STUDIA INFORMATICA 2009
Volume 30 Number 2B (84)

Michał KWAPISZ, Tomasz SPALENIAK, Adam PELIKANT
Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych

MIGRACJA SCHEMATÓW BAZ DANYCH Z ZASTOSOWANIEM WARSTWY TRANSPORTOWEJ XML

Streszczenie. Artykuł opisuje metodę automatycznego przenoszenia schematów relacyjnych baz danych. Zastosowane zostały pliki XML, które pozwalają na swobodne kształtowanie postaci dokumentu. Celem prowadzonych badań było stworzenie wydajnego algorytmu automatycznej migracji danych, który będzie gwarantował poprawną budowę docelowej struktury danych oraz bezpieczeństwo podczas przetwarzania. Prace prowadzono w środowisku Microsoft Windows 2003 z wykorzystaniem silnika bazy danych ORACLE 10g.

Słowa kluczowe: migracja danych, integracja danych, XML, Oracle.

THE DATABASE SCHEMA MIGRATION WITH USE XML TRANSPORTING LAYER.

Summary. The article describes method of the automatic transfer of relational databases schemas. Files XML were applied, which permit on free formation of document structure. The aim of investigations was creation of effective algorithm for automatic data migration which would guarantee correct target data structure as well as safety during processing. Works were guided in environment Microsoft Windows 2003 with utilization the engine of database ORACLE 10g.

Keywords: Database migration, data integration, XML, Oracle.

1. Wstęp

XML (ang. eXtensible Markup Language), czyli rozszerzalny język znaczników, to otwarty standard opracowany przez konsorcjum W3C. Język ten nie jest kolejnym standardem przechowywania i prezentacji danych, jak np. język HTML, który opisuje wygląd stron internetowych, lecz samoopisującym się językiem opisującym dane, metajęzykiem.

Wprowadzonych zostało wiele specjalizowanych standardów samego XML, jak chociażby język AIML[1], wykorzystywany do budowy inteligentnych systemów symulujących odpowiedzi człowieka. Innym przykładem mogą być standardy opisu przepływu dokumentów czy wprowadzany ostatni standard epacjent. W naszym opracowaniu chcemy się skupić na zastosowaniu XML do zagadnień związanych z migracją i integracją danych.

W świetle baz danych XML można wykorzystać do opisu danych zawartych w systemach baz danych. Każda baza danych składa się z szeregu tabel, ich atrybutów oraz danych w nich zawartych. Tabele, atrybuty oraz relacje je wiążące, przy użyciu standardu XML można przedstawić w postaci tabelaryczno – drzewiastej, zachowując przy tym hierarchię opisywanych danych. Dokument wykorzystujący technologię XML można wykorzystać do przenoszenia lub przechowywania struktury danych dowolnej relacyjnej bazy danych.

Celem pracy było utworzenie narzędzia, mechanizmu do analizy i odwzorowywania dokumentów opisanych standardem XML na schematy baz danych, tj.: tabel, atrybutów oraz danych w nich zawartych przez zastosowanie wydajnych i bezpiecznych algorytmów parsowania danych, powodujących zwiększenie wydajności oraz skrócenie czasu analizy. Większość algorytmów opiera się na zmiennych w postaci tabel indeksowanych, zbudowanych z rekordów, które są proste w obsłudze, mają duży rozmiar (do 2 GB). Górna i dolna granica jest ograniczona jedynie przez typ BINARY_INTEGER, którego wartości mieszczą się w zakresie od –2 147 483 647 do +2 147 48 3647. Przechowywanie oraz przetwarzanie danych znajdujących się w zmiennej tabelarycznej odbywa się w pamięci, dzięki czemu wzrasta wydajność przetwarzania, na skutek pominięcia operacji odczytu i zapisu do fizycznych lokalizacji, które są co najmniej o kilka rzędów wolniejsze od operacji na danych w pamięci.

