ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1999
Seria: INFORMATYKA z. 37 Nr kol. 1421

Piotr BAJERSKI
Politechnika Śląska, Instytut Informatyki

MOŻLIWOŚCI ZAPISU W JĘZYKU SQL ZAPYTAŃ PRZESTRZENNYCH WYKORZYSTUJĄCYCH APROKSYMACJE OBIEKTÓW

Streszczenie. W artykule omówiono możliwości formułowania w języku SQL zapytań dotyczących relacji przestrzennych pomiędzy aproksymacjami obiektów. Przedstawiono dwie metody zapisu w relacyjnej bazie danych aproksymacji obiektów przestrzennych drzewem czwórkowym uporządkowanym krzywą fraktalną N-Peano.

ON USING SQL QUERY LANGUAGE FOR WRITING SPATIAL QUERIES BASED ON OBJECTS APPROXIMATION

Summary. The paper presents a conception of using SQL for formulation of queries concerning spatial relations among objects approximations. Two methods of storing spatial objects approximated by means of a quadtree ordered by N-Peano fractal curve in a relational database are described.

1. Wstęp

We współczesnej nauce i technice często występuje konieczność badania przestrzennych relacji między obiektami. Jakkolwiek zapytania mogą pochodzić z różnych dziedzin, to wykorzystywane algorytmy i struktury danych są podobne. Zapytania takie mogą również dotyczyć rozkładu zadanej cechy w przestrzeni, nazywanej dalej rozkładem przestrzennym.

Dane opisujące położenie obiektów w przestrzeni oraz ich kształt (geometrię) będą w dalszej części nazywane danymi przestrzennymi. Dane przestrzenne dotyczące danego obiektu są nazywane jego atrybutami przestrzennymi. Zbiór obiektów przestrzennych posiadających ten sam zestaw atrybutów definiuje relację przestrzenną.

Zapytania dotyczące atrybutów przestrzennych nazywane są zapytaniami przestrzennymi. Ze względu na złożoność dostępu do danych są one dzielone na dwie grupy [2, 8]:

- wymagające pojedynczego dostępu do danych w trakcie wykonywania zapytania (ang. single-scan). Charakteryzują się liniową lub lepszą zależnością czasu wykonania zapytania od liczby obiektów w tablicy. Najbardziej znanymi przykładami tej grupy zapytań są:
  - zapytania typu położenie punktu względem wielokąta (ang. point-in-polygon query): mając dany punkt P znajdź wszystkie prostokąty R, zawierające P,
  - zapytania typu okno (ang. window query): mając dany prostokąt S znajdź wszystkie prostokąty R przecinające (zawierające, należące do) S,
- wymagające wielokrotnego dostępu do niektórych obiektów w trakcie wykonywania zapytania (ang. multiple-scan). Najbardziej znanym przykładem takiego zapytania jest złączenie przestrzenne (ang. spatial join), które znajduje wszystkie pary obiektów wskazanych typów, spełniające zadaną własność. Na przykład, dla tablic *lasy* i *miasta*, gdzie atrybut przestrzenny reprezentuje granice lasów i miast, złączenie przestrzenne pozwala odpowiedzieć na pytanie: „Które lasy leżą na terenach miejskich?". Wyznaczenie odpowiedzi w pesymistycznym przypadku wymaga porównania wszystkich obiektów z pierwszego zbioru ze wszystkimi obiektami z drugiego zbioru.

W zapytaniach przestrzennych często stosuje się przetwarzanie dwuetapowe. Na pierwszym etapie wykorzystuje się specjalne struktury danych oraz proste i szybkie algorytmy umożliwiające wyszukanie obiektów, które mogą spełniać warunki zapytania. Na drugim etapie wykonuje się dokładne, bardziej czasochłonne obliczenia, aby określić, które obiekty rzeczywiście spełniają kryteria zapytania.

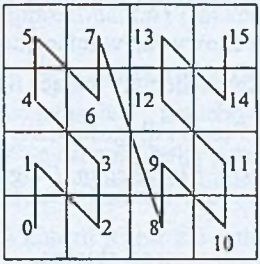
Można wyróżnić dwie metody zapamiętania rozkładu przestrzennego opisanego za pomocą izolinii przedstawiających granice zadanych przedziałów:

- wektorowa – za pomocą wektorów zapamiętywany jest przebieg granic przedziałów, wymaga rozszerzenia języka SQL o predykaty badające relacje przestrzenne między obiektami, podejście to zostało wykorzystane w przestrzennym rozszerzeniu serwera Oracle (ang. Spatial Data Option) [10],
- powierzchniowa – zapamiętywane są aproksymacje obszarów odpowiadających poszczególnym przedziałom.

Praca poświęcona jest drugiej metodzie.

