STUDIA INFORMATICA 2011
Volume 32 Number 2B (97)

Adam DUSZEŃKO, Aleksandra WERNER
Politechnika Śląska, Instytut Informatyki

ORGANIZACJA DANYCH O CHARAKTERZE TOPOLOGICZNYM NA BAZIE MODELU RELACYJNEGO

Streszczenie. Podstawową abstrakcją organizacji danych w tradycyjnych systemach bazodanowych jest zbiór. Model relacyjny, bazując na zbiorze, organizuje dane bez uwzględnienia jakiegokolwiek ich uporządkowania. Generalnie przynosi to swobodę indeksacji danych, jednak powoduje, iż uporządkowanie wprowadzane indeksami jest uporządkowaniem wtórnym. W przypadku danych o charakterze topologicznym, brak takiego naturalnego uporządkowania powoduje istotne utrudnienia w poszukiwaniu punktów sąsiednich wg zadanych kryteriów sąsiedztwa. Niniejszy artykuł koncentruje się na przedstawieniu organizacji danych, opartej na modelu relacyjnym, pozwalającej bezpośrednio wyznaczać podzbiory punktów sąsiednich.

Słowa kluczowe: relacyjna baza danych, dane topologiczne

ORGANIZATION OF TOPOLOGICAL DATA, BASED ON RELATIONAL DATA MODEL

Summary. The relational model can be considered as a generalization of the set data model. So, the basic organization of a data in traditional – relational – database system is a set of data with no defined any positional orderings of them. At first glance, this approach is accurate, because it brings unlimited possibilities of data indexing. However such forced order of data organization is the secondary one and doesn't reflect inner topology of data in a set. In the case of topological data, it can cause significant difficulties in finding the neighboring points, according to defined neighborhood criteria. This chapter presents the organization of a topological data modeled in the relational database system that allows direct designating the subsets of neighboring points.

This article describes recursive SQL queries and presents its syntax with explanation of used clauses. Besides, the design of performance cluster by the use of DB2 Distributed Partitioning Feature is discussed

Keywords: database, relational model, topological data, recursive queries

1. Wstęp

U podstaw relacyjnego modelu danych, zaproponowanego w 1970 r. przez E. Codda, leży teoria zbiorów. W związku z tym, że z definicji zbiorów matematycznych wynika m.in. zasada, iż relacja powinna być uniezależniona od uporządkowania zbiorów, nad którymi jest ona rozważana, również i w relacyjnym modelu danych uporządkowanie elementów zbioru nie jest istotne. To powoduje, że są one przechowywane w porządku dowolnym, który nie musi odzwierciedlać ani kolejności ich wprowadzania, ani kolejności ich przechowywania. Oznacza to również, że – w odróżnieniu od struktur listowych, sekwencji czy drzew – w zbiorze danych, w ujęciu relacyjnym, nie ma zdefiniowanego porządku pomiędzy samymi elementami. Taki brak uporządkowania z jednej strony pozwala swobodniej zarządzać danymi (dodawanie, modyfikacje, usuwanie nie powodują konieczności wykonywania dodatkowych operacji, przywracających wprowadzone uporządkowanie), ale z drugiej strony często utrudnia wyszukiwanie elementów o zadanych właściwościach. Jest tak np. wtedy, gdy cechy, według których następuje wyszukiwanie, wymagają wykorzystania informacji o położeniu lub o relacjach przestrzennych, w jakich te obiekty pozostają – czyli ich topologii. Tym samym można stwierdzić, że jeżeli kryterium wyszukiwania oparte jest na pewnej relacji porządkującej, wyznaczającej kolejność elementów w zbiorze, to samo uporządkowanie pozwala szybciej zlokalizować poszukiwany obiekt. W relacyjnym modelu danych wprowadza się w tym celu dodatkowe struktury danych w postaci indeksów, które umożliwiają szybkie wyszukiwanie na podstawie wartości pozycji indeksu. Tak więc wprowadzenie indeksu jest niczym innym, jak wprowadzeniem na zadanych cechach elementów uporządkowania, co pozwala uniknąć powolnego, sekwencyjnego przeszukiwania całego zbioru. Dzięki takiemu uporządkowaniu można zarówno szybko wyszukiwać elementy, jak i odpowiadać na pytania typu, ile elementów ma tę samą wartość lub który element jest najbliższy w kontekście wprowadzonego uporządkowania i definicji odległości w zbiorze?

Wprowadzanie indeksów na całym zbiorze jest operacją ogólną, to znaczy wprowadza uporządkowanie ogólne dla wszystkich elementów tego zbioru.

Problem pojawia się jednak, gdy zadanie wyszukiwania sprowadza się do kryterium względnego, gdzie nie zawsze możliwe jest wykorzystanie odpowiedniego uporządkowania obiektywnego, ogólnego, nieuzależnionego od wartości punktu odniesienia.

2. Cechy ogólne i względne

Z matematycznego punktu widzenia topologia rozumiana jest jako rodzina wyróżnionych podzbiorów (tzw. zbiorów otwartych) danego zbioru X, zwanego przestrzenią [1, 2]. Takimi

zbiorami otwartymi mogą być nie tylko punkty przestrzeni, ale nawet np. funkcje czy krotki relacji. Dlatego często okazuje się, że poznanie struktury zbiorów otwartych jest równie użyteczne, co badanie przestrzeni za pomocą metryki (odległości).