Podczas pracy nad rozwiązaniem przyjęto kilka dodatkowych założeń:

- Dokument poddawany analizie, oprócz podstawowych znaczników, może zawierać dodatkowo atrybuty, instrukcje przetwarzania, sekcje CDATA oraz komentarze.
- Odwzorowanie każdego atrybutu opisującego elementy XML, na atrybuty encji (odpowiedników elementów) oraz utworzenie dla każdej encji identyfikatora ID, który ma pełnić rolę klucza głównego zgodnie z pierwszą postacią normalną oraz identyfikatorów dla kluczy obcych, odwołujących się do encji nadrzędnych zgodnie z drugą postacią normalną.
- Możliwość zapisania wielu obiektów (np. tabel) w jednym dokumencie XML.
- Utworzenie funkcji analizujących dane, których zadaniem jest zwrócenie ich typu wraz z optymalną długością.
- Utworzenie funkcji sprawdzających, czy wygenerowane atrybuty encji istnieją fizycznie w bazie danych, zapobiegając efektowi duplikacji atrybutów, co z kolei prowadzi do generowania błędów. Zadaniem tej funkcji miało być także generowanie instrukcji

[1] AIML - Artificial Intelligence Markup Language

modyfikujących, dodających encje lub atrybuty, które wygenerowano z dokumentu XML, a nie znajdujące się fizycznie w bazie danych.

- Utworzenie całego rozwiązania w postaci jednego obiektu – pakietu PL/SQL dostępnego dla każdego użytkownika systemu Oracle, dzięki synonimowi publicznemu, zawierającego wszystkie funkcje i procedury umożliwiające kompletną analizę dokumentu.

2. Projekt

2.1. Operacje na plikach

Pakiet UTL_FILE jest wbudowanym pakietem stworzonym przez firmę Oracle Corporation dostępnym od wersji 7.3.4, dostarczającym interfejs API dla operacji I/O na plikach. Dzięki niemu programy napisane w języku PL/SQL, wykorzystujące jego interfejs, posiadają możliwość odczytywania i zapisywania plików w lokalnym systemie plików systemu operacyjnego. Pliki muszą znajdować się fizycznie na tym samym komputerze, na którym znajduje się serwer bazy danych, lub muszą być dostępne za pośrednictwem odwzorowania katalogów.

Dostęp do pliku przez pakiet UTL_FILE odbywa się przez nazwę katalogu oraz nazwę pliku. Operacje na plikach są realizowane tylko i wyłącznie wtedy, gdy katalog jest dostępny dla bazy danych. Dostęp jest możliwy na dwóch poziomach bezpieczeństwa:

- Użytkownik, właściciel procesów Oracle, musi mieć możliwość odczytu lub zapisu do katalogu.
- Katalog musi być zdefiniowany przez parametr UTL_FILE_DIR w pliku init.ora. Parametr ten jako jedyny (spośród wszystkich parametrów pliku konfiguracyjnego) może przyjmować listę katalogów, które będą uznawane za poprawne, lub posiadać wartość w postaci znaku *, oznaczającym, że wszystkie katalogi są poprawne.