2. Krzywa N-Peano

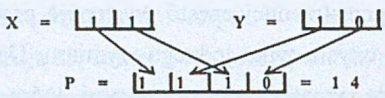
W systemach informacji przestrzennej często występuje problem dostępu do obiektów dwu- i trójwymiarowych przy użyciu tylko jednego wymiaru. Uporządkowanie takie można uzyskać konstruując ścieżkę w rozpatrywanej przestrzeni, która przechodzi przez każdy jej punkt dokładnie raz. Aby taka ścieżka miała skończoną długość, punkty tej przestrzeni muszą zostać zastąpione elementami o niezerowych rozmiarach. W dalszej części opis zostanie ograniczony do przestrzeni dwuwymiarowej, chociaż przedstawione koncepcje można uogólnić na przestrzeń o większej liczbie wymiarów. Przyjmijmy, że elementami wypełniającymi przestrzeń są kwadraty elementarne o jednostkowej długości boku. Przy takich założeniach nie można rozróżnić elementów mniejszych od kwadratów elementarnych. Długość boku kwadratów elementarnych jest nazywana rozdzielczością. W dalszej części, jeżeli nie będzie to prowadziło do nieporozumień, kwadrat elementarny będzie w skrócie nazywany kwadratem. Ścieżka przechodząca przez wszystkie kwadraty wypełniające zadaną przestrzeń odpowiada krzywej wypełniającej tę przestrzeń. Kwadrat jest identyfikowany numerem kroku, w którym do niego dochodzimy, nazywanym kluczem. Istnieje wiele krzywych posiadających własność wypełniania przestrzeni [7, 8, 9], różniących się takimi parametrami, jak długość ścieżki, średnia odległość między sąsiadami, łatwość przejścia ze współrzędnych na klucz i odtworzenia współrzędnych na podstawie klucza czy możliwość powiększania obszaru bez modyfikacji wykorzystanych dotąd kluczy.



Rys. 1. Krzywa N-Peano
Fig. 1. N-Peano curve

Krzywa, która wykorzystuje fraktale do wypełnienia przestrzeni, jest nazywana krzywą fraktalną. Krzywą fraktalną zapewniającą proste przejście ze współrzędnych na klucz jest krzywa N-Peano [8, 9], nazywana również sekwencją Mortona, krzywą N i krzywą Z (rys. 1). Klucz N-Peano, określający położenie kwadratu na krzywej, tworzony jest poprzez przeplot bitów poszczególnych współrzędnych. Dla przypadku dwuwymiarowego parzyste bity klucza Peano odpowiadają współrzędnej X, a nieparzyste współrzędnej Y (rys. 2). Krzywa N-Peano pozwala efektywnie zapamiętać obiekty o różnej wielkości. Gdy obszar przestrzeni, dla

którego są zapamiętywane obiekty, jest powiększany, krzywa nie wymaga zmiany wcześniej wykorzystanych kluczy.



Rys. 2. Związek miedzy kluczem Peano i współrzędnymi X, Y
Fig. 2. The relation between Peano key and X, Y co-ordinates

Drzewo czwórkowe (ang. quadtree) jest strukturą danych klasy drzewo [6, 11, 13], powstałą poprzez rekurencyjny podział przestrzeni 2-wymiarowej na cztery części o takich samych wymiarach. Metoda ta daje najlepsze wyniki, jeżeli drzewo zbudowane jest z kwadratów o boku będącym potęgą liczby 2. Podczas aproksymacji obiektu drzewem proces podziałów kończy się, gdy cały kwadrat leży na jego obszarze lub gdy osiągnie założoną granicę rozdzielczości. Liniowym drzewem czwórkowym będzie nazywane drzewo czwórkowe, którego elementy zostały uporządkowane krzywą fraktalną.

3. Relacja Peano

3.1. Postacie relacji Peano

Aproksymację obiektów przestrzennych liniowym drzewem czwórkowym uporządkowanym krzywą N-Peano można przechowywać w relacyjnej bazie danych w relacjach o różnych schematach. Poniżej omówiono dwa schematy mające największe znaczenie praktyczne [8].

- schemat 1 – minimalny klucz Peano i długość boku:

  SchemRelPeano1 (IdObiektu, KluczPeano, DługoscBoku, Atrybuty)

  gdzie:
  - *IdObiektu* - identyfikator kodowanego obiektu,
  - *KluczPeano* - klucz Peano lewego dolnego kwadratu elementarnego,
  - *DługoscBoku* - długość boku kwadratu, wyrażona liczbą kwadratów elementarnych,
  - *Atrybuty* - atrybuty opisowe, związane z danym fragmentem obiektu o identyfikatorze *IdObiektu*.
- schemat 2 – zakres kluczy Peano:

  SchemRelPeano2 (Id obiektu, KluczPeanoPoczatku, KluczPeanoKonca, Atrybuty)

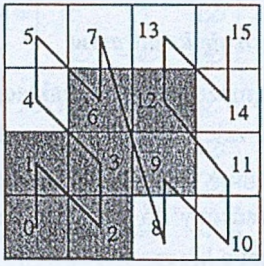
  gdzie:
  - *KluczPeanoPoczatku* - klucz Peano lewego dolnego kwadratu elementarnego (minimalna wartość klucza Peano odpowiadająca kwadratowi),

- *KluczPeanoKonca* - klucz Peano prawego górnego kwadratu elementarnego (maksymalna wartość klucza Peano odpowiadająca kwadratowi),
- pozostałe atrybuty mają takie samo znaczenie jak w schemacie 1.

Każda krotka relacji o przedstawionych schematach opisuje jeden kwadrat tworzący aproksymację.

