realizację operacji porównania zbiorów zastępuje zbory wartości lub związków specjalnymi wartościami numerycznymi nazywanymi sygnaturami.

Dla każdego elementu zbiorów składowanych w obiektach, czyli pojedynczej wartości atrybutu wielowartościowego lub pojedynczej referencji związku wielokrotnego, jest wyznaczana unikalna sygnatura. Równość elementów oznacza równość ich sygnatur.

Następnie za pomocą logicznego sumowania sygnatur wszystkich elementów zbiorów są tworzone zbiorcze sygnatury dla całych zbiorów. Sumaryczne sygnatury są prostymi wartościami liczbowymi, dzięki czemu można je łatwo porównywać i porządkować, co pozwala na zastosowanie wydajnych metod łączenia, takich jak wspomniane metody sort-merge lub hash-join.

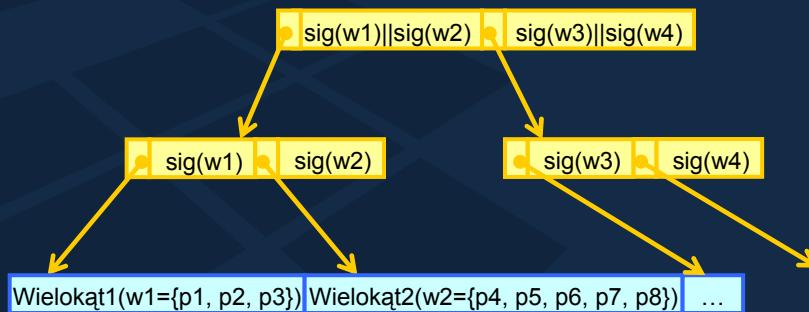
Specyficzne zależności zachodzące między wartościami sygnatur zbiorów oznaczają określone zależności zachodzące między samymi zbiorami. Na slajdzie pokazano zależność, która musi zachodzić między sygnaturami zbiorów, by między tymi zbiorami zachodziła relacja podzbioru. W tym konkretnym wypadku zależność, w której iloczyn logiczny sygnatury zbioru „s”, z negacją sygnatury zbioru „t” jest równa zero, oznacza, że zbiór „t” jest podzbiorem zbioru „s”.

Na dole slajdu pokazano przykład utworzenia sygnatury dla zbioru punktów A i B, którym przypisano dwie dowolnie ustalone sygnatury. Dla uzyskanej sygnatury zbioru punktów A i B można zweryfikować na podstawie zdefiniowanej wyżej zależności, zastosowanej dla sygnatury zbioru punktów A i B oraz sygnatury punktu A, że zbiór punktów A i B zawiera w sobie punkt A.



Indeks RD-drzewo

Wartościami indeksowanymi przez indeks RD-drzewo są sygnatury związków wielokrotnych lub atrybutów wielowartościowych.



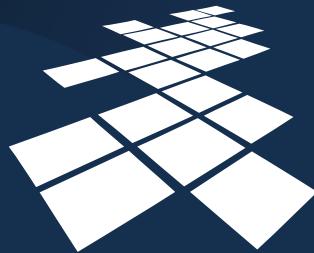
Obiektowe bazy danych – Implementacja obiektowych baz danych (24)

Dla wydajnej realizacji zapytań odwołujących się do atrybutów wielowartościowych lub związków wielokrotnych stosuje się dodatkowe struktury danych bazujące na sygnaturach zbiorów. Jedną z takich struktur jest indeks nazywany RD-drzewem od angielskiego Russian Doll. Indeks ten jest zakładany na sygnaturach atrybutów wielowartościowych lub związków wielokrotnych. Indeks ten ma strukturę hierarchiczną i jest wzorowany na B-drzewie. Wartościami kluczyc przechowywanymi w węzłach wewnętrznych są sygnatury utworzone przez operację sumy logicznej wszystkich sygnatur przechowywanych w węźle niższego poziomu wskazywanym przez wskaźnik skojarzony z daną zbiorczą sygnaturą.

Na rysunku pokazano fragment dwupoziomowego RD-drzewa założonego na atrybucie Wierzchołki obiektów z kolekcji Wielokąty. W liściach indeksu znajdują się sygnatury utworzone dla zbiorów wierzchołków poszczególnych wielokątów. Na przykład sygnatura $\text{sig}(w2)$ reprezentuje zbiór pięciu wierzchołków: $p4, p5, p6, p7$ i $p8$ wielokąta o nazwie Wielokąt2. Sygnatury w korzeniu drzewa są utworzone z sygnatur przechowywanych w liściach drzewa. Na przykład pierwsza sygnatura składowana w korzeniu jest sumą logiczną sygnatur dwóch zbiorów wierzchołków $w1$ i $w2$.

Multimedialne bazy danych

Wykład prowadzi:
Marek Wojciechowski



UCZELNIA
ONLINE

Multimedialne bazy danych

Wykład poświęcony jest problematyce multimedialnych baz danych, ich specyfice oraz technikom wyszukiwania, składowania i prezentacji danych wykorzystywanym w ich kontekście.

Podstawowym celem niniejszego wykładu jest zapoznanie studenta z problematyką multimedialnych baz danych, ze szczególnym naciskiem na opis i wyszukiwanie treści multimedialnych.

Do zrozumienia treści wykładu niezbędna jest znajomość systemów baz danych i języka SQL.



Plan wykładu

- Wprowadzenie: multimedia, podstawowe definicje
- Specyfika systemów multimedialnych baz danych
- Metadane
- Modele danych dla multimedialnych baz danych
- MPEG-7
- Przetwarzanie zapytań w multimedialnych bazach danych
- Wyszukiwanie w oparciu o zawartość
- Składowanie danych multimedialnych
- Prezentacja i transmisja danych multimedialnych

Multimedialne bazy danych (2)

Wykład rozpocznie się od wprowadzenia do multimediiów i przedstawienia podstawowych definicji. Następnie wskazane będą cechy i wymagania stawiane systemom multimedialnych baz danych, które odróżniają je od tradycyjnych systemów baz danych. W kolejnej części wykładu omówione będą metadane, ich klasyfikacja i rola w opisie zawartości multimedialnej. Kolejny temat to modele danych dla multimedialnych baz danych. Następnie omówiony będzie standard opisu zawartości multimedialnej MPEG-7. Po nim przedstawiona będzie problematyka przetwarzania zapytań w multimedialnych bazach danych, ze szczególnym uwzględnieniem wyszukiwania obrazów w oparciu o zawartość (ang. Content-Based Image Retrieval). Na zakończenie, krótko nakreślone będą problemy składowania, prezentacji i transmisji danych multimedialnych, ze wskazaniem rozwiązań stosowanych w tym zakresie.



Multimedia – podstawowe definicje

- Obiekt medialny – informacja zrozumiała dla komputera, zakodowana w jednym medium
 - media dyskretne: tekst, obraz
 - media ciągłe: audio, wideo
- Obiekt multimedialny – informacja zrozumiała dla komputera, zakodowana w jednym lub więcej mediów
- Dane multimedialne – zbiór obiektów multimedialnych
- System multimedialny – system komputerowy wspierający wymianę informacji z użytkownikiem za pomocą kilku różnych mediów

Multimedialne bazy danych (3)

Obiekt medialny to informacja zrozumiała dla komputera, zakodowana w jednym medium. Wykorzystywane obecnie media to media wizualne i akustyczne: tekst, obrazy nieruchome, audio i wideo. Systemy wykorzystujące w komunikacji z człowiekiem zapach, smak i dotyk są na razie przedmiotem prac badawczych i prototypowych. Z punktu widzenia prezentacji multimedialnych istotny jest podział na media dyskretne (niezmienne w czasie, statyczne), obejmujące tekst i obrazy nieruchome oraz media ciągłe (zmienne w czasie, dynamiczne), obejmujące audio i wideo.

Obiekt multimedialny to informacja zrozumiała dla komputera, zakodowana w jednym lub więcej mediów. Często dodawany jest jeszcze warunek, że co najmniej jedno medium powinno być nie-alfanumeryczne. W zasadzie termin „multimedia”, z racji przedrostka „multi-”, powinien być używany do opisu danych zakodowanych w więcej niż jednym medium. W praktyce przyjmuje się jednak, że obrazy czy muzyka w postaci cyfrowej również stanowią dane multimedialne, gdyż jest to zgodne z odczuciami użytkowników.

Dane multimedialne można najprościej zdefiniować jako zbiór obiektów multimedialnych.

System multimedialny (wg Meyera-Wegenera) to system komputerowy wspierający wymianę informacji z użytkownikiem za pomocą kilku różnych mediów. W zasadzie brak jest jednoznacznej, „ostrej”, powszechnie akceptowanej definicji systemu multimedialnego.

Przykładowo, istnieje inna definicja mówiąca, że system multimedialny to system komputerowy umożliwiający w sposób zintegrowany tworzenie, przetwarzanie, prezentację i przesyłanie informacji zakodowanych w co najmniej jednym medium ciągłym i jednym dyskretnym.



Charakterystyka typów multimedialnych (1/2)

- Obrazy
 - reprezentowane jako 2-wymiarowa tablica pikseli, gdzie wartość piksela reprezentuje jego kolor
 - zazwyczaj skompresowane (GIF, JPEG, PNG)
- Wideo
 - sekwencja klatek prezentowanych w odpowiednim tempie
 - podzielone na segmenty (fizyczne, logiczne)
 - zazwyczaj skompresowane (MPEG)
 - zazwyczaj uzupełnione o dane audio

Multimedialne bazy danych (4)

Podstawowe typy multimedialnych to obrazy nieruchome, dane wideo, dane audio i złożone obiekty multimedialne.

Obrazy są reprezentowane w postaci 2-wymiarowej tablicy pikseli (ang. pixel – picture element). Wartość przypisana danemu pikselowi reprezentuje jego kolor. Przykładowe liczby bitów przeznaczanych na zapamiętanie koloru to 8 dla obrazów w skali odcieni szarości i 24 dla obrazów kolorowych. Zazwyczaj obrazy są zapisane w postaci skompresowanej w celu redukcji ich rozmiaru. Przykładowe formaty to: GIF i PNG (kompresja bez utraty jakości) i JPEG (kompresja z utratą jakości, wykorzystująca właściwości ludzkiego zmysłu wzroku).

Dane wideo mają postać sekwencji klatek (ang. frame), gdzie każda klatka jest nieruchomym obrazem. Klatki muszą być prezentowane w odpowiednim ustalonym tempie, np. 25 klatek na sekundę. Sekwencja wideo może być podzielona na segmenty. Podstawowy podział to podział na segmenty fizyczne, gdzie każdy segment to sekwencja klatek pochodząca z jednego ujęcia kamery. Innym możliwym podziałem jest podział na segmenty logiczne reprezentujące fragmenty wideo o konkretnym znaczeniu. Dane wideo w postaci nieskompresowanej, ze względu na swój rozmiar, mogą być wykorzystywane jedynie na etapie profesjonalnej obróbki. Udostępniane użytkownikom w postaci cyfrowej są zawsze skompresowane.

Kompresja wideo wykorzystuje przestrenną i czasową nadmiarowość danych. Redukcja nadmiarowości przestrzennej polega na kompresji klatek metodą dla obrazów statycznych (np. JPEG). Redukcja nadmiarowości czasowej opiera się na reprezentacji niektórych klatek poprzez zapamiętanie zmian w stosunku do klatki poprzedzającej lub interpolację klatki z wykorzystaniem jednej klatki poprzedzającej i jednej następującej.

Podstawowe standardy kompresji wideo zostały opracowane przez grupę MPEG (Moving Pictures Experts Group): MPEG-1 zapewniający jakość porównywalną z VHS, wykorzystywany w VCD; MPEG-2 wykorzystywany w DVD i telewizji wysokiej rozdzielczości HDTV; MPEG-4 wykorzystywany w dystrybucji wideo poprzez Internet.

Dane wideo zazwyczaj uzupełnione są o dane audio, tworząc złożony obiekt multimedialny.



Charakterystyka typów multimedialnych (2/2)

- Audio
 - cyfrowy zapis muzyki lub mowy
 - wynik próbkowania sygnału analogowego (PCM, MP3) lub dźwięk syntezyowany (MIDI)
- Złożone dane multimedialne
 - filmy wideo z dźwiękiem (i opcjonalnie napisami)
 - prezentacje multimedialne
 - poszczególne media zapisane niezależnie lub razem tworząc nowy format

Multimedialne bazy danych (5)

Dane audio to muzyka i mowa. Mowa jest szczególnym przypadkiem danych audio, którym towarzyszą inne metadane niż muzyce i często również tekstowy zapis treści. Dane audio w postaci cyfrowej mogą być wynikiem próbkowania sygnału analogowego (np. z mikrofonu) albo być syntezowane na podstawie zapisu podobnego do nutowego. Dane będące wynikiem próbkowania mogą być zapisane w postaci nieskompresowanej (format PCM) lub skompresowanej (np. MP3 – kompresja z utratą jakości, wykorzystująca właściwości ludzkiego zmysłu słuchu). Najpopularniejszym formatem dla dźwięku syntezowanego jest MIDI.

Oprócz samodzielnych obrazów, audio i wideo, w praktyce często występują dane rzeczywiście multimedialne, czyli zakodowane w kilku różnych mediach. Typowym przykładem są filmy, w których sekwencji wideo towarzyszy zsynchronizowana z nią ścieżka dźwiękowa, a opcjonalnie też napisy (ang. subtitles). Innym przykładem danych integrujących wiele mediów są prezentacje multimedialne, gdzie tekstowo-obrazkowym slajdom towarzyszy dźwięk z komentarzem i opcjonalnie animacje. W przypadku złożonych obiektów multimedialnych, składające się na nie media mogą być składane niezależnie lub razem, tworząc nowy format. Przykładem złożonego formatu jest choćby powszechnie znany format AVI, w którym przeplecione ze sobą są niezależnie kompresowane wideo i audio.



Przyczyny gwałtownego rozwoju systemów multimedialnych

- Rzeczywistość postrzegana przez człowieka ma charakter multimedialny
- Dostępność mocy obliczeniowej
- Dostępność urządzeń do składowania danych o dużej pojemności
- Szybkie sieci komputerowe
- Efektywne standardy kodowania i transmisji
- Rozwój technologii mobilnych

Multimedialne bazy danych (6)

Podstawową motywacją dla systemów multimedialnych jest niepodważalny fakt, że rzeczywistość postrzegana przez człowieka ma charakter multimedialny. Na obraz rzeczywistości składają się informacje przekazane za pomocą wielu mediów, odbieranych za pomocą zmysłów (obraz, dźwięk, smak, zapach, dotyk). Zadaniem multimedialnych systemów informatycznych jest imitacja otaczającej człowieka rzeczywistości i sposobu jej postrzegania.

Jednakże dopiero od niedawna istnieją warunki do tworzenia systemów multimedialnych powszechnego użytku ze względu na:

- (a) dostępność tanich stacji roboczych zdolnych do przetwarzania danych multimedialnych (wydajne procesory i sprzętowe kodery/dekodery);
- (b) dostępność tanich urządzeń i nośników do składowania danych o dużej pojemności;
- (c) dostęp do szybkich sieci komputerowych (wzrost prędkości transmisji, rozwój infrastruktury, spadek cen nośników np. światłowodów, szybkie technologie transmisji danych np. ATM);
- (d) opracowanie efektywnych standardów kodowania i transmisji danych;
- (e) rozwój technologii mobilnych (telefony komórkowe umożliwiające odbiór zawartości multimedialnej, PDA, przewidywane nadanie UMTS).



Zastosowania systemów multimedialnych

- Wideo na żądanie
- Muzyczne bazy danych
- Bazy danych z obrazami (nauka, sztuka, medycyna)
- Edukacja
- Walka z przestępcością i terroryzmem

Multimedialne bazy danych (7)

Przykładowe zastosowania systemów multimedialnych to:

- (a) wideo na żądanie (ang. video on demand);
- (b) muzyczne bazy danych;
- (c) bazy danych z obrazami, przykładowo: kolekcje zdjęć rentgenowskich do wspierania diagnostyki medycznej, kolekcje fotografii dzieł malarskich, wirtualne galerie, itp.;
- (d) edukacja: interaktywne kursy multimedialne, zdalne nauczanie, instrukcje obsługi, itp.;
- (e) walka z przestępcością i terroryzmem: multimedialne bazy danych o przestępcaach i terrorystach wykorzystane w połączeniu z infrastrukturą do monitoringu i technikami rozpoznawania obrazu, w tym ludzkich twarzy.

W większości zastosowań systemów multimedialnych mamy do czynienia z dużą kolekcją danych multimedialnych, które muszą być w sposób efektywny składowane, przeszukiwane i udostępniane. Stanowi to motywację dla ważnej klasy systemów multimedialnych – systemów multimedialnych baz danych.



SMBD = SZMBD + MBD

- SMBD – System multimedialnej bazy danych
- SZMBD – System zarządzania multimedialną bazą danych
 - wysoce wydajny SZBD
 - obsługujący dane alfanumeryczne i multimedialne
- MBD – Multimedialna baza danych
 - baza danych o dużej pojemności
 - zawierająca dane multimedialne
- SMBD a bazy danych z multimedialną zawartością

Multimedialne bazy danych (8)

System multimedialnej bazy danych (SMBD) składa się z bazy danych o dużej pojemności (MBD) z zawartością multimedialną i wysoce wydajnego systemu zarządzania bazą danych (SZBD), który wspiera i obsługuje obok alfanumerycznych typów danych również dane multimedialne w zakresie ich składowania, wyszukiwania i przetwarzania zapytań.

Należy zwrócić uwagę, że systemy oparte o bazę danych z multimedialną zawartością, nierealizujące funkcjonalności SZBD i wyszukiwania w oparciu o zawartość nie stanowią jeszcze systemów multimedialnych baz danych. Przykłady baz danych z multimedialną zawartością to:

- (a) płyty CD-ROM z multimediami,
- (b) katalogi obrazków z ikonkami prowadzącymi do obrazów w większej rozdzielcości,
- (c) systemy video on demand z wyszukiwaniem wg typowych słów kluczowych (tytuł, obsada, itp.),
- (d) systemy zarządzania dużymi zbiorami złożonych dokumentów.



Systemy zarządzania multimedialną bazą danych (SZMBD)

- SZMBD =
 - SZBD
 - architektury do efektywnego składowania dużych wolumenów danych multimedialnych
 - techniki information retrieval (IR) – wyszukiwanie w oparciu o zawartość
- SZMBD integruje wiele technologii
- Podzbiorem funkcjonalności SZMBD jest funkcjonalność tradycyjnego SZBD

Multimedialne bazy danych (9)

System zarządzania multimedialną bazą danych (SZMBD) to wysoce wydajny system zarządzania bazą danych (SZBD), wykorzystujący architektury do efektywnego składowania dużych wolumenów danych multimedialnych oraz techniki Information Retrieval (IR) do wyszukiwania w oparciu o zawartość.

Systemy zarządzania multimedialnymi bazami danych integrują wiele technologii, w tym: bazy danych, Information Retrieval i przetwarzanie sygnałów. Podzbiorem ich funkcjonalności jest funkcjonalność tradycyjnych SZBD. W szczególności, system zarządzania multimedialną bazą danych, tak jak każdy system zarządzania bazą danych powinien oferować:

- (a) przetwarzanie transakcyjne,
- (b) zarządzanie współbieżnością,
- (c) odtwarzanie po awarii,
- (d) wersjonowanie danych,
- (e) ochronę zawartości bazy danych przed nieuprawnionymi użytkownikami i niedozwolonymi operacjami.



Specyfika systemów multimedialnych baz danych

- Duży rozmiar danych
- Złożone struktury danych
- Zaawansowane przetwarzanie danych
- Zapytania różniące się od tradycyjnych
- Problemy prezentacji danych

Multimedialne bazy danych (10)

Właściwości, które sprawiają, że implementacja systemów zarządzania multimedialną bazą danych wymaga nowych rozwiązań znanych w systemach baz danych problemów to:

- (a) duży rozmiar danych - Przykładowo, pojedynczy film wideo może zajmować kilka gigabajtów. Dlatego też systemy zarządzania multimedialną bazą danych oprócz składowania danych multimedialnych wewnętrz bazy danych w postaci dużych obiektów binarnych (ang. binary large object) muszą pozwalać na składowanie danych poza bazą danych (np. w systemie plików serwera, na serwerze WWW, na dedykowanych serwerach audio/wideo przygotowanych do transmisji strumieniowej). Duży rozmiar danych może też skłaniać do wykorzystania hierarchicznych struktur składowania oraz specjalnych organizacji danych na urządzeniach pamięci masowej.
- (b) złożone struktury danych - Dane multimedialne są w praktyce uzupełnione opisem formatu i zawartości. Dla umożliwienia ścisłego wiązania metadanych z danymi multimedialnymi bardziej odpowiedni niż "czysty" model relacyjny jest obiektowy lub obiektowo-relacyjny model danych.
- (c) zaawansowane przetwarzanie danych - Systemy zarządzania multimedialną bazą danych powinny realizować takie operacje jak np. konwersja formatu, skalowanie i inne transformacje obrazów. Dla wygody użytkowników i bezpieczeństwa danych ważne jest, aby realizacja tego typu operacji nie wymagała wydobycia danych z bazy danych i wykorzystania do ich obróbki innego oprogramowania.



Specyfika systemów multimedialnych baz danych (cd.)

- Duży rozmiar danych
- Złożone struktury danych
- Zaawansowane przetwarzanie danych
- Zapytania różniące się od tradycyjnych
- Problemy prezentacji danych

Multimedialne bazy danych (11)

(d) zapytania różniące się od tradycyjnych - Charakterystyczne dla multimedialnych baz danych jest wyszukiwanie oparte o zawartość, rozumiane jako wyszukiwanie, którego kryteria nie odnoszą się do zapisu danych w bazie danych, ale do właściwości wizualnych lub akustycznych, które muszą być wyekstrahowane z obiektu multimedialnego. Kryteria wyszukiwania w oparciu o zawartość najczęściej są specyfikowane poprzez podanie przykładu. Charakterystyczne dla wyszukiwania obiektów podobnych są: możliwość przypisywania wag poszczególnym właściwościom oraz tolerowanie przybliżonych lub niekompletnych wyników.

(e) problemy prezentacji danych - Problem ten dotyczy danych wrażliwych na opóźnienia, czyli sekwencji audio i wideo. Występuje w związku z dużym rozmiarem danych, ograniczoną przepustowością serwera i kanałów transmisyjnych oraz ograniczonym buforem danych klienta. W szczególności dane wideo powinny być prezentowane w oparciu o transmisję strumieniową. Serwer musi być więc przygotowany do strumieniowej transmisji danych, ze współbieżną obsługą wielu klientów z zachowaniem odpowiedniej jakości usług.



Systemy rozproszonych multimedialnych baz danych

- Motywacje:
 - wymagania pamięci i przepustowości sieci
 - często naturalne rozproszenie
- Zalety systemów rozproszonych:
 - efektywność systemu (moc obliczeniowa, większe pojemności, zrównoleglenie pracy)
 - krótsza ścieżka od serwera do użytkownika
- Klasyczne problemy rozproszonych baz danych

Multimedialne bazy danych (12)

W kontekście systemów multimedialnych (w tym multimedialnych baz danych) często rozwija się architektury rozproszone. Są dwie główne motywacje dla multimedialnych systemów rozproszonych. Po pierwsze, systemy skoncentrowane wymagają olbrzymich pamięci masowych do składowania danych i dużych przepustowości sieci do ich przesyłania. Szczególnie dla baz danych wideo może się okazać, że skoncentrowany system udostępniania filmów wideo przy aktualnym stanie technologii i dostępnej infrastrukturze dla konkretnego planowanego obciążenia nie będzie mógł zapewnić odpowiedniej jakości usług. Po drugie, często dane multimedialne stanowiące multimedialną bazę danych są naturalnie rozproszone np. gdy poszczególne typy multimediiów (audio, wideo, obrazy) są składowane na serwerach dedykowanych do udostępniania konkretnego typu danych.

Zaleta systemów rozproszonych to możliwość zwiększania efektywności i pojemności systemu poprzez dodawanie nowych węzłów. Pozwala to zrównoleglić pracę i rozłożyć obciążenie sieci. Ponadto, w połączeniu z odpowiednim rozmieszczeniem i replikacją danych na poszczególnych węzłach systemu można skrócić ścieżkę między serwerem a użytkownikiem.

Systemy rozproszonych multimedialnych baz danych wymagają jednak rozwiązywania problemów znanych z klasycznych systemów rozproszonych baz danych takich jak:

- (a) wybór sposobu rozproszenia (alokacji) danych między serwery,
- (b) wybór danych, które mają być replikowane,
- (c) integracja danych z różnych serwerów przy przetwarzaniu zapytań,
- (d) zarządzanie rozproszonymi transakcjami,
- (e) pielęgnacja rozproszonych metadanych.



- Metadane sygnałowe
 - wyodrębnione właściwości sygnału stanowiącego zapis zawartości multimedialnej
- Metadane semantyczne
 - informacje o obiektach, postaciach, zdarzeniach, itp. reprezentowanych przez treści multimedialne
- Metadane zewnętrzne
 - informacje nieodnoszące się bezpośrednio do zawartości medium, tylko do sposobu jego tworzenia, przechowywania, udostępniania, klasyfikacji

Multimedialne bazy danych (13)

Metadane to „dane o danych”. Metadane mają szczególne znaczenie w opisie i wyszukiwaniu danych multimedialnych. Zależnie od ich charakteru, niektóre metadane mogą być automatycznie wyekstrahowane z zawartości binarnej, a niektóre muszą być wprowadzone przez człowieka i towarzyszyć zawartości multimedialnej. Ze względu na charakter metadanych można pokusić się o ich następującą klasyfikację:

Metadane sygnałowe to wyodrębnione właściwości sygnału stanowiącego zapis zawartości multimedialnej. Mogą być stosunkowo łatwo wygenerowane automatycznie w oparciu o specjalne algorytmy. Przykłady metadanych sygnałowych to: histogram kolorów czy tekstura dla obrazów, dynamika ruchu dla sekwencji wideo, melodia dla sekwencji audio.