W bieżącym rozdziale rozważania będą oparte na 2-wymiarowej przestrzeni euklidesowej. Posługiwanie się jednowymiarową przestrzenią, choć uprościłoby przykłady, mogłoby jednak nie w pełni oddać cechy proponowanego rozwiązania. Zaproponowane metody modelowania danych są jednak w pełni rozszerzalne na wyżej wymiarowe przestrzenie.

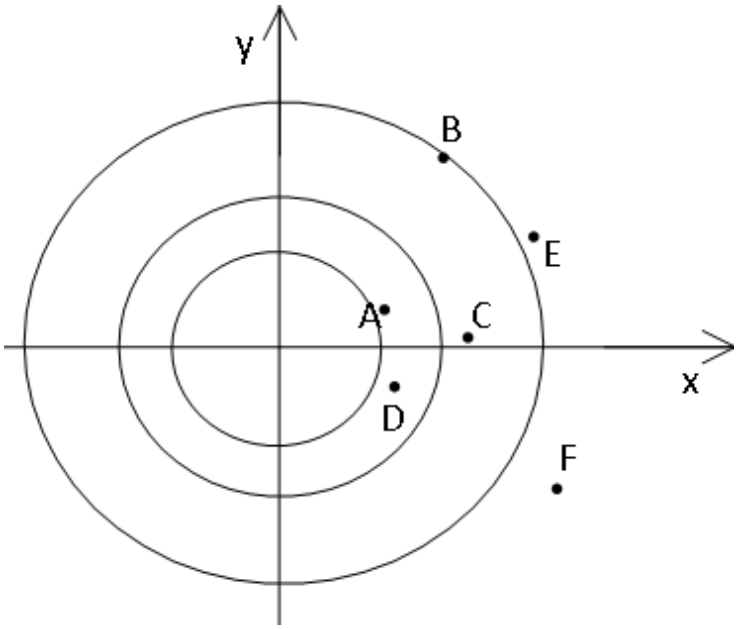
Wspomniane wcześniej ogólne uporządkowanie dla elementów przestrzeni euklidesowej może być realizowane na przykład przy uwzględnieniu ich odległości od początku układu współrzędnych. Jest to jedno z najczęściej realizowanych uporządkowań. W zależności od zastosowania, może być ono ograniczane do wybranych wymiarów (z pominięciem pozostałych – zwłaszcza w wielowymiarowych przestrzeniach).

Przykładowo, jeżeli punkty ułożone są jak na rys. 1, ich uporządkowanie, wynikające z odległości od początku układu współrzędnych, ma postać:

A, *D*, *C*, *B*, *E*, *F*.

Dla tego samego zbioru punktów, uporządkowanie tylko na podstawie wymiaru *X* daje kolejność:

A, *D*, *B*, *C*, *E*, *F*.



Rys. 1. Przykładowe rozmieszczenie punktów w przestrzeni
Fig. 1. Sample elements layout

Jak widać, zaprezentowany powyżej przykład uporządkowania ogólnego może zostać wykorzystany do wyszukiwania na podstawie ogólnej cechy, jaką jest lokalizacja punktu względem wspólnego punktu odniesienia. Takim punktem odniesienia może być na przykład początek układu współrzędnych lub określony punkt na wybranej osi.

Osobnym problemem jest organizacja procesu wyszukiwania na podstawie cech względnych, tj. zależnych od punktu odniesienia, który z założenia jest zmienny. Przykładem takiej cechy jest najbliższe sąsiedztwo (lub najbliższy sąsiad).

Każdy punkt przestrzeni euklidesowej posiada nieskończenie wiele otoczeń, z których niektóre zawierają się w innych. Z definicji otoczenia wynika, że jeżeli punkt *A* należy do otoczenia *U* punktu *B*, to istnieje takie otoczenie punktu *A*, które zawiera się w *U*. To zawieranie się otoczeń jest jedynym odpowiednikiem informacji o odległości danych punktów pomiędzy sobą [1].

Dla celów badawczych, w dalszej analizie problemu posłużono się pojęciem sąsiedztwa punktu, rozumianego jako otoczenie punktu z wyłączeniem jego samego.

Przyjmując standardową definicję odległości *d* pomiędzy dwoma punktami *A* i *B* w prostokątnym układzie współrzędnych:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \tag{1}$$

gdzie x_1, y_1 oraz x_2, y_2 oznaczają współrzędne punktu, odpowiednio *A* i *B*,

można wprowadzić uporządkowania względne, szeregujące punkty według ich sąsiedztwa. W odróżnieniu od jednego, ogólnego, uporządkowania zbioru punktów na płaszczyźnie, takich uporządkowań względnych może być wiele. W szczególnym przypadku może ich być tak dużo, jak duża jest moc (liczność) zbioru. Ta właśnie mnogość uporządkowań względnych stanowi główny problem w ich modelowaniu na poziomie relacyjnej bazy danych.

Odnosząc się do przykładu z rys. 1, względne uporządkowanie względem punktu *C* ma postać:

C, *D*, *A*, *E*, *B*, *F*,

a względem punktu D:

D, *A*, *C*, *E*, *B*, *F*.

