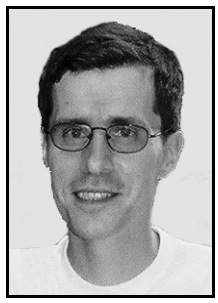
Jarosław GRAMACKI, Artur GRAMACKI
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Dane przestrzenne w bazach relacyjnych. Wykorzystanie danych przestrzennych, systemy zarządzania danymi przestrzennymi

Dr inż. Jarosław GRAMACKI



Pracuje w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego na stanowisku adiunkta. Jego zainteresowania koncentrują się wokół zagadnień związanych z bazami danych, hurtowniami danych oraz kopalniami danych. Pracuje głownie w środowisku bazy Oracle, jednak z dużą sympatią odnosi się do baz danych związanych z ruchem Open Source. Swoje doświadczenie stara się wykorzystywać zarówno w pracy dydaktycznej, jaki i aktywnie uczestnicząc w różnych projektach bazodanowych.

e-mail: j.gramacki@iie.uz.zgora.pl

Dr inż. Artur GRAMACKI

Pracuje w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego na stanowisku adiunkta. Interesuje się zagadnieniami związanych z bazami danych, w szczególności firmy Oracle ale też rozwiązaniami Open Source. Oprócz prowadzenia zajęć dydaktycznych stara się wykorzystywać swoją wiedzę uczestnicząc w różnych projektach informatycznych z tego zakresu. Brał udział w czterech projektach, których celem było przygotowanie systemów wspomagających działalność UZ.

e-mail:

Streszczenie

W artykule przedstawiono wybrane zagadnienia dotyczące pracy z tzw. mapami cyfrowymi (ang. Digital Maps). Pokazano podstawową funkcjonalność wybranych narzędzi dedykowanych do zarządzania danymi przestrzennymi. Pokazane rozwiązania korzystają z przestrzennych rozszerzeń bazy danych dostępnych w systemie komercyjnym Oracle. Zasady gromadzenia w bazach relacyjnych danych przestrzennych oraz korzystania z nich pokazano w pierwszej części pracy „Dane przestrzenne w bazach relacyjnych. Model danych, zapytania przestrzenne".

Słowa kluczowe: mapy cyfrowe, dane przestrzenne w bazach relacyjnych, bazy relacyjne, systemy klasy GIS, Oracle Spatial

Spatial data in relational databases. Spatial Data Management Systems

Abstract

In the paper some selected aspects of working with digital maps are presented. A basic functionality of systems dedicated to management of spatial data is shown. Those systems use spatial extensions of commercially available Oracle database. Background information on spatial data in relational databases is presented in the first part of the work: "Spatial Data in Relational Databases. Data Model and Spatial Queries".

Keywords: digital maps, spatial data in relational databases, relational databases, GIS systems, Oracle Spatial

1. Wstęp

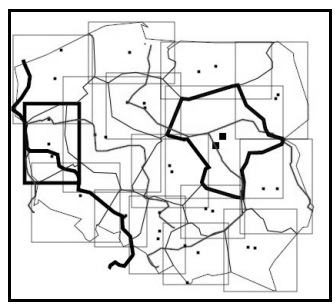
Sposoby gromadzenia w bazach relacyjnych danych przestrzennych pokazano w pierwszej części pracy „Dane przestrzenne w bazach relacyjnych. Model danych, zapytania przestrzenne". Moduł *Spatial* tam opisany udostępnia zaawansowane narzędzia programistyczne, które dopiero właściwie wykorzystane tworzą użyteczną z punktu widzenia użytkownika aplikację przestrzenną (GIS-ową). Końcowy użytkownik aplikacji GIS-owej wymaga jednak wygodnego interfejsu graficznego, który jest jedynie formą nakładki na bardzo rozbudowaną funkcjonalność modułu *Spatial*. W tej części pracy pokazano przykłady takich systemów.

2. Obiekty przestrzenne

W dalszej części pracy pokazano przykłady (wykorzystano w nich bazę Oracle, choć podobną funkcjonalność udostępniają bazy niekomercyjne, na przykład MySQL [2]) tworzenia obiektów przestrzennych oraz manipulowania nimi [4]. Zdefiniowano je w trzech tabelach: opisujących miasta (obiekty typu *point*), rzeki (obiekty typu *line string*), województwa (obiekty typu *n-point polygons*). Z uwagi na dużą ilość danych, w poniższych zapytaniach SQL przedstawiono jedynie niewielki fragment danych.