Ponieważ schematy *SchemRelPeano1* i *SchemRelPeano2* nie spełniają wymagań pierwszej postaci normalnej [14] - jedna krotka zawierająca opis obiektu zakodowanego krzywą Peano może odpowiadać wielu innym krotkom - wprowadzono w [9, 10] pojęcie relacji Peano. Zdefiniowano dla niej poziomy poprawności rozszerzające postacie normalne podane dla relacji o reguły integralności, jakie musi spełniać zapis liniowego drzewa czwórkowego.

3.2. Poziomy poprawności relacji Peano



IdOb	KP	Bok
A	0	1
A	1	1
A	2	1
A	3	2
A	9	1

Rys. 3. Niepoprawna aproksymacja obiektu przestrzennego
Fig. 3. An incorrect approximation of an spatial object

Rozważmy aproksymację obiektu A przedstawioną na rys. 3. Z lewej strony rysunku przedstawiono zawartość tablicy, o schemacie 1, przechowującej aproksymację obiektu A. Klucz Peano jest oznaczany przez *KP*, a długość boku przez *Bok*. Widać, że kwadraty zapisane w tabeli nie mogły powstać w trakcie budowy drzewa czwórkowego poprzez rekurencyjny podział przestrzeni. Kwadrat o kluczu 3 jest źle położony, dodatkowo pokrywa się z kwadratem o kluczu 9.

3.2.1. Pierwszy poziom poprawności - prawidłowo położone kwadraty

Aby móc wykonywać operacje na aproksymacjach obiektów, tworzące je kwadraty muszą mieć poprawną wielkość i być dobrze położone. Wielkość kwadratu jest poprawna, jeżeli długość jego boku jest potęgą liczby 2 i zakres kluczy Peano odpowiadający temu kwadratowi zawiera się w dopuszczalnym zakresie kluczy. Kwadrat jest dobrze położony, jeżeli w zapisie bitowym, licząc od najmniej znaczącego bitu, liczba zer w kluczu jest co najmniej dwukrotnie większa od liczby zer w długości boku. Na rys. 4 tabela z lewej strony przedsta-

wia przykładowy obiekt z rys. 3 doprowadzony do pierwszego poziomu poprawności. Kwadrat o kluczu 3 i boku 2 został rozbity na mniejsze kwadraty.

IdOb	KP	Bok
A	0	1
A	1	1
A	2	1
A	3	1
A	6	1
A	9	1
A	9	1
A	12	1

IdOb	KP	Bok
A	0	1
A	1	1
A	2	1
A	3	1
A	6	1
A	9	1
A	12	1

IdOb	KP	Bok
A	0	2
A	6	1
A	9	1
A	12	1

Rys. 4. Wynik normalizacji aproksymacji obiektu z rys. 3 – pierwszy poziom poprawności (po lewej),drugi (w środku), trzeci (z prawej)

Fig. 4. The result of normalisation of the object from fig. 3 – first conformance level (left), second (middle), third (right)

3.2.2. Drugi poziom poprawności - brak nakładających się kwadratów

Relacja Peano na drugim poziomie poprawności nie może zawierać nakładających się kwadratów. Przy przejściu na ten poziom sprawdzana jest zależność, czy zakres kluczy jednego kwadratu nie pokrywa się z zakresem kluczy innego kwadratu. Na rys. 4 tabela w środku przedstawia przykładowy obiekt z rys. 3 doprowadzony do drugiego poziomu poprawności - wyeliminowany został powtarzający się kwadrat o kluczu 9.

3.2.3. Trzeci poziom poprawności - zwarte kwadraty

Kwadraty tworzące większy kwadrat są łączone. W przykładzie można połączyć kwadraty o kluczach 0, 1, 2 i 3, tworząc jeden kwadrat o kluczu 0 i boku 2. Na rys. 4 tabela z prawej strony przedstawia przykładowy obiekt z rys. 3 doprowadzony do trzeciego poziomu poprawności.

Poziomy poprawności relacji Peano i postacie normalne relacji ułatwiają utrzymanie integralności danych. Schemat 2 wymaga, aby kwadraty spełniały przynajmniej pierwszy poziom poprawności – zakres kluczy Peano odpowiadający kwadratowi musi być ciągły.

4. Przestrzenne relacje topologiczne dwuwymiarowych obiektów

Przedstawione określenie binarnych przestrzennych relacji topologicznych (ang. binary topological spatial relation) opiera się na badaniu dla dwóch zbiorów punktów, oznaczonych tutaj przez A i B, relacji między ich wnętrzami (oznaczonymi przez A° i B°), brzegami (oznaczonymi przez ∂A i ∂B) i dopełnieniami (oznaczonymi przez A^- i B^-) [3, 4, 5]. Istnieje

dziewięć możliwych przecięć wyróżnionych podzbiorów punktów (1). Każde z nich może być niepuste (oznaczone przez $\neg\varnothing$) lub puste (oznaczone przez $\varnothing$). Dana relacja topologiczna jest charakteryzowana przez występowanie konkretnego układu przecięć. Możliwych jest 512 kombinacji, jednakże pomiędzy wielokątami bez dziur, składającymi się tylko z jednego kawałka, może wystąpić osiem binarnych przestrzennych relacji topologicznych.

$$I(A,B)=\begin{pmatrix} A^{\circ}\cap B^{\circ} & A^{\circ}\cap \partial B & A^{\circ}\cap B^{-} \\ \partial A\cap B^{\circ} & \partial A\cap \partial B & \partial A\cap B^{-} \\ A^{-}\cap B^{\circ} & A^{-}\cap \partial B & A^{-}\cap B^{-} \end{pmatrix} \quad (1)$$

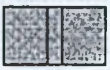
Dla każdej relacji podano charakteryzującą ją macierz przecięć *I(A, B)* i ilustrację graficzną.