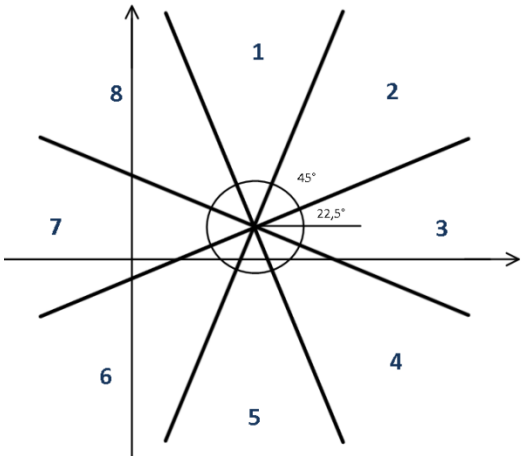
Cechy względne i związane z nimi uporządkowania są bardzo ważne przy układach samoorganizujących się, jak na przykład sieci neuronowe lub modele służące do symulacji zjawisk i procesów naturalnych. W tych zastosowaniach propagacja zmian i wzajemne oddziaływania realizowane są sukcesywnie, począwszy od punktu zainicjowania, według najbliższych sąsiadów. Wydajne realizowanie operacji w takim środowisku wymaga możliwości szybkiego wskazywania w bardzo dużym zbiorze obiektów, tych najbliższych zadanemu. Jak pokazano wcześniej, w relacyjnym modelu danych nie jest to możliwe z wykorzystaniem obiektywnych kryteriów porządkowania, czyli np. indeksów, a musi być realizowane na podstawie pewnych struktur pomocniczych, odzwierciedlających uporządkowania względne.

3. Model danych

Rejestrowanie i trwałe zapisywanie w modelu danych pełnych uporządkowań względnych wprowadzałoby bardzo duży narzut przestrzeni składowania. Dla uniknięcia tego można zastosować przechodniość pewnych cech, realizując odtwarzanie pełnego uporządkowania względnego przez złożenie kolejnych, częściowych uporządkowań względnych. Ograniczenie wielkości – a bardziej precyzyjnie zasięgu – uporządkowań względnych, pozwoli zasadniczo ograniczyć rozmiar struktur porządkujących z N^2 do wartości $N*k$, gdzie k jest małe, znacznie mniejsze od N, a N jest licznością zbioru elementów (punktów) przestrzeni.

Na potrzeby badań zostało zaproponowane ograniczenie względnego uporządkowania według względnej odległości punktów od ośmiu najbliższych sąsiadów w ośmiu arbitralnie narzuconych kierunkach. Liczba osiem została dobrana bez dodatkowych testów, a jedynie wg zasady uzyskania wystarczającej ekspresji. Weryfikacja słuszności tej wartości lub wskazanie lepszej w kontekście konkretnego porządkowania i wyszukiwania oraz, być może, charakteru zbioru danych, stanowi przedmiot osobnych badań.

Zgodnie z tym założeniem, przestrzeń wokół każdego punktu, względem którego wyznaczane jest względne uporządkowanie, dzielona jest na 8 stref, tak jak to zaprezentowano na rys. 2.



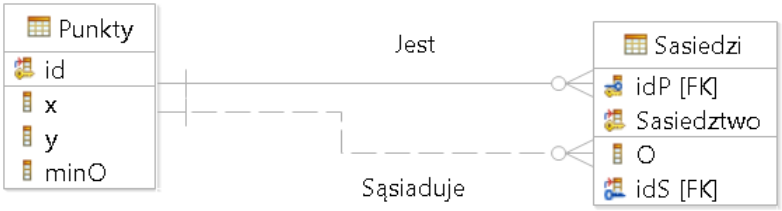
Rys. 2. Ośmiosąsiedztwo pojedynczego punktu
Fig. 2. Eight-neighbouring of a single point

W każdym z ośmiu obszarów wyszukiwany jest jeden najbliższy punkt. W ten sposób sąsiedztwo każdego punktu reprezentowane jest przez 8 punktów najbliższych w danym kierunku. Kolejne punkty sąsiednie mogą być wyznaczane na zasadzie przechodniości sąsiedztwa, czyli według zasady: sąsiedzi sąsiada są kolejnymi sąsiadami. Ta zasada posłużyła w dalszej części prac do budowy zapytań wyszukujących kolejne poziomy sąsiedztwa.

Tak zdefiniowane ośmiosąsiedztwo można zamodelować w relacyjnym modelu danych za pomocą jednej relacji, reprezentującej punkty na płaszczyźnie powiązanej z samą sobą

związkiem wiele-do-wielu, wskazującym na punkty sąsiednie. Krotność związku wynika z faktu, iż pojedynczy punkt może mieć wielu sąsiadów oraz może być sąsiadem wielu innych punktów.

Transformacja modelu logicznego danych do modelu fizycznego prowadzi do powstania dwóch tabel reprezentujących punkty oraz związek sąsiedztwa. Związek sąsiedztwa realizowany jest tabelą *Sasiedzi*, powiązaną dwiema relacjami z tabelą *Punkty* – relacjmi *Jest* oraz *Sasiedzi* – tak jak to pokazano na rys. 3. Tak więc każdy wpis w tabeli *Sasiedzi* dotyczy pewnego punktu wskazanego związkiem *Jest* i określa jego punkt sąsiedni, który wskazany jest z kolei relacją *Sąsiaduje*.