Na rysunkach poniżej pokazano dla ilustracji całe „spektrum" zgromadzonych danych w różnych ujęciach. Dla wygody „wyciągnięte" z bazy dane zapisano jako pliki graficzne w formacie SVG (ang. *Scalable Vector Graphics*) [3]. Format SVG umożliwia zakodowanie w tekstowym pliku XML dowolnych danych, w tym rysunków, zdjęć, szkiców itp. Na rysunku 1 pokazano zapisaną w systemie mapę Polski z zaznaczonymi:

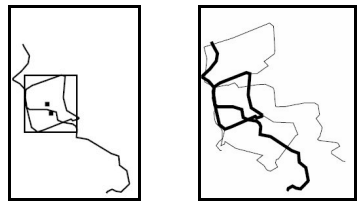
- obrysami województw oraz miastami wojewódzkimi,
- głównymi rzekami,
- „punktami ciężkości" województw,
- MBR-ami województw (ang. *minimum bounding rectangle*; patrz pierwsza część pracy).



Rys. 1. Przykładowe dane przestrzenne
Fig. 1. Sample spatial data

Kreską pogrubioną zaznaczono: rzekę Odrę, MBR województwa lubuskiego, obrys województwa mazowieckiego, lokalizację Warszawy (kwadracik na rzece Wiśle), „punkt ciężkości" województwa mazowieckiego (drugi kwadracik). Pozostałe obiekty, aby nie zaciemniać rysunku, są zaznaczone dużo cieńszymi kreskami i małymi punktami.

Zgromadzone dane można oczywiście prezentować i wybierać w dowolnych porcjach. Na rysunku 2 pokazano dwa przykładowe zestawy danych. Na rysunku z lewej strony pokazane są: obrys oraz MBR województwa lubuskiego, lokalizację Zielonej Góry oraz punkt ciężkości województwa. Na rysunku z prawej strony kreską pogrubioną pokazano rzekę Odrę oraz obrys województwa lubuskiego a kreską cienką dwa województwa ościenne oraz rzekę Wartę.



Rys. 2. Możliwości selektywnego pobierania obiektów przestrzennych
Fig. 2. A subset of objects taken from a whole set

3. Przykłady obiektów przestrzennych, ich indeksowanie oraz zapytania przestrzenne

Poniżej pokazano przykłady tworzenia obiektów przestrzennych oraz manipulowania nimi [4]. Przy dużej ilości danych symbol '...' (trzy kropki) oznacza, że pominięto ich dużą część (koordynaty przestrzenne poszczególnych punktów danej geometrii). Czasami jest ich bardzo dużo, gdyż kształty obiektów są skomplikowane. Przykładowo aby podać wszystkie współrzędne poligonu opisującego województwo mazowieckie należy użyć ok. 40 punktów.

Szczegóły dotyczące wykorzystywanych typów przestrzennych pokazano w pierwszej części pracy „Dane przestrzenne w bazach relacyjnych. Model danych, zapytania przestrzenne".

W pierwszej kolejności tworzymy tabele z kolumnami typu `SDO_GEOMETRY`. Opisują one obiekt przestrzenne typu poligon, odcinek łamany oraz punkt:

```
CREATE TABLE so_wojewodztwa (
  wo_id      NUMBER(3) NOT NULL,
  wo_nazwa   VARCHAR2(100), wo_ksztalt sdo_geometry );
CREATE TABLE so_rzeki, [so_miejscowosci] ...
```

Następnie wprowadzamy do nich dane opisujące (m.in.) topografie obiektów przestrzennych:

```
INSERT INTO so_wojewodztwa
VALUES (          -- 1-szy NULL, koordynata własna
  6, 'lubuskie',
  mdsys.sdo_geometry(2003, NULL, NULL,
  mdsys.sdo_elem_info_array(1, 1003, 1),
  mdsys.sdo_ordinate_array(31,83, ... ,60,28)) );
INSERT INTO so_rzeki, [so_miejscowosci] ...
```