1. obiekty są rozłączne (ang. disjoint)

$$I(A,B)=\begin{pmatrix} \varnothing & \varnothing & \neg\varnothing \\ \varnothing & \varnothing & \neg\varnothing \\ \neg\varnothing & \neg\varnothing & \neg\varnothing \end{pmatrix}$$

2. obiekty nakładają się (ang. overlap)

$$I(A,B)=\begin{pmatrix} \neg\varnothing & \neg\varnothing & \neg\varnothing \\ \neg\varnothing & \neg\varnothing & \neg\varnothing \\ \neg\varnothing & \neg\varnothing & \neg\varnothing \end{pmatrix}$$



3. obiekty mają wspólną granicę (ang. meet)

$$I(A,B)=\begin{pmatrix} \varnothing & \varnothing & \neg\varnothing \\ \varnothing & \neg\varnothing & \neg\varnothing \\ \neg\varnothing & \neg\varnothing & \neg\varnothing \end{pmatrix}$$

4. obiekty równają się sobie (ang. equal)

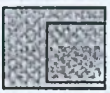
$$I(A,B)=\begin{pmatrix} \neg\varnothing & \varnothing & \varnothing \\ \varnothing & \neg\varnothing & \varnothing \\ \varnothing & \varnothing & \neg\varnothing \end{pmatrix}$$

5. obiekt A zawiera obiekt B (ang. contains)

$$I(A,B)=\begin{pmatrix} \neg\varnothing & \neg\varnothing & \neg\varnothing \\ \varnothing & \varnothing & \neg\varnothing \\ \varnothing & \varnothing & \neg\varnothing \end{pmatrix}$$

6. obiekt A pokrywa obiekt B (ang. cover)

$$I(A,B)=\begin{pmatrix} \neg\varnothing & \neg\varnothing & \neg\varnothing \\ \varnothing & \neg\varnothing & \neg\varnothing \\ \varnothing & \varnothing & \neg\varnothing \end{pmatrix}$$



7. obiekt A jest zawarty w obiekcie B (ang. inside)

$$I(A,B) = \begin{pmatrix} \neg\varnothing & \varnothing & \varnothing \\ \neg\varnothing & \varnothing & \varnothing \\ \neg\varnothing & \neg\varnothing & \neg\varnothing \end{pmatrix}$$

8. obiekt A jest pokryty przez obiekt B (ang. coveredBy)

$$I(A,B) = \begin{pmatrix} \neg\varnothing & \varnothing & \varnothing \\ \neg\varnothing & \neg\varnothing & \varnothing \\ \neg\varnothing & \neg\varnothing & \neg\varnothing \end{pmatrix}$$

5. Badanie relacji topologicznych aproksymacji obiektów dwuwymiarowych w języku SQL

Rozważmy obiekty dwuwymiarowe, których identyfikatory i nazwy zostały zapisane w relacji:

DaneOb (IdOb, NazwaOb),

Ich aproksymacje liniowym drzewem czwórkowym uporządkowanym krzywą N-Peano zostały zapisane w relacji Peano o schemacie 2:

AproksOb*P2 (IdOb, KPpocz, KPkon)*

Kwadraty tworzące aproksymacje spełniają wymagania trzeciego poziomu poprawności. Schemat 2 został wybrany w celu skrócenia zapisu. Przedstawione zapytania można zapisać dla relacji o schemacie 1 korzystając z następującej równości:

*KluczPeanoKonca = KluczPeano + DługoscBoku * DługoscBoku – 1*

Warunek, czy dwa obiekty nakładają się na siebie, można sprawdzić testując, czy dowolne dwa kwadraty z aproksymacji tych obiektów nakładają się na siebie. Sprawdzenie nakładania się kwadratów, spełniających pierwszy poziom poprawności, sprowadza się do sprawdzenia, czy istnieje niepuste przecięcie odpowiadających im odcinków na krzywej Peano (czyli istnieje przynajmniej jeden kwadrat elementarny, który należy do obydwu odcinków). Jeżeli kwadrat nie byłby prawidłowo położony, mogłoby mu odpowiadać kilka zakresów klucza Peano i nie można by było zastosować podanych algorytmów. Niech *k.KPpocz* oznacza minimalną wartość klucza Peano kwadratu *k*, a *k.KPkon* wartość maksymalną. Oznaczmy dwa kwadraty przez *k1* i *k2*. Warunkiem koniecznym i wystarczającym istnienia niepustej części wspólnej prawidłowo położonych kwadratów *k1* i *k2* jest, aby minimalny klucz Peano *k1* był mniejszy lub równy maksymalnemu kluczowi Peano *k2* i minimalny klucz

Peano *k2* był mniejszy, lub równy maksymalnemu kluczowi Peano *k1*. Można to zapisać za pomocą następującego predykatu:

NakładKwKP (k1, k2) ≡ k1.KPpocz <= k2.KPkon i k1.KPkon >= k2.KPpocz

Podane poniżej zapytania w języku SQL, sprawdzające zachodzenie wyróżnionych powyżej relacji topologicznych między parami obiektów, zwracają nazwy obiektów, pomiędzy którymi zachodzi dana relacja.