Rys. 3. Fizyczny model danych
Fig. 3. Physical Data Model

Znaczenie poszczególnych atrybutów jest następujące:

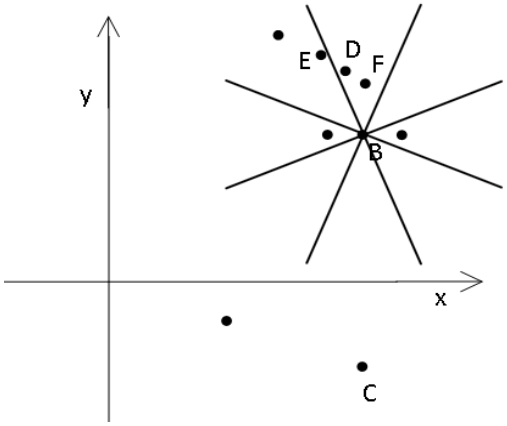
`Id` – identyfikator punktu,
`x` – współrzędna *X* punktu,
`y` – współrzędna *Y* punktu,
`minO` – minimalna odległość punktu od jednego z ośmiu sąsiadów,
`idP` – identyfikator punktu,
`idS` – identyfikator sąsiada,
`O` – odległość od punktu `idP` do `idS`,
`Sasiedztwo` – numerycznie zapisany identyfikator sąsiedztwa o wartościach ze zbioru {1, 2,…,8}.

4. Wyszukiwanie sąsiadów

Przykładem takiego zadania może być wyszukiwanie punktów położonych najbliżej zadanego punktu na płaszczyźnie. Jeżeli zostanie przyjęty układ kartezjański i każdy punkt będzie opisany przez dwie współrzędne *x* i *y*, to realizacja tego wyszukiwania sprowadzi się do weryfikacji względnej cechy, jaką jest odległość danego punktu od pozostałych. Bazując na opisanym wyżej zapisie, możliwe jest uzyskanie bezpośredniej informacji o punktach znajdujących się najbliżej, bez przeszukiwania całego zbioru i wyliczania odległości między nimi. Proces ten można rekurencyjnie kontynuować, uzyskując informacje o kolejnych punktach najbliższych punktowi wyjściowemu.

Jeżeli poszukiwane są punkty najbliższe punktowi znajdującemu się już w zbiorze, postępowanie jest zgodne z opisem podanym w poprzednim akapicie. Natomiast, jeżeli zadany punkt nie jest obecny w zbiorze, możliwe jest albo jego dodanie do opisu zbioru, albo odnalezienie jego punktu najbliższego i rozpoczęcie poszukiwań od właśnie znalezionego punktu.

Sposób konstrukcji opisu sąsiedztwa sprawia, że na każdym poziomie rekurencji zostanie uzyskanych najwyżej k*8 punktów, gdzie k to wartość oznaczająca poziom rekurencji (wartość początkowa: k=1).



Rys. 4. Najbliższe ośmiosąsiedztwo punktu *B*
Fig. 4. The nearest eight-neighbouring of point *B*

Poszukując punktów najbliższych, nie można jednak bazować tylko na przechodniości najbliższego sąsiedztwa w zadanych kierunkach, gdyż punkt najbliższy w jednym ze wskazanych ośmiu kierunków może być znacznie dalszy niż punkty w obrębie innego kierunku sąsiedztwa. Obrazuje to rys. 4, na którym punkt *C* znajduje się w znacząco większej odległości od punktu *B* niż np. punkty *E* czy *F*.

Dla przypadku zilustrowanego na rys. 4, w ramach bezpośredniego sąsiedztwa punktu *B* w obszarze 1 i 8 (por. rys. 2), zapisane zostaną punkty *F* i *E*, gdyż w ramach odpowiednich kierunków są najbliższymi sąsiadami. Tak więc np. punkt *D* nie zostanie uwzględniony w sąsiedztwie 1 punktu *B*, ale dopiero w drugim kroku rekurencji, jako sąsiad punktu *F* lub *E* (lub obu jednocześnie). Oparcie rekurencji tylko na samym sąsiedztwie punktu *B* spowodowałoby wysnucie błędnego wniosku, że nie ma punktów bliższych punktowi *B*, jak *F* i *E*. Tymczasem, aby uzyskać pełną informację o najbliższych punktach, należy kontynuować rekurencyjne wyznaczanie sąsiedztwa.

W przypadku tego rodzaju wyszukiwania, kryterium zatrzymania wyszukiwania rekurencyjnego jest wykrycie sytuacji, iż kolejny krok rekurencji nie wprowadził żadnych nowych bliższych punktów od już znanych.

5. Implementacja

Implementacja powyższego modelu fizycznego zrealizowana została na platformach Systemu Zarządzającego Bazą Danych IBM DB2 Oracle 11g oraz PostgreSQL 9.0. Wybór tych właśnie systemów podyktowany był między innymi faktem, iż posiadają one dobrze zdefiniowane struktury tabel tymczasowych, które były planowane do wykorzystania przy pozyskiwaniu kolejnych poziomów sąsiedztwa, wynikających z przyjętej przechodniości sąsiedztwa.

