Wykłady z OSZBD

Michał Bugno Antek Piechnik

29 stycznia 2009

Spis treści

1 Ten plik

Plik to zapis wykładów dra Roberta Marcjana na Akademii Górniczo–Hutniczej w Krakowie na kierunku Informatyka.

- inicjatywa: msq (michal.bugno<at>gmail.com)
- pomoc: tph (antek.piechnik<at>gmail.com).
- źródła można znaleźć w repozytorium Git–a pod adresem: http://github.com/michalbugno/wyklady-oszdb/

2 Wstęp

2.1 Literatura

- Silberschatz "Database System Concepts"
- Garcia-Molina, Ullman, Widom "Database systems The Complete Book"
- (!)Garcia-Molina, Ullman, Widom "Implementacja Systemow Baz Danych"
- Ramakrishan, Gecherke "Database Managment Systems"
- Date "Wprowadzenie Do Baz Danych" (uzupełniająca, raczej nie)
- Celko "SQL zaawansowane techniki programowania"

2.2 Omawiane bazy danych

2.2.1 Komercyjne

- Oracle
- IBM INFORMIX
- IBM DB2
- MS SQL
- Sybase

2.2.2 Darmowe

- PostgreSQL
- MySQL

2.3 Linki

- `http://www.sqlservercentral.com`
- `http://www.databaseweekly.com`
- `http://www.oracle.com/oramag/index.html`
- `http://www.db2mag.com`
- `http://www.iiug.org`

2.4 Plan

2.4.1 Semestr zimowy

1. SZBD
   - pamięć, procesy, dysk
   - moduły funkcjonalne SZBD, architektura klient/serwer
2. Fizyczna struktura danych
   - struktura fizyczna zbiorów
   - metody przechowywania danych, organizacja dysków, RAID
   - fragmentacja/partycjonowanie
3. Metody dostępu do danych, indeksowanie
   - dostęp sekwencyjny
   - indeksy, B–Tree, hash, bitmap, siatkowe (struktury wielowymiarowe)
4. Przetwarzanie transakcyjne
   - pojęcie transakcji
   - Atomicity, Consistency, Isolation, Durability (ACID)

5. Metody realizacji przetwarzania transakcyjnego
    - ACD
6. Równoczesny dostęp do danych
    - poziomy izolacji
    - metody zarządzania równoczesnym dostępem
    - blokady, wielowersyjność
7. Realizacja operacji, optymalizacja
    - algebra relacji
    - SQL, PL/SQL i podobne
    - sposoby realizacji operacji (SELECT, JOIN itp.)
8. Replikacja danych
    - metody replikacji danych
    - synchronizacja danych
9. Bezpieczeństwo
    - integralność, transakcje
    - archiwizacja, odtwarzanie
    - fault tolerance (RAID, klastry, replikacja)
10. Administracja i optymalizacja wydajności serwera
    - konfiguracja (specyficzne cechy SZBD)
    - strojenie
11. Przetwarzanie informacji przestrzennych, GIS
    - metody, algorytmy, struktury danych
    - Oracle spatial, DB2, INFORMIX, MS SQL 2008
12. Przykłady SZBD
    - IBM/Informix

- IBM/DB2
- Oracle
- MSSQL Server

2.4.2 Semestr letni

1. Hurtownie danych
2. Analiza danych
3. Exploracja danych

2.5 Laboratoria

- podział na grupy na 2 semestry
- Prowadzący
  - Anna Zygmunt (pon. 11.oo - 12.3o, pt. 15.3o - 17.oo)
  - Robert Marcjan (pon. 9.3o - 11.oo)
  - Leszek Siwik (pt. 15.3o - 17.oo)

Model danych zestaw pojęć używanych do opisu danych

Modele danych:

- związków encji
- sieciowy
- hierarchiczny
- relacyjny (mocna teoria (algebra relacji), udany, prosty)
- obiektowy (nie tak wydajny jak relacyjny)
- obiektowo-relacyjny

3 Relacyjny model danych

3.1 Cechy

- relacje, krotki, atrybuty
- tabele, wiersze, kolumny
- algebra relacji (prosta, przejrzysta)
  - wejście: relacja1, relacja2 ..., wyjście: nowa relacja
  - operacje: selekcja, projekcja, złączenie (join), suma, różnica, przecięcie etc.
- język SQL