Sprawdzenie zachodzenia relacji 1 (obiekty są rozłączne) sprowadza się do przetestowania, czy żaden z kwadratów pierwszej aproksymacji nie ma części wspólnej z dowolnym kwadratem drugiej aproksymacji.

```
SELECT d1.NazwaOb, d2.NazwaOb
FROM DaneOb d1, DaneOb d2
WHERE d1.IdOb < d2.IdOb AND NOT EXISTS (
        SELECT *
        FROM AproksObP2 a1, AproksObP2 a2
        WHERE a1.IdOb = d1.IdOb AND a2.IdOb = d2.IdOb AND a1.KPpocz <= a2.KPkon
            AND a1.KPkon >= a2.KPpocz);
```

Stosunkowo łatwo jest sprawdzić zachodzenie relacji 2 (obiekty nakładają się) lub relacji 3 (obiekty mają wspólną granicę). Wystarczy przetestować, czy jednocześnie istnieje niepuste przecięcie aproksymacji obydwu obiektów (pierwsze pytanie zagnieżdżone) i obydwie aproksymacje zawierają wartości klucza Peano, które nie należą do drugiej aproksymacji (dwa kolejne pytania zagnieżdżone).

```
SELECT d1.NazwaOb, d2.NazwaOb
FROM DaneOb d1, DaneOb d2
WHERE d1.IdOb < d2.IdOb AND EXISTS (
        SELECT *
        FROM AproksObP2 a1, AproksObP2 a2
        WHERE a1.IdOb = d1.IdOb AND a2.IdOb = d2.IdOb AND a1.KPpocz <= a2.KPkon
            AND a1.KPkon >= a2.KPpocz)
    AND EXISTS (
        SELECT *
        FROM AproksObP2 a3
        WHERE a3.IdOb = d1.IdOb AND NOT EXISTS (
            SELECT *
            FROM AproksObP2 a4
            WHERE a4.IdOb = d2.IdOb AND
            a3.KPpocz >= a4.KPpocz AND a3.KPkon <= a4.KPkon))
    AND EXISTS (
        SELECT *
        FROM AproksObP2 a5
        WHERE a5.IdOb = d2.IdOb AND NOT EXISTS (
            SELECT *
            FROM AproksObP2 a6
```

```
                WHERE a6.IdOb = d1.IdOb AND
                a5.KPpocz >= a6.KPpocz AND a5.KPkon <= a6.KPkon));
```

Podstawowym problemem przy rozróżnieniu, czy zachodzi relacja 2 czy 3, jest sprawdzenie, czy dany kwadrat elementarny należy do brzegu obiektu. Efektywne zbadanie zachodzenia tej własności wymaga zastosowania języka proceduralnego, jednoczesnego dostępu do wszystkich kwadratów tworzących aproksymacje badanego obiektu i zdekomponowania ich na kwadraty elementarne.

Aby stwierdzić zachodzenie relacji 4 (obiekty równają się sobie), wystarczy sprawdzić, czy składają się z takich samych elementów.

```
SELECT d1.NazwaOb, d2.NazwaOb
FROM DaneOb d1, DaneOb d2
WHERE d1.IdOb < d2.IdOb AND NOT EXISTS (
        SELECT *
        FROM AproksObP2 a1
        WHERE a1.IdOb = d1.IdOb AND NOT EXISTS (
                SELECT *
                FROM AproksObP2 a2
                WHERE a2.IdOb = d2.IdOb AND
                a1.KPpocz = a2.KPpocz AND a1.KPkon = a2.KPkon))
    AND NOT EXISTS (
        SELECT *
        FROM AproksObP2 a3
        WHERE a3.IdOb = d2.IdOb AND NOT EXISTS (
                SELECT *
                FROM AproksObP2 a4
                WHERE a4.IdOb = d1.IdOb AND
                a3.KPpocz = a4.KPpocz AND a3.KPkon = a4.KPkon));
```

Przy badaniu zachodzenia relacji 5 (obiekt A zawiera obiekt B) i 6 (obiekt A pokrywa obiekt B) problem jest podobny do badania zachodzenia relacji 2 i 3 – łatwo jest sprawdzić, czy zachodzi któraś z tych relacji, trudno je natomiast rozróżnić, ze względu na kłopoty związane z zapisem warunków na brzeg obiektu. W celu przetestowania zachodzenia relacji 5 lub 6 należy sprawdzić, czy kwadraty tworzące aproksymację obiektu B są pokrywane przez kwadraty tworzące aproksymację obiektu A (pierwsze pytanie zagnieżdżone) oraz czy aproksymacja obiektu A zawiera kwadraty nie należące do aproksymacji obiektu B (drugie pytanie zagnieżdżone).

```
SELECT d1.NazwaOb, d2.NazwaOb
FROM DaneOb d1, DaneOb d2
WHERE d1.IdOb <> d2.IdOb AND NOT EXISTS (
        SELECT *
        FROM AproksObP2 a1
        WHERE a1.IdOb = d2.IdOb AND NOT EXISTS (
                SELECT *
```

```
                FROM AproksObP2 a2
                WHERE a2.IdOb = d1.IdOb AND
                a1.KPpocz >= a2.KPpocz AND a1.KPkon <= a2.KPkon))
    AND EXISTS (
            SELECT *
            FROM AproksObP2 a3
            WHERE a3.IdOb = d1.IdOb AND NOT EXISTS (
                    SELECT *
                    FROM AproksObP2 a4
                    WHERE a4.IdOb = d2.IdOb AND
                    a3.KPpocz >= a4.KPpocz AND a3.KPkon <= a4.KPkon));
```

Dla relacji 7 i 8 problem wygląda analogicznie jak dla relacji 5 i 6, z tym że obiekty są zamienione miejscami.