Metadane semantyczne to metadane opisujące znaczenie danych, czyli zawierające informacje o obiektach, postaciach, zdarzeniach, itp. reprezentowanych przez treści multimedialne.

Metadane semantyczne są trudne bądź niemożliwe do w pełni automatycznego uzyskania.

Powstają one w wyniku tworzenia adnotacji przez człowieka, który może być wspomagany przez narzędzia dokonujące automatycznej segmentacji obrazu, podziału sekwencji wideo na ujęcia itp.

Metadane zewnętrzne to informacje nieodnoszące się bezpośrednio do zawartości medium, tylko do sposobu jego tworzenia, przechowywania, udostępniania, czy klasyfikacji. Przykłady metadanych zewnętrznych to reżyser filmu, nazwa pliku źródłowego, typ kompresji pliku, tytuł zdjęcia, czas i miejsce wykonania zdjęcia.



Rodzaje metadanych - Przykład



Metadane semantyczne:
łąka pełna maków
na dole z prawej kobieta z parasolką i dziecko

Metadane zewnętrzne:

Autor: Claude Monet
Tytuł: Maki w pobliżu
Argenteuil
Styl: impresjonizm
Format: jpeg
Rozmiar: 640 x 480

Metadane sygnałowe:



Multimedialne bazy danych (14)

Slajd przedstawia przykłady poszczególnych rodzajów metadanych dla fotografii dzieła malarstwa w postaci cyfrowej.

Metadane sygnałowe w tym wypadku to histogram reprezentujący udział kolorów w obrazie i macierz pokazująca dominujące kolory w poszczególnych regionach obrazu. Metadane semantyczne obejmują ogólny opis zawartości („łąka pełna maków”) oraz opis najważniejszych obiektów wraz z ich położeniem („na dole z prawej kobieta z parasolką i dziecko”). Metadane zewnętrzne to informacje o autorze, tytule i stylu dzieła oraz informacje o cyfrowej reprezentacji tj. format graficzny i rozmiar w pikselach.



Modele danych dla multimedialnych baz danych

- Model relacyjny
 - nieodpowiedni
 - problemy z reprezentacją złożonych metadanych i wiązaniem ich z zawartością binarną
- Model obiektowy
 - odpowiedni, elastyczny, ale nie odniosł sukcesu
- Model obiektowo-relacyjny
 - odpowiedni, praktyczny kompromis
- Model semistrukturalny (XML)
 - odpowiedni do reprezentacji metadanych

Multimedialne bazy danych (15)

Modele danych wykorzystywane w dzisiejszych systemach baz danych to model relacyjny, model obiektowy, model obiektowo-relacyjny i model semistrukturalny.

Model relacyjny to model, w którym struktury danych mają postać zbioru wzajemnie ze sobą powiązanych relacji (tabel). Czysty model relacyjny nie przewidywał obsługi danych binarnych. Obecne systemy relacyjne obok alfanumerycznych typów danych, oferują również typ BLOB (Binary Large OBject), umożliwiający składowanie dużych obiektów binarnych w komórce tabeli. Mimo tego, model relacyjny nie jest odpowiedni do reprezentacji danych multimedialnych, gdyż oferowana przez niego organizacja danych nie pozwala na reprezentację złożonych metadanych semantycznych. Nie jest możliwe również ścisłe wiązanie metadanych z zawartością binarną.

Model obiektowy stanowi odpowiedź na ubóstwo modelu relacyjnego. Umożliwia modelowanie złożonych struktur i obiektów oraz wiąże dane z operacjami na nich. Model ten jest potencjalnie atrakcyjny dla multimedialów: obiekt multimedialny może być reprezentowany poprzez obiekt w bazie danych, którego jednym z atrybutów jest zawartość binarna, a pozostałe reprezentują wszelkiego rodzaju metadane. Problemem modelu obiektowego jest to, że nie odniosł dużego sukcesu gdyż mimo 20 lat rozwoju ciągle nie zaproponowano satysfakcjonujących rozwiązań problemów: współbieżnego dostępu, optymalizacji zapytań, odtwarzania po awarii, itp.



- Model relacyjny
 - nieodpowiedni
 - problemy z reprezentacją złożonych metadanych i wiązaniem ich z zawartością binarną
- Model obiektowy
 - odpowiedni, elastyczny, ale nie odniosł sukcesu
- Model obiektowo-relacyjny
 - odpowiedni, praktyczny kompromis
- Model semistrukturalny (XML)
 - odpowiedni do reprezentacji metadanych

Multimedialne bazy danych (16)

Model obiektowo-relacyjny to odpowiedź „świata relacyjnego” na model obiektowy. Doczekał się standaryzacji wraz z wersją standardu SQL99. Jest to model relacyjny uzupełniony o: złożone typy danych, dziedziczenie, kolekcje, referencje do obiektów, itp. Model obiektowo-relacyjny jest powszechnie implementowany gdyż większość systemów relacyjnych przeszła ewolucję stając się systemami obiektowo-relacyjnymi. Model ten jest odpowiedni dla multimedialnych baz danych i stał się podstawą dla standardu SQL/MM, standaryzującego sposób reprezentacji i przetwarzania zaawansowanych rodzajów danych (w tym multimediiów) w bazach danych.

Model semistrukturalny umożliwia reprezentację danych posiadających pewną strukturę, ale niekoniecznie na tyle określoną, aby odpowiedni dla nich był model relacyjny lub obiektowy. Podstawowym formatem reprezentacji danych semistrukturalnych jest obecnie XML. W kontekście multimedialnych baz danych, XML jest atrakcyjnym formatem zapisu metadanych. Może być więc wykorzystany w multimedialnych bazach danych w połączeniu z modelem relacyjnym, obiektowym lub obiektowo-relacyjnym. Format XML stał się podstawą dla standardu reprezentacji metadanych o danych multimedialnych MPEG-7.



- Systemy komercyjne (Oracle, IBM DB2):
 - wykorzystanie rozszerzeń obiektowo-relacyjnych
 - rozwiązania firmowe dla metadanych sygnałowych
 - prawie brak wsparcia dla metadanych semantycznych
- Standardy:
 - SQL/MM – model obiektowo-relacyjny; brak wsparcia dla metadanych semantycznych
 - MPEG-7 – reprezentacja wszystkich typów metadanych w postaci XML; na razie słabe wsparcie przez systemy zarządzania bazą danych

Multimedialne bazy danych (17)

Wiodące systemy zarządzania bazami danych ogólnego przeznaczenia (Oracle i IBM DB2) obsługują dane multimedialne w oparciu o mechanizmy obiektowo-relacyjne, a przede wszystkim mechanizm definiowania złożonych typów danych. Reprezentacja i ekstrakcja metadanych sygnałowych oraz wyszukiwanie w oparciu o zawartość są realizowane tylko dla obrazów, w oparciu o rozwiązania firmowe, przy czym Oracle jako alternatywę dla swoich rozwiązań oferuje interfejs zgodny ze standardem SQL/MM. Żaden z systemów zarządzania bazą danych ogólnego przeznaczenia nie wspiera w szczególny sposób reprezentacji metadanych semantycznych. Ogólne wsparcie sprowadza się do możliwości definiowania złożonych struktur obiektowych i wsparcia dla XML. Metadane zewnętrzne nie stanowią problemu dla istniejących systemów zarządzania bazami danych, gdyż jako proste dane alfanumeryczne mogą być reprezentowane w strukturach relacyjnych i obiektowo-relacyjnych.

Obowiązujące standardy dla multimedialnych baz danych to SQL/MM i MPEG-7, rozwijane niezależnie przez dwa różne środowiska.

SQL/MM jest oparty o model obiektowo-relacyjny i uzupełnia SQL o typy danych do reprezentacji zaawansowanych rodzajów danych. Z multimediów w tej chwili obejmuje jedynie obrazy, przewidując dla nich ograniczony zestaw metadanych zewnętrznych, podstawowe metadane sygnałowe i wyszukiwanie w oparciu o zawartość z ich uwzględnieniem. Standard SQL/MM nie porusza kwestii metadanych semantycznych.

MPEG-7 standaryzuje sposób reprezentacji wszelkiego rodzaju metadanych o wszelkiego typu danych multimedialnych i jest oparty o XML. Nie standaryzuje metod ekstrakcji i wykorzystywania metadanych. Na razie MPEG-7 jest słabo wspierany w systemach zarządzania bazą danych. Wsparcie dla opisów MPEG-7 nie wykracza w nich poza ogólne wsparcie dla XML.



MPEG-7

- MPEG-7 = Multimedia Content Description Interface
- Standard normujący opisy zawartości multimedialnej



- Adresatami opisu mają być ludzie lub programy komputerowe
- Przewidywane zastosowania:
 - wyszukiwanie w multimedialnych bazach danych
 - filtracja treści multimedialnych

Multimedialne bazy danych (18)

W przeciwieństwie do standardów MPEG-1, MPEG-2 i MPEG-4 dotyczących reprezentacji (sposobu kodowania) zawartości obiektów multimedialnych, standard MPEG-7 (Multimedia Content Description Interface) dotyczy opisu zawartości multimedialnej. Standard MPEG-7 normuje jedynie samą formę opisu, nie standaryzując technik generacji i konsumpcji opisów. Przyczyny wyłączenia generacji opisów, ekstrakcji właściwości oraz metod wykorzystywania opisów z zakresu standardu były następujące:

- (a) mechanizmy te mają stanowić obszar swobodnej konkurencji, sprzyjającej rozwojowi technologicznemu;
- (b) standaryzacja algorytmów nie jest konieczna dla możliwości współdziałania różnych systemów;
- (c) charakter i funkcjonalność wyszukiwarek zwykle zależy od zastosowania, a format MPEG-7 ma z założenia być niezależny od zastosowań.

Zakłada się, że odbiorcami opisów MPEG-7 będą zarówno ludzie jak i programy komputerowe oraz inteligentne urządzenia elektroniczne. Opisy MPEG-7 powinny towarzyszyć danym multimedialnym składowanym w bazach danych, ale również udostępnianym przez radio, telewizję, media. Przewidywane zastosowania MPEG-7 to:

- (a) wyszukiwanie informacji w multimedialnych bazach danych;
- (b) filtracja treści multimedialnych odbieranych za pomocą telewizji, radia itp. w oparciu o opisy MPEG-7 przesyłane wraz z zawartością i ustawieniami preferencji użytkownika.



Założenia standardu MPEG-7

- Standaryzacja opisu metadanych
- Niezależność od sposobu kodowania medium
- Elastyczność
- Rozszerzalność
- Oparcie o język XML
 - język definicji składni (DDL) oparty o XML Schema
- Binarny format BiM do kompresji i transmisji opisów

Multimedialne bazy danych (19)

Podstawowym założeniem przyświecającym twórcom standardu MPEG-7 była standaryzacja opisu metadanych. Wcześniej pojawiało się wiele niezgodnych ze sobą propozycji formatów opisów multimedialów, MPEG-7 korzysta z ich doświadczeń i normuje format opisu jako powszechnie uznany i oficjalny standard.

MPEG-7 jest całkowicie niezależny od sposobu kodowania i przechowywania medium i może dotyczyć np. obrazów narysowanych na papierze. Jeśli jednak dany format kodowania może zawierać informacje o strukturalnej dekompozycji obiektu multimedialnego (jak np. MPEG-4), informacje te mogą być wykorzystane przy tworzeniu opisów MPEG-7.

MPEG-7 jest standardem elastycznym i rozszerzalnym. Jako efekt analizy szerokiego spektrum potencjalnych zastosowań, MPEG-7 jest formatem ogólnego przeznaczenia. Jednocześnie daje on możliwość rozszerzenia składni języka metadanych o elementy przydatne w specyficznych zastosowaniach.

MPEG-7 jest oparty o język XML. Wybór formatu do tekstowych opisów był w zasadzie oczywisty, biorąc pod uwagę elastyczność, prostotę i pozycję na rynku języka XML. Dzięki temu, że twórcy MPEG-7 postanowili nie opracowywać od podstaw nowego formatu, ale oparli się o XML, automatycznie dostępnych jest szereg narzędzi do parsowania i przeszukiwania opisów. Język definicji składni opisów MPEG-7 (Description Definition Language - DDL) został zdefiniowany w oparciu o XML Schema, który jest silnym narzędziem opisu struktury (gramatyki) dokumentów XML.

Mimo niekwestionowanych zalet XML, ma on też pewne wady. XML jest nieodpowiedni dla transmisji strumieniowej, choćby ze względu na nieoszczędny sposób reprezentacji informacji. Dlatego też, w ramach standardu MPEG-7 z myślą o transmisji (szczególnie strumieniowej) opisów opracowany został binarny format opisów MPEG-7 o nazwie BiM.

BiM umożliwia bardzo dobrą kompresję opisów XML-owych z zachowaniem pewnych możliwości nawigacji po strukturze opisu na poziomie binarnym.



Zawartość opisów MPEG-7

- Zbudowane z deskryptorów i schematów opisu
- Dekompozycja przestrzenna i czasowa
 - segmenty, regiony
- Opisy znaczenia zawartości (metadane semantyczne)
 - opisy zdarzeń, osób, obiektów, czynności
- Niskopoziomowe opisy zawartości (metadane sygnałowe)
 - kształt, tekstura, kolor, lokalizacja, trajektoria ruchu, melodia
- Informacje uzupełniające (metadane zewnętrzne)
 - forma, dostępność materiału, klasyfikacja, kontekst

Multimedialne bazy danych (20)

Od strony technicznej opisy MPEG-7 są tworzone za pomocą deskryptorów (reprezentujących poszczególne właściwości obiektów multimedialnych) i schematów opisu (specyfikujących strukturę i semantykę związków między elementami opisu tj. deskryptorami i innymi schematami opisu).

Treść dokumentów MPEG-7 oprócz ogólnego opisu całego obiektu multimedialnego, zawiera typowo opisy poszczególnych elementów składowych obiektu, wyodrębnionych w drodze dekompozycji przestrzennej i/lub czasowej. Przykładem dekompozycji czasowej jest podział sekwencji wideo na segmenty. Z kolei dekompozycja przestrzenna może dotyczyć samodzielnych obrazów nieruchomych lub klatek filmu wideo i polega na wskazaniu na obrazie regionów zawierających interesujące obiekty.

Opisy MPEG-7 mogą obejmować wszystkie rodzaje metadanych. Za pomocą MPEG-7 można opisać semantykę obiektu i jego elementów (opisy zdarzeń, osób, obiektów, czynności) oraz własności niskopoziomowe (zależnie od typu danych: kształt, rozmiar, tekstura, kolor, lokalizacja, trajektoria ruchu, tempo, melodia). Opisy typowo obejmują również informacje uzupełniające na temat formy (format, rozmiar, rozdzielcość), dostępności materiału (cena, ochrona praw autorskich), klasyfikacji treści (kategoria tematyczna, kategoria wiekowa odbiorców) i kontekstu (czas i miejsce nagrania, wydarzenie).

MPEG-7 – Przykład

```

<Mpeg7 ...>
<Description xsi:type="ContentEntityType">
<MultimediaContent xsi:type="ImageType">
<Image id="a102">
<MediaLocator><MediaUri>kuszczak.jpg</MediaUri></MediaLocator>
<TextAnnotation><FreeTextAnnotation>
    Polska - Kolumbia (1:2). Polski bramkarz Tomasz
    Kuszczak wpuszcza gola na 0:2 po strzale bramkarza Kolumbi.
</FreeTextAnnotation></TextAnnotation>
<SpatialDecomposition>
<StillRegion>
<TextAnnotation><FreeTextAnnotation>
    Polski bramkarz Tomasz Kuszczak w wysokim
    skoku myślał, że piłka przechodzi nad poprzeczką.
</FreeTextAnnotation></TextAnnotation>
<SpatialLocator>
<Polygon><Coords dim="2 4"> 45 20 0 -20 25 0 60 0
</Coords></Polygon>
</SpatialLocator>
</StillRegion>
...
</Mpeg7>

```

Multimedialne bazy danych (21)

Na slajdzie pokazano przykład opisu MPEG-7 dla obrazka przedstawiającego scenę z meczu piłki nożnej. (Aby przykład zmieścił się na slajdzie niektóre fragmenty zostały zastąpione wielokropkiem.). Znaczenie oznaczonych fragmentów dokumentu jest następujące:

1. Informacja o tym, że opisywany obiekt multimedialny jest obrazem. Następnie początek elementu z opisem obrazu, a w nim jako pierwszy, element zawierający informację o źródle (lokalizacji) obrazu.
2. Ogólny opis treści obrazu w formie adnotacji swobodnym tekstem.
3. Dekompozycja przestrzenna obrazu, umożliwiająca szczegółowy opis poszczególnych jego regionów. W tym wypadku wyróżniony i opisany jest jeden region (zaznaczony na obrazku żółtym obramowaniem).
4. Opis zawartości regionu obrazu w formie adnotacji swobodnym tekstem.
5. Informacje o lokalizacji opisywanego regionu na obrazie. W tym wypadku jest on zdefiniowany jako wielokąt o czterech wierzchołkach. Pierwsze cztery liczby opisują współrzędne X wierzchołków, a kolejne cztery – współrzędne Y. Współrzędne pierwszego wierzchołka są określone w sposób bezwzględny, pozostałych – inkrementalnie względem poprzedzającego wierzchołka.



Zapytania do multimedialnych baz danych

- Zapytania o zawartość i metadane ją opisujące
 - mechanizm content-based retrieval
- Duża rola interfejsów wizualnych
- Podejście Information Retrieval (IR):
 - wagi przypisywane kryteriom selekcji
 - wyszukiwanie jako proces iteracyjny
 - tolerancja dla niekompletnych wyników

Multimedialne bazy danych (22)

Zapytania do multimedialnych baz danych odwołują się do metadanych zewnętrznych opisujących obiekty multimedialne i do rzeczywistej ich zawartości, reprezentowanej przez metadane semantyczne i sygnałowe. Kluczowe znaczenie mają algorytmy wyszukiwania w oparciu o zawartość (ang. content-based retrieval) operujące na metadanych sygnałowych.

Wałą rolę w przeszukiwaniu multimedialnych baz danych pełnią interfejsy wizualne, za pomocą których zapytania tekstowe są uzupełnione lub zastąpione przez wybranie, dostarczenie lub naszkicowanie wzorca do przeszukiwania kolekcji danych.

Charakterystyczne dla zapytań do multimedialnych baz danych jest podejście w stylu Information Retrieval (IR). Po pierwsze, użytkownik powinien mieć możliwość przypisywania wag poszczególnym kryteriom selekcji np. odwołującym się do koloru i tekstu. Po drugie, zakłada się, że wyszukiwanie ma charakter procesu iteracyjnego, w którym użytkownik modyfikuje parametry zapytania aż do uzyskania satysfakcjonującego rezultatu. Jest to uzasadnione, gdyż np. przy wyszukiwaniu obiektów podobnych do zadanego, trudno jest za pierwszym razem dobrać odpowiednie wagi i próg dla miary podobieństwa. Po trzecie, tolerowane są niekompletne wyniki, w których nie znalazły się niektóre obiekty spełniające zadane kryteria, a aby móc pojawiły się obiekty tych kryteriów nie spełniające. Takie podejście jest znaczco odmienne od klasycznych zapytań do baz danych, ale w sytuacji gdy zapytania odwołują się niekiedy do subiektywnego odczucia podobieństwa, trudno o obiektywną kompletność wyników wyszukiwania.



Wyszukiwanie w oparciu o zawartość

- Wyszukiwanie w oparciu o automatycznie wyekstrahowane metadane sygnałowe
- Technologia najlepiej dopracowana dla obrazów
- Alternatywa lub uzupełnienie (!) dla wyszukiwania w oparciu o alfanumeryczne opisy wprowadzone „ręcznie”
- Zaleta: opis słowny czasochłonny i często trudny
- Wiele praktycznych zastosowań
 - wyszukiwanie poprzez podanie przykładu
 - wykrywanie plagiatów

Multimedialne bazy danych (23)

Wyszukiwanie w oparciu o zawartość (ang. content-based retrieval) polega na wyszukiwaniu w oparciu o automatycznie wyekstrahowane metadane sygnałowe. Są to metadane reprezentujące takie właściwości jak:

- (a) dla obrazów: średni kolor, udział i lokalizacja kolorów, tekstura;
- (b) dla danych audio: melodia, rytm;
- (c) dla danych wideo: dynamika ruchu, właściwości obrazów stanowiących klatki filmu.

Wyszukiwanie w oparciu o zawartość jest najlepiej dopracowane dla obrazów. Tylko dla obrazów jest ono przedmiotem standardu SQL/MM. Podobnie, tylko w zakresie przeszukiwania kolekcji obrazów ten mechanizm wspierają dostępne obecnie (wiosna 2006) systemy zarządzania bazą danych ogólnego przeznaczenia (Oracle, IBM DB2 i Informix).

Wyszukiwanie w oparciu o zawartość może stanowić alternatywę lub uzupełnienie dla wyszukiwania w oparciu o alfanumeryczne opisy wprowadzone „ręcznie”.

Zaletą wyszukiwania w oparciu o zawartość jest to, że tworzenie alfanumerycznych opisów zawartości danych multimedialnych jest czasochłonne, a na razie nie może być wykonane bez udziału człowieka. Ponadto, nie wszystkie właściwości można w sposób naturalny opisać słownie, np. tekstury i kształty na obrazach.

Wyszukiwanie w oparciu o zawartość ma szereg zastosowań. Przede wszystkim są to wszelkie interfejsy do zapytań poprzez przykład, a z zastosowań ścisłe praktycznych - wykrywanie plagiatów utworów muzycznych, zastrzeżonych znaków towarowych, itp.



Przykłady zapytań

- Odwołujące się do metadanych zewnętrznych:
 - utwory wykonawcy X dostępne w formacie MP3
 - komedia reżysera X o czasie trwania poniżej 1,5h
- Odwołujące się do znaczenia (metadane semantyczne):
 - przemówienia na temat bezrobocia
 - fotografie przedstawiające prezydenta z premierem
- Odwołujące się do zawartości (metadane sygnałowe):
 - obrazy/utwory podobne do narysowanego/zanuconego
 - filmy z aktorem przedstawionym na zdjęciu
 - obrazy, w których dominuje kolor czerwony

Multimedialne bazy danych (24)

Slajd przedstawia przykładowe zapytania do multimedialnych baz danych, pogrupowane wg typu metadanych, do których się odwołują.

Przykłady zapytań odwołujących się do metadanych zewnętrznych to:

- (a) „Znajdź wszystkie utwory wykonawcy X dostępne w formacie MP3”;
- (b) „Znajdź wszystkie komedia reżysera X o czasie trwania poniżej 1,5h”.

Przykłady zapytań odwołujących się do znaczenia, realizowanych w oparciu o metadane semantyczne to:

- (a) „Znajdź wszystkie przemówienia na temat bezrobocia”;
- (b) „Znajdź wszystkie fotografie przedstawiające prezydenta z premierem”.

Przykłady zapytań odwołujących się do zawartości, realizowanych w oparciu o metadane sygnałowe to:

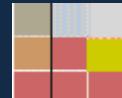
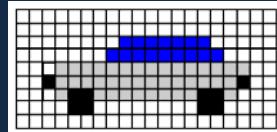
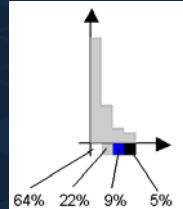
- (a) „Znajdź wszystkie obrazy podobne do narysowanego (naszkicowanego)”;
- (b) „Znajdź wszystkie utwory podobne do zanuconego do mikrofonu”;
- (c) „Znajdź wszystkie filmy z aktorem przedstawionym na zdjęciu”;
- (d) „Znajdź wszystkie obrazy, w których dominuje kolor czerwony”.

Oczywiście możliwe są też zapytania odwołujące się jednocześnie do różnych typów metadanych, np. „Znajdź wszystkie fotografie w formacie JPEG, w rozdzielcości co najmniej 1024 x 768 pikseli, przedstawiające krajobraz wysokogórski o udziale i lokalizacji kolorów zbliżonej do wskazanego, przykładowego obrazka”.



Metadane sygnałowe dla obrazów

- Średni kolor
- Histogram kolorów
- Kontury segmentów obrazu (kształty)
- Tekstury
- Lokalizacja kolorów, kształtów i tekstur na obrazie



Multimedialne bazy danych (25)

Jak wspomniano wcześniej, technika wyszukiwania w oparciu o zawartość w multimedialnych bazach danych jest w chwili obecnej najlepiej dopracowana i dostępna dla obrazów. Typowe właściwości obrazów wykorzystywane przy wyszukiwaniu obrazów w oparciu o zawartość to:
 Średni kolor – wyznaczany w oparciu o podział obrazu na n próbek i niezależne uśrednianie komponentów RGB koloru;

Histogram kolorów – reprezentujący udział procentowy kolorów w obrazie, najczęściej wyznaczany po zredukowaniu liczby kolorów np. do 256;

Kształty - odkrywane w procesie tzw. segmentacji obrazu, gdzie segmenty są rozumiane jako spójne regiony o tym samym kolorze;

Tekstury - opisujące chropowatość, kontrast, kierunkowość wzorców wypełniających kształty i tło obrazu; reprezentowane w postaci wektorów czy macierzy współczynników liczbowych, wyznaczanych specjalistycznymi algorytmami;

Lokalizacja kolorów, kształtów i tekstur na obrazie – wyznaczana poprzez podział obrazu na siatkę regionów i następnie wyznaczenie dominujących kolorów, kształtów i tekstur dla każdego regionu.



Interfejsy do wyszukiwania obrazów w oparciu o zawartość

- Najczęściej wyszukiwanie poprzez podanie przykładu
 - wybranego z predefiniowanego zbioru
 - dostarczonego przez użytkownika
 - naszkicowanego za pomocą edytora graficznego
- Możliwość wyboru właściwości i przypisania wag
- Przykład (<http://www.heritagemuseum.org/>):



Multimedialne bazy danych (26)

Kryteria wyszukiwania obrazów ze względu na zawartość typowo są podawane w formie przykładowego obrazka. Wyszukiwanie sprowadza się do znalezienia obrazów podobnych do podanego przykładu. Często użytkownik ma możliwość wskazania, które właściwości wizualne mają być uwzględnione w testach podobieństwa z opcją przypisania im wag.