3.2 Niezależność

- możliwość modyfikacji definicji (schematu) na danym poziomie bez konieczności zmian na wyższym poziomie (poziom fizyczny, logiczny, prezentacji)
- aplikacje są niezależne od tego jak dane są strukturalizowane i przechowywane

3.3 Architektury

- mainframe
- file sharing, ISAM (na plikach)
- client-server (klient nie “dotyka“ serwera)
- architektury wielowarstwowe (3 warstwowe)
- architektury rozproszone
  - rozproszenie danych
  - powielanie (replikacja) danych
  - transakcje rozproszone
- architektury równoległe (w tym widzi się skok wydajności, dotychczasowe rozwiązania są już bardzo dobrze zoptymalizowane)

3.4 Client–server

- operacje “klienta“
- operacje “serwera“
- Structured Query Language (SQL)
- stored procedures – każdy ma własny język, nie ma zgodności (zmienne, pętle, nie ma struktur danych)
- SQL + możliwość wykonania programu na serwerze (SUM(kolumna) vs SELECT * a potem sumuj, programy w Java, C)

3.5 Dlaczego SZBD?

- redundancja danych
- spójność danych (dlaczego nie tylko w aplikacji? bo z reguły SZBD żyje DŁUŻEJ niż aplikacja)
- równoczesny dostęp do danych (nie trzeba przejmować się równoczesny dostępem)
- transakcje (ACID)
- bezpieczeństwo danych
  - ochrona przed niepowołanym dostępem (bardzo dokładna kontrola np. konkretne kolumny)
  - odporność na awarię

Przetwarzanie powinno być *efektywne*.

4 Pamięć, dyski

W SZBD dane przechowywane są na dyskach. Ma to znaczenie przy ich projektowaniu.

4.1 Hierarchia szybkości

1. cache (najszybszy)
2. main memory
3. flash memory
4. magnetic disk
5. optical disk
6. magnetic tapes (najwolniejszy)

4.2 Podstawowe operacje

READ transfer z dysku do RAM

WRITE zapis z RAM na dysk

Te operacje są kosztowne w porównaniu do operacji w pamięci

4.3 Komponenty tworzące SZBD

- dysk
  - struktury przechowujące dane
  - struktury przechowujące opis danych
  - struktury zapewniające przetwarzanie transakcyjne
  - struktury zapewniające bezpieczeństwo
- procesy
  - operacje związane z funkcjonowaniem serwera
  - operacje wykonywane “na zlecenie“ aplikacji klienckich
- pamięć
  - pula buforów
  - struktury kontrolne

Typowy system ma dużo więcej danych na dysku niż może przechowywać w pamięci.

4.4 Sposoby zabezpieczeń

4.4.1 RAID

- backup danych, backup logów (raz na jakiś czas zapisujemy dane, ciągle backupujemy logi)
- w przypadku awarii restore danych a potem wykonujemy logi od ostatniego dobrego backupu

4.4.2 Replikacja

- zabezpieczenie przed utratą wszystkiego (serwer wykonuje log i przesyła go gdzieś indziej, do innego serwera)
- przybliżenie użytkownikowi danych (coraz mniejsza rola tej funkcji, przepustowość sieci rośnie)

4.4.3 Struktury dyskowe

- czas dostępu do dysku jest bardzo wolny
- operujemy na dużych zbiorach danych
- powinny zapewniać wydajność i bezpieczeństwo

4.5 Dysk

4.5.1 Budowa

- głowica
- talerz
- ścieżka
- cylinder
- blok
- sektor

4.5.2 Czas odczytu

time = seek time (head)(longest?) + rotation delay + transfer time

4.5.3 Sposób dostępu

- random (1o-krotni wolniejsze)
- następny blok
- sekwencyjny

Przykład

```
transfer rate = 4000 KB
avg seek time = 0.01 s
page          = 2KB
```

Random 1 strona

```
2 KB / (0.01 s + (2 KB / ( 4000 KB/s ))) = 190 KB/s
```

Sekwencyjny 3o stron

```
60 KB / (0.01 s + (60 KB / ( 4000 KB/s ))) = 2400 KB/s
```