Ponieważ przetwarzanie odbywa się na płaszczyźnie dyskretnej, ważne wydaje się badanie, czy dwa obiekty graniczą ze sobą. Wprowadzone relacje topologiczne nie pozwalają rozróżnić obiektów dotykających się od obiektów leżących z dala od siebie. Wykonanie tego testu wymaga wywołania w zapytaniu dodatkowej funkcji (nazwanej tutaj *Granicza ()*). Dla danych zakresów kluczy badanych kwadratów musiałaby ona przejść na współrzędne XY dla wszystkich kwadratów elementarnych i sprawdzić, czy istnieją takie, których współrzędne różnią się o 1.

```
SELECT d1.NazwaOb, d2.NazwaOb
FROM DaneOb d1, DaneOb d2
WHERE d1.IdOb < d2.IdOb AND NOT EXISTS (
            SELECT *
            FROM AproksObP2 a1, AproksObP2 a2
            WHERE a1.IdOb = d1.IdOb AND a2.IdOb = d2.IdOb AND a1.KPpocz <= a2.KPkon
                    AND a1.KPkon >= a2.KPpocz)
    AND EXISTS (
            SELECT *
            FROM AproksObP2 a3, AproksObP2 a4
            WHERE a3.IdOb = d1.IdOb AND a4.IdOb = d2.IdOb AND
                    Granicza (a3.KPpocz, a3.KPkon, a4.KPpocz, a4.KPkon));
```

Pierwsze zapytanie zagnieżdżone sprawdza, czy obiekty są rozłączne, drugie, czy graniczą ze sobą. Funkcja *Granicza ()* w języku C++ może wyglądać następująco:

```
BOOL Granicza (int KPpocz1, int KPkon1, int KPpocz2, int KPkon2) {
        if (KPpocz1 <= KPkon2 && KPkon1 >= KPpocz2)
                return FALSE;           // nakładają się na siebie
        for (int viKP1 = KPpocz1; viKP1 <= KPkon1; viKP1++)
                for (int viKP2 = KPpocz2; viKP2 <= KPkon2; viKP2++)
                        if (abs (WspXzKP (viKP1) - WspXzKP (viKP2)) <= 1 &&
                          abs (WspYzKP (viKP1) - WspYzKP (viKP2)) <= 1
                                        return TRUE;
                return FALSE;
```

Założono, że obiekty są ośmiospójne [6, 11]. Funkcje *WspXzKP ()* i *WspYzKP ()* zwracają odpowiednio wartości *x* i *y* odpowiadające podanemu kluczowi Peano.

6. Rozszerzenie zapytań o wielkość obszaru

W zapytaniach dotyczących rozkładów przestrzennych zarówno wśród zwracanych wartości, jak i w warunkach zawartych we frazie *WHERE* powinna być możliwość zapisu wyrażeń dotyczących wielkości obszarów. Wielkość części wspólnej aproksymacji dwóch obiektów jest równa sumie części wspólnej nakładających się kwadratów tworzących te aproksymacje. Wielkość części wspólnej dwóch kwadratów, wyrażona w liczbie wspólnych kwadratów elementarnych, jest równa:

WlkCzWspKw (k1, k2) ≡ *Min (k1.KPkon, k2.KPkon) – Max (k1.KPpocz, k2.KPpocz) + 1*

W przypadku gdy kwadraty są rozłączne, wyrażenie przybiera wartość 0 lub wartość ujemną. Aby otrzymać wielkość obszaru w jednostkach powierzchni używanych w modelowanej rzeczywistości, należy otrzymaną liczbę kwadratów elementarnych przemnożyć przez obszar odpowiadający jednemu kwadratowi. W dalszej części dla uproszczenia przyjęto, że kwadratowi elementarnemu odpowiada pole o powierzchni 1 jednostki umownej, tak więc szukana wielkość obszaru będzie równa liczbie kwadratów elementarnych. Nie jest to żadnym ograniczeniem, ponieważ do zapytania można wprowadzić mnożnik odpowiadający wielkości kwadratu elementarnego.