W kolejnym kroku aktualizujemy widok `USER_SDO_GEOM_METADATA` zawierający potrzebne *metadane*. Jest to wymagane przed zbudowaniem indeksów przestrzennych. Obiekty składowane będą w kwadracie o umownym rozmiarze `135 x 125`. Korzystamy więc z własnej koordynaty typu kartezjańskiego:

```
INSERT INTO user_sdo_geom_metadata VALUES (
 'so_wojewodztwa',
-- [analogicznie dla so_miejscowosci, so_rzeki]
 'wo_ksztalt', mdsys.sdo_dim_array(
  mdsys.sdo_dim_element('x', 0, 135, 0.01),
  mdsys.sdo_dim_element('y', 0, 125, 0.01) ),
  NULL ); -- srid = NULL, własna koordynata
```

Następnie tworzymy indeks przestrzenny dla każdego typu obiekt przestrzennego:

```
CREATE INDEX rz_rtree ON so_rzeki
-- [analogicznie dla so_miejscowosci, so_wojewodztwa]
(rz_ksztalt) INDEXTYPE IS mdsys.spatial_index;
```

Wszystkie obiekty są już utworzone. Dalej podano przykłady zapytań przestrzennych operujących na utworzonych w poprzednich krokach danych.

Powierzchnie województw. Użyto funkcji z pakietu `SDO_GEOM`:

```
SELECT  wo_nazwa, sdo_geom.sdo_area(wo_ksztalt, 1) pow
FROM      so_wojewodztwa
ORDER BY  powierchnia DESC;

WO_NAZWA              POW
-------------------- -------------
mazowieckie              1318,2897
wielkopolskie           1106,73465
warmińsko-mazurskie      920,96865
...
opolskie                 335,99505
```

Długości rzek. Użyto funkcji z pakietu `SDO_GEOM`:

```
SELECT   rz_nazwa, sdo_geom.sdo_length(rz_ksztalt, 1) dl
FROM     so_rzeki
ORDER BY dlugosc DESC;

RZ_NAZWA        DL
--------------- ----------
Wisła           168,828583
Odra            137,903376
...
Nysa Łużycka    31,7182609
```

Topologiczna suma dwóch obszarów wyliczona z topografii obiektów przestrzennych:

```
SELECT
  sdo_geom.sdo_union(A.wo_ksztalt, B.wo_ksztalt,0.05)
  suma
FROM    so_wojewodztwa A, so_wojewodztwa B
WHERE   A.wo_nazwa = 'lubuskie' AND
        B.wo_nazwa = 'wielkopolskie';
```

Odnajdujemy miasta w promieniu 10 (dla przykładowych danych jednostki są nieokreślone) od rzeki Odry:

```
SELECT /*+ ORDERED */ MI.mi_nazwa
FROM   so_miejscowosci MI, so_rzeki RZ
WHERE  RZ.rz_nazwa = 'Odra' AND
       sdo_within_distance(MI.mi_ksztalt,
       RZ.rz_ksztalt,'distance = 10') = 'TRUE';

MI_NAZWA
-----------------
Opole
Wrocław
Zielona Góra
Szczecin
```

Odległości miast od rzeki Odra. Użyto operatora `SDO_NN`:

```
SELECT   MI.mi_nazwa,sdo_nn_distance(1) odleglosc
FROM     so_miejscowosci MI, so_rzeki RZ
WHERE    RZ.rz_nazwa = 'Odra' AND
         sdo_nn(MI.mi_ksztalt,
         RZ.rz_ksztalt,'SDO_NUM_RES=5',1) = 'TRUE'
ORDER BY odleglosc;

MI_NAZWA       ODLEGLOSC
-------------  -----------
Wrocław                  0
Szczecin       0,144829628
Opole          0,978158505
Zielona Góra    1,46493526
Gorzów Wlkp.    10,3627795
```

4. Zaawansowane narzędzia

Oprócz „niskopoziomowego" dostępu do danych z użyciem pokazanych wcześniej funkcji przestrzennych API, operatorów przestrzennych, itp. dostępne są gotowe narzędzia, które ukrywają przed użytkownikiem szczegóły korzystania z rozbudowanego interfejsu programistycznego. Należą do nich udostępniane przez Oracle takie narzędzia jak *MapViewer* czy *Oracle Network Data Model Editor* [5]. Oczywiście zakłada się, że dane przestrzenne, które oglądamy z ich użyciem już znajdują się w bazie.