Wspomniana przechodniość zastosowanego sąsiedztwa powoduje, iż proces odnajdywania kolejnych sąsiadów ma charakter rekurencyjny. Pierwszym krokiem rekurencji inicjującym ją, jest wyszukanie wszystkich sąsiadów punktu znajdującego się najbliżej zadanego miejsca. Do tego zadania wystarczy wykorzystanie tabeli *Punkty*. Może do tego posłużyć zapytanie agregacyjne w postaci:

```
SELECT min((X-p.x)*(X-p.x)+(Y-p.y)*(Y-p.y))
FROM PUNKTY as p;
```

wyznaczające najmniejszy kwadrat odległości istniejący w zbiorze punktów o zadanej lokalizacji (X,Y). Posiadając tę informację, można znaleźć odpowiedni punkt, ponownie wyszukując punkty odległe od zadanej lokalizacji o wyznaczoną odległość:

```
SELECT id
FROM PUNKTY AS p
WHERE ((X-p.x)*(X-p.x)+(Y-p.y)*(Y-p.y)) =
  (SELECT MIN((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y))
   FROM PUNKTY AS p);
```

Powyższe zapytanie nie czerpie korzyści z zastosowanego sposobu opisu punktów, a zastosowanie w nim funkcji agregujących, przetwarzających cały zbiór punktów, wiąże się z kosztem liniowo zależnym od wielkości zbioru. Podstawową możliwością ograniczenia tego kosztu jest zastosowanie klasteryzacji przestrzeni ze stałym oknem lub – w przypadku zbiorów bardzo rzadkich, w których elementy są rozproszone na całej płaszczyźnie i nie tworzą skupisk – wielowymiarowej klasteryzacji zorganizowanej w strukturze drzewiastej.

Ze względu na to, iż inicjowanie rekurencji odbywa się jednokrotnie, nie były na chwilę obecną podejmowane kroki mające na celu przyspieszenie tej części wyszukiwania.

W wyniku inicjalizacji uzyskuje się identyfikator jednego lub wielu punktów, znajdujących się najbliżej wskazanej lokalizacji, a w kolejnych krokach rekurencji wyszukiwane są ich punkty sąsiednie. W tym procesie, w pełni wykorzystywane są opisane związki sąsiedztwa, przechowywane w tabeli *Sąsiedzi*. Identyfikatory punktów odnalezionych w procesie inicjalizacji rekurencji, przechowywane są tabeli *temp*. Krok rekurencji ma w tym momencie następującą postać:

```
SELECT s.ids
FROM PUNKTY AS p , temp AS t, SASIEDZI AS s
WHERE t.id=s.idp AND
      p.id = s.ids;
```

Wyznaczone w każdym kroku identyfikatory kolejnych sąsiednich punktów wprowadzane są do tabeli *temp*.

Wyszukanie wszystkich sąsiadów lokalizacji o współrzędnych $(X,Y) = (52,52)$, ograniczone do 3 poziomów rekurencji, ma w systemach DB2 i PostgreSQL następującą postać [6, 4]:

```
WITH temp (id,x,y,l,distance) AS
(
SELECT id,x,y,1, ((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y))
FROM PUNKTY as p
WHERE ((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y)) =
      (SELECT min((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y))
       FROM PUNKTY AS p)

       UNION ALL

       SELECT s.ids,p.x,p.y,t.l+1,((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y))
       FROM PUNKTY AS p , temp AS t, SASIEDZI AS s
       WHERE t.id=s.idp AND
          p.id = s.ids AND
          t.l<4
)

SELECT UNIQUE id,x,y,distance FROM temp ORDER BY distance;
```

Jeżeli chodzi o system Oracle, to w celu umożliwienia zadawania zapytań o charakterze hierarchicznym, wykorzystano w zapytaniu `SELECT` niestandardową konstrukcję w postaci:

```
START WITH <warunek_1>,
CONNECT BY <warunek_2>,
```

gdzie `<warunek_1>` specyfikuje korzeń hierarchii (tu: punkt, od którego należy zacząć poszukiwanie sąsiadów), natomiast `<warunek_2>` pozwala zbudować hierarchię powiązań pomiędzy wierszem rodzica (nadrzędnym) a jego potomkami [5, 3]. Dla omawianego przykładu jest to powiązanie pomiędzy punktem a najbliższym sąsiadem w każdym z ośmiu kierunków.

Aby znaleźć najbliższych sąsiadów danego punktu, Oracle przetwarza warunek `s.ids = s.idp`, określając wartości `ids` dla poszczególnych wierszy-rodziców i wartości `idp` dla wierszy ich dzieci. Wiersze (punkty), dla których warunek jest spełniony, są szukanymi węzłami potomnymi (sąsiadami). Dodatkowo klauzula `CONNECT BY` zawiera również inne warunki, służące do późniejszej filtracji wyselekcjonowanych już danych – w szczególności warunek `level<4`, kontrolujący poziom (głębokość) rekurencji.