Dostarczenie systemowi wzorcowego obrazu może polegać na:

- wyborze z predefiniowanego zbioru obrazków reprezentujących poszczególne kategorie,
- załadowaniu do systemu obrazka dostarczonego przez użytkownika,
- naszkicowaniu obrazu za pomocą prostego, wbudowanego w interfejs edytora graficznego. Edytor ten może być zorientowany na konkretną dziedzinę zastosowań i udostępniać pewne predefiniowane kształty.

Przykładem istniejącego, ogólnodostępnego serwisu umożliwiającego wyszukiwanie obrazów ze względu na zawartość jest serwis internetowy rosyjskiego muzeum Ermitaż (<http://www.heritagemuseum.org/>). Serwis udostępnia dwie wyszukiwarki oparte o technologię QBIC firmy IBM, która jest obecnie częścią funkcjonalności systemu zarządzania bazą danych IBM DB2:

- QBIC Color Search (<http://www.heritagemuseum.org/cgi-bin/db2www/qbicColor.mac/qbic?selLang=English>) umożliwiającą wyszukanie obrazów o zadanym udziale kolorów,
- QBIC Layout Search (<http://www.heritagemuseum.org/cgi-bin/db2www/qbicLayout.mac/qbic?selLang=English>) umożliwiającą wyszukanie obrazów o wskazanej lokalizacji kolorów.

Na slajdzie pokazany jest przykład zapytania o lokalizację kolorów przygotowanego za pomocą interfejsu QBIC Layout Search i przykładowy obrazek z kolekcji obrazów zwróconych przez system dla tego zapytania.



Ekstrakcja i indeksowanie właściwości

- Ekstrakcja:
 - a priori
 - dynamiczna
 - a priori dla części właściwości, dynamiczna dla innych
- Indeksowanie:
 - właściwości reprezentowane jako punkty w przestrzeni wielowymiarowej
 - wykorzystywane struktury indeksów wielowymiarowych:
 - R-drzewa, pliki kratowe, k-d-B-drzewa, itp.
 - zapytania punktowe, przedziałowe, najbliższy sąsiad

Multimedialne bazy danych (27)

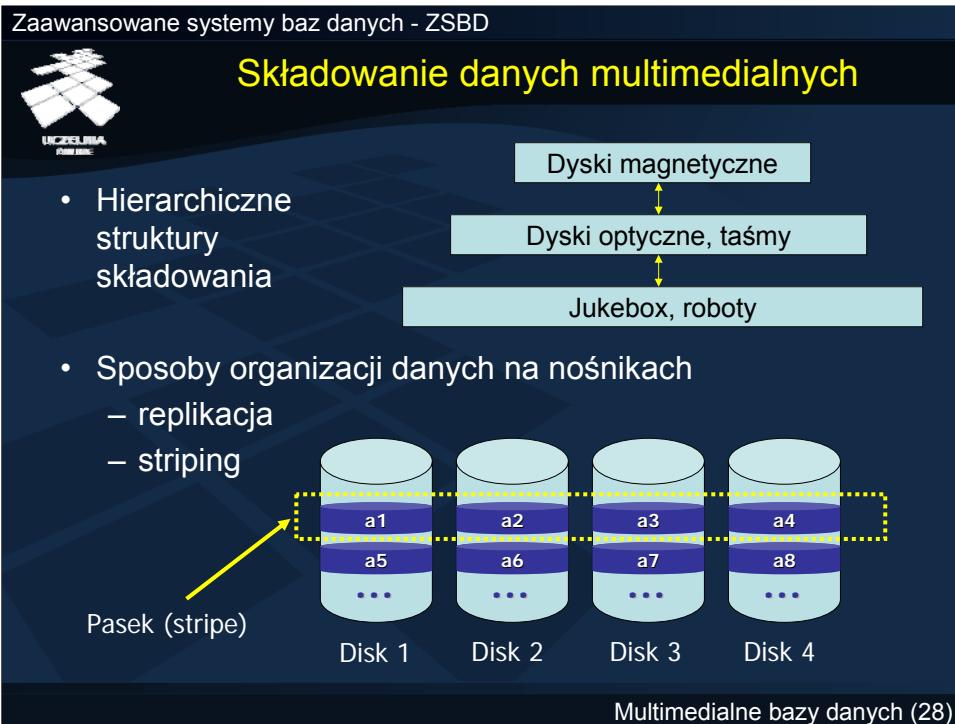
Ekstrakcja właściwości wykorzystywanych do wyszukiwania w oparciu o zawartość z obiektów multimedialnych może odbywać się: a priori – w momencie ładowania obiektów do bazy danych lub dynamicznie – w momencie gdy pojawia się zapytanie, które się do nich odwołuje.

Zaletą ekstrakcji a priori jest większa szybkość realizacji zapytań, gdyż właściwości dla obiektów w bazie danych są już wyznaczone, a być może również poindeksowane. W przypadku ekstrakcji a priori wydłuża się jednak czas ładowania obiektów do bazy danych i rosną wymagania na pamięć dyskową. Z kolei ekstrakcja dynamiczna prowadzi do spowolnienia zapytań, gdyż algorytmy ekstrakcji właściwości są czasochłonne.

Ogólnie, lepszym rozwiązaniem jest ekstrakcja a priori, gdyż pamięć dyskowa nie jest dziś droga, a ważniejsza jest redukcja czasu zapytań (wykonywanych wielokrotnie) niż ładowania danych do bazy (wykonanego raz). Ekstrakcja a priori daje też możliwość poindeksowania właściwości, co dodatkowo może przyspieszyć realizację zapytań. Poleganie na samej tylko ekstrakcji a priori wiąże się jednak z koniecznością wskazania, które właściwości będą mogły być wykorzystywane w zapytaniach, już na etapie projektowania struktur danych. Dlatego też, możliwym kompromisem jest ekstrakcja a priori dla najważniejszych w danej aplikacji właściwości, z możliwością ekstrakcji dynamicznej tych rzadziej wykorzystywanych.

Wyszukiwanie w oparciu o zawartość może być wspierane za pomocą struktur indeksowych. Właściwości obiektów multimedialnych mogą być reprezentowane jako punkty w przestrzeni wielowymiarowej. Z tego powodu, multimedialne bazy danych wykorzystują struktury indeksów wielowymiarowych, takie jak R-drzewa, pliki kratowe, k-d-B-drzewa, itp. Od indeksów wymaga się wspierania zapytań:

- (a) punktowych – do wyszukiwania obiektów o dokładnie takich samych właściwościach;
- (b) przedziałowych - do wyszukiwania obiektów o właściwościach zbliżonych;
- (c) typu najbliższy sąsiad – do wyszukiwania obiektu najbardziej podobnego do danego.



Składowanie danych stanowi istotny problem dla danych multimedialnych o szczególnie dużym rozmiarze, czyli przede wszystkim filmów wideo. Pierwszym ograniczeniem jakie może napotkać system składowania danych wideo jest ograniczona pamięć dyskowa. Tradycyjnie problem ten był rozwiązywany przez wykorzystanie hierarchicznych struktur składowania. Idea polegała na przechowywaniu najczęściej żądanych filmów na dyskach magnetycznych (dostępnych on-line), a wszystkich filmów na nośnikach typu dyski optyczne czy taśmy (dostępnych off-line), udostępnianych przez urządzenia typu jukebox albo archiwów obsługiwane przez roboty. Dane zawsze były serwowane z pamięci dyskowej, co pociągało za sobą konieczność migracji danych z pamięci off-line do on-line w przypadku żądania mało popularnego filmu.

Drugim problemem, który miał znaczenie przede wszystkim w przeszłości, była zbyt mała szybkość transmisji danych z dysków. Rozwiązaniem tego problemu to replikacja i striping. Replikacja polega na duplikowaniu zawartości poszczególnych dysków, dzięki czemu kolejne fragmenty filmu mogą być równolegle odczytywane z kolejnych dysków. Striping to technika umożliwiająca osiągnięcie tego samego efektu, ale bez duplikowania danych i marnowania przestrzeni dyskowej. W tym wypadku dane rozmieszczone są na kolejnych dyskach w formie tzw. pasków (ang. stripe). Fragmenty jednego paska mogą być odczytywane równolegle, zwiększając przepustowość transmisji danych filmu z macierzy dyskowej maksymalnie tyle razy, ile dysków obejmuje pasek.

Rozwiązania przedstawione na slajdzie w ostatnich latach z pewnością straciły na znaczeniu w związku z ciągłym wzrostem pojemności i szybkości dysków magnetycznych połączonym ze spadkiem ich cen. Niemniej, pojemność macierzy dysków serwera w dalszym ciągu jest mniejsza od praktycznie nieograniczonej pojemności archiwów taśmowych obsługiwanych przez roboty, a szybkość transmisji danych z dysków magnetycznych w dalszym ciągu stanowi wąskie gardło w porównaniu z szybkością odczytu danych z pamięci operacyjnej, a nawet szybkością transmisji danych przez nowoczesne sieci komputerowe.



Prezentacja danych multimedialnych (1/2)

- Problemy z danymi wrażliwymi na opóźnienia
- Transmisja strumieniowa danych wideo
- Warunki dla pojedynczego strumienia
- Obsługa wielu strumieni
 - balans między liczbą żądań a jakością usług
 - szeregowanie strumieni

Multimedialne bazy danych (29)

Problemy prezentacji danych multimedialnych dotyczą danych wrażliwych na opóźnienia, czyli danych audio i wideo. Szczególnie problemy uwidaczniają się dla danych wideo, ze względu na ich duży rozmiar. W praktyce dla danych wideo stosowana jest transmisja strumieniowa z dwóch podstawowych powodów. Po pierwsze, klient może nie dysponować wolną przestrzenią dyskową umożliwiającą pobranie i składowanie całego filmu po swojej stronie przed rozpoczęciem odtwarzania. Po drugie, w przypadku transmisji strumieniowej prawie natychmiast po rozpoczęciu przesyłania danych może rozpocząć się odtwarzanie filmu po stronie klienta, bez konieczności oczekiwania na przesłanie całego filmu.

Dla pojedynczego strumienia musi być spełniony oczywisty i z pozoru łatwy do spełnienia warunek, tj. ilość produkowanych danych musi w każdej chwili przewyższać ilość danych konsumowanych. Sytuacja niestety jest o tyle skomplikowana, że poziom konsumpcji danych może być zmienny w czasie ze względu na kompresję danych, a poziom produkcji danych może być zmienny ze względu na konieczność zmiany ścieżek przy odczycie z dysku, zmienne prędkości transmisji danych dla różnych regionów nośnika, itp.

W przypadku obsługi wielu klientów przez wiele strumieni, system musi zadbać, aby do każdego strumienia w odpowiednim momencie wysłać odpowiednią ilość danych. Przyjmując kolejne żądanie system musi zapewnić, aby nie spowodowało to obniżenia poziomu jakości usług obsługiwanych już użytkownikom. Strumienie obsługiwane są w tzw. rundach – w każdej rundzie do każdego strumienia przekazywana jest odpowiednia ilość danych.



Prezentacja danych multimedialnych (2/2)

- Optymalizacja obsługi wielu klientów
 - batching
 - bridging
 - piggybacking

Multimedialne bazy danych (30)

Optymalizacja strumieniowej transmisji dla wielu klientów sprowadza się przede wszystkim do redukcji liczby strumieni wykorzystywanych do transmisji tego samego filmu.
Zaproponowano w tym zakresie następujące techniki:

Batching – polega na odsunięciu w czasie rozpoczęcia transmisji – być może nadzieją identyczne żądania i grupę żądań obsłuży jeden strumień;

Bridging – dla strumieni dotyczących tego samego filmu, nieznacznie przesuniętych w czasie, polega na przechowywaniu danych z okna czasowego między strumieniami w pamięci cache;

Piggybacking – dla strumieni dotyczących tego samego filmu, nieznacznie przesuniętych w czasie, polega na przyspieszeniu drugiego filmu (poprzez pominięcie niektórych klatek) lub spowolnieniu pierwszego (poprzez wstawienie klatek interpolowanych) tak aby oba żądania były od pewnego momentu obsługiwane przez jeden strumień (zaobserwowano, że wahania prędkości transmisji poniżej 5% są niezauważalne dla człowieka).



Podsumowanie

- SZMBD = SZBD obsługujący dane multimedialne
- Dane multimedialne znacząco różnią się od tradycyjnych
- Kluczową rolę w opisie i wyszukiwaniu danych multimedialnych odgrywają metadane
- Typowe w multimedialnych bazach danych jest wyszukiwanie w oparciu o zawartość
- Szczególne wyzwanie dla systemów baz danych stanowią dane wideo ze względu na duży rozmiar i wrażliwość na opóźnienia

Multimedialne bazy danych (31)

Od systemów zarządzania multimedialną bazą danych oczekuje się, aby oferowały one tradycyjną funkcjonalność systemów zarządzania bazą danych dla danych multimedialnych. Problemy w implementacji takich systemów wynikają ze specyfiki danych multimedialnych w zakresie modeli danych, przetwarzania zapytań, fizycznych struktur składowania i metod prezentacji.

Kluczową rolę w opisie i wyszukiwaniu danych multimedialnych odgrywają różnego rodzaju metadane. W związku z tym dla multimedialnych baz danych wskazane są modele danych umożliwiające modelowanie obiektów multimedialnych jako obiektów wiążących zawartość binarną i opisujące ją metadane.

Typowe w multimedialnych bazach danych jest wyszukiwanie w oparciu o zawartość, w praktyce najczęściej realizowane poprzez podanie przykładu i wyszukiwanie obiektów podobnych. Zapytania tego typu odwołują się niskopoziomowych właściwości obiektów multimedialnych, automatycznie ekstrahowanych z danych specjalistycznymi algorytmami.

Szczególne wyzwanie dla systemów baz danych stanowią dane wideo, ponieważ podobnie jak audio są wrażliwe na opóźnienia w transmisji, a do tego typowo ich rozmiar jest znacznie większy od innych typów danych. Stwarza to problemy odpowiedniej organizacji danych na nośnikach oraz wymaga transmisji strumieniowej przy prezentacji danych.



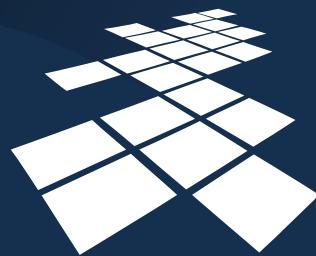
Materiały dodatkowe

- V.S. Subrahmanian, Principles of Multimedia Database Systems, Morgan Kaufmann
- B. Thuraisingham, Managing and Mining Multimedia Databases, CRC Press
- H. Kosch, Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21, CRC Press
- J. M. Martínez, MPEG-7 Overview
- J. Yiu-bun Lee, Distributed Video Systems

Multimedialne bazy danych (32)

Standard SQL/MM

Wykład prowadzi:
Marek Wojciechowski



UCZELNIA
ONLINE

Standard SQL/MM

Wykład poświęcony jest standardowi SQL/MM, który jest nowym standardem uzupełniającym język SQL o biblioteki do obsługi specjalistycznych danych i aplikacji.

Celem wykładu jest przedstawienie głównych idei standardu SQL/MM, a w szczególności jego części poświęconych przetwarzaniu danych tekstowych, przestrzennych i obrazów w bazach danych.

Do zrozumienia treści wykładu niezbędna jest znajomość systemów baz danych i języka SQL oraz podstawowej problematyki multimedialnych baz danych.



Plan wykładu

- Wprowadzenie do standardu SQL/MM
- Przegląd części standardu SQL/MM
- Omówienie specyfikacji SQL/MM Full Text
- Omówienie specyfikacji SQL/MM Spatial
- Omówienie specyfikacji SQL/MM Still Image
- Podsumowanie

Standard SQL/MM (2)

Wykład rozpocznie się od krótkiego wprowadzenia do standardu SQL/MM, przedstawienia jego genezy i zakresu. Następnie szczegółowo omówione i zilustrowane przykładami będą jego specyfikacje składowe dotyczące danych tekstowych, przestrzennych i obrazów. Dla kompletności, krótko przedstawione będą również pozostałe części standardu, dotyczące eksploracji danych i obsługi danych historycznych.



Standard SQL/MM

- SQL/MM: SQL Multimedia and Application Packages
- Opracowywany i publikowany przez ISO
- Standard obejmujący wiele części:
 - części poświęcone szeroko pojmowanym multimediami
 - części poświęcone specjalistycznym zastosowaniom
- Oparty o SQL i jego typy definiowane przez użytkownika (typy obiektowe SQL99)

Standard SQL/MM (3)

Podobnie jak SQL, SQL/MM jest standardem ISO i składa się z wielu części, przy czym części SQL/MM są ze sobą raczej luźno związane w porównaniu ze specyfikacjami SQL.

Pełna nazwa standardu SQL/MM brzmi: SQL Multimedia and Application Packages, co oznacza że jego części niekoniecznie muszą dotyczyć przetwarzania szeroko pojętych danych multimedialnych, ale również specjalistycznych zastosowań systemów baz danych. Należy też zwrócić uwagę, że z punktu widzenia standardu termin multimedia obejmuje również dane przestrzenne i tekstowe.

SQL/MM stanowi uzupełnienie języka SQL i jest oparty o mechanizmy obiektowo-relacyjne, które pojawiły się w standardzie SQL99.



Geneza standardu SQL/MM

- Wnioski z prac nad rozszerzeniami SQL dla danych tekstowych, przestrzennych, multimedialnych
 - konflikty nazw np. CONTAINS
 - przewidywane trudności w implementacji
- Dostępność obiektowych typów danych od SQL99
 - zamiast rozszerzeń SQL – biblioteki typów obiektowych

Standard SQL/MM (4)

Ponieważ standard języka SQL nie zawiera konstrukcji do obsługi takich danych jak multimedia, duże obiekty tekstowe, czy też dane przestrzenne, środowiska zajmujące się tworzeniem oprogramowania do przetwarzania tych specjalistycznych typów danych rozpoczęły pracę nad propozycjami rozszerzenia języka SQL o potrzebne im elementy. Niestety okazało się, że ewentualne rozszerzenia SQL dedykowane dla poszczególnych rodzajów danych mogą być niekompatybilne ze sobą. Najłatwiejszym do zauważenia potencjalnym konfliktom był konflikt słów kluczowych. Przykładowo, słowo kluczowe CONTAINS, używane jest zarówno w kontekście tekstowych baz danych (do wskazania, że dane słowo lub fraza zawiera się w danym fragmencie tekstu), jak i baz przestrzennych i multimedialnych (do wskazania, że jeden obiekt zawiera w sobie inny obiekt).

Ze względu na wspomniane wyżej problemy zarzucono koncepcję rozszerzania języka SQL w zakresie wsparcia dla baz danych tekstowych, przestrzennych i multimedialnych. Zwyciężyła koncepcja opracowania nowego standardu, obejmującego specyfikacje bibliotek opartych o typy obiektowe SQL99, przeznaczonych do obsługi poszczególnych specjalistycznych rodzajów danych i aplikacji. Nowy standard natychmiast stał się znany pod nazwą SQL/MM („MM” od „MultiMedia”). Dzięki oparciu specyfikacji SQL/MM o obiektowe typy SQL, funkcjonalność bibliotek jest w sposób naturalny dostępna z poziomu poleceń języka SQL, np. poprzez wywołania metod bibliotecznych typów obiektowych w wyrażeniach języka SQL. Z myślą o użytkownikach niechętnie korzystających z mechanizmów obiektowo-relacyjnych, dla użytkowych metod typów SQL/MM standard specyfikuje odpowiadające im funkcje SQL.



Części standardu SQL/MM

- Part 1: Framework (baza dla pozostałych części)
- Part 2: Full-Text (tekstowe bazy danych)
- Part 3: Spatial (przestrzenne bazy danych)
- Part 5: Still Image (obrazy)
- Part 6: Data Mining (eksploracja danych)
- Part 7: History (dane historyczne)

Standard SQL/MM (5)

W chwili obecnej (wiosna 2006) standard SQL/MM obejmuje pięć części (1, 2, 3, 5 i 6) o statusie oficjalnego standardu oraz jedną (7) w stadium Working Draft. Nie ma w standardzie SQL/MM części czwartej. Miała ona dotyczyć ogólnych operacji matematycznych (General Purpose Facilities), ale prace nad nią zarzucono kilka lat temu.

Część pierwsza – Framework ma charakter ogólny. Zawiera ona informacje o zakresie standardu oraz definicje i koncepcje wspólne dla pozostałych, specjalistycznych części. Część pierwsza dotyczy między innymi sposobu, w jaki inne części standardu SQL/MM wykorzystują mechanizm obiektowych typów SQL.

Części druga, trzecia i piąta poświęcone są multimediom w rozumieniu standardu SQL/MM, czyli odpowiednio danym tekstowym, przestrzennym i obrazom (nieruchomym).

Części 6 i 7 dotyczą specjalistycznych zastosowań (Application Packages). Część szósta poświęcona jest eksploracji danych, a siódma obsłudze danych historycznych w bazach danych.

Zwraca uwagę brak części poświęconych danym audio i wideo, których można było się spodziewać w standardzie, którego nazwa sugeruje nacisk na multimedia. Nie wyklucza się opracowania tych części w przyszłości, w zależności od odzewu środowiska na części już istniejące.

Istniejące specyfikacje są ciągle rozwijane. Niektóre części doczekały się już drugiej edycji, a dla niektórych z nich trwają prace nad trzecią edycją.



SQL/MM Full-Text

- Dotyczy danych tekstowych różniących się od znakowych:
 - rozmiarem
 - strukturą (zdania, akapity)
 - typami operacji
- Zakres SQL/MM Full-Text:
 - typy danych dla dokumentów tekstowych oraz wzorców wyszukiwania
 - metody dopasowywania dokumentów do wzorca
 - konwersja do/z typów znakowych
- SQL/MM Full-Text zakłada obsługę wielu języków

Standard SQL/MM (6)

SQL/MM Full-Text dotyczy przetwarzania dokumentów tekstowych, które od tradycyjnych łańcuchów znaków odróżnia przede wszystkim znacznie większy rozmiar, ale również obecność struktury (podział na zdania i akapity) oraz charakter operacji wyszukiwania. O ile dla prostych łańcuchów znaków wyszukiwanie sprowadza się do prostego dopasowywania do wzorca, w przypadku danych tekstowych kryteria wyszukiwania dotyczą obecności słów kluczowych lub fraz i uwzględniają odmianę, wymowę, a nawet znaczenie poszczególnych słów czy fraz w kontekście konkretnego języka.

Standard SQL/MM Full-Text definiuje typy danych dla dokumentów tekstowych i wzorców wyszukiwania, a także metody dopasowywania dokumentów tekstowych do wzorca. W zakres standardu wchodzą również metody konwersji danych tekstowych do/z typów znakowych.

SQL/MM Full-Text zakłada obsługę wielu języków. Niemniej, mechanizmy przeszukiwania tekstów objęte standardem SQL/MM są szczególnie odpowiednie dla języków, które łatwo poddają się analizie komputerowej w zakresie wyodrębniania poszczególnych słów i zdań. Do tej klasy języków należą języki zachodnie (w tym np. angielski, niemiecki, polski), w przypadku których słowa oddzielone są odstępami, a zdania znakami interpunkcyjnymi. Inaczej jest np. w języku japońskim, gdzie wyodrębnienie słów z tekstu wymaga analizy kontekstu.



Typy danych SQL/MM Full-Text

- Dokumenty tekstowe reprezentuje typ FullText
- Metody typu FullText:
 - Contains – test binarny (0/1) czy dokument pasuje do wzorca
 - Score –miara zgodności ze wzorcem
- Wzorce reprezentowane jako
 - wystąpienia typu FT_Pattern
 - łańcuchy znaków (CHARACTER VARYING)

Standard SQL/MM (7)

SQL/MM Full-Text definiuje kilka typów obiektowych, z których podstawowe znaczenie ma typ FullText, służący do reprezentacji dokumentów tekstowych. Spośród metod typu FullText największe znaczenie mają: Contains i Score. Contains umożliwia sprawdzenie czy dokument pasuje do wzorca i zwraca wartość 1 gdy pasuje a 0 w przeciwnym wypadku. Metoda Score dla danego dokumentu i wzorca zwraca miarę zgodności (ang. relevance) dokumentu z wzorcem w formie liczby zmienoprzecinkowej. Metoda Score może być wykorzystana do generowania rankingu dokumentów wg dopasowania do zadanej wzorcowej. Parametrem obu metod jest wzorzec. Wzorce mogą być reprezentowane w postaci wystąpień dedykowanego do tego celu typu FT_Pattern albo jako zwykłe łańcuchy znaków (typu CHARACTER VARYING).



Klasy wzorców (1/3)

- Wystąpienie konkretnego słowa lub frazy
 - tresp.Contains(' "standard" ')
 - tresp.Contains(' "standard języka" ')
- Wzorce ze znakami _ i %
 - tresp.Contains(' "sport_" ')
 - tresp.Contains(' "standard%" ')
- Wystąpienie co najmniej jednego słowa/frazy z listy
 - tresp.Contains(' "standard", "język%" ')
- Wystąpienie formy danego słowa/frazy wg reguł odmiany danego języka (domyślny jest angielski)
 - tresp.Contains('STEMMED FORM OF POLISH "standard języka" ')

Standard SQL/MM (8)

Ten slajd i dwa kolejne przedstawiają klasy wzorców, które wg standardu SQL/MM Full-Text mogą być wykorzystane do przeszukiwania kolekcji dokumentów. Dla każdej klasy wzorców przedstawiono przykład (przykłady) jego wykorzystania w metodzie Contains na rzecz dokumentu reprezentowanego przez zmienną „tresp”.

Pierwsza klasa wzorców żąda wystąpienia konkretnego słowa lub frazy. Druga przy dopasowaniu wzorca do dokumentu wykorzystuje znaki specjalne „_” i „%” o takim samym znaczeniu jak dla operatora LIKE w języku SQL. Trzecia klasa wzorców to wzorce wymagające wystąpienia co najmniej jednego słowa/frazy z listy słów/fraz. Ostatnia przedstawiona na slajdzie klasa wzorców to wzorce wymagające wystąpienia słowa lub frazy lub jego/jej formy gramatycznej, będące(-go)(-j) wynikiem odmiany zgodnie z regułami danego języka.