4.5.4 Wydajność

Poprawa prędkości odczytu

- odpowiednia organizacja przestrzeni dyskowej
- szeregowanie operacji we/wy (sterowanie ruchem głowicy)
- bufor dla zapisu (np. w pamięci RAM)
- bufor na dysku (zapis sekwencyjny do bufora a potem porozdzielanie w odpowiednie miejsca)

Mean Time To Failure (MTTF) Średni czas spodziewanej bezawaryjnej pracy dysku

Typowy MTTF – kilka-kilkadziesiąt lat

Prawdopodobieństwo awarii jest stosunkowo niskie

Bez nadmiarowości

- 1 dysk: $100.000h \sim 11lat$
- 100 dysków: $1.000h \sim 41dni$

Mirroring

- średni czas $(100.000^2)h/10 \sim 100.000lat$
- ale prawdopodobieństwo awarii rośnie mocno z wiekiem
- rzeczywisty czas rzędu kilkudziesięciu lat

Bit–level striping

- np macierz 8 dysków, i-ty bit umieszczony na i-tym dysku
- 8x szybszy dostęp

Block–level striping

4.5.5 RAID

Co to i do czego służy?

- poprawa wydajności - zrównoleglenie operacji we/wy
- poprawa bezpieczeństwa - nadmiarowość

Typy RAID

1. RAID0 – wysoka wydajność, utrata danych nie może być problemem
2. RAID1 – wydajny ale kosztowny
3. RAID3 – szybki transfer danych
4. RAID5 – lepszy niż RAID3 dla dostępu typu “random“
5. RAID > 5 – raczej niestetosowany

Jeśli stać nas na RAID0+1 to jest najlepsze rozwiązanie, jeśli nie to RAID5

4.5.6 Inne sposoby poprawy wydajności

- mirroring na poziomie SZBD i systemu
- sterowanie położeniem danych na dysku (bliżej środka)
- fragmentacja/grupowanie danych
  - klastry
  - fragmentacja tabel
  - separowanie danych na poziomie fizycznym

4.6 Sposoby przechowywania danych

- plik, rekord, blok
- tabela, wiersz, indeks, baza
- relacja między poziomem logicznym a fizycznym
  - plik składa się z bloków (a nie pojedynczych bitów [odczyt])
  - format rekordu: stały lub zmienny

4.6.1 Stały format rekordu

Zawartość

- liczba pól
- typ każdego pola
- kolejność pól w rekordzie
- definiuje rekord

Zalety

- umożliwia łatwy dostęp
- prostota implementacji

Wady

- tylko jeden typ rekordy w danym pliku
- dopisywanie na koniec
- usuwanie rekordów
  - flaga "deleted"
  - lista usuniętych rekordów (dodawanie może ją wykrzoystywać)
  - jako że więcej INSERT niż DELETE to nie jest problem
- jeśli rozmiar bloku nie jest wielokrotnością rozmiaru rekordu to niektóre rekordy będą wymagały dwóch odczytów

4.6.2 Zmienny format rekordu

Zawartość

- każdy rekord zawiera opis formatu (self describing)

Zalety

- rzadkie rekordy
- formaty zmienne w czasie
- rekordy różnych typów w jednym pliku
- oszczędność miejsca
  - sposoby przechowywania NULLi
  - wskaźniki
  - rezerwowanie miejsca dla rekordów (np rezerwowanie 255 dla VARCHAR *zawsze*)

Wady

- marnuje ilość miejsca (więcej miejsca == więcej czasu do przetwarzania danych)
- trudniejszy w implementacji

Slotted page

- plik składa się ze stron które odpowiadają blokom na dysku (2kB (Oracle), 4kB, 8kB)
- nagłówek (timestamps etc.)
- tablica slotów (wskaźnik do początku wolnego miejsca, wskaźnik – rozmiar, wskaźnik – rozmiar ...)
- elastyczny
- powszechnie stosowany w SZBD
- w praktyce
  - jedna strona == jedna tabela
  - sekwencyjne przetwarzanie danych z tabeli
- wiele tabel w jednym pliku
- jedna tabela w wielu plikach
- reguła: jeżeli wiersz mieści się na stronie, to jest zawsze na jednej stronie
  - więcej zajętego miejsca
  - efektywniejsze przetwarzanie danych

