Paweł DRZYMAŁA, Henryk WELFLE

Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych

doi:10.15199/48.2017.06.09

Przetwarzanie dużych zbiorów danych XML z użyciem struktur relacyjno-hierarchicznych w systemie IBM DB2

Streszczenie. *W pracy zaprezentowano natywne przechowywanie oraz przetwarzanie danych XML z użyciem struktur relacyjno-hierarchicznych, w oparciu o technologię pureXML dostarczaną z systemem IBM DB2. Artykuł ten ma na celu przedstawienie mechanizmów składowania danych XML i metod ich pozyskiwania z wykorzystywaniem serwera baz danych DB2 przy pracy z dużymi wolumenami danych XML. Zaprezentowana została efektywna metoda przetwarzania danych XML składowanych po stronie bazy danych w postaci natywnej z pominięciem operacji ich dekompozycji na fragmenty, możliwe do składowania w postaci relacyjnej (czyli pominięcie operacji „cięcia", tzw. „shredding"-u). Omówiono również możliwości dynamicznego sprzęgania danych relacyjnych z danymi składowanymi w formacji XML, tak by na wyjściu zapytania uzyskać rezultaty w postaci struktur „płaskich" lub hierarchicznych.*

Abstract*. The paper presents the native storage and processing XML data using relational-hierarchical structures, based on the pureXML technology provided by IBM DB2 server. The article shows the mechanisms for storing and methods of obtaining large volumes of XML data using DB2 database server. It presents an analysis of productivity growth during processing XML data stored on the database side in native form without the operation of decomposition into fragments. It also discusses the possibility of a dynamic coupling of relational data with the stored data in XML structures.(****Processing large volumes of XML data using relational-hierarchical structures in IBM DB2****).*

Słowa kluczowe: DB2 pureXML, IBM DB2, bazy relacyjno-hierarchiczne.
Keywords: DB2 pureXML, IBM DB2, relational-hierarchical structures.

Wprowadzenie

Pliki XML oprócz danych przenoszą również metadane, które opisują znaczenie samych danych. Z tych powodów są bardzo chętnie stosowanym formatem przeznaczonym do wymiany informacji. Duże wolumeny dokumentów XML coraz częściej przechowywane są po stronie baz danych. Sposób ten wydaje się atrakcyjny z powodów wydajnościowych oraz samej nadmiarowości formatu XML jako struktury hierarchicznej. Nadmiarowość (redundancja) hierarchii węzłów XML skutkuje wzrostem składowanego wolumenu oraz skutkuje obniżeniem efektywności przetwarzania pliku przechowywanego poza bazą danych.

W artykule podano wytyczne dotyczące efektywnego składowania danych w formacie XML. Natywne przechowywanie oraz przetwarzanie danych XML z użyciem struktur relacyjno-hierarchicznych, omówiono na przykładzie technologii pureXML, która dostarczana jest w środowiskach systemu IBM DB2 od wersji 9. Należy również zaznaczyć, że po stronie serwera DB2 składować można tylko dokumenty typu „well-formed XML".

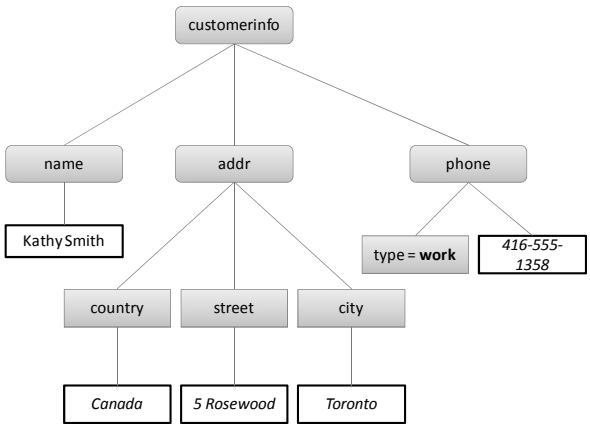
Składowanie danych XML w strukturach relacyjno-hierarchicznych

Natywne składowanie danych XML w strukturach relacyjno-hierarchicznych pozwala na uzyskanie wzrostu wydajności przetwarzania informacji po stronie bazy danych. Składowanie takie możliwe jest z pominięciem operacji dekompozycji XML na fragmenty, które możliwe są do przechowywania w postaci relacyjnej. Eliminacja kodu mapującego i mało elastycznej metody dekompozycji, która z punktu widzenia przetwarzania, polega na stosowaniu operacji „cięcia" (tzw. „shredding"-u) dokumentu XML przynosi wymierne korzyści w postaci bardziej uniwersalnego modelu składowania dokumentów XML. Istotne jest to zwłaszcza w przypadku, w którym nie jest możliwa do zastosowania ściśle zdefiniowana hierarchia węzłów elementów dokumentu XML.