Standard SQL/MM Full-Text zakłada, że każdy dokument, słowo, czy fraza posiada przypisany język. W definicji wzorców, każde słowo lub fraza mogą być poprzedzone nazwą języka (ENGLISH, GERMAN, POLISH, ...). W przypadku gdy słowo (lub fraza) nie jest poprzedzone we wzorcu nazwą języka, przyjmowany jest język domyślny, jakim wg standardu jest język angielski. W ramach jednego złożonego wzorca mogą występować słowa lub frazy z różnych języków.



- Wystąpienie słów/fraz w sąsiedztwie
 - `tresc.Contains(' standard' NEAR ("SQL", "MPEG") WITHIN 2 PARAGRAPHS)`
- Wystąpienie słów/fraz w tym samym kontekście
 - `tresc.Contains(' standard' IN SAME SENTENCE AS "SQL")`
- Wzorce złożone za pomocą operatorów logicznych AND, OR i NOT
 - `tresc.Contains(' SQL/MM' & ("Text" | "Spatial") & NOT "Image")`

Standard SQL/MM (9)

Pierwsza z klas wzorców wymienionych na slajdzie wymaga wystąpienia dwóch słów lub fraz w sąsiedztwie ograniczonym podaną liczbą znaków (CHARACTERS), słów (WORDS), zdań (SENTENCES) lub akapitów (PARAGRAPHS) z możliwością wskazania, że podane słowa/frazy muszą wystąpić w wyspecyfikowanej we wzorcu kolejności (IN ORDER). Druga klasa wzorców wymaga wystąpienia dwóch słów/fraz w tym samym kontekście, przy czym kontekstem może być zdanie (SENTENCE) lub akapit (PARAGRAPH). Trzecia klasa wzorców to wzorce złożone, zbudowane z prostych wzorców połączonych logicznymi operatorami AND („&”), OR („|”) i NOT („NOT”). Budując wzorce złożone można wykorzystać nawiasy do wskazania kolejności ewaluacji operatorów. W przypadku braku nawiasów, obowiązuje priorytet operatorów logicznych w następującym porządku (od najwyższego): „NOT”, „&”, „|”.



Klasy wzorców (3/3)

- Wzorce odwołujące się do tematyki tekstu
 - tresp.Contains('IS ABOUT "język zapytań"')
- Wzorce odwołujące się do brzmienia w danym języku (domyślny jest angielski)
 - tresp.Contains(' SOUNDS LIKE "sequel" ')
- Dopasowanie przybliżone (odporne na „literówki”)
 - tresp.Contains(' FUZZY FORM OF "teacher" ')
- Wzorce dopuszczające synonimy
 - tresp.Contains(THESAURUS "computer science" EXPAND SYNONYM TERM OF "list" ')

Standard SQL/MM (10)

Pierwsza z klas wzorców wymienionych na slajdzie służy do wyszukania dokumentów na dany, ogólnie wyspecyfikowany temat. Druga klasa wzorców odwołuje się do brzmienia słowa/frazy w danym języku (domyślny jest angielski). Kolejna klasa wzorców to dopasowanie odporne na tzw. „literówki”, które może odnaleźć teksty, w których popełniono błędy ortograficzne lub drobne błędy edycyjne. Ostatnia z przedstawionych na slajdzie klas wzorców służy do wyszukania dokumentów zawierających słowa/frazy wywiedzione z podanego słowa lub frazy (np. synonimy). W definicji wzorca tego typu przede wszystkim należy wskazać nazwę słownika wyrazów bliskoznacznych (THESAURUS), który ma być wykorzystany do generacji związkanych terminów. Dostępne klauzule wskazujące metodę generacji wyszukiwanych słów i fraz to:

- (a) EXPAND SYNONYM TERM OF – synonimy,
- (b) EXPAND BROADER TERM OF – słowa/frazy o szerszym znaczeniu,
- (c) EXPAND NARROWER TERM OF – słowa/frazy o węższym znaczeniu,
- (d) EXPAND TOP TERM OF – najbardziej ogólne terminy z słów/fraz o szerszym znaczeniu,
- (e) EXPAND PREFERRED TERM OF – preferowane terminy ze zbioru synonimów,
- (f) EXPAND RELATED TERM OF – terminy powiązane.



SQL/MM Full-Text – Przykład

```
CREATE TABLE raporty (
    numer INTEGER,
    tresc FULLTEXT
)
```

```
SELECT numer
FROM raporty
WHERE tresc.CONTAINS('STEMMED FORM OF "standard"
    IN SAME PARAGRAPH AS SOUNDS LIKE "sequel"') = 1
```

Standard SQL/MM (11)

Na slajdzie pokazano przykład wykorzystania możliwości SQL/MM Full-Text z poziomu języka SQL. Pierwsze polecenie tworzy tabelę RAPORTY, której pierwsza kolumna typu INTEGER zawiera numery raportów, a druga kolumna typu FULLTEXT treści raportów. Drugie polecenie to zapytanie zwracające numery raportów zawierających w tym samym akapicie dowolną formę słowa „standard” wg reguł odmiany języka angielskiego (język nie został podany jawnie, a angielski jest domyślny) i słowo brzmiące jak „sequel”.



- Dotyczy danych przestrzennych
 - figury geometryczne, lokalizacja i topologia obiektów
- Zakres SQL/MM Spatial:
 - typy danych dla danych przestrzennych
 - tworzenie i porównywanie geometrycznych obiektów przestrzennych, obliczanie wartości miar
 - konwersje do/z typów znakowych i binarnych
- Przede wszystkim zastosowania geograficzne
- Jeden z kilku standardów dla danych przestrzennych

Standard SQL/MM (12)

SQL/MM Spatial definiuje obiektowe typy danych i ich metody do przetwarzania danych przestrzennych tj. dotyczących geometrii, lokalizacji i topologii obiektów. Przetwarzanie danych przestrzennych obejmuje tworzenie i porównywanie obiektów geometrycznych oraz obliczanie wartości miar takich jak np. pole powierzchni. W zakres standardu wchodzą również metody konwersji między typami SQL/MM Spatial a innymi znakowymi i binarnymi reprezentacjami danych przestrzennych.

Specyfikacja SQL/MM Spatial jest szczególnie zorientowana na przetwarzanie danych w systemach informacji geograficznej (GIS), ale obszar jej zastosowań wykracza poza przetwarzanie informacji o obiektach na powierzchni Ziemi i obejmuje również np. projektowanie układów elektronicznych.

SQL/MM Spatial jest związany z dwoma innymi, rozwijanymi równolegle, standardami dla danych przestrzennych opracowywanymi przez ISO Technical Committee TC 211 i Open GIS Consortium.



Możliwości SQL/MM Spatial

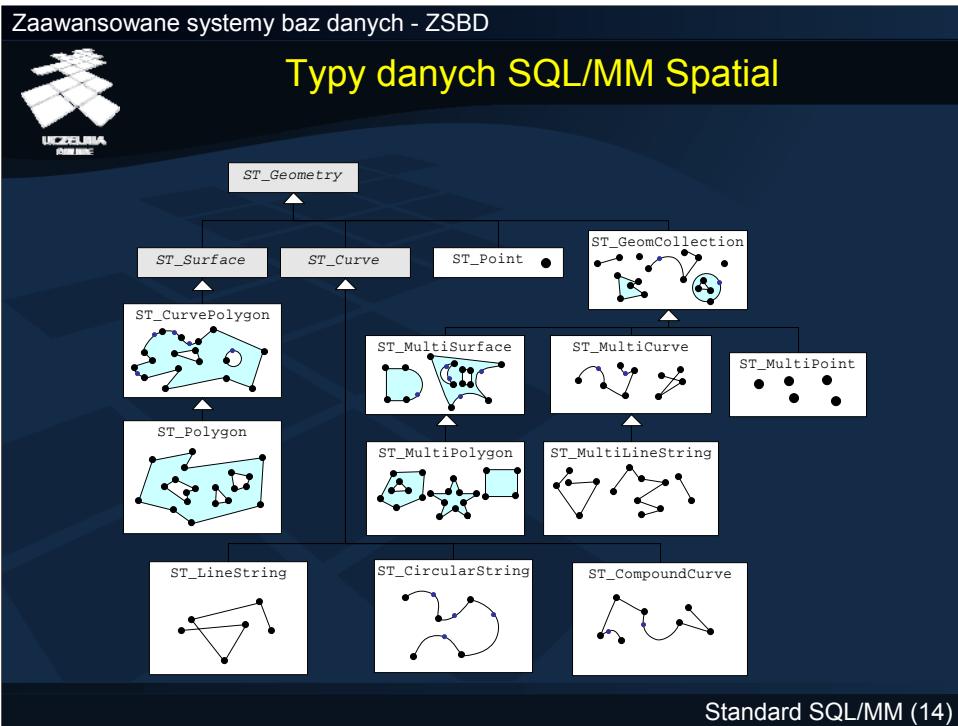
- W chwili obecnej standard wspiera dane:
 - 0-wymiarowe (punkty)
 - 1-wymiarowe (linie)
 - 2-wymiarowe (figury płaskie)
- Wsparcie dla testów przecinania, pokrywania, itp.
- Wsparcie dla różnych przestrzennych układów odniesienia
 - przede wszystkim opisujących geografię naszej planety i jej regionów (krzywizna Ziemi)

Standard SQL/MM (13)

SQL/MM Spatial obecnie wspiera dane 0-wymiarowe (punkty), 1-wymiarowe (linie) i 2-wymiarowe (figury płaskie). W rzeczywistych zastosowaniach pojawiają się oczywiście również obiekty 3-wymiarowe, niestety obecna specyfikacja SQL/MM (edycja druga) ich nie uwzględnia.

Dla obiektów geometrycznych standard przewiduje testy wzajemnego położenia obiektów np. przecinania, pokrywania, zawierania.

Dużą wagę specyfikacja SQL/MM Spatial przykłada do przestrzennych układów odniesienia (ang. spatial reference system). Większość z nich opisuje geografię naszej planety i jej regionów, co jest konsekwencją faktu, że typowymi użytkownikami wymagającymi przetwarzania danych przestrzennych są władze państwowie i lokalne oraz duże korporacje, operujące na danych geograficznych. Wybór układu wpływa m.in. na metodę wyznaczania odległości.



Podstawowy zbiór typów danych SQL/MM Spatial to typy reprezentujące poszczególne figury (kształty) geometryczne. Typy te tworzą hierarchię przedstawioną na slajdzie. Korzeniem tej hierarchii jest abstrakcyjny typ `ST_Geometry`. Inne typy abstrakcyjne, pełniące rolę węzłów pośrednich w hierarchii to `ST_Curve` (krzywa) i `ST_Surface` (2-wymiarowa figura). Typy `ST_MultiCurve` (kolekcja krzywych) i `ST_MultiSurface` (kolekcja figur 2-wymiarowych) mogą być abstrakcyjne lub nie – zależnie od implementacji. W hierarchii występują ponadto następujące nieabstrakcyjne podtypy: `ST_Point` (punkt), `ST_LineString` (sekwencja odcinków prostych), `ST_CircularString` (sekwencja łuków – odcinków okręgu), `ST_CompoundCurve` (sekwencja połączonych krzywych), `ST_CurvePolygon` (figura 2-wymiarowa o jednym brzegu i opcjonalnych dziurach w formie zamkniętej sekwencji połączonych krzywych), `ST_Polygon` (wielokąt z opcjonalnymi dziurami w kształcie wielokątów), `ST_GeomCollection` (kolekcja figur geometrycznych), `ST_MultiPoint` (kolekcja punktów), `ST_MultiLineString` (kolekcja `ST_LineString`) i `ST_MultiPolygon` (kolekcja `ST_Polygon`).

Spośród pozostałych typów SQL/MM Spatial należy wymienić `ST_SpatialRefSystem` służący do opisu przestrzennego układów odniesienia oraz typy `ST_Angle` i `ST_Direction` reprezentujące odpowiednio kąty i kierunki.



Metody typów SQL/MM Spatial

- Metody do odczytu właściwości i miar obiektów
 - np. ST_Boundary, ST_Length, ST_Area
- Metody do porównywania obiektów geometrycznych
 - ST_Equals, ST_Disjoint, ST_Intersects, ST_Crosses, ST_Overlaps, ST_Touches, ST_Within, ST.Contains i ST_Distance
- Metody do tworzenia nowych obiektów geometrycznych
 - ST_Difference, ST_Intersection i ST_Union
- Metody do konwersji z/do zewnętrznych formatów danych

Standard SQL/MM (15)

Metody typów SQL/MM Spatial można podzielić na 4 grupy:

- (a) metody do odczytu właściwości i miar dla obiektów geometrycznych,
- (b) metody do porównywania obiektów geometrycznych,
- (c) metody do tworzenia nowych obiektów geometrycznych,
- (d) metody do konwersji z/do zewnętrznych formatów danych (tekstowych, binarnych).

Przykłady metod do odczytu właściwości i miar dla obiektów geometrycznych to:
ST_Boundary zwracająca brzeg figury geometrycznej, ST_Length zwracająca długość krzywej, ST_Area zwracająca pole powierzchni figury 2-wymiarowej.

Metody do porównywania obiektów geometrycznych to ST_Equals (do testowania czy figury są równe), ST_Disjoint (do testowania czy figury są rozłączne), ST_Intersects, ST_Crosses i ST_Overlaps (trzy bardzo podobne metody do testowania czy wnętrza figur mają część wspólną), ST_Touches (do testowania czy figury stykają się brzegiem), ST_Within i STContains (do testowania czy jedna figura zawiera się w drugiej) oraz ST_Distance (do wyznaczania minimalnej odległości między punktami dwóch figur).

Metody do tworzenia nowych obiektów geometrycznych to przede wszystkim ST_Difference, ST_Intersection i ST_Union zwracające odpowiednio różnicę, część wspólną i sumę figur.

Przykładowe metody do konwersji danych to metody do konwersji między typami SQL/MM Spatial a formatem GML (Geography Markup Language), np. ST_MPointFromGML, ST_LineFromGML, ST_AsGML.



SQL/MM Spatial – Przykład

```
CREATE TABLE kraje (
    nazwa_kraju VARCHAR(30),
    lokalizacja ST_Geometry )
```

```
SELECT lokalizacja.area
FROM kraje
WHERE nazwa_kraju = 'POLSKA'
```

```
SELECT nazwa_kraju FROM kraje
WHERE lokalizacja.ST_Touches(
    SELECT lokalizacja FROM kraje
    WHERE nazwa_kraju = 'POLSKA' )
```

Standard SQL/MM (16)

Na slajdzie pokazano przykład wykorzystania możliwości SQL/MM Spatial z poziomu języka SQL. Pierwsze polecenie tworzy tabelę KRAJE, której pierwsza kolumna zawiera nazwy krajów, a druga kolumna figury geometryczne reprezentujące ich lokalizacje i kształty w postaci obiektów ST_GEOMETRY. Drugie polecenie to zapytanie zwracające pole powierzchni Polski. Trzecie polecenie to zapytanie zwracające nazwy krajów graniczących z Polską.



- Dotyczy obrazów nieruchomych (np. fotografii)
 - obrazy bitmapowe, wsparcie dla różnych formatów
- Zakres SQL/MM Still Image:
 - typy danych dla obrazów i ich właściwości
 - metody do modyfikacji obrazów
 - wyszukiwanie w oparciu o zawartość
- Nie obejmuje semantycznych opisów zawartości
- Specyfikacja zaimplementowana w Oracle10g

Standard SQL/MM (17)

Specyfikacja SQL/MM Still Image dotyczy obrazów nieruchomych, takich jak np. fotografie. SQL/MM Still Image przyjmuje, że obraz jest dwuwymiarową tablicą pikseli (obraz bitmapowy). Zakłada się, że implementacje standardu powinny wspierać wiele różnych formatów graficznych np. JPEG, GIF, TIFF.

Standard definiuje strukturalne typy SQL umożliwiające składowanie obrazów w bazie danych, reprezentację właściwości wizualnych obrazów, podstawowe operacje przetwarzania obrazów (skalowanie, obroty) oraz wyszukiwanie obrazów w oparciu o zawartość.

SQL/MM dla obrazów nie standaryzuje metod opisu zawartości semantycznej. Nie stanowi więc w tym zakresie konkurencji dla standardu MPEG-7.

Still Image jest obecnie (wiosna 2006) jedyną częścią standardu SQL/MM, która doczekała się implementacji w rzeczywistym systemie zarządzania bazą danych. SQL/MM Still Image jest wspierany przez system Oracle od wersji 10g.



Wyszukiwanie w oparciu o zawartość w SQL/MM Still Image

- Content-Based Image Retrieval
- Realizowane poprzez wyszukiwanie obrazów podobnych do wzorca cech wizualnych
- Uwzględniane właściwości wizualne obrazów:
 - średni kolor
 - histogram kolorów (udział kolorów w obrazie)
 - lokalizacja kolorów
 - tekstura
- Możliwość przypisania wag poszczególnym właściwościom
- Miara odległości obrazów

Standard SQL/MM (18)

SQL/MM Still Image specyfikuje typy danych i ich metody umożliwiające wyszukiwanie obrazów w oparciu o zawartość (ang. Content-Based Image Retrieval). Wyszukiwanie jest realizowane względem wzorca cech wizualnych, typowo wywiedzionego z obrazu podanego jako przykład. SQL/MM Still Image umożliwia uwzględnienie następujących właściwości wizualnych przy wyszukiwaniu:

- (a) średni kolor,
- (b) histogram kolorów (udział kolorów w obrazie),
- (c) lokalizacja kolorów,
- (d) tekstura.

Standard przewiduje możliwość przypisywania wag poszczególnym właściwościom i specyfikuje miarę podobieństwa (a właściwie odległości) obrazów.



Typy danych SQL/MM Still Image

- SI_StillImage – reprezentuje obraz
- SI_AverageColor – reprezentuje średni kolor obrazu
- SI_ColorHistogram – reprezentuje histogram kolorów
- SI_PositionalColor – reprezentuje lokalizację kolorów obrazu
- SI_Texture – reprezentuje teksturę obrazu
- SI_Color – reprezentuje kolor
- SI_FeatureList – reprezentuje listę właściwości wizualnych obrazu

Standard SQL/MM (19)

W standardzie SQL/MM obrazy są reprezentowane za pomocą typu SI_StillImage. Atrybuty typu SI_StillImage to: SI_content (obraz w postaci binarnej, typu Binary Large Object), SI_contentLength (rozmiar w bajtach), SI_reference (referencja typu DATALINK do zewnętrznego źródła danych przechowującego obraz), SI_format (format graficzny obrazu), SI_height (wysokość obrazu w pikselach), SI_width (szerokość obrazu w pikselach). Metody typu SI_StillImage pozwalają m.in. na skalowanie i obrót obrazu, konwersje między formatami graficznymi oraz generację miniaturki obrazu w mniejszej rozdzielczości (ang. thumbnail).

Oprócz podstawowego typu SI_StillImage, SQL/MM Still Image definiuje również kilka typów służących do reprezentacji właściwości obrazu. Typ SI_AverageColor reprezentuje średni kolor obrazu, SI_ColorHistogram reprezentuje udział kolorów w obrazie w formie histogramu, SI_PositionalColor reprezentuje lokalizację kolorów na obrazie, a SI_Texture służy do zapamiętania informacji o teksturze obrazu. Właściwości odnoszące się do kolorystyki obrazu reprezentują poszczególne kolory jako wartości pomocniczego typu danych SI_Color.

Ponadto, SQL/MM Still Image przewiduje jeszcze typ danych SI_FeatureList do reprezentacji listy właściwości wizualnych obrazu wraz z przypisanymi im wagami. Typ ten jest wykorzystywany do testów podobieństwa uwzględniających więcej niż jedną właściwość obrazu.



Metody typu SI_StillImage

- Konstruktory
- Metody do odczytu atrybutów
- Metody do zmiany zawartości binarnej obrazu
- Metody przetwarzania obrazu
- Metody do testów podobieństwa

Standard SQL/MM (20)

Metody typu SI_StillImage można podzielić na następujące grupy:

1. Konstruktory, m.in.: `SI_StillImage(BINARY LARGE OBJECT)` – tworzący obiekt `SI_StillImage`, dla podanego obiektu binarnego zawierającego obraz; `SI_StillImage(DATALINK)` – tworzący obiekt `SI_StillImage`, dla podanej referencji do obiektu binarnego zawierającego obraz.
2. Metody do odczytu atrybutów: `SI_Content` – zwracająca zawartość binarną w postaci `BINARY LARGE OBJECT`, `SI_ContentLength` – zwracająca rozmiar zawartości binarnej w bajtach, `SI_Height` – zwracająca wysokość obrazu w pikselach, `SI_Width` – zwracająca szerokość obrazu w pikselach, `SI_Format` – zwracająca format graficzny obrazu.
3. Metody do zmiany zawartości binarnej obrazu: `SI_SetContent` – ustawiająca nową zawartość binarną (podmiana obrazu), `SI_ChangeFormat` – dokonująca konwersji formatu obrazu.
4. Metody do przetwarzania obrazu: `SI_Scale`, `SI_Resize`, `SI_Thumbnail`, `SI_Rotate` (omówione na następnym slajdzie).
5. Metody do testów podobieństwa: przeciążona metoda `SI_Score` (omówiona na jednym z kolejnych slajdów).

Przetwarzanie obrazów



Slajd ilustruje działanie metod typu `SI_StillImage` do przetwarzania obrazu: `SI_Resize` – dokonującej zmiany rozmiaru obrazu, bez zachowania proporcji; `SI_Scale` – dostępnej w dwóch wersjach; skalującej obraz z zachowaniem proporcji w ramach podanego okna lub wg podanego współczynnika skalowania; `SI_Thumbnail` – generującej miniaturę obrazu; `SI_Rotate` – obracającej obraz o zadany kąt.



Właściwości wizualne obrazu

SI_ColorHistogram



SI_StillImage



SI_Texture



SI_AverageColor



SI_PositionalColor



Standard SQL/MM (22)

Slajd ilustruje znaczenie czterech dostępnych w SQL/MM Still Image typów właściwości wizualnych obrazu.

Każdą z właściwości można wyznaczyć dla obrazu za pomocą konstruktora typu właściwości, przyjmującego jako parametr obiekt SI_StillImage: SI_AverageColor(StillImage), SI_ColorHistogram(StillImage), SI_PositionalColor(StillImage), SI_Texture(StillImage).

Jako alternatywę dla powyższych konstruktorów (wygodną szczególnie w zapytaniach SQL) standard definiuje poniższe cztery funkcje SQL, których działanie sprowadza się do wywołania stosownych konstruktorów: SI_findAvgColor(StillImage), SI_findClrHstgr(StillImage), SI_findPstnlClr(StillImage), Si_findTexture(StillImage).

Sposób opisu tekstury, algorytm jej wyznaczania oraz algorytm określania podobieństwa względem tekstury nie są zawarte w standardzie i będą zależne od implementacji. Dla pozostałych trzech właściwości standard wskazuje sposób ich reprezentacji i ogólne algorytmy ich ekstrakcji z obrazu.



Algorytm wyznaczania średniego koloru

- Obraz jest próbkowany za pomocą siatki referencyjnej
- Komponenty koloru R, G i B z poszczególnych próbek są niezależnie od siebie sumowane i dzielone przez liczbę próbek

$$(\bar{R}, \bar{G}, \bar{B}) = \left(\frac{\sum_{i=1}^{SI_width} \sum_{j=1}^{SI_height} R(i, j)}{SI_width * SI_height}, \frac{\sum_{i=1}^{SI_width} \sum_{j=1}^{SI_height} G(i, j)}{SI_width * SI_height}, \frac{\sum_{i=1}^{SI_width} \sum_{j=1}^{SI_height} B(i, j)}{SI_width * SI_height} \right)$$

- Wynikowy średni kolor jest reprezentowany jako jedna właściwość typu SI_Color

Standard SQL/MM (23)

SQL/MM proponuje następujący algorytm wyznaczania średniego koloru dla obrazu. Najpierw obraz jest próbkowany tzw. siatką referencyjną, wskazującą piksele, z których będzie uśredniany kolor. Następnie komponenty koloru z poszczególnych próbek są niezależnie uśredniane poprzez zsumowanie ich i podział sum przez liczbę próbek. Na slajdzie przedstawiono wzór na średni kolor przy założeniu, że siatka referencyjna jest tworzona przez wszystkie piksele obrazu. Wynik uśredniania jest zapisany w obiekcie SI_AverageColor w jego właściwości typu SI_Color.



Algorytm generacji histogramu kolorów

- Przestrzeń kolorów dzielona jest na obszary
- Każdy obszar przestrzeni kolorów obejmuje pewien zbiór kolorów i jest reprezentowany przez jeden kolor C_i
- Dla każdego obszaru kolorów określana jest jego częstotliwość F_i w ramach obrazu poprzez iterację po wszystkich pikselach
- Każdy piksel zwiększa wartość F_i o 1 dla tego zakresu, do którego należy jego kolor
- Wartości F_i są normalizowane, aby zawierały się w zakresie od 0 do 100
- Wynikowy histogram jest sekwencją par (kolor, częstotliwość) w postaci dwóch tablic (ARRAY)

Standard SQL/MM (24)

SQL/MM proponuje następujący algorytm generacji histogramu kolorów dla obrazu.

Przestrzeń kolorów dzielona jest na pewną liczbę obszarów, zależną od implementacji. Każdy obszar przestrzeni kolorów obejmuje pewien zbiór kolorów i jest reprezentowany przez jeden kolor C_i . Dla każdego obszaru kolorów określana jest jego częstotliwość F_i w ramach obrazu, poprzez iterację po wszystkich pikselach. Każdy piksel zwiększa wartość licznika F_i o 1 dla tego obszaru kolorów, do którego należy jego kolor. Po zliczeniu częstotliwości, wartości F_i są normalizowane, aby zawierały się w zakresie od 0 do 100. Wynikowy histogram, logicznie będący sekwencją par (kolor, częstotliwość), fizycznie zapamiętywany jest w postaci dwóch tablic (ARRAY). Pierwsza z tablic zawiera kolory, a druga odpowiadające im częstotliwości. Tablice mają oczywiście taką samą liczbę elementów, zależną od implementacji, odpowiadającą maksymalnej dostępnej w implementacji długości histogramu.