4.6.3 Reprezentacja danych, nomenklatura

Sposoby reprezentacji

- każdej tabeli odpowiada plik – mniejsze SZBD
- rezerwowany obszar dysku – większe SZBD

Przetrzeń dyskowa SZBD

- Oracle – tablespace
- Informix – Dbspace
- MSSQL, Sybase – file group

Informix

- database, dbspace, chunk
- przestrzeń dyskowa z jednego lub więcej chunków
- chunk to jeden plik lub obszar dyskowy
- przestrzeń dyskowa podzidzelona na jednostki logiczne (dbspace), każda składa się z jednego lub więcej plików
- na serwerze może być wiele BD (w określonych Dbspaceach)

MSSQL

- database, filegroup, datafile
- wiele baz danych
- każdej bazy danych mamy przyzielaną przestrzeń dyskową
- dla każdej bazy można stworzyć kilka grup plików (file group)

ORACLE

- tablespace, datafile
- przestrzeń dyskowa podzielona na tablespace'y
- tablespace to jeden lub więcej plików

Tabela

- w momencie tworzenia serwer przydziela jej przestrzeń dyskową składającą się z obszaru stron na dysku nazywanego zakresem (extent)
- extent: ciągly obszar stron na dysku
- rozmiar zwykle kilka lub kilkanaście stron (można nim sterować)
- dopisywanie wierszy do tabeli powoduje wypełnianie stron w ramach extentu
- jeżeli brakuje miejsca to przydzielany jest kolejny extent

Rozmiar extentu może być zmieniony (od danego momentu extenty większe). Można też automatycznie zwiększać rozmiar extentu (np. co 16 extent podwajamy)

Extenty nie są zwalniane automatycznie (wyjątek: tabele tymczasowe) – dlatego że zakładamy, że z reguły tabele się rozrastają a nie kurczą.

Cluster Przechowywanie dwóch (lub wiecej) tabel łącznie na poziomie fizycznym (razem). Wydajne, gdy najczęstszym sposobem ich przeglądania jest JOIN.

Fragmentacja tabel Dystrybucja danych z jednej tabeli pomiędzy wiele dbspace'ów.

Podsumowanie

- dążymy do tego żeby tabele były przechowywane w ciągłych obszarach na dysku
  - w ogólnym przypadku – trudne
  - pewnym rozwiązaniem jest pamiętanie danych w ciągłych extentach
  - warto czasem zreorganizowac strukturę extentów aby uzyskać ciągły obszar
- czasami sens ma partycjonowanie/fragmentacja danych
  - zrównoleglanie operacji we/wy
  - zysk w przypadku wielu małych transakcji
  - zysk w przypadku długich operacji
- w pewnych przypadkach sens ma tworzenie struktur typu cluster
  - gdy wiekszosc operacji dla tych tabel to JOIN
  - tabele nie są przeglądane samodzielnie ("podatek")

5 Indeksy

Indeks może być plikiem płaskim, strukturą drzewiastą, hashem, który pozwala na przyspieszenie dostępu do danych.

Można używać go na dwa sposoby:

- w celu uzyskania dostępu o charakterze bezpośrednim

- w celu uzyskania dostępu sekwencyjnego (dostęp zgodny z porządkiem wyznaczonym przez indeks)

5.1 Podział

- indeksy płaskie – uporządkowane zbiory indeksujące
- indeksy bazujące na B-drzewach
- indeksy - struktury typu hash
- indeksowanie po kilku atrybutach

5.2 Budowa

- klucz (search key) – atrybut/zbiór atrybutów, wg których wyszukujemy
- zbior indeksujacy – rekordy postaci search–key => pointer
- indeks jest mniejszy od zbioru, który podlega indeksowaniu

5.3 Podstawowe typy indeksów

- uporządkowane – klucze są uporządkowane
- hash – klucze są rozmieszczone równomiernie (zgodnie z funkcją haszujacą)