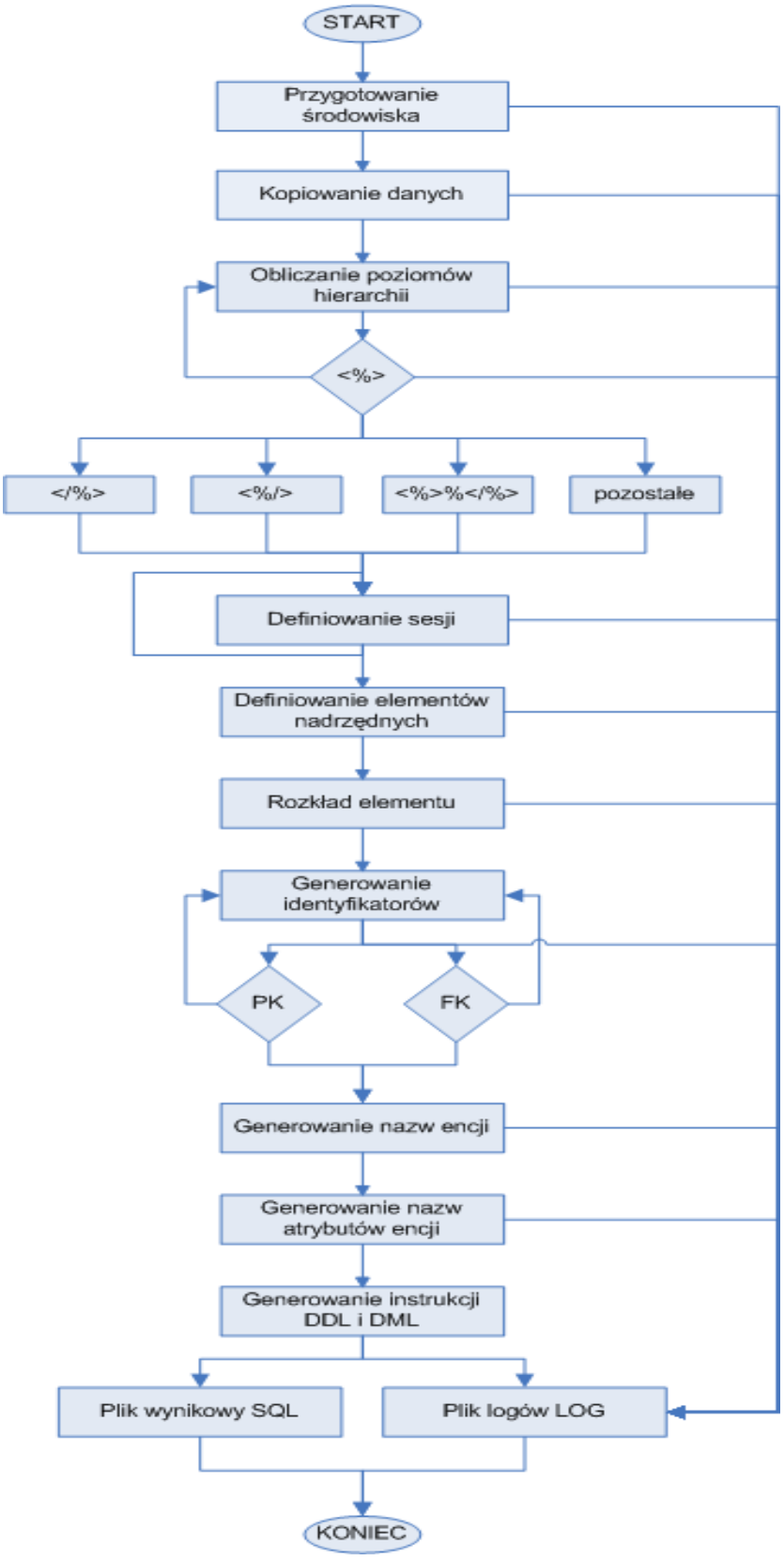
2.2. Pakiet XMLMS – silnik rozwiązania

Pakiet XMLMS zawiera wszystkie procedury i funkcje potrzebne do wygenerowania pliku ze strukturą bazodanową na podstawie dokumentu XML. Poniżej znajduje się szczegółowy opis poszczególnych jego obiektów oraz algorytmy działania wraz z przykładami. W skład pakietu wchodzą następujące obiekty:

- Funkcja IS_DIGIT – przyjmuje dowolny znak p_sign i zwraca wartość liczbową dla odpowiedniego typu danych. Typem zwracanym może być 1 dla wartości od 0 do 9 (odpowiednik NUMBER) oraz 0 dla pozostałych znaków (odpowiednik VARCHAR2).