Do innych powodów przechowywania plików XML w bazach danych zaliczyć można:

- zagwarantowane wsparcie dla trwałości danych w obrębie przeprowadzanych transakcji przy współbieżnym ich przetwarzaniu,
- zagwarantowana możliwość skalowania i wzrostu wydajności w środowiskach rozproszonych w oparciu o klastry i centra danych,
- wsparcie mechanizmów tworzenia kopii zapasowych i odzyskiwania danych nie tylko w oparciu o strategie backupów lecz również z zastosowaniem środowiska serwerów zapasowych (np. HADR),
- możliwości wydajnego ich przechowywanie poza formatem binarnym zapisu (tzw. BLOB, CLOB) i odczytywania.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- comment -->
3  <customerinfo>
4      <name>Kathy Smith</name>
5      <addr>
6          <country>Canada</country>
7          <street>5 Rosewood</street>
8          <city>Toronto</city>
9      </addr>
10     <phone type="work">416-555-1358</phone>
11 </customerinfo>
```



Rys.1. Dokument XML w środowisku ECLIPSE (IBM DataStudio) oraz w postaci drzewa węzłów.

Na uwagę zasługuje tu również spójny model przechowywania poprzez przechowywanie zależnych od siebie dokumentów XML. Pozwala to na proste operacje łączenia

nowych danych XML do już istniejących oraz możliwość generacji nowych dokumentów XML z zastosowaniem dialektu zapytania SQL z typowym dla hierarchii przetwarzania w oparciu o technologię XPath i XQuery.

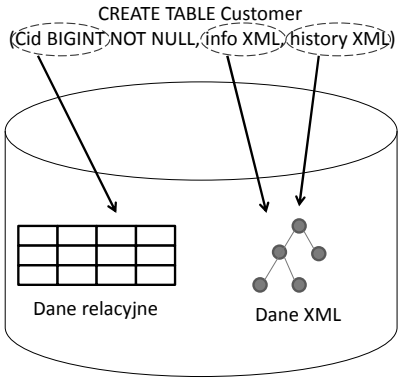
Nie bez znaczenia jest fakt, że dane relacyjne mogą być publikowane jako XML i vice versa. Dzięki integracji z bazami danych pozwala to na wsparcie aplikacji WEB i usług sieciowych (WEB Services).

Fizyczny sposób przechowywania dokumentów XML

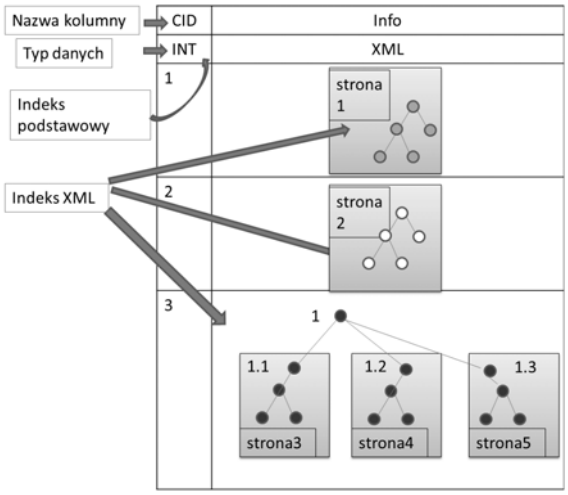
W DB2 począwszy od wersji 9 zaproponowano hierarchiczny sposób składowania dokumentów XML (nazwa technologii: pureXML). W momencie wstawienia węzły dokumentu XML są umieszczane w hierarchicznych strukturach. Fizyczny sposób przechowywania dokumentów XML opowiada modelowi XML. Takie podejście pozwala na szybką nawigację po odpowiednich elementach dokumentu XML bez konieczności parsowania całego dokumentu w momencie wykonywania zapytania. Tworzenie bazy danych do przechowywania danych XML w DB2 nie różni się od tworzenia bazy danych dla danych relacyjnych. Oba typy danych mogą być przechowywane i udostępniane jednocześnie w tej samej bazie danych. Przykład instrukcji która tworzy tabelę przeznaczoną do składowania danych hierarchicznych przedstawiony został na rysunku (Rys.2). Dane XML są zazwyczaj składowane jako Unicode chociaż od wersji serwera DB2 9.5 i 9.7 można wykorzystywać w bazach danych także ze stroną kodową.

Hierarchiczny silnik daje bardzo dobrą wydajność przy zachowaniu elastyczności obsługi. Po wstawieniu dokumentu XML do tabeli z kolumną typu XML (Rys. 3), dane są automatyczne dekomponowane do struktur hierarchicznych oraz tworzony jest słownik znaczników, który wykorzystywany jest do identyfikacji węzłów dokumentu XML. W poszczególnych węzłach pamiętane są wskaźniki do wartości ze słownika, a nie całe znaczniki. Organizacja danych w oparciu o słownik znaczników pozwala zredukować rozmiar składowanych danych wynikający z nadmiarowej natury samego formatu XML. Należy pamiętać, że dokument XML w postaci hierarchii węzłów przechowywany jest na stronach, zaś sama strona jest najmniejszą jednostką przechowującą w systemie DB2. Można przechowywać na serwerze DB2 v10.5 dokumenty XML do 2 gigabajtów wielkości każdy.. Dopuszczalne rozmiary stron to: 4K, 8K, 16K oraz 32K. Z tego powodu bardzo często istnieje konieczność posługiwania się zakresem stron (regionem stron) i w takim przypadku dokumenty, których rozmiar przekracza rozmiar zdefiniowanej strony są dzielone na regiony i przechowywane na wielu stronach. Jak podaje dokumentacja serwera DB2 10.5 ilość miejsca, które zajmuje dokument XML w bazie danych DB2 na danej stronie jest zależna od: początkowej wielkości dokumentu i szeregu innych czynników: struktury dokumentu, strony kodowej dokumentu, nazw znaczników węzłów, stosunku liczby atrybutów do liczby elementów itp.