5.4 Miary efektywności indeksów

- dostęp do danych
  - rekordy o określonej wartości atrybutu (rekordy które spełniają warunek)
  - rekordy które nie spełniają warunku dla określonego atrybutu
- czas dostępu do danych
- czas potrzebny na dopisanie rekordu
- czas potrzebny na usunięcie rekordu
- narzut wynikający z przestrzeni zajmowanej przez indeks

5.5 Indeksy primary/secondary

5.5.1 Primary

- uporządkowanie wg indeksu zgodne z fizycznym uporządkowaniem danych
- czasami nazywane grupującymi (clustering index)
- indeks gesty (dense index) – rekord indeksu dla każdej wartości klucza
- indeks rzadki (sparse index)
- indeksy wielopoziomowe (płaski indeks do płaskiego indeksu)
- duplikaty (reprezentowanie tylko raz wartości klucza w pliku – optymalne rozwiązanie, wciąż indeks gęsty)

5.5.2 Secondary

- uporządkowanie nie jest zgodne z fizycznym uporządkowaniem
- nazywane non–clustering index
- indeksy rzadkie – nie mają sensu
- duplikaty (buckets)

5.6 Uwagi

- generalnie indeks przyspiesza dostęp do danych
- dodatkowy koszt związany z modyfikacją danych
- jeśli indeks jest duży to spada wydajność dostępu do danych (dla płaskich plikłw indeksujących)
- dostęp sekwencyjny
  - przy użyciu indeksu primary – efektywny
  - przy użyciu indeksu secondary – o wiele bardziej kosztowny
- indeksy rzadkie są mniejsze ale trzeba dodatkowo realizować wyszukiwanie na poziomie danych

- w przypadku indeksów gęstych niektóre operacje mogą być zrealizowane przy pomocy samego indeksu
- w przypadku rzadkich w zasadzie też, ale jest zdecydowanie mniej możliwości

5.7 B–Tree

Wygląd węzła: $K_1, P_1, K_2, P_2, \ldots, K_n, P_n$

K klucz

P pointer

n rozmiar

Klucze są uporządkowane.

5.7.1 B+ drzewa

Głębokość B+ drzewa: $\log_{\frac{n}{2}} K$

K liczba kluczy

- B-drzewa to najczęstszy sposób organizacji
- wydajny jeśli chodzi o dostęp *random*
- wydajny jeśli chodzi o dostęp sekwencyjny

5.8 Hash

- funkcja haszujaca: $key => h(key)$
- statyczne lub dynamiczne (linear hashing, extensible hashing)
- dają lepsze efekty jeśli chodzi o dostęp bezpośredni
- problemy przy zapytaniach o zakres wartości

6 ACID

6.1 Atomicity, Consistency

6.1.1 Dlaczego?

- możliwość awarii systemu podczas transakcji
- równoczesny dostęp do danych
- równoczesne wykonywanie transakcji

6.1.2 Własności transakcji

Consistency

```
1. READ(A)
2. A = A - 50
3. WRITE(A)
4. READ(B)
5. B = B + 50
6. WRITE(B)
7. A + B = const
```

Na przykład awaria między 3 i 4 powoduje, że baza powróci do stanu pierwotnego.

Trwałość Zmiany mają charakter trwały.

Izolacja Transakcje są wykonywane w całowitej izolacji (dla innych transakcji nie są widoczne do czasu zatwierdzenia.

6.1.3 W jaki sposób zapewnić transakcyjność?

- shadow database: kopiujemy bazę, i po udanej transakcji usuwamy starą bazę i wskaźnik wskazuje na nową
  - prosta implementacja
  - nieefektywna (kopiowanie *całej* bazy danych?!)
  - krótkie transakcje czekają na jedną długą
- algorytmy zapewniające niepodzielność i trwałość

- metody pozwalające odtworzyć stan bazy w przypadku awarii, powszechnie stosujemy system logów

6.1.4 Awarie transakcji

- użytkownik – rollback (jawnie)
- serwer – na przykład naruszenie warunku integralnościowego
- awarie aplikacji (awaria połączenia itp.)
- awarie na poziomie serwera baz danych
  - system crash (utrata danych w pamięci)
  - uszkodzenie dysku (utrata danych na dysku)