Algorytm wyznaczania lokalizacji kolorów

- Obraz jest dzielony na „siatkę” $m \times n$ prostokątów
- Dla każdego z prostokątów wyznaczany jest dominujący kolor
- Wynikowa lokalizacja kolorów jest reprezentowana w postaci tablicy (ARRAY) obiektów SI_Color

Standard SQL/MM (25)

SQL/MM proponuje następujący algorytm ekstrakcji lokalizacji kolorów dla obrazu. Najpierw obraz dzielony jest na siatkę $m \times n$ prostokątów, o wymiarach zależnych od implementacji. Następnie dla każdego z prostokątów wyznaczany jest kolor dominujący, jako kolor o największej częstotliwości w histogramie kolorów wygenerowanym dla prostokąta (algorytmem przedstawionym na poprzednim slajdzie). Wynikowa lokalizacja kolorów jest reprezentowana w postaci tablicy (ARRAY) obiektów SI_Color.



Typ danych SI_FeatureList

- Służy do reprezentacji zbioru właściwości wizualnych
- Wykorzystywany do testów podobieństwa obrazów z uwzględnieniem więcej niż jednej właściwości
- Reprezentuje złożony wzorzec do testów podobieństwa
- Umożliwia przypisanie wag poszczególnym właściwościom

Standard SQL/MM (26)

Zbiór właściwości wizualnych obrazu można zapamiętać jako jeden obiekt typu SI_FeatureList. Podstawowym przeznaczeniem tego typu danych jest umożliwienie przeprowadzania testów podobieństwa obrazów z uwzględnieniem więcej niż jednej właściwości. Obiekty typu SI_FeatureList stanowią złożone wzorce do testów podobieństwa obrazów, w których każdej uwzględnionej właściwości przypisana jest waga. Waga jest liczbą zmiennoprzecinkową z przedziału <0.0, 1.0>.

Złożony wzorzec można utworzyć następującym konstruktorem typu SI_FeatureList:
SI_FeatureList(SI_AverageColor, DOUBLE PRECISION, SI_ColorHistogram, DOUBLE PRECISION, SI_PositionalColor, DOUBLE PRECISION, SI_Texture, DOUBLE PRECISION). Parametry typu zmiennoprzecinkowego to wagi przypisane poprzedzającym je na liście argumentów właściwościom. W przypadku gdy dana właściwość ma nie być zawarta we wzorcu, należy podać NULL jako wartość właściwości i jej wagi lub przypisać właściwości wagę 0.

Wzorzec reprezentowany przez obiekt SI_FeatureList można modyfikować za pomocą metod SI_SetFeature, z których każda ustawia wartość danej właściwości i przypisuje jej wagę:
SI_SetFeature(SI_AverageColor, DOUBLE PRECISION), SI_SetFeature(SI_ColorHistogram, DOUBLE PRECISION), SI_SetFeature(SI_PositionalColor, DOUBLE PRECISION),
SI_SetFeature(SI_Texture, DOUBLE PRECISION).



Testowanie podobieństwa obrazów

- Metody `SI_Score` typu `SI_StillImage` z argumentem typu pojedynczej właściwości lub `SI_FeatureList`
- Metody `SI_Score` typów pojedynczych właściwości lub `SI_FeatureList` z argumentem typu `SI_StillImage`
- Brak metody do bezpośredniego porównywania dwóch obiektów `SI_StillImage` ze sobą!
- Wartości `SI_Score >= 0` (0 – najlepsze dopasowanie)
- Dla testów poprzez `SI_FeatureList` wynik `SI_Score` to suma ważona wyników testów dla właściwości składowych

$$\frac{\sum_{i=1}^N F_i \cdot SI_Score(image) \cdot W_i}{\sum_{i=1}^N W_i}.$$

Standard SQL/MM (27)

Do realizacji testów podobieństwa na potrzeby wyszukiwania obrazów w oparciu o zawartość służą metody:

1. Przeciążona metoda `SI_Score` typu `SI_StillImage`: `SI_Score(SI_AverageColor)`, `SI_Score(SI_ColorHistogram)`, `SI_Score(SI_PositionalColor)`, `SI_Score(SI_Texture)`, `SI_Score(SI_FeatureList)`.
2. Metody `SI_Score(SI_StillImage)` typów `SI_AverageColor`, `SI_ColorHistogram`, `SI_PositionalColor`, `SI_Texture` i `SI_FeatureList`.

Zwraca uwagę brak metody do bezpośredniego porównywania dwóch obiektów `SI_StillImage` ze sobą. W celu porównania dwóch obrazów należy najpierw z jednego z nich wyekstrahować właściwość lub właściwości wizualne, które mają być uwzględnione w teście podobieństwa. Jeśli test ma dotyczyć więcej niż jednej właściwości, należy w kolejnym kroku utworzyć obiekt `SI_FeatureList` przypisując poszczególnym właściwościom wagę.

Wynikiem zwracanym przez metody `SI_Score` jest nieujemna liczba zmiennoprzecinkowa. Im mniejsza wartość tym lepiej wzorzec właściwości reprezentuje porównywany z nim obraz. Wartość 0 oznacza najlepsze dopasowanie do wzorca.

W przypadku testów realizowanych z wykorzystaniem obiektu `SI_FeatureList`, końcowy wynik metody `SI_Score` jest średnią ważoną wyników metod `SI_Score` dla poszczególnych właściwości, co ilustruje wzór przedstawiony na slajdzie. N jest liczbą właściwości zawartych w obiekcie `SI_FeatureList`, F_i to i -ta właściwość, a W_i jej waga.



SQL/MM Still Image – Przykład 1

```
CREATE TABLE flagi
(panstwo VARCHAR2(40),
 flaga SI_StillImage);
```

```
SELECT f1.flaga
FROM flagi f1
WHERE SI_findAvgClr(
    (SELECT f2.flaga FROM flagi f2
     WHERE f2.panstwo = 'Polska')
).SI_Score(f1.flaga) < 30;
```

Standard SQL/MM (28)

Na slajdzie pokazano przykład wykorzystania możliwości SQL/MM Still Image z poziomu języka SQL. Pierwsze polecenie tworzy tabelę FLAGI, której pierwsza kolumna zawiera nazwy państw, a druga kolumna obrazki reprezentujące ich flagi w postaci obiektów SI_StillImage. Drugie polecenie to zapytanie zwracające flagi podobne do polskiej w sensie średniego koloru. W podzapytaniu z bazy danych jest odczytywany obrazek z polską flagą, następnie wyznaczany jest dla niego funkcją SQL SI_findAvgClr średni kolor jako wzorzec do testów podobieństwa. W warunku WHERE zapytania zewnętrznego testowana jest odległość poszczególnych obrazów z bazy danych od tego wzorca średniego koloru. Za podobne uznawane są obrazy, dla których odległość od wzorca jest mniejsza niż 30.

Należy podkreślić, że dobór progu podobieństwa (w przykładzie: 30) jest zadaniem trudnym. Selektynośc danej wartości progowej zależy od natury przeszukiwanej kolekcji obrazów. Dlatego też aby odpowiednio dobrać wartość progu odległości, może być konieczne wykonanie zapytania dla kilku różnych wartości progu aż do uzyskania satysfakcjonujących wyników.



SQL/MM Still Image – Przykład 2 (1/2)

```

DECLARE
    tempimage      SI_StillImage;
    tempAvgColor   SI_AverageColor;
    tempTexture    SI_Texture;
    myFeatureList  SI_FeatureList;
    score          DOUBLE PRECISION;
BEGIN
    SELECT flaga INTO tempimage FROM flagi
    WHERE państwo = 'Polska';

    tempAvgColor := NEW SI_AverageColor(tempimage);
    tempTexture := NEW SI_Texture(tempimage);
    myFeatureList := NEW SI_FeatureList(
        tempAvgColor, 0.7, NULL, NULL,
        NULL, NULL, tempTexture, 0.3);
    ...

```

1

2

3

4

Standard SQL/MM (29)

Ten i następny slajd pokazują przykład wyszukania obrazów podobnych do zadanego z uwzględnieniem więcej niż jednej właściwości. W takim wypadku niezbędne jest najpierw utworzenie złożonego wzorca podobieństwa obrazów w postaci obiektu SI_FeatureList, a następnie wyznaczenie wartości odległości poszczególnych obrazów od tego wzorca. Odpowiednią sekwencję operacji wygodnie można zapisać w proceduralnym języku pozwalającym na zagnieżdżanie poleceń SQL, takim jak np. PL/SQL na platformie Oracle.

Niniejszy slajd pokazuje pierwszą część anonimowego bloku PL/SQL, którego zadaniem jest wyświetlenie nazw państw, których flagi są podobne do polskiej w sensie średniego koloru i tekstury, przy czym średni kolor ma mieć większą wagę (0.7) niż tekstura (0.3).

W sekcji deklaracji zmiennych (1) deklarowane są kolejno zmienne, w których pamiętane będą: obrazek z polską flagą, jego średni kolor, jego tekstura, wzorzec do testów podobieństwa, wynik bieżącego testu podobieństwa. Działanie bloku kodu rozpoczyna się pobraniem z bazy danych obrazka z polską flagą do zmiennej w programie (2). Następnie za pomocą konstruktorów SI_AverageColor i SI_Texture wyznaczane są z obrazka jego właściwości średniego koloru i tekstury (3). Wreszcie z tych dwóch właściwości tworzony jest obiekt SI_FeatureList z uwzględnieniem zadanych wag (4).



SQL/MM Still Image – Przykład 2 (2/2)

```
...
FOR c IN (SELECT panstwo, flaga FROM flagi) LOOP
    score := myFeatureList.SI_Score(c.flaga);
    IF score < 10 THEN
        dbms_output.put_line(c.panstwo);
    END IF;
END LOOP;
END;
/
```

5

6

7

Standard SQL/MM (30)

Po przygotowaniu złożonego wzorca do testów podobieństwa, w pętli FOR z podzapytaniem przeglądane są kolejne flagi odczytane z bazy danych (5). Dla każdej z nich metodą SI_Score wyznaczana jest jej odległość od wzorca (6). Jeśli wartość ta jest mniejsza niż 10, nazwa państwa wyświetlana jest na konsoli (7).



- Biblioteka typów i ich metod do eksploracji danych
- Ma umożliwiać wymianę modeli między systemami
- Obecnie wspiera cztery metody eksploracji danych:
 - odkrywanie reguł asocjacyjnych
 - klasyfikację
 - analizę skupień
 - regresję
- Jeden z wielu standardów dotyczących eksploracji danych

Standard SQL/MM (31)

Eksploracja danych (ang. data mining) to odkrywanie wcześniej nieznanych, nietrywialnych, potencjalnie interesujących wzorców, reguł, modeli z dużych zbiorów danych. SQL/MM Data Mining stanowi próbę dostarczenia standardowych interfejsów do algorytmów eksploracji danych na gruncie obiektowo-relacyjnych baz danych. Ma na celu umożliwienie wymiany modeli eksploracji danych między różnymi systemami. Standard nie określa algorytmów implementujących poszczególne modele ani formatu przechowywanych danych dla eksploracji danych.

Obecnie SQL/MM Data Mining wspiera cztery podstawowe metody eksploracji danych: odkrywanie reguł asocjacyjnych, klasyfikację, analizę skupień i regresję.

SQL/MM Data Mining jest jednym z wielu standardów dotyczących eksploracji danych. Rozważane jest uzgodnienie go w przyszłości ze standardem specyfikującym interfejs do eksploracji danych dla języka Java (Java Data Mining API).



- Obsługa historii zmian dla tabel bazy danych
 - składowanie danych historycznych
 - zarządzanie danymi historycznymi
 - udostępnianie danych historycznych
- Historia tabeli rozumiana jako sekwencja operacji INSERT, UPDATE i DELETE wykonanych na niej w przeszłości

Standard SQL/MM (32)

Ostatnią obecnie częścią standardu SQL/MM jest część siódma – History. Nie doczekała się ona jeszcze statusu obowiązującego standardu. Aktualnie (wiosna 2006) jest w stadium Working Draft.

SQL/MM History dotyczy obsługi historii zmian dla tabel bazy danych, obejmującej składowanie danych historycznych, zarządzanie nimi i udostępnianie ich z poziomu zapytań SQL.

Historia tabeli rozumiana jako sekwencja operacji INSERT, UPDATE i DELETE wykonanych na niej w przeszłości.

SQL/MM History ma umożliwiać np. sprawdzenie jaką wartość miał atrybut wiersza tabeli w danym momencie w przeszłości.



Podsumowanie

- SQL/MM uzupełnia SQL o biblioteki do obsługi specjalistycznych danych i aplikacji
- SQL/MM: SQL Multimedia and Application Packages
- Jeszcze za wcześnie by mówić o akceptacji standardu
 - jedyna istniejąca implementacja to SQL/MM Still Image w Oracle 10g
 - funkcjonalność definiowana przez standard dostępna w istniejących SZBD, ale inna składnia

Standard SQL/MM (33)

SQL/MM uzupełnia język SQL o biblioteki do obsługi specjalistycznych danych i aplikacji. Ma z założenia dotyczyć szeroko pojętych danych multimedialnych i zaawansowanych zastosowań systemów baz danych. Obecnie obejmuje obsługę danych tekstowych, przestrzennych i obrazów oraz eksplorację danych i obsługę danych historycznych.

W chwili obecnej (wiosna 2006) trudno jeszcze mówić o sukcesie standardu. SQL/MM jest ciągle rozwijany, ale prawie nie pojawiają się jego implementacje. Jedyną zaimplementowaną częścią standardu jest SQL/MM Still Image, wspierany w Oracle 10g. Dostępne systemy zarządzania bazami danych w dużym stopniu oferują już funkcjonalność przewidywaną przez standard SQL/MM. Pozostaje jednak kwestia dostosowania interfejsu do wyznaczonego przez standard.



Materiały dodatkowe

- ISO/IEC 13249, Information Technology – Database Languages – SQL Multimedia and Application Packages (specyfikacja standardu ISO)
- Melton J., Eisenberg A.: SQL Multimedia and Application Packages (SQL/MM). SIGMOD Record 30(4), 2001
- Stolze K.: SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems. BTW 2003

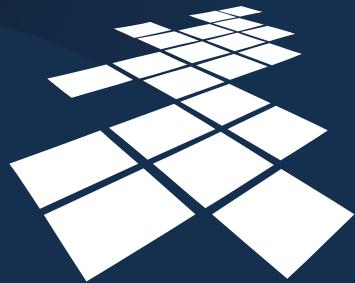
Standard SQL/MM (34)

Hurtownie danych – 1

Problematyka hurtowni danych

Wykład przygotował:

Robert Wrembel



UCZELNIA
ONLINE

ZSBD – wykład 12 (1)



Plan wykładu

- Problematyka integracji danych
- Integracja danych za pomocą hurtowni danych
- Przetwarzanie analityczne OLAP
- Model wielowymiarowy
- Implementacje modelu wielowymiarowego
 - ROLAP
 - MOLAP

ZSBD – wykład 12 (2)

Celem niniejszego wykładu jest wprowadzenie do zagadnień integracji danych i technologii hurtowni danych. W ramach wykładu zostaną omówione następujące zagadnienia:

- wprowadzenie do problematyki integracji rozproszonych i heterogenicznych baz danych,
- podstawowa architektura integracji danych oparta o hurtownię danych,
- wprowadzenie i charakterystyka przetwarzania analitycznego (On-Line Analytical Processing - OLAP),
- wielowymiarowy model danych,
- implementacja modelu wielowymiarowego w serwerach relacyjnych (ROLAP) i wielowymiarowych (MOLAP).



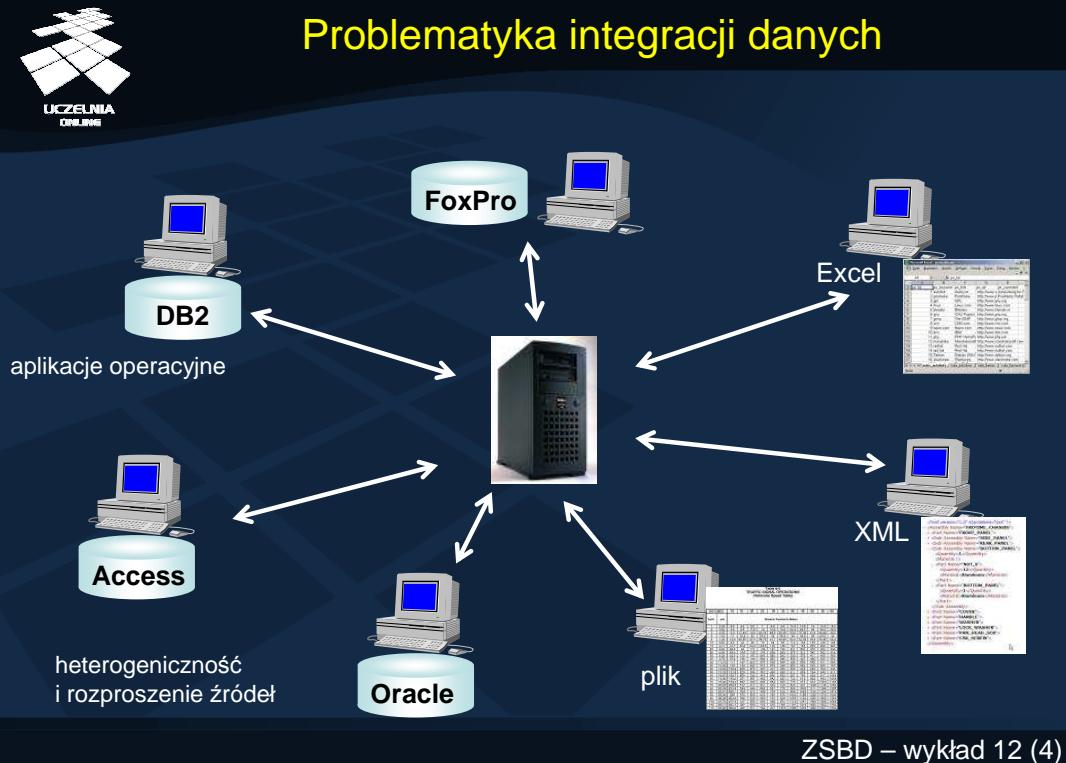
Cechy systemów informatycznych

- Składowanie, przetwarzanie i zapewnienie efektywnego dostępu do danych w przedsiębiorstwie
- Akutalny stan technologiczny systemów informatycznych
 - heterogeniczność
 - rozproszenie
- Heterogeniczność
 - wielość struktur danych
 - różna funkcjonalność
 - różne modele danych
- Rozproszenie
 - dane są rozmieszczone w geograficznie różnych lokalizacjach

ZSBD – wykład 12 (3)

Główym zadaniem systemów informatycznych jest przyspieszenie i ułatwienie pracy poprzez efektywny dostęp, przetwarzanie i składowanie danych. Dane gromadzone przez firmy i instytucje są często przechowywane w heterogenicznych i rozproszonych systemach informatycznych.

Heterogeniczność oznacza, że systemy posiadają różne struktury, funkcjonalność i wykorzystują różne modele danych (np. hierarchiczne, relacyjne, obiektowe, semistrukturalne), przechowują dane w dokumentach tekstowych, czy arkuszach kalkulacyjnych. Często nawet w ramach tej samej instytucji wykorzystuje się różne systemy informatyczne. Dodatkowym problemem w dostępie do informacji jest geograficzne rozproszenie źródeł danych.



Problematykę integracji danych ilustruje niniejszy slajd.



SI we wspomaganiu zarządzaniem

- Wspomaganie zarządzaniem bazują na analizie danych
- Dane opisują bieżący stan i historię działania firmy
- Analizie powinny podlegać możliwie wszystkie dane opisujące analizowany fragment działalności firmy
- Problem
 - dane są składowane w systemach heterogenicznych
 - dane są składowane w systemach rozproszonych
 - nieefektywny dostęp i analiza danych
- Rozwiązanie
 - integracja danych na platformie hurtowni danych

ZSBD – wykład 12 (5)

Racjonalizacja działania firm i instytucji wymaga stosowania procesów wspomagania decyzji kadry zarządzającej. Procesy wspomagania decyzji bazują na danych analitycznych opisujących bieżący stan i historię działania danej firmy. Programowe narzędzia analityczne, na podstawie danych elementarnych będących wynikiem pracy pojedynczych pracowników, powinny udostępniać informacje statystyczne o bieżącym stanie firmy, występujących trendach i korelacjach między różnymi czynnikami. Dzięki szybkiej analizie bazującej na pełnej i aktualnej informacji o stanie firmy, kadra zarządzająca może podejmować trafniejsze decyzje o strategicznym znaczeniu dla rozwoju danego przedsiębiorstwa.

W celu zapewnienia decydentom pełnego dostępu do heterogenicznych i rozproszonych danych, buduje się systemy integrujące. Jedną z najczęściej stosowanych technik integracji danych jest ich transformacja do wspólnego modelu i składowanie w centralnym systemie, zwany hurtownią (magazynem) danych (ang. data warehouse).



Hurtownia danych

- Bardzo duża baza danych (dziesiątki, setki TB)
- Charakterystyka:
 - dane są wyłącznie odczytywane przez użytkowników
 - zawiera dane historyczne i bieżące
 - zawiera dane zagregowane na wielu poziomach
 - zawartość jest zorientowana tematycznie

ZSBD – wykład 12 (6)

Hurtownia danych jest bardzo dużą bazą danych (często rzędu dziesiątek czy setek terabajtów). Hurtownia danych posiada następującą charakterystykę:

- dane w niej składowane nie są modyfikowane przez użytkowników, są natomiast wyłącznie przez nich odczytywane,
- zawiera wszystkie dane historyczne i bieżące,
- zawiera dane zagregowane na wielu poziomach szczegółowości,
- zawartość hurtowni jest zorientowana tematycznie, np. hurtownia danych nt. ruchu telefonicznego i opłat klientów sieci komórkowej, hurtownia danych nt. sprzedaży pojazdów.



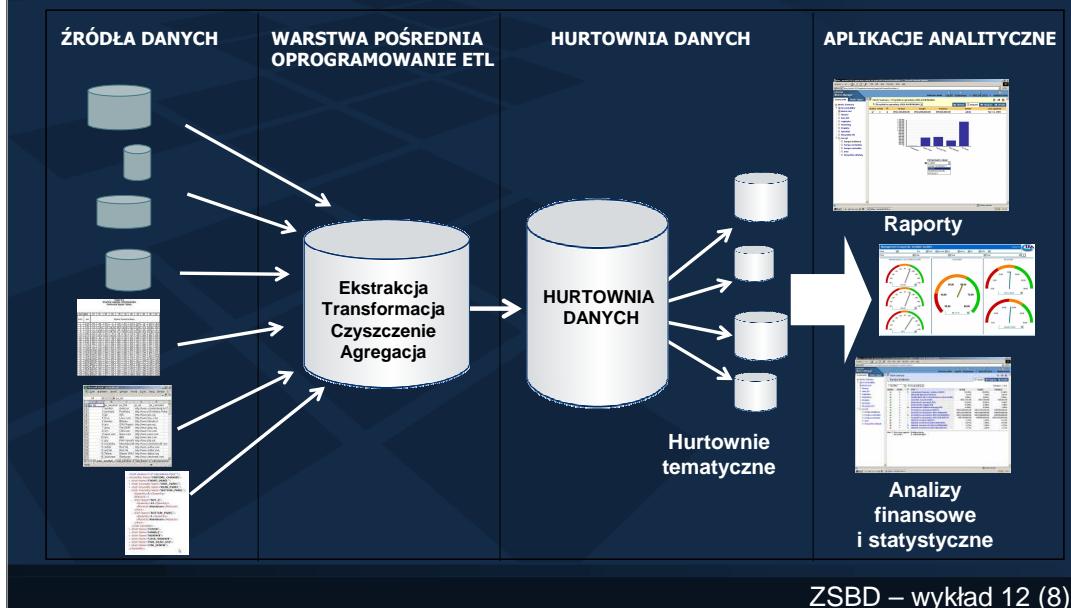
Systemy komercyjne

- Oracle8i, Oracle9i, Oracle10g – Oracle Corporation,
- DB2 UDB – IBM,
- SybaseIQ – Sybase, Inc.,
- SAS Enterprise BI Server - SAS Institute
- MS SQL Server – Microsoft,
- SAP Business Warehouse – SAP,
- Adabas C i Adabas D – Software AG,
- Teradata – NCR Corporation,
- Hyperion Essbase OLAP Server – Hyperion Solutions Corporation
- Red Brick Warehouse – Red Brick Systems

ZSBD – wykład 12 (7)



Podstawowa architektura HD

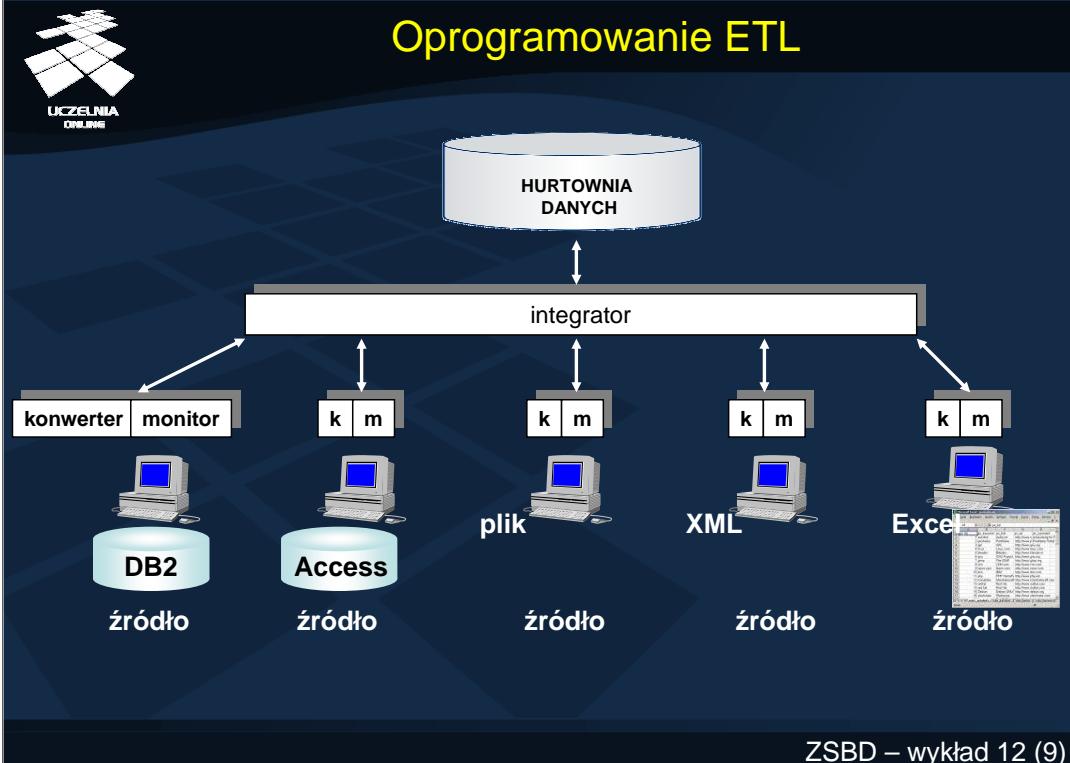


ZSBD – wykład 12 (8)

Podstawową poglądową architekturę hurtowni danych przedstawiono na slajdzie. Heterogeniczne i rozproszone źródła danych zasilają hurtownię danymi za pośrednictwem warstwy oprogramowania ETL. Jego podstawowymi zadaniami są wykrywanie zmian w źródłach, transformacja danych do wspólnej postaci, uspójnianie i czyszczenie danych, agregowanie danych. Uspójnione dane są następnie ładowane do centralnej hurtowni danych.