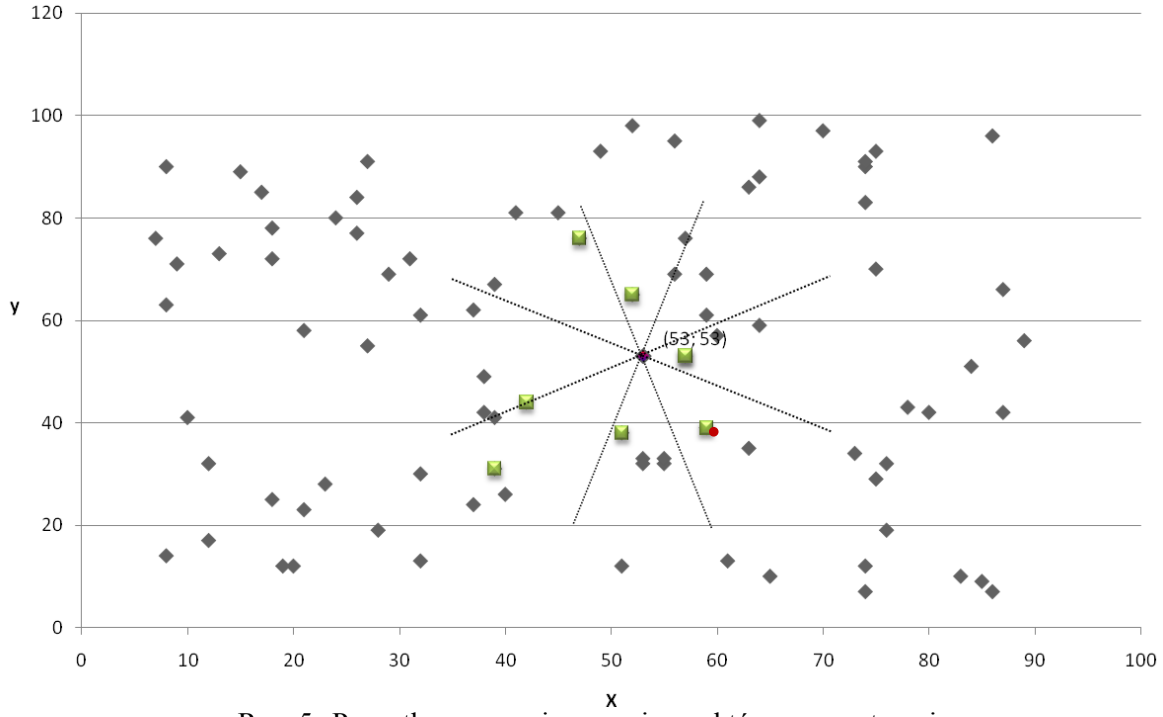
Ostatecznie zapytanie wyświetlające rekurencyjnie wszystkich sąsiadów danego punktu, zostało sformułowane w systemie Oracle następująco:

```
SELECT level, p.id, p1.id, s.idp, s.ids, s.o, s.sasiedztwo
FROM PUNKTY p, SASIEDZI s, PUNKTY p1
```

```
WHERE s.idp=p.id and s.ids = p1.id
START WITH ((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y)) =
           (SELECT MIN((52-p.x)*(52-p.x)+(52-p.y)*(52-p.y))
           FROM PUNKTY p)
CONNECT BY PRIOR s.ids = s.idp and p.id=s.idp and level<4
ORDER SIBLINGS BY p.id;
```

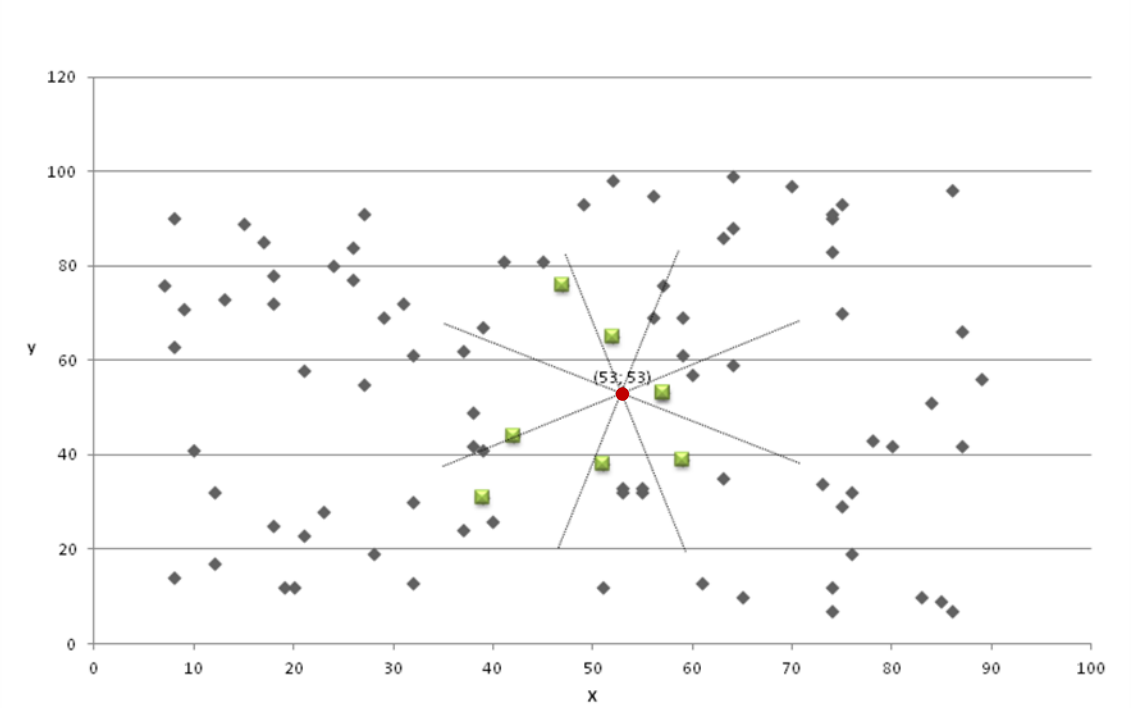
Unarny operator `PRIOR` określa w tym zapytaniu kierunek przechodzenia po drzewie. W związku z tym, że występuje on przed atrybutem `ids`, po odwiedzeniu rekordu z pewną wartością `ids` odnajdywane są rekordy z odpowiadającą im wartością `idp` (kierunek „od dołu po drzewie").