INPUT(X) blok z dysku zawierający element X jest kopiowany do pamięci

READ(X, t) jeżeli w pamięci nie ma bloku X, to INPUT(X), następnie t := x

WRITE(X, t) jeżeli bloku zwierającego X nie ma w pamięci to INPUT(X), następnie X := t

OUTPUT(X) zawartość bufora kopiowania na dysk

	t	MA	MB	DA	DB
READ(A, t)	8	8		8	8
t := t * 2	16	8		8	8
WRITE(A, t)	16	16		8	8
READ(B, t)	8	16	8	8	8
t := t * 2	16	16	8	8	8
WRITE(B, t)	16	16	16	8	8
OUTPUT(A)	16	16	16	16	8
OUTPUT(B)	16	16	16	16	16

6.1.5 Logi

Struktura

<**START T**> rozpoczęcie transakcji T

<**T, X, v**> T – transakcja, X – el. danych, v – wartość przed modyfikacją

<**COMMIT T**> zakończenie transakcji

Log może służyć do operacji undo (wycofania transakcji)

Implementacja

- rekord logu dla każdej operacji
- rekord logu musi osiągnąć dysk zanim zmodyfikowany bufor zostanie zapisany na dysk (Write-Ahead Logging (WAL))
- przed commitem wszystkie operacje muszą być zapisane na dysk

Podczas awarii

- wpisz <ABORT T> do logu
- jeśli transakcja zapisuje informacje na dysk dopiero po zakończeniu wszystkich operacji to nie trzeba robić odtwarzania w przypadku błędu (wystarczy abort do logu)
- transakcje niekompletne: jeśli jest <START T> a nie ma <COMMIT T> ani <ABORT T>
- podczas procesu odtwarzania awaria nie jest problemem: po prostu powtarzamy proces odtwarzania

Checkpoint Operacja odtwarzania wymaga przeglądnięcia całego logu. Rozwiązaniem są *checkpointy*.

- zatrzymaj "nowe transakcje"
- poczekaj aż bieżące transakcje się skończą (COMMIT lub ABORT)
- zapisz bufory na dysk
- zapisz informacje o CHECKPOINTcie do logu (<CHECKPOINT>)

Wtedy możemy zacząc odwarzanie od *checkpointu*. Typowy system ma checkpoint co kilka minut.

Minusy algorytmu CHECKPOINT

- żadnych nowych transakcji w tym czasie
- czekanie na koniec transakcji

Rozwiązanie:

Nonquiscent checkpoint

- <START CHECKPOINT(AT1, AT2, ...)> (ATN – aktywana transakcja N)
- nie wstrzymujemy nowych transakcji
- po zakończeniu transakcji z listy <END CHECKPOINT>

Awaria w trakcie *checkponitu*, trzeba:

- odtworzyć transakcje, które rozpoczęły się po <START CHECKPOINT>
- wszystkie transakcje z listy przy <START CHECKPOINT>

Awaria po <END CHECKPOINT>

- odtworzyć transakcje po <START CHECKPOINT>

Redo/undo logging.

- rekord logu zawiera starą i nową wartość: <T, X, v, w>
- rekord logu dla każdej operacji
- zapisz log na dysk zanim bufor X zostanie zapisany na dysk
- zapisz log po każdym <COMMIT T>
- rolling back
  - konstruuj listę zatwierdzonych transakcji – S
  - jeśli transakcja nie jest na liście S wykonaj UNDO
- rolling forward
  - wykonaj operację REDO dla transakcji z listy S

6.2 Isolation

Najprostsze rozwiazanie: seryjne wykonywanie transakcji. Trudniej: równolegle (ale szybciej)

6.2.1 Właściwości transakcji

- zużycie I/O
- zużycie CPU
- różnorodność: dużo krótkich transakcji lub "długie" + "krotkie" – potrzebne rozwiązanie

6.2.2 Integralność danych

- transakcja zapewnia integralność
- seryjne transakcje – nie ma problemu
- równoległe transakcje – problem

Problemy wynikające z równoczesnego dostępu

- problem utraconej modyfikacji (lost update)
- zależność od niezatwierdzonej wartości (dirty read, uncommited dependency)
- niespójna analiza (inconsistent analysis, nonrepeatable read) – transakcja dwukrotnie odczytuje tą samą jednostkę danych (wiersz tabeli) i dostaje różne dane (w międzyczasie była modyfikacja)
- phantom read – odczyt danych, które pojawiły się podczas transakcji (INSERT)

Zapewnienie wszystkich reguł jest *bardzo* kosztowne i zazwyczaj są pewne odstępstwa.