Centralna hurtownia danych najczęściej zawiera dane dla wszystkich grup decydentów. Ze względu na ilość danych, wygodniej jest zbudować na bazie hurtowni centralnej, małe hurtownie tematyczne (ang. data marts), zawierające dane opisujące wąskie dziedziny funkcjonowania firmy. Taka hurtownia tematyczna zawiera często dane na jeszcze wyższym poziomie agregacji niż hurtownia centralna. Przykładowo, hurtownia danych nt. ruchu telefonicznego i opłat klientów sieci komórkowej może posłużyć do zbudowania dwóch data marts opisujących odpowiednio natężenie ruchu telefonicznego w ciągu dnia i ranking klientów ze względu na płacone rachunki.

Na hurtowni danych pracują tzw. aplikacje analityczne wspomagania decyzji (ang. decision support), czy odkrywania wiedzy (ang. data mining), których celem jest wspomaganie pracy kadry zarządzającej poprzez analizę trendów, anomalii, poszukiwanie reguł zachowań.



ZSBD – wykład 12 (9)

Oprogramowanie ETL (Extraction Translation Loading) realizuje tzw. procesy ETL, składające się z trzech następujących faz:

- odczytu danych ze źródeł (Extraction),
- transformacji ich do wspólnego modelu wykorzystywanego w magazynie wraz z usunięciem wszelkich niespójności (Translation),
- wczytanie danych do magazynu (Loading). Na slajdzie przedstawiono podstawowe komponenty oprogramowania ETL.

Obiekty oznaczone jako źródło reprezentują heterogeniczne i rozproszone źródła danych. Z każdym z takich źródeł jest związana dedykowana dla niego warstwa oprogramowania o nazwie konwerter/monitor.

Zadaniem modułu konwertera jest transformowanie danych z formatu wykorzystywanego w źródle, do formatu wykorzystywanego w hurtowni. Dlatego, dla każdego modelu danych źródłowych konieczne jest zastosowanie specyficznego modułu konwertera. Przykładowo, jeśli źródło przechowuje dane w dokumentach tekstowych, a hurtownia została zaprojektowana z wykorzystaniem modelu relacyjnego, to konwerter musi zapewnić poprawne odwzorowanie danych z plików w strukturę modelu relacyjnego.

Zadaniem modułu monitora jest wykrywanie zmian w danych źródłowych i ich przekazywanie do warstwy oprogramowania integratora (po uprzedniej konwersji do modelu danych hurtowni). Sposób wykrywania zmian w danych źródłowych zależy od własności samych źródeł.



Wykrywanie zmian w źródłach

- Źródła aktywne
- Źródła utrzymujące dzienniki operacji
- Źródła przepływalne
- Źródła wspierające migawki

ZSBD – wykład 12 (10)

Wyróżnia się cztery następujące rodzaje źródeł danych:

- aktywne (ang. active sources),
- utrzymujące dzienniki operacji wykonywanych na danych źródłowych (ang. logged sources),
- przepływalne (ang. queryable sources),
- wspierające mechanizm migawek (ang. snapshot sources).

Źródła aktywne posiadają zaimplementowane mechanizmy wyzwalaczy, które informują monitor o zmianach zachodzących w danych źródłowych.

W źródłach utrzymujących dzienniki operacji zmiany są wykrywane poprzez analizę zawartości dziennika w module monitora.

Źródła przepływalne umożliwiają wydawanie zapytań i w celu wykrycia zmian w danych źródłowych monitor okresowo wydaje zapytania do takich źródeł.

Źródła wspierające mechanizm migawek umożliwiają tworzenie migawek, czyli obrazów stanu źródła z określonego momentu. Porównanie migawek z kolejnych momentów umożliwia wykrycie zmian.



Przetwarzanie analityczne OLAP (1)

- Aplikacje analityczne
 - wspomagania decyzji
 - eksploracji danych
- Zorientowane na wspieranie procesów decyzyjnych
 - wykonywanie zaawansowanych analiz, wspomagających zarządzanie przedsiębiorstwem, np.
 - analiza trendów sprzedaży
 - analiza nakładów reklamowych i zysków
 - analiza ruchu telefonicznego

ZSBD – wykład 12 (11)

Na hurtowni danych pracują tzw. aplikacje analityczne wspomagania decyzji (ang. decision support), czy eksploracji danych (ang. data mining). Aplikacje analityczne (ang. On-Line Analytical Processing - OLAP) są zorientowane na wspieranie procesów decyzyjnych, czyli przetwarzanie danych historycznych i zagregowanych. Przykładami takich zapytań mogą być: *Jaki jest trend sprzedaży towarów z branży AGD w ostatnich kilku tygodniach? Jaki jest rozkład sprzedaży lodówek w województwie wielkopolskim?*



Przetwarzanie analityczne OLAP (2)

- Operacje
 - łączenia (kilka, kilkanaście, kilkadziesiąt tabel),
 - filtrowania,
 - agregowania (np. suma, średnia)
- Dostęp do ogromnych wolumenów danych (miliony, dziesiątki, setki milionów rekordów)
- Czas realizacji zapytań analitycznych: godziny, dziesiątki godzin
- Problem efektywności, czasu odpowiedzi na zapytanie OLAP

ZSBD – wykład 12 (12)

Jak wspomniano, większość operacji realizowanych przez aplikacje analityczne obejmuje złożone zapytania wykorzystujące łączenie wielu tabel, filtrowanie, agregowanie, operujące na milionach rekordów. W konsekwencji, czasy przetwarzania danych sięgają nawet dziesiątek godzin. Kluczowym parametrem efektywności tego typu systemów jest czas odpowiedzi na zapytania analityczne.



Wielowymiarowy model danych (1)

- Dane zorganizowane w postaci wielowymiarowego modelu danych (ang. Multidimensional Data Model)
 - fakty
 - wymiary
- Fakty
 - informacje podlegające analizie
 - sprzedaż, rozmowy telefoniczne, ubezpieczenia
 - charakteryzowane ilościowo za pomocą miar
 - liczba sprzedanych sztuk towaru, czas trwania rozmowy, kwota ubezpieczenia

ZSBD – wykład 12 (13)

Dane w magazynie są zorganizowane w postaci tzw. modelu wielowymiarowego (ang. multidimensional data model), w którym wyróżnia się dwie podstawowe kategorie danych, tj. fakty i wymiary.

Fakty (ang. facts) reprezentują informacje podlegające analizie, np. fakt sprzedaży produktu, fakt wykonania rozmowy telefonicznej, fakt ubezpieczenia pojazdu. Fakty są charakteryzowane ilościowo za pomocą cech zwanych **miarami** (ang. measures). Przykładowo, miarą jest liczba zakupionych produktów, czas trwania rozmowy, kwota ubezpieczenia.



Wielowymiarowy model danych (2)

- Wymiary
 - ustalają kontekst analizy
 - sprzedaż czekolady (produkt) w Auchan (sklep) w poszczególnych miesiącach roku (czas)
 - składają się z poziomów, które tworzą hierarchię
 - zależności hierarchiczne między poziomami tworzą tzw. strukturę wymiaru



ZSBD – wykład 12 (14)

Wymiary (ang. dimensions) ustalają kontekst analizy. Przykładowo, analiza sprzedaży czekolady w Auchan, w poszczególnych miesiącach roku jest dokonywana w wymiarze *Produktu*, *Sklepu* i *Czasu*. Inne typowe wymiary to *Lokalizacja* lub *Klient*. Wymiary składają się z **poziomów**, które tworzą hierarchie. Jako przykład można podać wymiar *Lokalizacja* złożony z trzech następujących poziomów: *Sklepy*, *Miasta* i *Województwa*. Wymiar ten ustala hierarchię, w której sklepy należą do miast, a miasta do województw.



Implementacje wielowymiarowego MD

- ROLAP - implementacja w serwerach relacyjnych
 - fakty przechowywane w tabelach faktów
 - wymiary przechowywane w tabelach wymiarów
- MOLAP - implementacja w serwerach wielowymiarowych
 - dane przechowywane w wielowymiarowych tabelach (ang. data cubes), zwanych potocznie kostkami
- HOLAP - implementacja hybrydowa (relacyjno-wielowymiarowa)
 - dane elementarne przechowywane w tabelach
 - dane zagregowane przechowywane w kostkach

ZSBD – wykład 12 (15)

Model wielowymiarowy może być implementowany albo w serwerach relacyjnych (tzw. ROLAP) albo w serwerach wielowymiarowych (tzw. MOLAP). W pierwszym przypadku, dane są składowane w tabelach relacyjnych. W drugim przypadku, są one składowane w wielowymiarowych tablicach. Często w tej samej bazie danych reprezentuje się informacje częściowo w implementacji ROLAP, a częściowo w MOLAP. Taki sposób reprezentacji nazywa się hybrydowym – HOLAP (ang. Hybrid OLAP).



Implementacja ROLAP

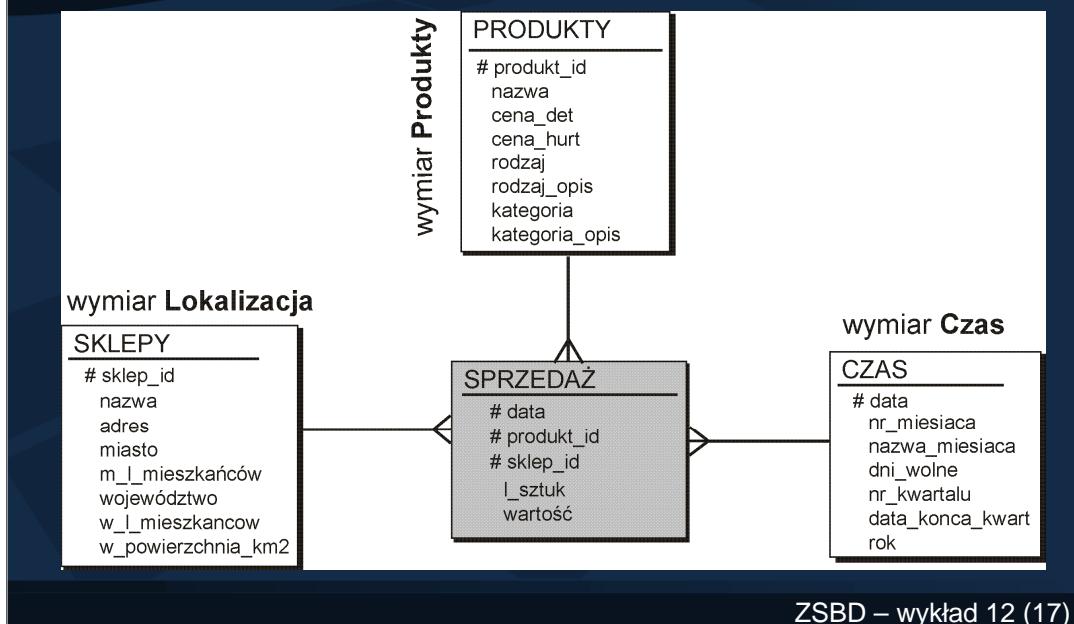
- Schematy podstawowe
 - gwiazda (ang. star schema)
 - płatek śniegu (ang. snowflake schema)
- Schematy pochodne
 - konstelacja faktów (ang. fact constellation schema)
 - gwiazda-płatek śniegu (ang. starflake schema)

ZSBD – wykład 12 (16)

Magazyn danych w technologii ROLAP jest implementowany w postaci tabel, których schemat posiada najczęściej strukturę *gwiazdy* (ang. star schema) lub *płatka śniegu* (ang. snowflake schema) lub *konstelacji faktów* (ang. fact constellation schema) lub strukturę *gwiazda–płatek śniegu* (ang. starflake schema).



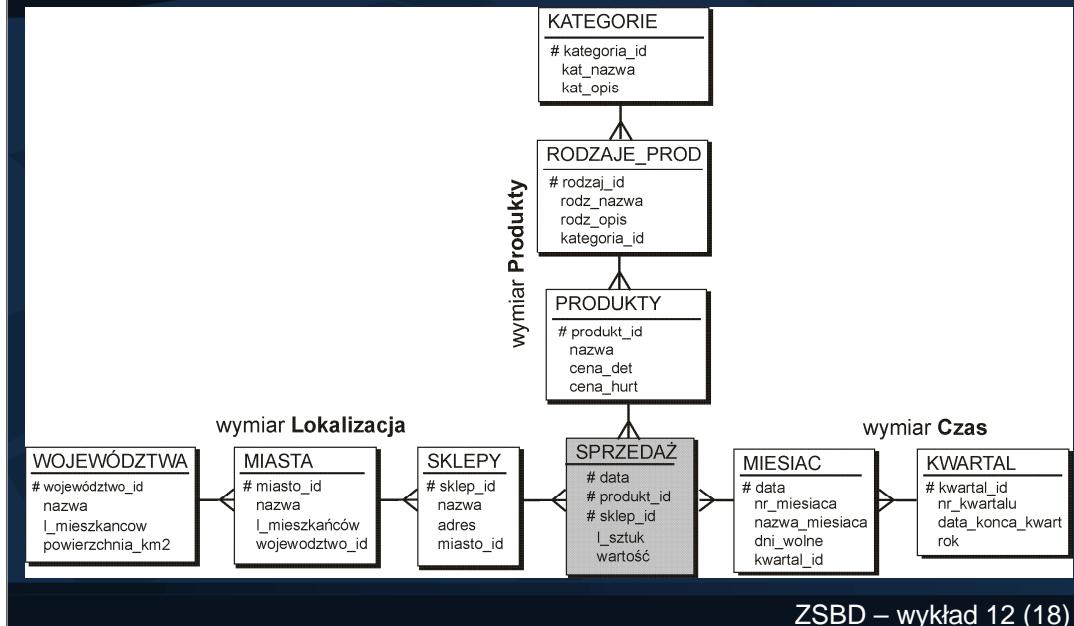
Schemat gwiazdy



Przykładowy schemat gwiazdy przedstawia slajd. Centralna tabela *Sprzedaż* zawiera informacje o sprzedaży pewnych produktów, w pewnych sklepach, w określonym czasie. Tabele *Sklepy*, *Produkty* i *Czas* są nazywane tabelami wymiarów (ang. dimension tables), natomiast tabela centralna jest nazywana tabelą faktów (ang. fact table). Atrybuty miar tabeli *Sprzedaż* to *wartość* i *l_sztuk*. Tabela faktów zawiera również atrybuty *produkt_id*, *sklep_id*, *data*. Są to klucze obce, których wartości wskazują na odpowiednie wymiary. W takim schemacie tabele wymiarów, tj. *Sklepy*, *Produkty* i *Czas* są zdenormalizowane (nie spełniają przynajmniej 3 postaci normalnej).



Schemat płatka śniegu

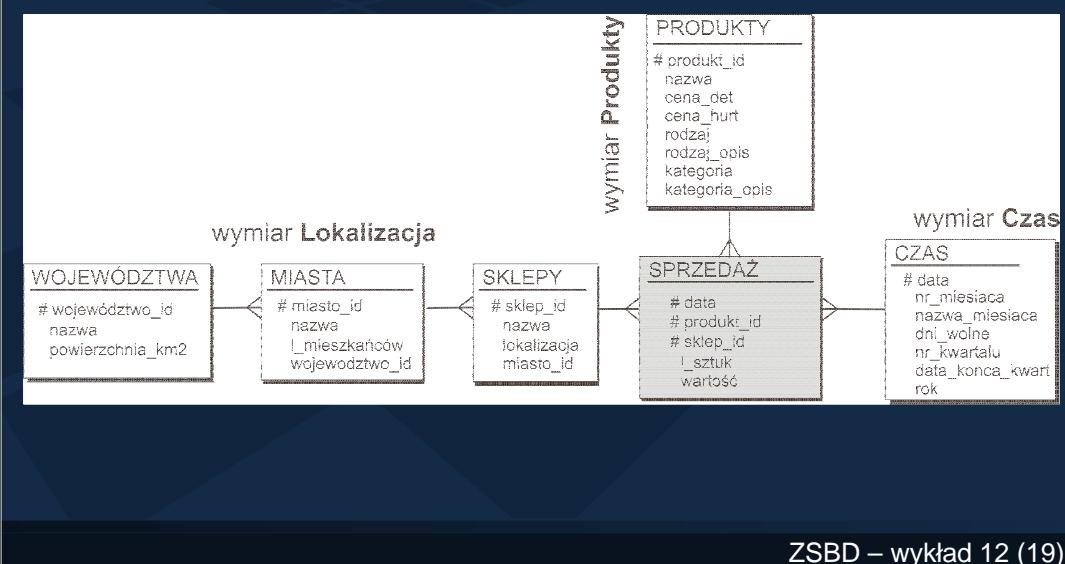


ZSBD – wykład 12 (18)

Jeśli wymiary są znormalizowane (spełniają przynajmniej 3 postać normalną), wówczas schemat magazynu danych ma postać *platka śniegu*. Przykładowy schemat o takiej strukturze został przedstawiony na slajdzie. W tym przypadku, wymiary *Lokalizacja*, *Produkty* i *Czas* mają postać hierarchii. Przykładowo, wymiarze *Lokalizacja* każdy sklep (tabela *Sklepy*) znajduje się w mieście (tabela *Miasta*), które z kolei znajduje się w województwie (tabela *Województwa*). Podobnie, w wymiarze produktów istnieje jawną hierarchię, w której produkty należą do rodzajów, a rodzaje do kategorii.



Schemat gwiazda-płatek śniegu



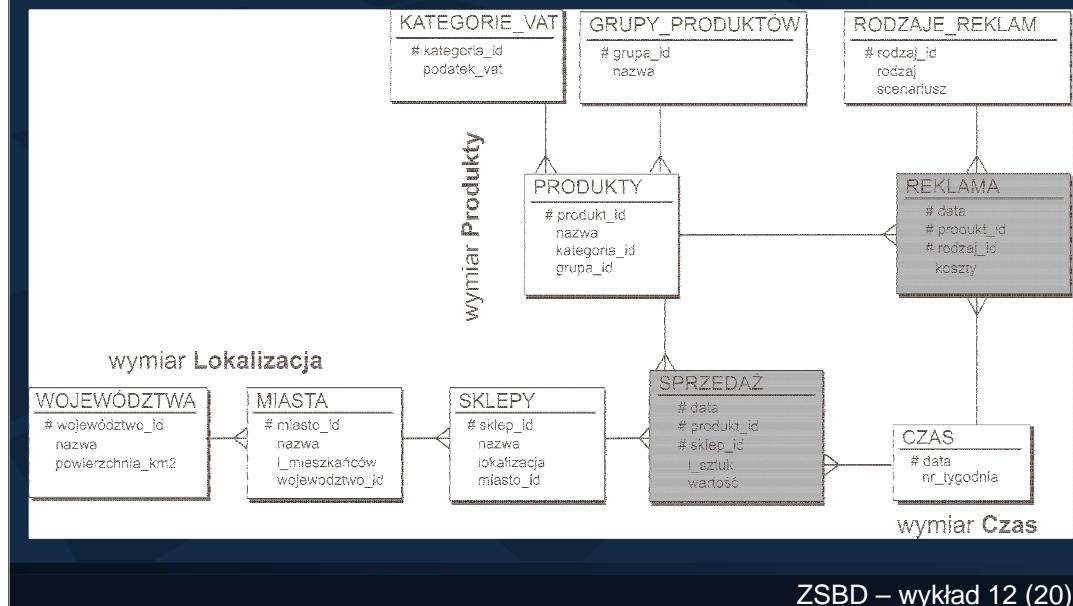
ZSBD – wykład 12 (19)

Natomiast schemat, w którym część wymiarów ma postać znormalizowaną, a część ma postać zdenormalizowaną nazywa się schematem *gwiazdy–płatka śniegu*.

Na slajdzie przedstawiono taki przykładowy schemat. W schemacie tym wymiar *Lokalizacja* jest znormalizowany, a pozostałe dwa wymiary są zdenormalizowane.



Schemat konstelacji faktów



ZSBD – wykład 12 (20)

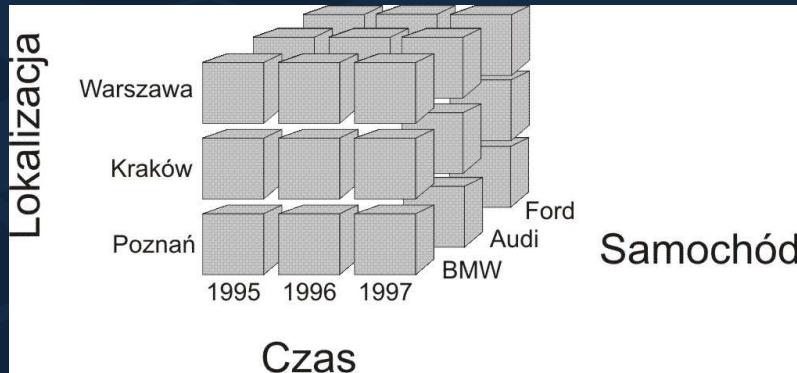
Schemat gwiazdy lub płatka śniegu, w którym ten sam wymiar jest powiązany z wieloma tabelami faktów nazywa się schematem *konstelacji faktów*.

Na slajdzie przedstawiono przykładowy schemat konstelacji faktów, w którym tabele faktów *Sprzedaż* i *Reklama* współdzielą wymiary *Produkty* i *Czas*.

W praktyce, ze względów efektywnościowych najczęściej stosuje się schematy gwiazdy lub gwiazdy–płatka śniegu.



Implementacja MOLAP



ZSBD – wykład 12 (21)

Hurtownia danych zaprojektowana w technologii MOLAP do przechowywania danych najczęściej wykorzystuje wielowymiarowe tablice (ang. multidimensional arrays, datacubes). Tablice te zawierają wstępnie przetworzone (m.in. zagregowane) dane pochodzące z wielu źródeł. Przykładowa wielowymiarowa tablica została przedstawiona na slajdzie. Zawiera ona trzy wymiary: *Lokalizacja*, *Czas* i *Samochód*. Komórki tablicy zawierają zagregowane informacje o sprzedaży wybranych samochodów (BMW, Audi, Ford) w poszczególnych latach (1995, 1996, 1997), w wybranych miastach (Poznań, Kraków, Warszawa).

Należy wspomnieć, że od strony implementacyjnej dane wielowymiarowe mogą być przechowywane nie tylko w wielowymiarowych tablicach. SZBD Oracle9i/10g do przechowywania danych wielowymiarowych wykorzystuje duże obiekty binarne (BLOB), a SQL Server2005 wykorzystuje tablice haszowe. Znane są również implementacje z wykorzystaniem struktur drzewiastych takich jak Quad-drzewa czy K-D-drzewa.



Operatory MOLAP (1)

- Wyznaczanie punktu centralnego (określanie kostki)
 - wskazanie miary
 - wskazanie wymiarów, w których miara będzie prezentowana

ZSBD – wykład 12 (22)

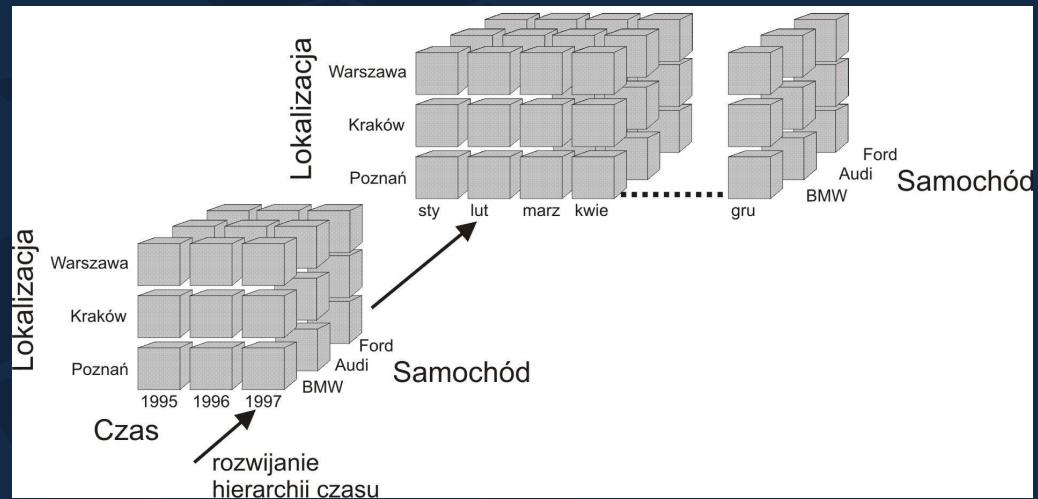
Analizę danych wielowymiarowych wspomagają specjalne operatory/operacje omówione na kolejnych slajdach. Pierwszym z nich jest wyznaczanie punktu centralnego (ang. pivoting).