Na rys. 5 przedstawiono jeden ze zbiorów punktów (tu: 100 punktów) analizowanych w pracy, z wyróżnionym punktem o współrzędnych (53, 53), od którego rozpoczynała się procedura rekurencyjnego poszukiwania sąsiadów.



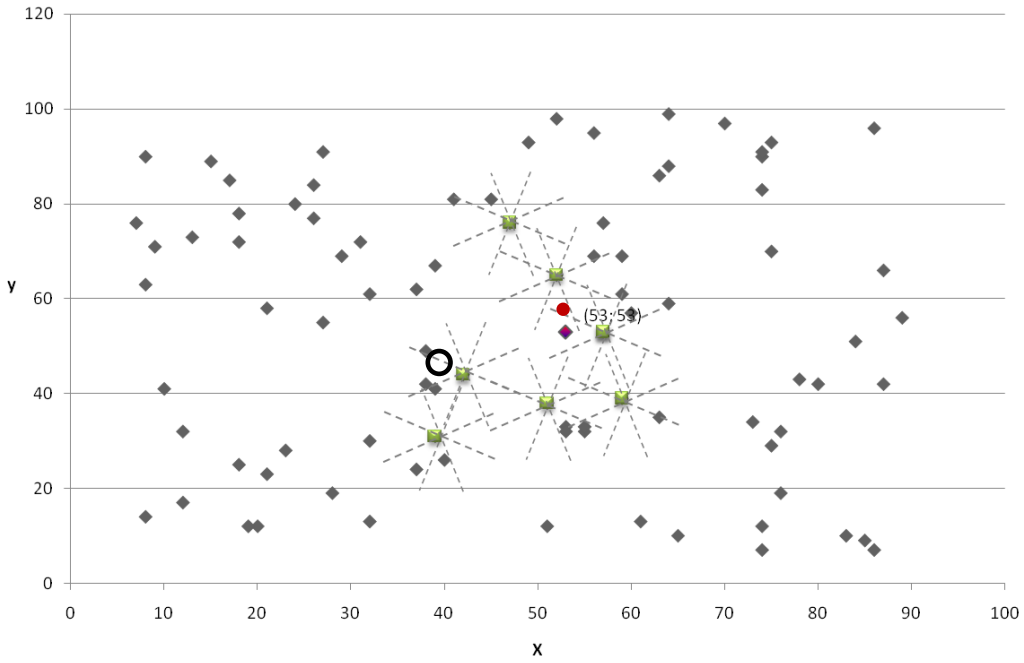
Rys. 5. Początkowe rozmieszczenie punktów w przestrzeni
Fig. 5. Initial distribution of topological points

Po pierwszym poziomie rekurencji (wymuszanym warunkiem `CONNECT BY…and level=1`), zostało znalezionych 8 punktów, które zaznaczono kwadratowymi symbolami na rys. 6.



Rys. 6. 8 najbliższych sąsiadów punktu startowego (1 poziom rekurencji)
Fig. 6. The 8 nearest neighbors of the starting point (1st level of recursion)

Natomiast włączenie drugiego poziomu rekurencji spowodowało wyszukanie kolejnych 8 najbliższych sąsiadów, według idei zilustrowanej na rys. 7.



Rys. 7. Wyszukiwanie najbliżej sąsiadujących punktów (2 poziom rekurencji)
Fig. 7. Searching for the closest neighboring points (2nd level of recursion)

Na wszystkich rysunkach (rys. 5 – rys. 7), symbolem wypełnionego kolorem kółka wyróżniono punkt startowy.

Przy zastosowaniu struktur dodatkowych, reprezentujących dane wykorzystywane w wyszukiwaniu, bardzo istotnym elementem jest koszt ich utrzymania. Już w przypadku standardowych struktur indeksowych, typowych dla baz danych, koszt utrzymania indeksu jest jednym z kryteriów jego przydatności.

W przypadku przedstawionego rozwiązania, typowe operacje usunięcia lub dodania nowego punktu do zbioru (zmianę traktuje się jako usunięcie starego i dodanie nowego elementu) obarczone są stałym kosztem, niezależnym od wielkości zbioru.

6. Wyniki

Wyszukanie najbliższych sąsiadów bez zastosowania dedykowanych struktur wymaga generalnie przejrzenia całego zbioru i dla każdego punktu wyliczenia odległości od zadanej lokalizacji i na tej podstawie określenia, czy punkt powinien należeć do zbioru wynikowego czy nie. W tej sytuacji koszt całego wyszukiwania liniowo zależy od wielkości zbioru.

Przy wykorzystaniu opisanej wyżej struktury opisu punktów w optymistycznym wariancie wyznaczenie N najbliższych sąsiadów, wymaga odczytania N rekordów z tabeli *Punkty* i N rekordów z tabeli *Sąsiedzi*. Odpowiednio organizując dane do inicjalizacji rekurencji, można zminimalizować jej koszt [7].

W celu znalezienia możliwości optymalizacji wcześniej zdefiniowanego zadania wyszukiwania pod względem czasowym, analizie poddano też sposób zapisu samego zapytania rekurencyjnego.