6.2.3 Poziom izolacji

Pewne ustawienie, które definiuje jakie zapewniamy rozwiązania izolacji (trzeba wybrać między kosztem obsługi a zapewnioną izolacją).

7 Dane przestrzenne

7.1 Zastosowania

- architektura
- planowanie przestrzenne
- transport
- systemy nawigacji
- zarządzanie sytuacjami kryzysowymi

7.2 Model danych

- przechowywane z reguły jako grafy
- *native* data type (np. *SDO_GEOMETRY* w Oracle)

7.2.1 Model rastrowy

Opiera się na zapisie macierzowym. Informacje reprezentowane są przez wartości numeryczne umieszczone w komórkach macierzy. Liczby umieszczone w miacerzy mogą reprezentować dowolne informacje (wysokość nad poziomem morza, wilgotność).

7.2.2 Model geometryczny

Wektory opisują pewne kształty (np. bryły), posiadamy pewien układ odniesienia dla punktów.

Występuje pewna hierarchia obketów. Na szczycie *warstwa przestrzenna*, której potomkami są na przykład: punkt, linia, ciąg linii, wielokąt.

Warstwa przestrzenna – zbiór geometrii związany w jakiś sposób znaczeniowo, na przykład sieć dróg, mapa miast, obszar leśny. Odpowiednikiem jest z reguły kolumna w bazie danych.

7.3 Indeksowanie

Standardowe indeksy się *nie sprawdzają*, dlatego potrzebne są dedykowane rozwiązania.

Indeks przestrzenny – umożliwia wykrzystywanie operatorów przestrzennych (niektóre operatory nie mogą być wykonane bez indeksu).

7.4 Przetwarzanie danych

Najczęściej dane są dwuwymiarowe (można przechowywać 3, 4 wymiarowe). Można je przetwarzać za pomocą bogatego zbioru opertorów.

SDO_FILTER w sposób przybliżony sprawdza czy dwa obiekty są w jakiejś zależności

SDO_NN szwraca najbliższe obiekty danego obiektu

SDO_NN_DISTANCE odległość od obiektu zwróconego przez *SDO_NN*

SDO_RELATE bada występowanie określonej relacji przestrzennej (ale dokładnie)

SDO_WITHIN_DISTANCE obiekty w pewnej odległości od danego obiektu

7.4.1 Przykłady zapytań

- W jakim państwie leży Praga?
- Czy Praga leży w tym samym państwie co Pilzno?
- Co leży bliżej Berlina: Gorzów Wielkopolski czy Szczecin?
- Czy Polska graniczy z Republiką Czeską?
- Jakie państwa sąsiadują z Polską?

7.5 Implementacja w Oracle 10g/11g

Podstawowy typ: *SDO_GEMOETRY* przechowywany w schemacie użytkowika *MDSYS*. Jest to struktura zbudowana z prostych elementów takich jak łuki, proste, punkty – reprezentuje pewien kształt. Jest niepodzielna z punktu widzenia operacji przestrzennych.

7.5.1 SDO_GTYPE

Określa typ geometrii w formacie *dltt*

d liczba wymiarów

l dotyczy systemu Linear Referencing System (LRS)

tt typ geometrii

- **00** unknown geometry
- **01** point
- **02** line or curve
- **03** polygon
- **04** collection
- **05** multipoint
- **06** multiline or multicurve
- **07** multipolygon

SDO_GEOMETRY in Oracle

7.6 Standard SQL/MM Part 3: Spatial

8 Akronimy

2PL Two-Phase Locking

ACID Atomicity, Consistency, Isolation, Durability

B–Tree Balanced Tree

GIS Geographic Information System

ISAM Index Sequential Access Method

LRS Linear Referencing System

LRU Least Recently Used

MTTF Mean Time To Failure

RAID Redundant Array of Independent Disks

SQL Structured Query Language

SZBD System Zarządzania Bazą Danych

WAL Write-Ahead Logging