Pivot polega na wskazaniu miary i określaniu wymiarów, w których wybrana miara będzie prezentowana. Przykładowo, w wymiarach Samochód, Miasto i Czas może być prezentowana liczba sprzedanych sztuk samochodu (miara).



Operatory MOLAP (2)

- Rozwijanie



ZSBD – wykład 12 (23)

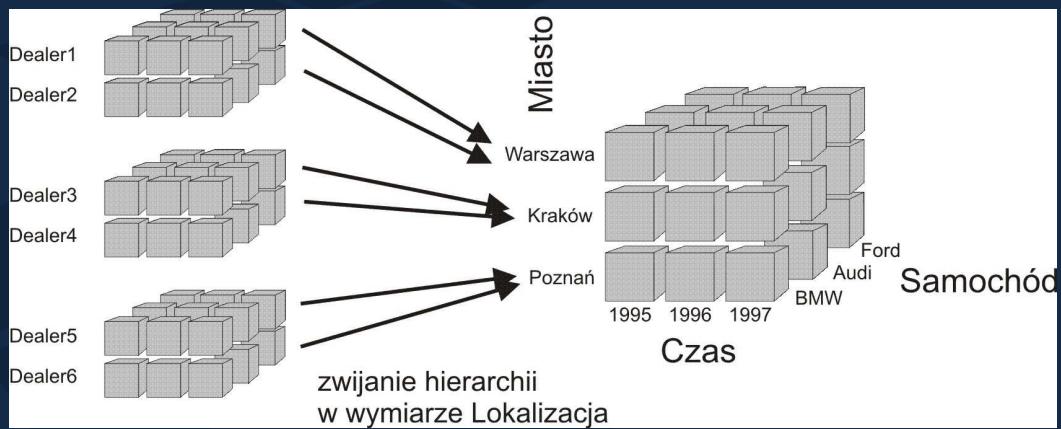
Kolejną operacją jest rozwijanie (ang. drilling down).

Rozwijanie polega na zagłębianiu się w hierarchię danego wymiaru w celu przeprowadzenia bardziej szczegółowej analizy danych. Jako przykład rozważmy informacje o sprzedaży samochodów BMW, Audi, Ford, w latach 95, 96 i 97, w poszczególnych miastach. W celu dokonania analizy sprzedaży w poszczególnych miesiącach roku 1997 należy rozwinąć hierarchię reprezentującą Czas.



Operatory MOLAP (3)

- Zwijanie



ZSBD – wykład 12 (24)

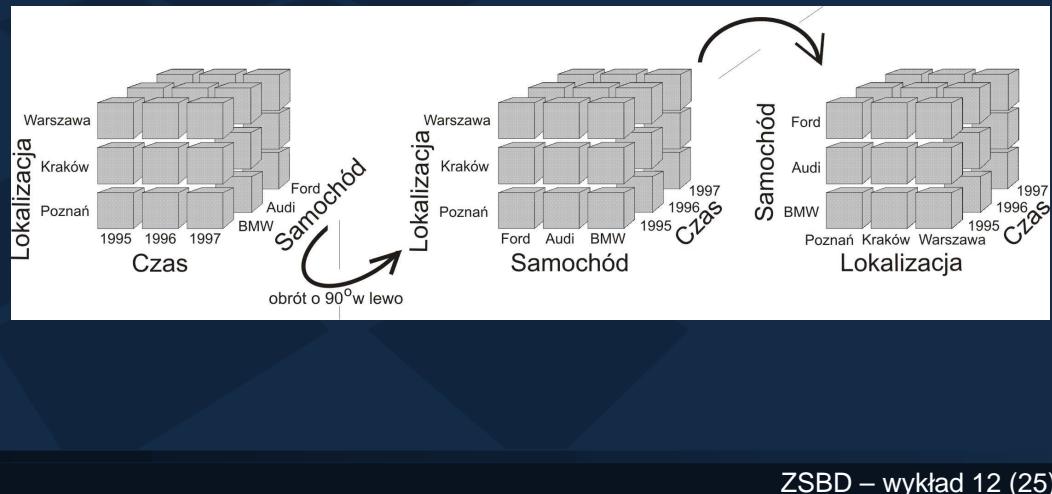
Kolejną operacją jest zwijanie (ang. rolling up).

Zwijanie jest operacją odwrotną do rozwijania i polega na nawigowaniu w górę hierarchii danego wymiaru. Dzięki tej operacji można przeprowadzać analizę danych zagregowanych na wyższym poziomie hierarchii wymiarów. Przykład na slajdzie ilustruje operację zwijania w wymiarze Lokalizacja. W tym przypadku, dane o sprzedaży przez dealerów z tego samego miasta są agregowane do sprzedaży w mieście, w którym ci dealerzy są zlokalizowani.



Operatory MOLAP (4)

- Obracanie



ZSBD – wykład 12 (25)

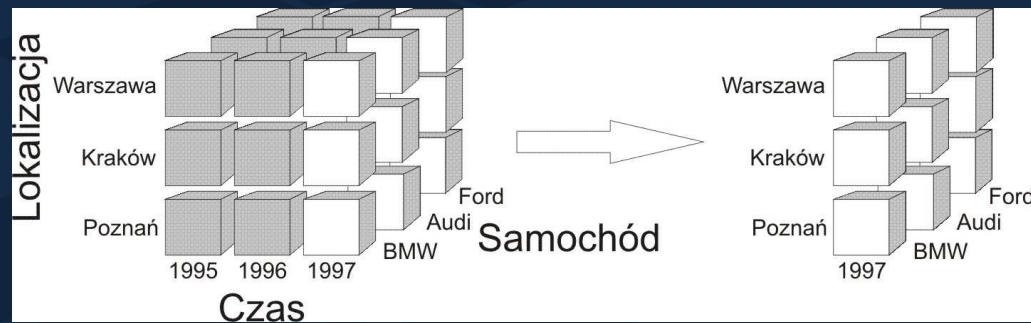
Kolejnym operatorem jest obracanie (ang. rotating).

Operacja obracania umożliwia prezentowanie danych w różnych układach. Celem jej jest zwiększenie czytelności analizowanych informacji.



Operacje MOLAP (5)

- Wycinanie



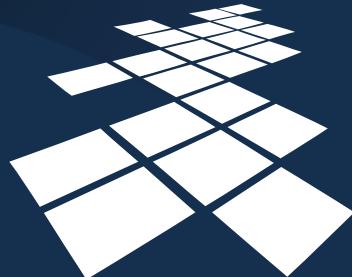
ZSBD – wykład 12 (26)

Ostatnią operacją modelu wycinanie (ang. slicing and dicing).

Operacja ta umożliwia zawężenie analizowanych danych do wybranych wymiarów, a w ramach każdego z wymiarów – zawężenie analizy do konkretnych jego wartości. Przykładowo, dyrektor do spraw marketingu będzie zainteresowany wielkością sprzedaży wszystkich produktów, we wszystkich miastach kraju, w wybranym roku. Natomiast kierownika oddziału firmy w Poznaniu będzie interesowała wielkość sprzedaży wszystkich produktów, w ciągu całego okresu działalności.

W przykładzie ze slajdu z kostki zostaje wycięty "plaster" opisujący sprzedaż samochodów BMW, Audi, Ford, w miastach Poznań, Kraków, Warszawa w roku 1997.

Bazy danych dokumentów XML wykład 1 – wprowadzenie



Wykład przygotował:
Krzysztof Jankiewicz

**UCZELNIA
ONLINE**

Bazy danych dokumentów XML – wykład 1 – wprowadzenie

Przez ostatnich kilkanaście lat znaczenie formatu danych XML stale rośnie. Popularność XML wynika z jego prostoty i elastyczności. Te cechy spowodowały, że XML stał się popularnym i dominującym standardem wymiany danych o złożonej, zmiennej i nieokreślonej strukturze. Dzięki tym cechom XML jest z powodzeniem wykorzystywany w takich dziedzinach jak: nauka, finanse, wymiana informacji, medycyna, grafika, kartografia, multimedia itp.

Początkowo dokumenty XML przechowywanie były przy wykorzystaniu systemów plików. Rosnące znaczenie i popularność XML-a a także stale rosnąca liczba dokumentów XML i konieczność ich wydajnego przetwarzania spowodowały, że w ostatnim czasie bardzo wiele komercyjnych systemów zarządzania obiektowymi i postrelacyjnymi systemami baz danych rozszerzyło swoją funkcjonalność o mechanizmy pozwalające na przechowywanie i przetwarzanie dokumentów XML. Takie rozwiązania pozwalają w znakomity sposób integrować dane mające różnych charakter, z jednej strony strukturalny (dane obiektowe i relacyjne) z drugiej strony semistrukturalne (dane w formacie XML). Oprócz adaptacji systemów obiektowych i postrelacyjnych obserwuje się także powstawanie licznych specjalizowanych rozwiązań pod postacią baz danych dokumentów XML (ang. native XML database systems). W tego typu bazach danych podstawową jednostką informacji jest dokument XML, użytkownicy przechowują w nich swoje dokumenty XML, przeglądają ich zawartość, generują na podstawie zawartości bazy danych nowe dokumenty XML.

Z punktu widzenia klasyfikacji, bazy danych dokumentów XML są przykładem semistrukturalnych baz danych.



Plan wykładu

- Typy dokumentów XML i ich wpływ na bazy danych
- Języki zapytań służące przetwarzaniu dokumentów XML
- Definicja bazy danych dokumentów XML
- Funkcjonalność baz danych dokumentów XML
- Sposoby przechowywania dokumentów XML
- Mechanizmy przetwarzania dokumentów
 - języki zapytań
 - sposoby modyfikacji

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (2)

Informacje dotyczące XML-owych baz danych zostaną przedstawione w trzech kolejnych wykładach.

Wykład pierwszy będzie wprowadzeniem do tematyki XML-owych baz danych. Zostanie przedstawiona definicja, funkcjonalność oraz typy XML-owych baz danych. Dodatkowo zostaną omówione te cechy XML-owych baz danych, które wyróżniają je na tle systemów relacyjnych i obiektowych.

Zadaniem wykładu drugiego będzie przedstawienie języka XQuery jako jednego z najbardziej zaawansowanych i najbardziej popularnego z języków zapytań stosowanych w XML-owych baz danych. Omówiona zostanie jego składania oraz funkcjonalność. Wykład zostanie zilustrowany szeregiem przykładów i porównań.

Wykład trzeci głównie będzie dotyczył języków przeznaczonych do modyfikacji dokumentów XML w XML-owych bazach danych. Ze względu na to, że nie ma obecnie standardu takiego typu języka, zostanie przedstawionych kilka podstawowych propozycji stosowanych w różnych bazach danych dokumentów XML.

W dzisiejszym wykładzie omówimy typy dokumentów XML i ich wpływ na bazy danych. Omówimy języki zapytań służące przetwarzaniu dokumentów XML. Zostanie przedstawiona jedna z możliwych definicji baz danych dokumentów XML. W dalszej części wykładu zostanie mówiona funkcjonalność baz danych dokumentów XML, sposoby przechowywania i mechanizmy przetwarzania dokumentów XML.



Typy dokumentów XML

- Zorientowane na dokumenty tekstowe
 - zawierają elementy o składowych mieszanych (węzłach elementów i węzłach tekstowych)
 - z reguły nie mają ścisłe określonej struktury
 - tworzone ze źródeł niestrukturalnych lub przez człowieka
- Zorientowane na dane
 - zawierają elementy zawierające węzły tekstowe lub elementy wewnętrzne
 - z reguły struktura jest z góry określona
 - tworzone zwykle ze źródeł strukturalnych

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (3)

Typ, budowa i funkcjonalność bazy danych dokumentów XML uzależniona jest ścisłe od typów dokumentów XML jakie będą w niej przechowywane.

W tym kontekście wyróżnia się dwa typy dokumentów XML:

Po pierwsze, dokumenty zorientowane na dokumenty tekstowe. Tego typu dokumenty zwykle zawierają elementy o składowych mieszanych (węzły elementów i węzły tekstowe jednocześnie). Ponadto, tego typu dokumenty, z reguły, nie mają ścisłe określonej struktury, nie ma narzuconego schematu, który ogranicza i definiuje strukturę dokumentu. Jest ona indywidualna dla każdego dokumentu. Często dokumenty takie są tworzone ze źródeł niestrukturalnych takich jak dokumenty tekstowe, dokumenty HTML. Mogą też być tworzone w sposób nieautomatyczny przez człowieka.

Drugim typem dokumentów XML są dokumenty zorientowane na dane. Tego typu dokumenty zawierają elementy, których składowe są ścisłe określone. Są to albo węzły tekstowe albo elementy wewnętrzne. Z reguły, struktura tego typu dokumentów zorientowana jest na przechowywanie danych, jest ona też z góry określona. Struktura ta wynika najczęściej ze struktury lub możliwości źródła pochodzenia informacji zawartej w dokumencie. Ponadto, tego typu dokumenty są zwykle tworzone ze źródeł strukturalnych takich jak bazy danych obiektowe, relacyjne itp.



Przykład dokumentu zorientowanego na informację tekstową

```
<OpisSpotkania>
  Dnia <DataSpotkania>01.05.2006</DataSpotkania>
  o godzinie <GodzSpotkania>09:15</GodzSpotkania>
  mam umówione spotkanie w miejscowości
  <MiejsceSpotkania>Zakopane</MiejsceSpotkania>,
  hotelu <MiejsceSpotkania>Kasprowy</MiejsceSpotkania>
</OpisSpotkania>
```

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (4)

Na slajdzie przedstawiono przykładowy dokument zorientowany na informację tekstową.

Element OpisSpotkania jest typowym przykładem elementu mieszanego, zawierającego w swoim wnętrzu zarówno węzły tekstowe (przykładowo węzeł z tekstem "Dnia ") jak i węzły będące podelementami (przykładowo element DataSpotkania). Można założyć, że inne dokumenty posiadające elementy opisujące spotkanie mogą mieć różną od powyższego przykładu zawartość. Dla przykładu element OpisSpotkania może w ich przypadku mieć tylko jeden podelement MiejsceSpotkanie, może nie mieć podelementów takich jak DataSpotkania lub GodzSpotkania (bo dla przykładu nie są one jeszcze znane). Ponadto element MiejsceSpotkania może być złożony z dodatkowych podelementów, dla przykładu, Miejscowość, Ulica, KodPocztowy itp.



Przykład dokumentu zorientowanego na dane

```
<EMP>
  <EMPNO>7369</EMPNO>
  <ENAME>SMITH</ENAME>
  <JOB>CLERK</JOB>
  <MGR>7902</MGR>
  <HIREDATE>17.12.1980</HIREDATE>
  <SAL>800</SAL>
  <DEPTNO>10</DEPTNO>
</EMP>
```

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (5)

Na tym z koleją slajdzie przedstawiono dokument zorientowany na dane. Informacje w nim zawarte mogą pochodzić z bazy danych relacyjnej lub obiektowej. Dokument ma wyraźną bardzo ściśle określona budowę.

Element ROW składa się tylko z podelementów. Nie ma bezpośrednich żadnych podwęzłów tekstowych. Podelementy elementu ROW, w przypadku dokumentu ze slajdu, są elementami prostymi. Ich zawartość jest prosta, może ona pochodzić z atrybutów tabeli lub obiektu. Można założyć, że budowa innych elementów EMP będzie bardzo podobna, każdy z nich będzie się składał z podelementów prostych wśród, których znajdą się elementy takie jak EMPNO czy ENAME.



Typy dokumentów a typy baz danych

- Bazy danych pozwalające na przechowywanie dokumentów XML (ang. XML-enabled database systems)
Niezorientowane na przetwarzanie dokumentów XML.
Z reguły pozwalają na przechowywanie określonego typu dokumentów XML
 - Plikowe i tekstowe bazy danych
 - Obiektowe i relacyjne bazy danych
- Bazy danych dokumentów XML (ang. native XML database systems)
Zorientowane na przetwarzanie dokumentów XML

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (6)

Typ wykorzystywanych dokumentów XML bardzo często ma zasadniczy wpływ na wybór bazy danych wykorzystywanej do ich przechowywania. Można wręcz powiedzieć, że możliwość przechowywania określonego typu dokumentów XML definiuje swego rodzaju klasyfikację baz danych wykorzystywanych do przechowywania dokumentów XML.

Z jednej strony mamy bazy danych, obiektowe lub obiektowo-relacyjne, które umożliwiają przechowywanie dokumentów XML. Niestety w większości przypadków jest to możliwe tylko w przypadku dokumentów XML o ścisłe określonej budowie. Wynika to faktu, że poszczególne fragmenty dokumentów XML są umieszczane we wcześniej zdefiniowanych tabelach, lub obiektach posiadających ścisłe określoną budowę. Jeśli takie ograniczenia występują mówimy o bazach danych pozwalających na przechowywanie dokumentów XML (ang. XML-enabled databases).

Z drugiej strony, mamy do czynienia z bazami danych, które pozwalają na przechowywanie dowolnych dokumentów XML, niezależnie od tego czy posiadają one określoną strukturę czy też nie. Jeżeli dodatkowo dokument XML-owy dla użytkownika stanowi podstawowy element przechowywany w bazie danych, i jego przetwarzanie jest realizowane głównie w oparciu o standardy związane z XML, wówczas możemy mówić o bazach danych dokumentów XML (ang. native XML database systems).



Przechowywanie dokumentów XML (1/3)

- Przechowywanie dokumentów XML
 - w systemie plików
 - w bazach danych w strukturach typu CLOB lub BLOB
 - w bazach danych w postaci zdekomponowanej
 - Mapowanie schematów XML na schematy baz danych
 - Generacja schematów XML ze schematów baz danych i odwrotnie
- Przechowywanie dokumentów w bazach danych dokumentów XML

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (7)

Istnieje kilka podstawowych sposobów przechowywania dokumentów XML.

Po pierwsze przy wykorzystaniu systemów plików. Składowanie dokumentów XML w systemach plików może być z powodzeniem wykorzystane w przypadkach małych zbiorów dokumentów XML. W przypadkach, gdy konieczne jest proste zarządzanie transakcjami można wykorzystać narzędzia, jakie jak CVS.

Nieco bardziej zaawansowanym sposobem jest wykorzystanie baz danych do przechowywania dokumentów XML w strukturach takich jak duże obiekty tekstowe lub binarne. Zalety takiego rozwiązania wywodzą się z funkcjonalności systemów baz danych takich jak: wielodostęp, obsługa transakcji, autoryzacja użytkowników itp. Kolejną zaletą jest możliwość wykorzystania indeksów do przeszukiwania dużych obiektów tekstowych.

Podstawowymi wadami obu podejść są:

- konieczność przetwarzania niestrukturalnego,
- brak rozróżnienia znaczników i ich zawartości.

Zaletą jest natomiast możliwość przechowywania dokumentów XML o dowolnej, nieokreślonej z góry strukturze.

Ponadto dokumenty XML możemy przechowywać:

- wykorzystując tabele lub obiekty składowane w obiektowo-relacyjnych bazach danych do przechowywania zdekomponowanych na składowe proste dokumentów XML, lub
- wykorzystując bazy danych dokumentów XML.



Przechowywanie dokumentów XML (2/3)

- w bazach danych w postaci zdekomponowanej
 - Mapowanie schematów XML na schematy baz danych
 - Generacja schematów XML ze schematów baz danych

```
<ROW>
  <EMPNO>7369</EMPNO>
  <ENAME>SMITH</ENAME>
  <JOB>CLERK</JOB>
  <MGR>7902</MGR>
  <HIREDATE>17.12.1980</HIREDATE>
  <SAL>800</SAL>
  <DEPTNO>10</DEPTNO>
</ROW>
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17.12.1980	800		20

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (8)

W przypadku wykorzystywania obiektowych lub relacyjnych baz danych, można przechowywać dokumenty XML w postaci zdekomponowanej. Dekompozycja polega na podziale dokumentu XML na mniejsze fragmenty, i przechowywanie ich w schemacie relacyjnym i obiektowych pozwalającym na późniejsze odtworzenie oryginalnej postaci dokumentu XML. Aby dekompozycja mogła być zrealizowana konieczna jest wcześniejsza znajomość struktury dokumentów XML i stworzenie schematów relacyjnych lub obiektowych zgodnych z tą strukturą. Przykładem dekompozycji dokumentu XML przedstawionego na slajdzie może być zawartość tabeli EMP w postaci również przedstawionej na slajdzie.

Dzięki elastyczności dokumentów XML możliwe jest mapowanie zarówno baz danych relacyjnych jak i obiektowych na postać dokumentów XML. Mapowanie takie jest stosunkowo powszechnie i daje możliwość udostępniania zawartości baz danych w postaci dokumentów XML. Przykładem standardu, który zajmuje się między innymi regułami dotyczącymi mapowania baz danych relacyjnych na dokumenty jest SQL/XML.

Mapowanie odwrotne, tzn. dokumentów XML na postać relacyjną lub obiektową, jest zagadnieniem nieco bardziej złożonym aczkolwiek i w tym przypadku mamy do dyspozycji konkretne rozwiązania, które wykorzystywane są dla przykładu przy budowie generatorów schematów. Mapowanie dokumentów XML na struktury obiektowe określone jest wiązaniem danych XML (ang. *XML data binding*). Mapowanie dokumentów XML na zawartość relacji może mieć różną postać:

1. Mapowanie strukturalne – możliwe jest tylko w przypadku znajomości struktury dokumentów XML. Istniejący schemat w bazie danych pozwala na przechowywanie tylko dokumentów zgodnych z tą strukturą. Przykładami takiego mapowania są przykładowo podejścia STORED czy XORator.
2. Mapowanie niestrukturalne – w tym przypadku nie jest konieczna znajomość struktury dokumentu ani jego schematu. Istniejący schemat w bazie danych pozwala na przechowywanie dokumentów o zróżnicowanej strukturze wewnętrznej. Propozycjami takiego mapowania dokumentów XML na zawartość tabel są przykładowo podejścia: Edge, XParent czy XRel.



Przechowywanie dokumentów XML (3/3)

- Przechowywanie dokumentów XML
 - w systemie plików
 - w bazach danych w strukturach typu CLOB lub BLOB
 - w bazach danych w postaci zdekomponowanej
 - Mapowanie schematów XML na schematy baz danych
 - Generacja schematów XML ze schematów baz danych i odwrotnie
- Przechowywanie dokumentów w bazach danych dokumentów XML

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (9)

Składowanie dokumentów XML w postaci zdekomponowanej nie jest pozbawione wad. W dalszym ciągu:

- przetwarzamy dokumenty XML w sposób niezorientowany na XML,
- dużym kosztem obarczona jest operacja odtworzenia dokumentu XML do jego oryginalnej postaci.

Z drugiej jednak strony:

- przetwarzanie jest w pełni strukturalne, co pozwala na wykorzystanie takich mechanizmów jak indeksy,
- występuje rozróżnienie pomiędzy schematem i strukturą dokumentu a jego zawartością.

Wykorzystanie specjalizowanych baz danych dokumentów XML stanowi rozwiązanie w większości przypadków pozbawione powyżej wymienianych wad.



Języki zapytań (1/5)

- Języki zapytań w obiektowo-relacyjnych bazach danych
 - oparte na konwersji zgodnej z przyjętym mapowaniem
 - oparte na szablonach XML
 - oparte na funkcjach SQL

Ich podstawowym zadaniem jest konstruowanie dokumentów XML w oparciu o zawartość bazy danych

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (10)

Istnieje wiele rozwiązań związanych z przeszukiwaniem, przetwarzaniem lub konstruowaniem dokumentów XML opartych na językach zapytań. Podstawowy podział pomiędzy tymi rozwiązaniami rysuje się na linii baz danych obiektowo-relacyjnych i baz danych dokumentów XML.

W pierwszym przypadku, mamy do czynienia z językami zapytań obiektowymi lub SQL-owymi, których podstawowym zadaniem jest konstruowanie dokumentów XML w oparciu o zawartość relacyjną lub obiektową baz danych. W najprostszym przypadku dokumenty XML, które powstają w wyniku zapytań są prostym mapowaniem zawartości tabel lub obiektów na ich XML-owe reprezentacje.

Bardziej zaawansowane rozwiązania są oparte o

- szablony,
- funkcje SQL.



Języki zapytań (2/5) przykład zapytania opartego o szablon

```
SELECT XMLGen (
    <PRACOWNIK id="${ID_PRAC}">
        <NAZWISKO>${NAZWISKO}</NAZWISKO>
        <ZAROBKI>
            <PODSTAWA>${PLACA_POD}</PODSTAWA>
            <DODATEK>${PLACA_DOD}</DODATEK>
        </ZAROBKI>
    </PRACOWNIK> ,
    p.ID_PRAC, p.NAZWISKO, p.PLACA_POD, p.PLACA_DOD) AS
    "wynik"
FROM pracownicy p
```

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (11)

Zastosowanie szablonów do konstruowania dokumentów XML daje możliwość umieszczenia wyników zapytań SQL w szablonach dokumentów XML. Szablony te posiadając w odpowiednich miejscach zmienne, wymieniają je na wartości kolumn pochodzące z zapytań SQL. Przykładem takiego rozwiązania jest funkcja XMLGen standardu SQL/XML.

Na slajdzie przedstawiono przykład zapytania SQL wykorzystującego funkcję XMLGen opartą na szablonie w celu wygenerowania dokumentów XML.

W wyniku zapytania powstaną elementy PRACOWNIK posiadające atrybut id oraz podelementy NAZWISKO i ZAROBKI. Element ZAROBKI będzie elementem złożonym. Elementy te i ich zawartość zostanie wygenerowana w oparciu o zawartość tabeli pracownicy.



Języki zapytań (3/5)

przykład dokumentu pełniącego rolę szablonu