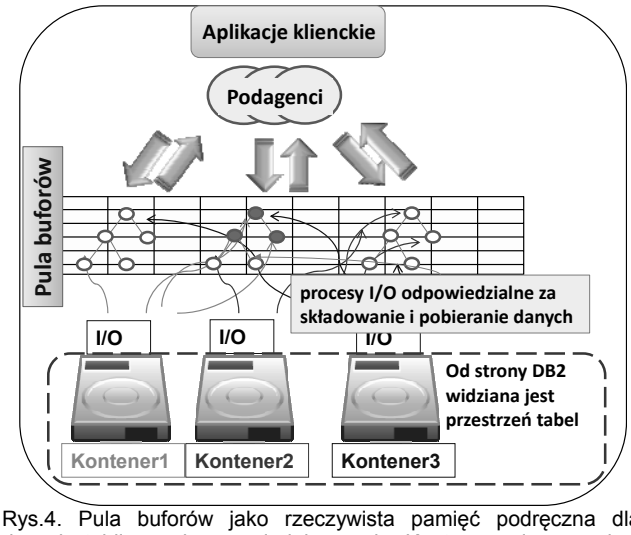
W przypadku konieczności przechowywania dokumentu XML na wielu stronach poszczególne gałęzie tego dokumentu są identyfikowane przez identyfikator węzła. Rysunek (Rys.3) przedstawia zasady przechowywanych poszczególnych gałęzi drzewa dokumentu XML na trzech stronach z użyciem trzech regionów (zakresów). Dostęp do danych XML umieszczonych w tym samym regionie jest szybszy niż między regionami i wynika ze specyfiki obsługi danych z użyciem puli buforów, która musi być dopasowana do rozmiaru strony przestrzeni tablicowej. Dodatkowo indeks regionów stron wspomaga nawigowanie pomiędzy zakresami drzewa. Pula buforów poprawia wydajność poprzez zmniejszenie bezpośrednich, sekwencyjnych operacji I/O oraz poprzez stosowanie asynchronicznych operacji czytania i zapisywania danych, podobnie jak w przypadku danych relacyjnych. Podczas definiowania puli buforów można również podawać wielkość bloku na stronach danych oraz liczbę stron, które mają być niezblokowane i zblokowane. Wprowadzenie zblokowania stron w puli buforów powoduje to, że sąsiednie strony na dysku, wewnątrz zblokowanego obszaru będą razem przeniesione do puli buforów. W pewnych sytuacjach podejście to może poprawić wydajność, chociaż obecnie w nowych wersjach systemu DB2 zalecane jest, aby system sam decydował o wykorzystania swoich zasobów pamięci (tzw. STMM - Self-Tuning Memory Manager).



Rys.2. Sposób definiowanie i przechowywania danych relacyjnych oraz XML.

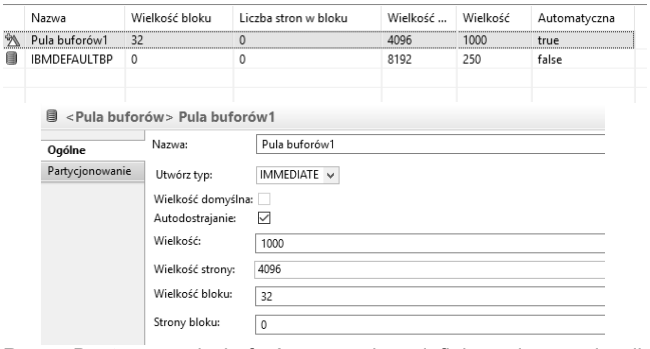


Rys.3. Hierarchiczny sposób składowania dokumentów XML z wykorzystaniem indeksów różnego typu oraz regionów stron.



Rys.4. Pula buforów jako rzeczywista pamięć podręczna dla danych tablicowych oraz indeksowych. Kontenerami mogą być pliki, katalogi lub urządzenia typu RAW

Można monitorować aktywność buforowania danych XML niezależnie od monitorowania aktywności buforowania dla danych relacyjnych tak, aby możliwa była ocena aktywność puli buforów dla obiektów składowanych w typie XML (Rys.4).



Rys.5. Dostępne pule buforów oraz okno definiowania nowej puli buforów w środowisku Data Studio.

Optymalizacja i wydajność

W dokumentacji pureXML środowiska DB2 [1],[2] podane są wytyczne dotyczące poprawy wydajności przetwarzania danych XML składowanych z wykorzystaniem technologii pureXML. Do wybranych wytycznych zaliczyć można:

- korzystanie z przestrzeni tablicowych zarządzanych automatycznie lub przestrzeni typu DMS (Database Managed Spaces) o większym rozmiarze stron np. 32kB. Im większy rozmiar strony tym mniejsza jest liczba regionów (a zatem również i podziałów) na dokumencie XML oraz tym większa wydajność przetwarzania, zwłaszcza dla operacji wstawiania i pobierania całych dokumentów,
- dla małych dokumentów XML należy korzystać z opcji przechowywania "inline", razem z całym wierszem rekordu oraz w sposób skompresowany. Dobrym pomysłem jest również przechowywanie danych XML w oddzielnej przestrzeni tabel.

Przykład instrukcji definiowania tabeli do przechowywania danych XML w trybie „inline":