Poniższe zapytanie zapisane w pseudo-SQL znajduje wielkość obszaru wspólnego parom nakładających się obiektów. Zwracane są identyfikatory nakładających się obiektów oraz wielkość części wspólnej wyrażona liczbą kwadratów elementarnych.

```
select k1.IdOb, k2.IdOb, sum (WlkCzWspKw (k1, k2)) as WielkObWsp
from AproksObP2 k1, AproksObP2 k2
where NaklKwKP (k1, k2)
group by k1.IdOb, k2.IdOb;
```

Warunek zawarty we frazie *WHERE* można wprost rozpisać na definicję predykatu. Więcej problemów sprawia wyznaczenie wielkości części wspólnej. W standardzie języka SQL w funkcji agregującej *SUM* nie ma możliwości wywołania funkcji *min* i *max* ani użycia instrukcji warunkowych. Większość implementacji języka SQL pozwala wywoływać podstawowe funkcje skalarne w zapytaniach, niektóre pozwalają wywołać funkcję zdefiniowaną przez użytkownika. Poniżej omówiono rozwiązanie problemu zapisu funkcji *WlkCzWspKw ()* oparte na funkcji charakterystycznej δ zaproponowanej w pracy [12]. Argumentem funkcji δ jest wyrażenie boolowskie. Funkcja δ zwraca wartość numeryczną 1, jeżeli wyrażenie jest spełnione, a wartość 0 w przeciwnym przypadku. Funkcję charakterystyczną można zakodo-

wać za pomocą standardowych operatorów i funkcji skalarnych na wiele różnych sposobów. Poniżej przedstawiono jej zapis przy użyciu operacji dodawania, odejmowania i mnożenia oraz funkcji *abs ()* – zwracającej wartość bezwzględną argumentu (2) i *sgn ()* – zwracającej znak argumentu (3).

$$abs\,(x) = \begin{cases} x, & gdy\ x >= 0, \\ -x, & gdy\ x < 0 \end{cases} \tag{2}$$

$$\operatorname{sgn}\,(x) = \begin{cases} -1, & gdy\ x < 0, \\ 0, & gdy\ x = 0, \\ 1, & gdy\ x > 0 \end{cases} \tag{3}$$

W poniższym zapisie symbole *x* i *y* oznaczają zmienne numeryczne, nie przyjmujące wartości *NULL*, a symbole *A* i *B* wyrażenia boolowskie, nie przyjmujące wartości *NULL*.

- δ (x = y) ≡ 1 – abs (sgn (x – y)
- δ (x <> y) ≡ abs (sgn (x – y)
- δ (x < y) ≡ 1 – sgn (1 + sgn (x – y))
- δ (x <= y) ≡ sgn (1 –sgn (x – y))
- δ (x > y) ≡ 1 – sgn (1 - sgn (x – y))
- δ (x >= y) ≡ sgn (1 + sgn (x – y))
- δ (NOT A) ≡ 1 – A
- δ (A AND B) ≡ A * B
- δ (A OR B) ≡ sgn (A + B)

Wybór tylko jednej wartości z kilku można uzyskać tworząc ważoną sumę tych wartości, gdzie wagami są funkcje charakterystyczne odpowiednich warunków. Wagi muszą być tak dobrane, aby dla dowolnej kombinacji argumentów tylko jedna z nich przybierała wartość 1, a reszta wartość 0. Funkcję znajdującą wielkość części wspólnej można zapisać wykorzystując funkcję charakterystyczną w następujący sposób:

```
WlkCzWspKw (k1, k1) ≡   k1.KPkon * δ (k1.KPkon <= k2.KPkon) +
                        k2.KPkon * δ (k1.KPkon > k2.KPkon) -
                        k1.KPpocz * δ (k1.KPpocz >= k2.KPpocz) -
                        k2.KPpocz * δ (k1.KPpocz < k2.KPpocz)
```

Po rozwinięciu funkcji charakterystycznej otrzymujemy:

```
WlkCzWspKw (k1, k1) ≡   k1.KPkon * SGN (1 – SGN (k1.KPkon – k2.KPkon)) +
                        k2.KPkon * (1- SGN (1 - SGN (k1.KPkon  - k2.KPkon))) -
                        k1.KPpocz * SGN (1 + SGN (k1.KPpocz – k2.KPpocz)) -
                        k2.KPpocz * (1 - SGN (1 + SGN (k1.KPpocz - k2.KPpocz))
```

W rezultacie zapytanie w języku SQL przybiera następującą postać:

```
SELECT k1.IdOb, k2.IdOb, SUM (
        k1.KPkon * SGN (1 – SGN (k1.KPkon – k2.KPkon)) +
        k2.KPkon * (1- SGN (1 - SGN (k1.KPkon  - k2.KPkon))) -
        k1.KPpocz * SGN (1 + SGN (k1.KPpocz – k2.KPpocz)) -
```

```
    k2.KPpocz * (1 - SGN (1 + SGN (k1.KPpocz - k2.KPpocz))) AS WielkObWsp
from AproksObP2 k1, AproksObP2 k2
where k1.KPpocz >= k2.KPpocz AND k1.KPkon <= k2.KPkon
group by k1.IdOb, k2.IdOb;
```

7. Podsumowanie

Jak pokazano w artykule, zakodowanie rozkładu za pomocą liniowego drzewa czwórkowego uporządkowanego krzywą N-Peano umożliwia przechowanie aproksymacji obiektów przestrzennych w relacyjnej bazie danych oraz wykorzystanie języka SQL do zadawania zapytań dotyczących przestrzennych relacji między tymi obiektami. Przedstawione zapytania należą do klasy zapytań wymagających wielokrotnego dostępu do danych i należy je traktować jako różne przypadki złączenia przestrzennego.