```
<?xml version="1.0"?>
<EmpInfo>
  <Emps>
    <Query><SelectStmt>
      SELECT ENAME, SAL FROM EMP</SelectStmt>
    <Emp>
      <Nr>$ENAME</Nr>
      <Name>$SAL</Name>
    </Emp>
  </Query>
  </Emps>
</EmpInfo>
```

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (12)

Innym podejściem również opartym o szablony są dokumenty XML z zatopionymi w ich wnętrzu poleceniami SQL-owymi. Po ewaluacji zapytań postać wynikowa dokumentu jest uzupełniania o obiekty XML pochodzące z zawartości bazy danych.

Na slajdzie przedstawiono przykładowy dokument, którego zawartość będzie odpowiednio modyfikowana w zależności od wyniku zawartego w nim zapytania opartego na tabeli EMP.

Komercyjnym rozwiązaniem wykorzystującym ten sposób użycia szablonów, będących dokumentami XML jest standard XSQL Pages.

Wykorzystanie szablonów jest bardzo elastyczne. Podstawowa funkcjonalność takich rozwiązań to:

- możliwość umieszczania wyników zapytań w dowolnym miejscu dokumentów posiadających dowolnie złożoną strukturę,
- możliwość wykorzystania konstrukcji programistycznych przypominających pętle oraz instrukcje warunkowe,
- możliwość wykorzystywania zmiennych oraz funkcji SQL,
- parametryzacja zapytań SQL w oparciu o parametry HTTP.



Języki zapytań (4/5)

przykład zapytania opartego o funkcje SQL

```
SELECT XMLElement (
    "PRACOWNIK",
    XMLAttributes(id_prac as "id"),
    XMLForest(    nazwisko as "NAZWISKO",
    XMLForest(    plac_a_pod as "PODSTAWA",
    plac_a_dod as "DODATEK")
    as "ZAROBKI"
) ) AS "wynik"
FROM pracownicy
```

wynik

```
<PRACOWNIK id="100">
    <NAZWISKO>WEGLARZ</NAZWISKO>
    <ZAROBKI>
        <PODSTAWA>1730</PODSTAWA>
        <DODATEK>420.5</DODATEK>
    </ZAROBKI>
</PRACOWNIK>
. . .
```

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (13)

Kolejnym sposobem na uzyskanie dokumentów XML w oparciu o zawartość bazy danych jest zastosowanie funkcji SQL. Podstawowym standardem wykorzystującym funkcje SQL do konstruowania dokumentów XML jest standard SQL/XML. Przykładowe zapytanie zostało przedstawione na slajdzie.

Wynikiem tego zapytania może być zbiór dokumentów w postaci zaprezentowanej na slajdzie wewnętrz ramki.

Standard SQL/XML, którego przykładem jest omawiane zapytanie, stał się w roku 2005 fragmentem standardu SQL. W związku z tym, można oczekwać, że będzie on, w przyszłości, powszechnie wykorzystywany w relacyjnych bazach danych.



Języki zapytań (5/5)

- Języki zapytań w bazach danych dokumentów XML

Są to języki zorientowane na przetwarzanie dokumentów XML. Działają na zbiorach dokumentów XML i konstruują dokumenty XML

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (14)

Innym zbiorem języków związanych z przetwarzaniem dokumentów XML w bazach danych są języki zapytań wykorzystywane w bazach danych dokumentów XML. Są to języki zorientowane na przetwarzanie dokumentów XML. Działają one na zbiorach dokumentów XML i konstruują dokumenty XML.

Języki zapytań wykorzystywane w bazach danych dokumentów XML zostaną omówione na późniejszych slajdach.



Definicja bazy danych dokumentów XML (ang. native XML database system)

- Baza danych dokumentów XML
 - Definiuje model dla dokumentów XML-owych
 - Dokumenty XML są jej podstawową jednostką składowania
 - Wykorzystuje dowolny sposób fizycznego składowania

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (15)

Przejdźmy teraz do zagadnień związanych bezpośrednio z bazami danych dokumentów XML.

Jedna z możliwych definicji (stworzona przez członków grupy dyskusyjnej XMLDB) mówi, że:

- Baza danych dokumentów XML definiuje model dla dokumentów XML-owych (w przeciwieństwie do danych zawartych wewnątrz dokumentów) i składa się oraz udostępnia dokumenty wg tego modelu.

- Dokumenty XML są podstawową jednostką składowania w bazach danych dokumentów XML, analogicznie do tego jaką jest krotka w bazach danych relacyjnych.

- Nie wymaga się stosowania jakiegoś określonego fizycznego modelu składowania. Bazy danych XML można, zatem, dla przykładu, budować w oparciu o bazy danych relacyjne, obiektowe, hierarchiczne, lub wykorzystywać poindeksowane, skompresowane pliki na poziomie systemu operacyjnego.

XML 1.0 nie definiuje, ani nie narzuca określonego modelu, dlatego XML-owe bazy danych definiują swój własny model. Model danych musi zawierać takie struktury XML jak elementy, atrybuty, tekst i definicję porządku. Przykładami stosowanych modeli mogą być modele wykorzystywane w XPath, DOM, SAX, XQuery. Dane zawarte w bazie danych dokumentów XML udostępniane są wg określonego modelu.

Dokument XML jest podstawową jednostką składowania – jest odpowiednikiem krotki w systemach relacyjnych. Dokument zawiera pojedynczy zbiór danych

Baza danych może wykorzystywać dowolny sposób składowania dokumentów. Dla przykładu mogą to być:

- tabele "obiektów" SAX w bazie danych relacyjnej,
- obiekty DOM w obiektowej bazie danych,
- plik binarny zoptymalizowany do modelu danych XPath,
- skompresowane i poindeksowane dokumenty XML przechowywane w systemie plików.

Stosowany sposób składowania często wpływa na możliwości przechowywania różnych typów dokumentów XML.



Funkcjonalność bazy danych dokumentów XML

- Składowanie dokumentów XML
- Definiowanie i przechowywanie schematów (DTD, XML Schema)
- Obsługa zapytań (XPath, XQuery, XML-QL, Quilt)
- Obsługa modyfikacji, wstawiania i usuwania dokumentów
- Obsługa interfejsów programistycznych (XML:DB API, XQuery API for Java – XQJ, SAX, DOM, JDOM)
- Funkcjonalność tradycyjnych SZBD

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (16)

Zadaniem baz danych dokumentów XML jest przede wszystkim:

1. Umożliwienie użytkownikom przechowywania zbiorów dokumentów XML. Jednocześnie nie jest określone gdzie i w jaki sposób te dokumenty mogą być składowane.
2. Definiowanie i przechowywanie schematów dokumentów XML. Tak jak, dla przykładu, w relacyjnych systemach baz danych schemat relacji narzuca postać przechowywanych w bazie danych krotek, tak samo wymaga się od baz danych dokumentów XML możliwości definiowania schematów ograniczających postać przechowywanych dokumentów. Użytkownik powinien mieć także możliwość ich modyfikacji i odczytu.
3. Obsługa zapytań definiowanych przez użytkowników w oparciu o jeden lub wiele języków zapytań przeznaczonych do przetwarzania dokumentów XML.
4. Obsługa interfejsów programistycznych. W związku z tym, że do przetwarzania dokumentów XML jest wykorzystywanych obecnie szereg interfejsów programistycznych należy oczekwać, że baza danych dokumentów XML będzie pozwalała na ich wykorzystanie. Daje to możliwość wykorzystania bazy danych w połączeniu z szeregiem popularnych narzędzi operujących na dokumentach XML z wykorzystaniem dla przykładu DOM API.
5. Ponadto, oczekuje się od baz danych dokumentów XML funkcjonalności tradycyjnych systemów zarządzania bazami danych. W szczególności chodzi tu o kwestie związane z wielodostępnem, obsługą transakcji, mechanizmami archiwizacji i odtwarzania po awarii, importem i eksportem danych itp.



Składowanie dokumentów XML w bazach danych dokumentów XML (1/2)

- Architektura baz danych dokumentów XML ściśle zależy od modelu bazy danych
 - oparte na obiektach tekstowych
 - oparte na strukturach (relacyjnych, obiektowych, oryginalnych)

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (17)

Architektura baz danych dokumentów XML ściśle zależy od modelu bazy danych. W tym kontekście możemy powiedzieć, że rozróżniamy bazy danych:

- oparte na obiektach tekstowych,
- oparte na strukturach (oparte na modelu).

Bazy danych oparte na obiektach tekstowych:

- dokumenty przechowują w całości, jako obiekty tekstowe,
- do składowania mogą wykorzystywać obiekty typu CLOB lub BLOB, systemy plików itp.,
- mogą wymagać parsowania dokumentów przed ich wstawieniem do bazy danych,
- mogą wykorzystywać indeksy w celu uniknięcia parsowania, a także w celu przyspieszenia przeszukiwania; indeksy pozwalają na dostęp do dowolnego fragmentu dokumentu XML-owego,
- pobieranie dokumentów XML lub ich fragmentów nie wymaga ich rekonstrukcji – są one pobierane po prostu jako ciąg bajtów,
- pobierane dokumenty mają identyczną postać w jakiej zostały w bazie danych składowane, nie ma utraty białych znaków, komentarzy, instrukcji przetwarzania itp.,
- przykłady baz danych wykorzystujących indeksowane pliki to TextML, a wykorzystujących duże obiekty tekstowe to: Oracle, DB2 itp.



- Architektura baz danych dokumentów XML ściśle zależy od modelu bazy danych
 - oparte na obiektach tekstowych
 - oparte na strukturach (relacyjnych, obiektowych, oryginalnych)

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (18)

Bazy danych oparte na strukturach (oparte na modelu):

- składają dokumenty w formie "obiektów",
 - parsują dokumenty w momencie ich wstawiania – wymagane jest to podczas procesu dekompozycji i tworzenia obiektowej reprezentacji dokumentów,
 - składają dokumenty w oparciu o struktury: relacyjne, obiektowe, hierarchiczne, oryginalne itp.,
 - wykorzystują indeksy przede wszystkim do przyspieszenia przeszukiwania dokumentów; wykorzystywane indeksy mogą być tradycyjnymi indeksami zależnym od wykorzystywanych struktur,
 - pobieranie dokumentów wymaga ich ponownej rekonstrukcji, co wymaga znacznie większej liczby odczytów, niż ma to miejsce w przypadku baz danych opartych na obiektach tekstowych,
 - z reguły wydajnie tworzą struktury oparte na dokumentach XML, dla przykładu drzewa DOM,
 - przykładowo mogą przechowywać obiekty DOM w OORDBMS.
- Przykładami baz danych opartymi na modelu i wykorzystującymi odpowiednie struktury są:
- Relacyjne: Xfinity, eXist, Sybase, DBDOM.
 - Obiektowe: eXcelon, X-Hive, Ozone/Prowler, 4Suite, Birdstep.
 - Oryginalne: Tamino, Xindice, Neocore, Ipedo, XStream DB, XYZFind, Infonyte, Virtuoso, Coherity, Luci, TeraText, Sekaiju, Cerisent, DOM-Safe, XDBM, i inne.



Kolekcje dokumentów w bazach danych dokumentów XML

- Bazy danych dokumentów XML, z punktu widzenia użytkownika, przechowują dokumenty zebrane w postaci tzw. kolekcji
- Kolekcje dokumentów
 - zawierają podobne lub powiązane ze sobą dokumenty
 - podobne są do katalogów w systemie plików
 - dokumenty mogą mieć dowolny schemat
 - mogą być zagnieżdżone
 - lub, podobne są do tabel w systemie relacyjnym
 - dokumenty muszą spełniać reguły określonego schematu
 - umożliwiają zaawansowane indeksowanie i optymalizację zapytań

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (19)

Bazy danych dokumentów XML, z punktu widzenia użytkownika, przechowują dokumenty zebrane w postaci tzw. kolekcji.

Kolekcje dokumentów zawierają podobne lub powiązane ze sobą dokumenty. Często kolekcje można przyczytać do tabel w relacyjnej bazie danych, które grupują obiekty o takim samym znaczeniu, lub do schematów, które grupują obiekty powiązane ze sobą referencjami, wykorzystaniem w jednej aplikacji itp.

Kolekcje dokumentów zawierają podobne lub powiązane ze sobą dokumenty, przykładowo mogą to być informacje o umówionych spotkaniach, o zespołach istniejących na poziomie firmy, o pracownikach, mogą to być dokumenty CV przesłane przez osoby ubiegające się o stanowisko w naszej firmie.

Kolekcje dokumentów:

- mogą być podobne do katalogów w systemie plików, w takich przypadkach:
 - * dokumenty składowane wewnątrz kolekcji mogą mieć zwykle dowolną strukturę.
 - * kolekcje mogą być wielokrotnie zagnieżdżone.
- mogą też być podobne są do tabel w systemie relacyjnym i wówczas:
 - * dokumenty przechowywane w kolekcji muszą spełniać reguły określonego schematu, bardzo często przypisanego do kolekcji,
 - * z reguły nie mogą być zagnieżdżane,
 - * umożliwiają zaawansowane indeksowanie oraz zaawansowaną optymalizację zapytań, co jest ściśle związane z istnieniem definicji schematu.



Schematy i indeksy w bazach danych dokumentów XML

- W bazach danych dokumentów XML oprócz samych dokumentów składowane są także:
 - schematy dokumentów XML
 - indeksy
- Typy indeksów
 - strukturalne – indeksowanie nazw elementów i atrybutów
 - oparte na wartościach – indeksowanie wartości elementów i atrybutów
 - indeksy pełnotekstowe – indeksowanie leksemów występujących w wartościach elementów i atrybutów

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (20)

W bazach danych dokumentów XML oprócz samych dokumentów składowane są także:

- schematy dokumentów XML,
- indeksy.

Zadaniem schematów jest przede wszystkim ograniczanie typów składowanych dokumentów oraz definiowane ograniczeń integralnościowych obowiązujących składowane dokumenty. W zależności od indywidualnych rozwiązań schematy XML mogą być składowane z punktu widzenia użytkownika albo w centralnym repozytorium, albo w ramach poszczególnych kolekcji.

Zadanie indeksów jest dwójakie. Tak jak wspomniano przy okazji omawiania sposobów składowania dokumentów XML, mogą one być wykorzystywane do parsowania składowanych dokumentów. Jednak głównym ich zadaniem, podobnie jak w przypadku relacyjnych czy obiektowych baz danych, jest zwiększenie wydajności przetwarzanych zapytań.

W bazach danych dokumentów XML możemy wyróżnić trzy podstawowe typy indeksów:

- indeksy strukturalne – indeksowanie nazw elementów i atrybutów, przyspiesza wyszukiwanie dokumentów posiadających określone elementy struktury,
- indeksy oparte na wartościach – indeksowanie wartości elementów i atrybutów, przyspiesza wyszukiwanie dokumentów posiadających określone wartości w określonych fragmentach dokumentu,
- indeksy pełnotekstowe – indeksowanie leksemów występujących w wartościach elementów i atrybutów, pozwala przyspieszyć wyszukiwanie dokumentów posiadających określone leksemy, często niezależnie od struktury, formy itd.

Należy zaznaczyć, że nie wszystkie typy indeksów występują w każdej bazie danych dokumentów XML.



Charakterystyka indeksów w bazach danych dokumentów XML

- Indeksy strukturalne
 - zawierają informacje o wszystkich ścieżkach, które występują w dowolnej instancji dokumentów XML
 - wspomaga przeszukiwanie dokumentów bez określonego schematu
 - może być wykorzystywany do walidacji zmian bez dostępu do schematu dokumentu
- Indeksy oparte na wartościach
 - wspomagają wyszukiwanie elementów (atrybutów) posiadających określone wartości
 - uwzględniają typy wartości
- Indeksy tekstowe
 - warunkują efektywne wyszukiwanie wartości tekstowych
 - indeksowane są słowa występujące w elementach lub atrybutach

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (21)

Indeksy strukturalne:

- skondensowana struktura indeksu zawiera informacje o wszystkich ścieżkach, które występują w dowolnej instancji określonego typu dokumentu,
- wspomaga przeszukiwanie dokumentu bez określonego schematu; w takich sytuacjach bez takiego indeksu podczas zapytania należałoby przeglądać cały dokument,
- dla dokumentów o określonym schemacie indeks ten może być wykorzystywany do walidacji zmian bez dostępu do schematu dokumentu,
- struktura indeksu oprócz faktu istnienia ścieżek, zawiera informacje o tym, które dokumenty tą ścieżkę zawierają; pozwala to znaczco przyspieszać zapytania, które dotyczą opcjonalnych fragmentów dokumentu.

Indeksy oparte na wartościach:

- wspomagają wyszukiwanie elementów (atrybutów) posiadających określone wartości,
- uwzględniają typy wartości.

Indeksy tekstowe:

- warunkują efektywne wyszukiwanie wartości tekstowych,
- indeksowane są słowa występujące w elementach lub atrybutach,
- indeksy tekstowe nie są definiowane tylko na liściach lecz także elementach zawierających podelementy – to pozwala na wyszukiwanie obiektów lub dokumentów mających w swoim poddrzewie określone słowa,
- podczas budowy indeksu podział na słowa może odbywać się za pomocą funkcji XQuery fn:tokenize (text()),
- dane nie XML-owe (jeśli mogą być przechowywane w bazie danych dokumentów XML) są często automatycznie indeksowane tym typem indeksu.



Języki zapytań w bazach danych dokumentów XML

- Oryginalne
- XPath – powszechnie wykorzystywany
- XQuery – obecny standard języka zapytań dla baz danych dokumentów XML

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (22)

Podobnie jak swego czasu w przypadku systemów baz danych relacyjnych czy obiektowych, również w przypadku systemów baz danych dokumentów XML opracowano odpowiednie języki zapytań. Proces definiowania standardu języka zapytań do przetwarzania dokumentów XML zakończył się stosunkowo nie dawno. Standardem rekomendowanym przez organizację W3C jest język XQuery.

W roku 2005 jedynie kilka baz danych dokumentów XML pozwalało na dostęp do swoich zasobów za pomocą języka XQuery. Najbardziej rozpowszechniony w owym czasie był interfejs pozwalający na wykonywanie zapytań z użyciem wyrażeń XPath. Niektóre z komercyjnych baz danych umożliwiano wykonywanie zapytań w oparciu o własne propozycje. Przykładowo, Tamino umożliwiano wykorzystywanie języka X-Query, będący rozszerzeniem wyrażeń XPath o nowe możliwości.

W chwili obecnej XQuery to powszechnie wykorzystywany standard języka zapytań implementowany w większości baz danych dokumentów XML.

Język XQuery zostanie omówiony w drugim wykładzie poświęconym bazom danych dokumentów XML.



Sposoby modyfikacji w bazach danych dokumentów XML

- Większość baz danych umożliwia tylko usuwanie i wstawianie kompletnych dokumentów XML
- Modyfikacja zawartości jest możliwa za pomocą:
 - operacji DOM
 - wyrażeń XPath, które wskazują węzły, na których można przeprowadzić operację:
 - wstawienie węzła przed lub po
 - modyfikacja węzła
 - usunięcie węzła
 - rozszerzeń języka XQuery

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (23)

Duża część baz danych dokumentów XML umożliwia tylko usuwanie i wstawianie kompletnych dokumentów XML. Jest to dalekie od standardów przyjętych w bazach danych obiektowych i relacyjnych.

Bazy danych, które umożliwiają modyfikację fragmentów dokumentów XML stosują następujące podejścia:

- umożliwiają wykonywanie operacji DOM na dokumentach w nich zawartych,
- umożliwiają wykorzystanie wyrażeń XPath, które wskazują węzły, na których można przeprowadzić jedną lub wiele operacji takich jak:
 - * wstawienie węzła przed lub po wskazywanych przez wyrażenia XPath fragmentach,
 - * modyfikacja wskazywanego węzła,
 - * usunięcie wskazywanego węzła,
 - * utworzenie zmiennej, której zawartość będzie identyczna ze wskazywanym węzłem,
 - * zmiana nazwy znacznika wskazywanego elementu;
- wykorzystanie rozszerzeń języka XQuery.

Modyfikacja dokumentów XML za pomocą interfejsu DOM wymaga podejścia proceduralnego. Jest ono często satysfakcjonujące np. w przypadku edytorów dokumentów XML zintegrowanych z bazami danych. Nie jest to jednak podejście satysfakcjonujące użytkownika przyzwyczajonego do języków deklaratywnych takich jak SQL.



Przykład polecenia modyfikacji zdefiniowanego za pomocą wyrażeń XPath

```
<xupdate:append select="/bib" child="last()">
    <xupdate:element name="book">
        <title>System zarządzania bazą danych Oracle 7</title>
    </xupdate:element>
</xupdate:append>
```

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (24)

Podejście drugie oparte na wyrażeniach XPath jest w chwili obecnej najbardziej rozpowszechnione. Językiem najczęściej wykorzystywanym i pozwalającym w ten sposób definiować operacje modyfikacji jest XUpdate.

Przykładowo, polecenie na slajdzie żąda dodania elementu book, jako ostatniego w elemencie bib, będącym korzeniem dokumentu XML.

Niestety również to podejście nie jest pozbawione wad. W szczególności dotyczą one definiowania modyfikacji, które mają dotyczyć szeregu elementów.

Rozwiązaniem wydaje się być adaptacja języka zapytań XQuery do możliwości wykonywania operacji modyfikacji. Szereg komercyjnych baz danych już kilka lat temu wprowadzało tego typu rozwiązania. Należy jednak podkreślić, że, jak dotąd, nie ma wyznaczonego standardu. Nie ma również gwarancji, że przyszłe rozwiązanie dotyczące języka modyfikacji dokumentów XML będzie oparte na tym podejściu.

Językom modyfikacji dokumentów XML poświęcony został trzeci wykład dotyczący baz danych dokumentów XML.



Interfejsy programistyczne w bazach danych dokumentów XML

- Zazwyczaj podobne do ODBC
 - Języki zapytań oddzielone są od API
 - Podstawowe polecenia: connect, execute query, get results, commit/rollback
 - Rezultat zapytań w postaci: ciągu znaków, drzewa DOM, zdarzeń SAX.
- Zazwyczaj dostępne przez HTTP
- Wiele baz danych wykorzystuje swoje oryginalne API
- XML:DB API i XQuery API for Java (XQJ) to rozwiązania niezależne od bazy danych

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (25)

Większość z baz danych dokumentów XML udostępnia interfejsy programistyczne podobne do ODBC. Ich zadaniem jest udostępnienie programistom metod pozwalających na łączenie się z bazą danych, eksploracje metadanych takich jak dla przykładu schematy dokumentów XML lub nazwy plików lub kolekcji składowanych w bazie danych, wykonywanie poleceń i zapytań, a także pobieranie wyników. Wyniki zapytań najczęściej przyjmują postać: ciągu znaków, drzewa DOM, parsera SAX. Wiele baz danych pozwala na uzyskiwanie wyników pochodzących z wielu dokumentów znajdujących się w jednej lub wielu kolekcjach.

Większość interfejsów udostępnianych przez bazy danych dokumentów XML jest unikalnych, możliwych do zastosowania tylko w przypadku jednej, ściśle określonej bazy danych. W roku 2004 dostępne były tylko dla interfejsy niezależne od bazy danych:

- XML:DB API – rozwijane przez XML:DB.org, która jest autorem także bazy danych dokumentów XML XMLDB oraz miała wkład w powstanie języka XUpdate,
- XQuery API for Java (XQJ) – interfejs oparty na języku JAVA i dający programistom dostęp do baz danych za pomocą zapytań XQuery. Interfejs rozwijany w ramach Sun's Java Community Process (JCP).

Ponadto wiele z baz danych dokumentów XML udostępnia interfejs pozwalający na przeglądanie dokumentów i wykonywanie zapytań za pomocą protokołu HTTP.



Dostęp do danych zewnętrznych w bazach danych dokumentów XML

- Zewnętrzne pliki XML
- Bazy danych relacyjne (za pomocą ODBC, JDBC itp.)
- Dane aplikacji (SAP, PeopleSoft, Excel, etc.)

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (26)

Kolejną funkcjonalnością baz danych dokumentów XML jest możliwość dostępu za ich pomocą do informacji zewnętrznych. Przykładami informacji zewnętrznych mogą być:

- pliki XML,
- źródła danych relacyjne lub obiektowe, udostępnianie w postaci wirtualnych plików XML przy wykorzystaniu odpowiedniego mapowania (dostęp za pomocą ODBC, JDBC itp.),
- dane aplikacji zewnętrznych (SAP, PeopleSoft, Excel, itp.).



- Większość baz danych stosuje transakcje
- W większości baz danych dostęp do dokumentu realizowany jest w sposób wyłączny
- Wymagany poziom współbieżności w rzeczywistości jest uzależniony od :
 - liczby użytkowników bazy danych
 - charakteru przechowywanych danych w dokumentach
- Pojawiają się propozycje nowych algorytmów pozwalających na kontrolę współbieżnego dostępu do baz danych dokumentów XML

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (27)

Większość baz danych dokumentów XML pozwala użytkownikom na wykorzystywanie transakcji przy dostępie do dokumentów i ich modyfikacji. Dużym problemem, w chwili obecnej, jest umożliwienie współbieżnego dostępu wielu użytkownikom do pojedynczego dokumentu. W większości przypadków w bazach danych dokumentów XML dostęp i modyfikacja dokumentów wymaga założenia blokady na poziomie dokumentu.

Takie podejście ma dwa źródła. Pierwsze z nich to fakt, że bazy danych dokumentów XML początkowo miały charakter repozytoriów udostępniających głównie operacje odczytu. Drugi powód to fakt, że dokument w bazach dokumentów XML przez wielu traktowany jest na równi z krotką w systemie relacyjnym.

Wymagany poziom współbieżności w rzeczywistości jest uzależniony od:

- liczby użytkowników bazy danych,
- charakteru przechowywanych danych w dokumentach.

Dla przykładu, jeśli dokumenty zawierają rozbudowane kompozycje graficzne zdefiniowane w oparciu o XML-owy standard SVG, to może się okazać, że zawłaszczenie dokumentu przez jednego użytkownika jest całkowicie niedopuszczalne.

Dlatego też coraz częściej pojawiają się propozycje nowych algorytmów pozwalających na kontrolę współbieżnego dostępu do baz danych dokumentów XML jednocześnie zapewniając odpowiedni poziom współbieżności.



Literatura

- <http://www.rpbourret.com/xml/>
- <http://xmlDb-org.sourceforge.net/index.html>
- <http://www.garshol.priv.no/download/xmltools/>
- <http://www.oasis-open.org/cover/xmlAndDatabases.html>
- *Wprowadzenie do systemów baz danych*, Ramez Elmasri, Shamkant B. Navathe, ISBN: 83-7361-716-7

Bazy danych dokumentów XML – wykład 1 – wprowadzenie (28)