```
CREATE TABLE product(pid BIGINT,
name VARCHAR(20), brand VARCHAR(35),
category INTEGER, price DECIMAL,
description XML INLINE length 3000) COMPRESS YES;
```

- większe dokumenty XML powinny być oddzielone od danych relacyjnych i jeżeli jest to możliwe to powinny być składowane w oddzielnych tabelach z odpowiednio dopasowanymi pulami buforów (Rys.6).

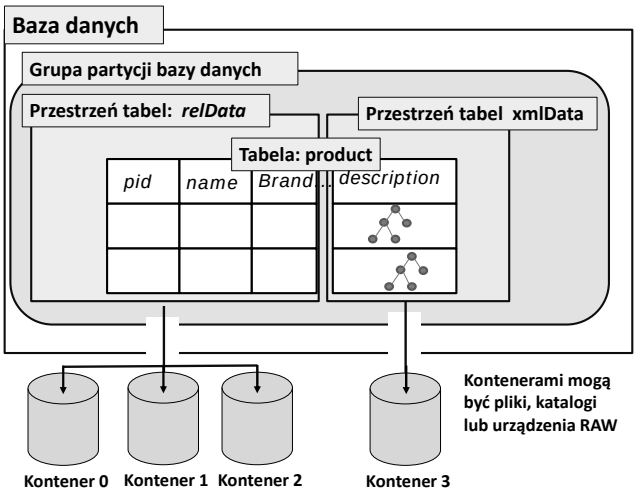
Przykład instrukcji rozdzielających składowanie danych relacyjnych i hierarchicznych:

```
CREATE BUFFERPOOL bp4k PAGESIZE 4k;
CREATE BUFFERPOOL bp32k PAGESIZE 32k;