Wykorzystując standardowe konstrukcje języka SQL, można testować, czy obiekty są rozłączne, czy mają część wspólną, czy są równe lub czy jeden obiekt zawiera drugi. Główny problem zastosowania języka SQL polega na trudności formułowania zapytań zawierających warunki na brzeg obszaru. Zapytania dotyczące wielkości obszarów wymagają powszechnie dostępnego rozszerzenia dopuszczającego wywoływanie w zapytaniu funkcji skalarnych.

Niedogodnością zaprezentowanego podejścia jest skomplikowane zadawanie zapytań typu okno. Wymagają one wygenerowania kluczy Peano odpowiadających obszarowi okna lub wykorzystania w zapytaniu funkcji przekształcających kod Peano na współrzędne. Z drugiej jednak strony, podejście to umożliwia zadawanie zapytań z dowolnie skomplikowanym oknem, które może składać się z rozłącznych części i zawierać dziury.

Ze względu na możliwości i ograniczenia przedstawionego podejścia jest ono predystynowane do zastosowań, w których przy formułowaniu zapytań nie wprowadza się rozróżnienia między wnętrzem i brzegiem obiektu. Podkreślić należy, że przedstawione podejście pozwala badać relacje topologiczne pomiędzy obiektami zawierającymi dziury oraz składającymi się z wielu rozłącznych części.

Osobnym problemem jest efektywność zaproponowanego w artykule podejścia do przetwarzania zapytań do rzeczywistych danych. Badania eksperymentalne oparte na przedstawionych koncepcjach zostały przedstawione w pracy [1].

LITERATURA

1. Bajerski P.: Efektywność przetwarzania w języku SQL zapytań przestrzennych wykorzystujących aproksymacje obiektów. Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka zeszyt 37, Gliwice 1999.
2. Brinhhoff T., Kriegel H., Seeger B.: Efficient Processing of Spatial Joins Using R-trees. ACM SIGMOD, 1993.
3. Bruns H., Egenhofer M.: Similarity of spatial scenes. Advances in GIS Research, Taylor & Francis 1997.
4. Egenhofer M., Franzoza R.: Point-Set Topological Spatial Relations. International Journal of Geographical Information Systems 5(2) 1991.
5. Egenhofer M., Herring J.: A Mathematical Framework for the Definition of Topological Relationships. in: Brassel K., Kishimoto H. (Ed.) Fourth International Symposium on Spatial Data Handling, Zurich, Switerland 1990.
6. Jankowski M.: Elementy grafiki komputerowej. WNT, Warszawa 1990.
7. Laurini R., Francoise M.: Spatial database queries: relational algebra versus computational geometry. Proceedings of the Fourth International Conference on Statistical and Scientific Database Management, Rome, Italy 1988, M. Rafamelli et al.,(eds) Berlin;Germeny: Springer Verlag. pp. 291-313.
8. Laurini R., Thompson D.: Understanding GIS, Academic Press Limited, third printing 1994.
9. Oosterom P., Vijlbrief T.: The Spatial Location Code in Advances in GIS Research II, edited by Kraak M.J. and Molenaar M., Taylor&Francis Ltd. 1997.
10. Oracle7 Spatial Data Option™ Overiew: Oracle April 1996.
11. Pavlidis T.: Grafika i przetwarzanie obrazów. WNT, Warszawa 1987.
12. Rozenshtein D., Abramovich A., Birger E.: Speeding Up SQL Queries – Characteristically, Database Programming & Design vol.8 no.10 1995.
13. Sammet H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reding 1989.
14. Ullman J.: Systemy baz danych. WNT, Warszawa 1988.

Recenzent: Prof. dr hab. inż. Alicja Wakulicz-Deja

Wpłynęło do Redakcji 17 grudnia 1998 r.

Abstract

The paper presents a conception of using SQL for formulation of queries concerning spatial relations among objects approximated by quadtrees. Ordering the quadtree by a space-filling curve makes it possible to store approximations in a relational database [7, 8]. Such a tree is called linear quadtree. A space filling curve is a fractal curve completely filling a tiled area. Tiles are identified by a key showing their position on the curve. Under the assumption that the considered area is built of elementary homogenous quadrants, the N-Peano curve (fig. 1) was chosen because of simple mapping between coordinates and Peano key – the key is created by bit interleaving of co-ordinates. Two methods of storing spatial objects approximated by means of a quadtree ordered by N-Peano fractal curve in a relational database are described. Such an approach makes it possible to perform spatial queries without using computational geometry algorithms.

The article gives a study of expressing in SQL language queries concerning topological spatial relations between approximations of two objects. The model of topological spatial relation is based on nine intersections between interior (denoted by A° and B°), boundary (denoted by ∂A and ∂B), and exterior (denoted by A^{-} and B^{-}) of two simple 2D regions without holes (denoted by A and B). With each of these nine intersections being empty or non-empty the model has eight distinct topological relations: disjoint, meet, equal, overlap, inside, contains, covers, and coveredBy. The only problem of this approach is connected with conditions on region boundary – distinguishing between relations: overlap and meet; contains and covers; inside and coveredBy. The article shows how to include in a SQL query conditions on region size and area on which the given spatial relation holds.

The results of experimental research investigating performance of queries based on the proposed conception are given in [1]. Some methods of speeding up these queries are also given there.