Na przykład, w celu określenia szacunkowego narzutu czasowego, który wiąże się z inicjalizacją rekurencji (tj. ze znalezieniem punktu startowego, dla którego są następnie wyszukiwani najbliżsi sąsiedzi), wykonano wiele zapytań, operujących na zbiorach punktów o zmiennej liczności. Uzyskane wyniki wskazały, że znajdowanie punktu startowego z wykorzystaniem wyrażenia agregacyjnego (fraza `START WITH` systemu Oracle i podzapytanie wyrażenia CTE systemu DB2), powodowało zwiększenie czasu wykonania zapytania rekurencyjnego o rząd wielkości. Przykładowo, w systemie Oracle całkowity czas wykonania zapytania z wykorzystaniem funkcji agregującej wynosił 59 – 64 ms dla zbioru 10000 punktów rozsianych na płaszczyźnie. Eliminacja funkcji agregującej z zapytania, przez wcześniejsze, jednorazowe wyznaczenie najmniejszego kwadratu odległości istniejącego w danym zbiorze punktów i porównanie odległości wszystkich sąsiadów punktu z odczytaną wartością, powodowało spadek czasu wykonania zapytania już do 4 – 6 ms.

Uzyskiwane wyniki wykonania zapytań wykazywały się dużym podobieństwem we wszystkich testowanych systemach (dla podanych wcześniej warunków w systemie PostgreSQL, czas wykonania zapytania początkowo mieścił się w granicach: 48 – 70 ms, aby, po eliminacji funkcji agregującej, spaść do poziomu 12 – 18 ms). Oznacza to, że wybór systemu zarządzania bazą danych jest nieznaczący i pozostaje właściwie bez wpływu na efektywność przetwarzania struktur hierarchicznych zamodelowanych do postaci relacyjnej.

Zwrócono również uwagę na pewne słabości proponowanego rozwiązania, będące bezpośrednio wynikiem zastosowania rekurencji do znajdowania najbliższego sąsiada. Otóż w trakcie znajdowania kolejnych sąsiadów zdarza się, że dany punkt jest wielokrotnie wskazywany jako najbliżej położony względem konkretnej lokalizacji. Można to zaobserwować na rys. 7, na którym taki właśnie punkt otoczono linią ciągłą (znajduje się w sąsiedztwie 1 i 4 dwóch różnych punktów). Oznacza to, że w opisywanym algorytmie rekurencyjnym występuje pewna, niepożądana w tym przypadku nadmiarowość, którą należy wyeliminować bądź ograniczyć.

Kontynuacją przedstawionych analiz będzie stworzenie pakietu rozszerzającego standardową funkcjonalność SZBD o zarządzanie dedykowane przestrzeniom *N*-wymiarowym. Podstawową funkcjonalnością rozszerzenia będzie samodzielne, niewidoczne dla użytkownika, tworzenie struktur, przechowujących sąsiedztwo i wykorzystujących je przy operacjach wyszukiwania.

BIBLIOGRAFIA

1. Frej B.: Topologia – wykład 4. Instytut Matematyki i Informatyki Politechniki Wrocławskiej, http://www.im.pwr.wroc.pl/~frej/top_4_wyklad.pdf.
2. Izdebski W.: Wykłady z przedmiotu SIT (Systemy Informacji o Terenie), http://www.izdebski.edu.pl/index.php?akcja=pokaz_kat&kat=18.
3. Dokumentacja techniczna Oracle 11g: Oracle Database SQL Language Reference, http://download.oracle.com/docs/cd/E17116_01/doc/server.112/e10881/chapter1.htm.
4. Jellema L.: Oracle RDBMS 11gR2 – goodbye Connect By or: the end of hierarchical querying as we know it, http://technology.amis.nl/blog/6104/oracle-rdbms-11gr2-goodbye-connect-by-or-the-end-of-hierarchical-querying-as-we-know-it.
5. IT Tips. Recursive subquery factoring using the SQL WITH clause. Burleson Enterprises, http://www.dba-oracle.com/t_recursive_subquery_factoring_with_clause.htm.
6. Centrum Informacyjne IBM DB2, https://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp.

7. Subieta K.: Tranzytywne domknięcia i równania stałopunktowe. Konstrukcja systemów obiektowych i rozproszonych, www.ipipan.eu/staff/k.subieta/SBA_SBQL/lectures-/SBA16.

Recenzent: Dr inż. Adam Piórkowski

Wpłynęło do Redakcji 31 stycznia 2011 r.

Abstract

The chapter concentrates on the important aspect that refers to relational data organization. At first it focused on the topological data with a formal discussion of some of its fundamental properties. It is pointed out, that the absence of a data positional orderings in a relational data model, may cause difficulties in search process. Especially, in finding neighbouring points of a specified point in a set. In order to solve this problem, the alternative approach was proposed, where the needed objects (points) can be localized only based on their relative position (location). This spatial topology of data resulted in making assumptions about the number of directions, in which neighbours will be looked for (Fig. 2). In Figures: 3 and 4, logical and physical data model for eight-neighbouring two-dimensional points is presented and the meaning of the tables attributes is explained also. Besides, the appropriate recursive queries for sample Oracle, DB2 and PostgreSQL implementations are formulated in a 5 point of a chapter and the possibilities of quicker access to topological data are discussed. The chapter is supplemented by conclusions.

Adresy

Adam DUSZEŃKO: Politechnika Śląska w Gliwicach, Akademicka 16, 44-100 Gliwice, Polska, adam.duszenko@polsl.pl.
Aleksandra WERNER: Politechnika Śląska w Gliwicach, Akademicka 16, 44-100 Gliwice, Polska, aleksandra.werner@polsl.pl.