MapViewer, cechy charakterystyczne

Aplikacja od strony **użytkownika** działa w środowisku przeglądarki WWW. W warstwie **środkowej** (aplikacyjnej) używamy albo serwera aplikacyjnego *Oracle Application Server*, bądź też wersji *standalone* kontenera J2EE [6] (*Oracle Application Server Containers for J2EE - OC4J*). Wersja *standalone* wykorzystywana bywa głównie do tworzenia oraz do testowania aplikacji. Wersja z serwerem aplikacyjnym używana jest w środowisku produkcyjnym. Funkcjonalnie obie wersje są identyczne. Trzecią warstwę tworzy **baza Oracle** z rozszerzeniem *Spatial* (lub *Locator*).

Do „wytworzenia" użytecznej aplikacji przestrzennej korzysta się ze zbioru funkcji API zbudowanego w technologii *JavaBeans.* Funkcje te zapewniają komunikację z właściwym serwisem działającym w warstwie środkowej (*MapViewer Service*). Komunikacja odbywa się na zasadzie wysyłania komunikatów w formacie XML. Możliwa jest też komunikacja ze wspomnianym serwisem z użyciem biblioteki znaczników JSP (*JSP Tag Library*) z pozio-

mu stron HTML. Dostarczane przez system znaczniki udostępniają jednak jedynie podzbiór funkcjonalności całego środowiska.

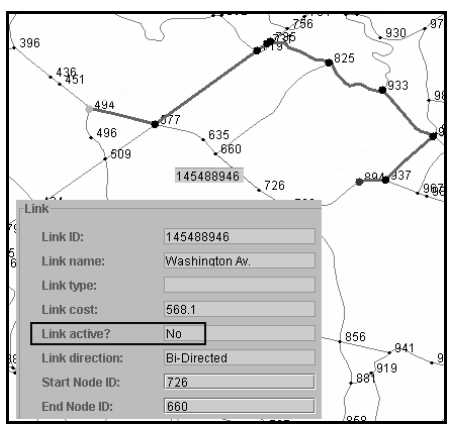
Możliwe sposoby bodowy aplikacji przestrzennej w środowisku *MapViewer* to: aplikacje Java, aplety Java, aplikacje JSP, servlety Java w zewnętrznym (w stosunku do *MapViewer Service*) środowisku J2EE.

Wydaje się jednak, że takie rozwiązanie jak *MapViewer*, będące *de facto* aplikacją uruchamianą z poziomu przeglądarki WWW, nie jest jeszcze na tyle technologicznie zaawansowane, aby mogło konkurować z bardziej „tradycyjnymi" narzędziami obsługującymi mapy cyfrowe.

Oracle Network Data Model Editor, cechy charakterystyczne

Aplikacja operuje ona na danych, które tworzą tzw. sieciowy model przestrzenny (ang. *Spatial Topology and Network Data Models)*. Model ten jest rozszerzeniem podstawowej funkcjonalności modułu *Spatial*. W części *Topology* dotyczy możliwości przechowywania struktury połączeń elementów. W części *Network* dotyczy możliwości przechowywania takich dodatkowych informacji, jak na przykład kierunek połączenia, koszt połączenia, koszt punktu, itp.

Na przykładzie (patrz rysunek 3) pokazano fragment hipotetycznej sieci dróg. Jest to niewielki fragment sieci liczącej 10505 dróg (licząc za drogę (*link*) każde połączenie pomiędzy dwoma punktami.. Obliczono najkrótszą drogę od punku 494 do punktu 894. Z uwagi na „zamkniętą" drogę pomiędzy punktami 660 a 726, wyliczony przebieg nie jest najkrótszym w sensie topograficznym. W tego typu zadaniach można również korzystać ze zdefiniowanego „kosztu" danego połączenia (atrybut *link cost*), celem zmiany kryterium wyszukiwania. Podobnie można uwzględniać lub nie jedno/dwu kierunkowość połączenia (atrybut *link direction*).



Rys. 3. Aplikacja wykorzystująca aspekt sieciowy obiektów przestrzennych
Fig. 3. Application based on the Network Model of spatial data

Topografia sieci (kolumna typu `SDO_GEOMETRY`) przechowywana jest w jednej tabeli wraz z informacjami topologicznymi [1].

```
CREATE TABLE NETWORK_LINK (
  LINK_ID        NUMBER,    LINK_NAME    VARCHAR2(200),
  START_NODE_ID  NUMBER,    END_NODE_ID  NUMBER,
  LINK_TYPE  VARCHAR2(200), ACTIVE       VARCHAR2(1),
  LINK_LEVEL     NUMBER,    GEOMETRY     SDO_GEOMETRY,
  COST           NUMBER);
```

Podobnie jak w przypadku podstawowej funkcjonalności modułu *Spatial*, tak i tu całości dopełniają struktury przechowujące tzw. metadane. Do budowy takiej aplikacji jak pokazano na rysunku 3 korzystać można z API opartego o PL/SQL lub opartego o język Java. Oba rodzaje API oferują tą samą funkcjonalność i umożliwiają tworzenie sieci topologicznej (w PL/SQL są to pakiety `SDO_NET` oraz `SDO_NET_MEM` a w Javie elementy pakietu `oracle.spatial.network.*`).