CREATE TABLESPACE relData PAGESIZE 4K
MANAGED BY AUTOMATIC STORAGE
BUFFERPOOL bp4k;

CREATE TABLESPACE xmlData pagesize 32K
MANAGED BY AUTOMATIC STORAGE
BUFFERPOOL bp32k;

CREATE TABLE product(pid BIGINT, name
VARCHAR(20), brand VARCHAR(35),
category INTEGER,
price DECIMAL,
description XML)  in relData  LONG in xmlData;
```



Rys.6 Przykład rozdzielania składowania danych relacyjnych i hierarchicznych w różnych przestrzeniach tabel.

- rozważne definiowanie indeksów XML i unikanie indeksowania wszystkiego,
- okresowo za pomocą elementów monitora migawek należy badać wydajność przetwarzania struktury XML
- w wyrażeniach XPath, należy korzystać z pełni określonych ścieżek nawigacji po dokumencie. Do badania wydajności zapytań XML, podobnie jak w przypadku danych relacyjnych wykorzystuje się te same narzędzia diagnostyczne, takie jak: db2exfmt oraz plan wykonania dla zapytania XML.

Poprawę wydajności zapytań można również uzyskać przez zastosowanie mechanizmów, partycjonowania bazy danych, papartycjonowania tabel oraz wielowymiarowego klastrowania.

Definiowanie indeksów na strukturze hierarchicznej

Indeksy założone na strukturze hierarchicznej pozwalają poprawić wydajność przetwarzania XML. Indeksy mogą być stworzone dla elementów, atrybutów lub wartości (węzłów tekstowych) (Rys. 1).

Przykład instrukcji która tworzy indeks na atrybucie @type - (tabela Customer, kolumna XML o nazwie info, struktura tabeli jak na Rys.2):

```
CREATE UNIQUE INDEX index1 ON customer(info)
GENERATE KEY USING xmlpattern
'/customerinfo/phones/phone/@type'
AS sql VARCHAR(10)
```

Przykład instrukcji, która tworzy indeks na elemencie o nazwie name:

```
CREATE INDEX index1 ON customer(info)
GENERATE KEY USING
xmlpattern '/customerinfo/name'
AS sql VARCHAR(10)
```

Przykład instrukcji, która definiuje indeks na wszystkich węzłach tekstowych (instrukcja nie jest zalecana dla dużych woluminów XML z racji tego, że indeks może być zbyt duży):

```
CREATE INDEX index1 ON customer(info)
GENERATE KEY USING
xmlpattern '//text()'
AS sql VARCHAR(10);
```

Dostęp do danych w strukturach relacyjno-hierarchicznych

W systemie DB2 10.5, istnieją cztery sposoby dostępu do danych składowanych w strukturach relacyjno - hierarchicznych:

- przy użyciu języka SQL do przeszukiwania danych relacyjnych,
- przy użyciu języka SQL z rozszerzeniem XML do przeszukiwania danych XML,
- przy użyciu składni języka XQuery do przeszukiwania dokumentów XML,
- przy użyciu języka XQuery do przeszukiwania danych relacyjnych.

Odpytywanie struktury bazy hierarchicznej w dialekcie języka SQL z rozszerzeniami XML polega na wykorzystaniu jednej z dostępnych funkcji. W dokumentacji [2], [3] w sposób obszerny opisano wykorzystanie takich funkcji. Funkcje takie pozwalają nie tylko na przeszukiwanie danych w składowanych strukturach XML-owych lecz również na konwersję danych XML do danych relacyjnych i odwrotnie. Do funkcji takich zaliczyć można między innymi: xmlquery, xmltable, xmlparse, xmlserialize, xmlvalidate, xmlexists, xmlcast, xmlelement, xmlattributes, xmlconcat, xmltext oraz inne.

Przykład wykorzystania funkcji XMLEXISTS, która sprawdza, czy wyrażenie XQuery zwraca sekwencję elementów:

```
SELECT CID, INFO
FROM XMLCUSTOMER WHERE
XMLEXISTS(
   '$d/customerinfo[name = "Nowak"]'
   passing INFO as "d");
```

Przykład wykorzystania funkcji XMLTABLE, która tworzy tabelę tymczasową SQL przy użyciu danych XML oraz pozwala na dalsze sprzęganie tak wygenerowanych danych z danymi składowanymi w schemacie relacyjnym:

```
SELECT T.* FROM XMLTABLE('db2 fn:xmlcolumn
( "CUSTOMER.INFO")/customerinfo' COLUMNS "NAME"
VARCHAR (20) PATH 'name', "STREET" VARCHAR (20)
PATH 'addr/street', "CITY" VARCHAR (20) PATH
'addr/city') AS T;
```

Odpytywanie struktury bazy hierarchicznej przy użyciu składni języka XQuery możliwe jest przy zastosowaniu zasadniczo dwóch funkcji:

- db2-fn: xmlcolumn oraz
- db2-fn: sqlquery

Pierwsza z nich pozwala na pozyskiwanie danych XML z kolumny typu XML druga natomiast na odpytywanie danych za pomocą składni języka SQL w XQuery.

Przykład instrukcji w której wykorzystana została funkcja db2-fn:xmlcolumn, która zwróci listę krajów i miast dla każdego klienta w nowym elemencie XML o nazwie custAddr:

```
xquery
for $customer in db2-fn:xmlcolumn
('CUSTOMER.INFO')/customerinfo
return
<custAddr> {$customer/addr/@country,
$customer/addr/city} </custAddr>;
```

Przykład instrukcji w której wykorzystana została funkcja db2-fn:xmlcolumn, która zwróci liczbę produktów w cenie mniejszej od 100. Funkcja count jest funkcją wbudowaną środowiska DB2.

```
xquery
count(db2-fn:xmlcolumn
('XMLPRODUCT.DESCRIPTION')/product/description[pri
ce < 100]);
```

Język XQuery oraz XPath (który jest podzbiorem XQuery) obsługuje ścieżki wyrażeń pozwalające przechodzić po hierarchicznej strukturze XML. XQuery wspiera wyrażenia FLWOR które odpowiadają wyrażeniu SELECT-FROM-WHERE. FLWOR jest akronimem słów:

FOR - przechodzenie po kolekcji, przypisywanie elementu do zmiennej
LET - przypisywanie elementów do kolekcji
WHERE - usuwanie elementów z kolekcji
ORDER - zmiana kolejności elementów w kolekcji
RETURN - zwracanie wyniku.

Przykład instrukcji z wyrażeniem FLWOR:

```
xquery
for $customer in db2-fn:xmlcolumn
('XMLCUSTOMER.INFO')/customerinfo
where $customer/addr/city = 'Lodz'
order by $customer/name
return ($customer/name, $customer/addr);
```

Podsumowanie

W artykule zaprezentowano natywne składowanie i przetwarzanie danych XML w strukturach relacyjno-hierarchicznych, które pozwala na uzyskanie wzrostu wydajności przetwarzania informacji po stronie bazy danych. Składowanie takie możliwe jest z pominięciem operacji dekompozycji XML na fragmenty. Podano wytyczne dotyczące poprawy wydajności przetwarzania danych XML składowanych z wykorzystaniem technologii pureXML.

***Autorzy**: dr inż. Paweł Drzymała, Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, Stefanowskiego 18/22, 90-924 Łódź, E-mail: pawel.drzymala@p.lodz.pl; dr inż. Henryk Welfle, Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, Stefanowskiego 18/22, 90-924 Łódź, E-mail: henryk.welfle@p.lodz.pl.*

LITERATURA

[1] IBM best practices for pureXML: http://www.ibm.com/developerworks/data/library/techarticle/dm-0610nicola/
[2] http://public.dhe.ibm.com/ps/products/db2/info/vr105/pdf/en_US/DB2pureXML-db2xge1050.pdf
[3] IBM DB2 10.5 dla systemów Linux, UNIX i Windows - Centrum informacyjne: http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp
[4] Matthias Nicola, Pav Kumar-Chatterjee: DB2 pureXML Cookbook: Master the Power of the IBM Hybrid Data Server, IBM Press book, 10 Aug 2009.
[5] Whei-Jen Chen, A. Sammartino, D. Goutev, F. Hendricks, I.Komi, Ming-Pang Wei, R. Ahuja: DB2 9 pureXML Guide, ibm.com/redbooks, January 2007.
[6] C.M. Saracco, D. Chamberlin, R. Ahuja: DB2 9: pureXML Overview and Fast Start, ibm.com/redbooks, June 2006.
[7] Korzeniewska E., Duraj A., Krawczyk A.; Detection of local changes in resistance by means of data mining algorithms, Przegląd Elektrotechniczny, 2014 R.90, nr 12, s. 229-232.