Ponadto udostępnia się w nich narzędzia do przeprowadzania analiz sieci (jak na przykład pokazanego wyżej szukania najkrótszej drogi). Aplikacja pokazana na rysunku 3 powstała z wykorzystaniem API Java. Przykładowo do wyliczenia wspominanej najkrótszej drogi służą funkcje `SHORTEST_PATH_DIJKSTRA` lub `SHORTEST_PATH`. (różnią się zaimplementowanym algorytmem):

```
path_ID := SDO_NET_MEM.NETWORK_MANAGER.SHORTEST_PATH(
Network_name, 494,894);

DBMS_OUTPUT.PUT_LINE('Najkrótsza droga z węzła 494 do
węzła 894 opisana numerem drogi: ' || path_ID);
-- Wynik otrzymujemy w postaci numeru wyznaczonej ścieżki
-- (ang. path)
```

W dalszej części jest oczywiście możliwe (poprzez użycie stosownych funkcji pakietu) pobranie na przykład numerów „odwiedzanych" węzłów, ich danych i w konsekwencji wyrysowanie wyliczonej najkrótszej drogi.

Przykładowo funkcja `SDO_NET_MEM.PATH.GET_LINK_IDS` ma za zadanie „wyrysowanie" obliczonej drogi:

```
res_array :=
  SDO_NET_MEM.PATH.GET_LINK_IDS(Network_name, path_id);
DBMS_OUTPUT.PUT_LINE( path_id || ' ma przebieg: ');
FOR indx IN res_array.FIRST..res_array.LAST
 LOOP DBMS_OUTPUT.PUT(res_array(indx) || ' '); END LOOP;
...
21 ma przebieg:
494 577 719 ... 894
```

W interfejsie Java skorzystamy z klasy `oracle.spatial.network.NetworkManager`. Znajdziemy tam metodę `shortestPath`. Dla ilustracji pokazujemy tu jedynie jej prototyp (`Path` i `Network` są stosownymi interfejsami Java):

```
public static Path shortestPath(
 Network network, int startNodeID, int goalNodeID)
```

Tworzenie rozbudowanych sieci, z uwagi na złożone informacje topograficzne, w praktyce odbywa się poprzez mniej lub bardzie automatyczne (wsadowe) wprowadzanie danych z zewnętrznych systemów udostępniających współrzędne geodezyjne danych obiektów. Nadmieńmy, że również popularny system GPS może być źródłem takich danych.

5. Podsumowanie

W artykule zaprezentowano podstawowe cechy wybranych narzędzi obsługujących dane przestrzenne przechowywane bezpośrednio w relacyjnych bazach danych. Pokazano to na przykładzie rozwiązań współpracujących z bazą danych Oracle. Jak na razie rozwiązania oferowane przez firmę Oracle są bardziej zaawansowane niż te, które możemy spotkać w systemach darmowych na przykład MySQL.

6. Literatura

[1] Oracle Spatial, Oracle Locator, Extensions for Oracle Database, http://www.oracle.com/technology/roducts/spatial/
[2] MySQL, MySQL 5.1 Reference Manual, Chapter 18, Spatial Extensions in MySQL, http://dev.mysql.com/doc/refman/5.1/en/index.html
[3] XML Graphics for the Web, Scalable Vector Graphics, http://www.w3.org/TR/SVG/
[4] Na podstawie danych przestrzennych przygotowanych przez Krzysztofa Jankiewicza z Politechniki Poznańskiej, http://www.cs.put.poznan.pl/kjankiewicz/
[5] Narzędzia dostępne z technologicznego portalu firmy Oracle http://otn.oracle.com/
[6] Oracle Containers for J2EE (OC4J), http://www.oracle.com/technology/tech/java/oc4j/index.html
[7] Open Geospatial Consortium, Inc. (OGC),
[8] http://www.opengeospatial.org/

Artykuł recenzowany