- Funkcja GET_DATA_TYPE – przyjmuje dowolny ciąg p_data_value i zwraca jego typ. Zwracanym typem może być NUMBER dla liczb całkowitych oraz VARCHAR2 dla pozostałych ciągów znakowych. W przypadku podania pustej wartości funkcja zwróci NULL.
- Funkcja CALCULATE_ATTRIBUTE – przyjmuje dowolny ciąg p_element zawierający definicje elementu oraz atrybutów i zwraca liczbę występujących w nim atrybutów.
- Procedura LOGGER – procedura wstawia log o treści określonej parametrem p_element do pliku p_file_name składowanego fizycznie na dysku w miejscu określonym przez parametr p_path_name. Dodatkowo log zawiera informacje, z jakiego obiektu p_source został wstawiony, dokładną datę wraz z godziną oraz typ logu p_log_type. Typ logu określamy przez podanie jednej z trzech wartości: I (Info), E (Error), W (Warning). Domyślną wartością dla tego parametru jest I (Info). Przykładowy tekst wyświetlany przez procedurę:

  [10/Sty/2009 22:24:32] [INFO] [xmlms.parser] Start parsera
- Procedura INSERT_BUFFER – procedura wstawia zawartość bufora p_buffer do wskazanego pliku p_file_name, znajdującego się w określonej lokalizacji p_path_name. Zawartość bufora jest dopisywana do obecnej zawartości pliku, nie powodując przy tym utraty danych istniejących już w pliku.
- Funkcja FORMAT_ROW – analizuje wiersze p_row_value pobrane z pliku, pobierając jedynie wartościowe (dla programu), a pomija te, które są instrukcjami przetwarzania, komentarzami oraz instrukcjami CDATA.
- Funkcja PARSE_TABLE_OBJECT – analizuje przekazany element p_element, będący przykładowym wierszem z pliku i zwraca tablicę zbudowaną na jego podstawie, tzn. zwraca nazwę tagu jako nazwę encji, atrybuty z wartościami jako nazwy atrybutów z przyporządkowanymi im wartościami.
- Funkcja GET_PARENT_NAME – zwraca nazwę elementu głównego z podanego Elementu p_element.
- Funkcja GET_TAB_COLL_EXISTS – funkcja sprawdza, czy podana tabela p_table_name lub para tabel i kolumna p_column_name istnieje fizycznie w bazie danych. W zależności od wyniku zwraca wartość EXISTS lub NOT EXISTS. Funkcja jest wykorzystywana do generowania instrukcji modyfikujących istniejące struktury na bazie.



Rys. 1. Schemat blokowy procedury PARSER
Fig. 1. Block schema of procedure PARSE

- Procedura PARSER – procedura odpowiada za wykonanie całej analizy pliku i zwrócenie jako wyniku, gotowego zestawu instrukcji DDL oraz DML w postaci pliku. Jako parametr wejściowy p_path, przyjmuje ścieżkę do pliku XML oraz flagi odpowiadające za ustawienie parsera. Flagi te decydują o generowaniu poszczególnych grup instrukcji. Flaga p_gen_prompt oznacza generowanie komentarzy, flaga p_gen_table oznacza generowanie definicji tabel, p_gen_pk oraz p_gen_fk oznaczają generowanie definicji kluczy podstawowych oraz obcych, p_gen_sequence oznacza generowanie sekwencji, p_gen_trigger oznacza generowanie wyzwalaczy, p_gen_tab_columns oznacza generowanie instrukcji modyfikujących istniejące struktury na bazie, a p_gen_insert oznacza generowanie instrukcji DML wstawiających dane (algorytm ilustruje rys. 1).

3. Testy wydajnościowe

Testy zostały zrealizowane na komputerze klasy PC o następujących parametrach:

- CPU AMD Semptron® 2800+ 2,00 GHz,
- 1GB RAM (DUAL 2x 512MB),
- HDD Maxtor 6Y080L0, 160 GB, 7200rpm.

3.1. Testy stałości czasu analizy

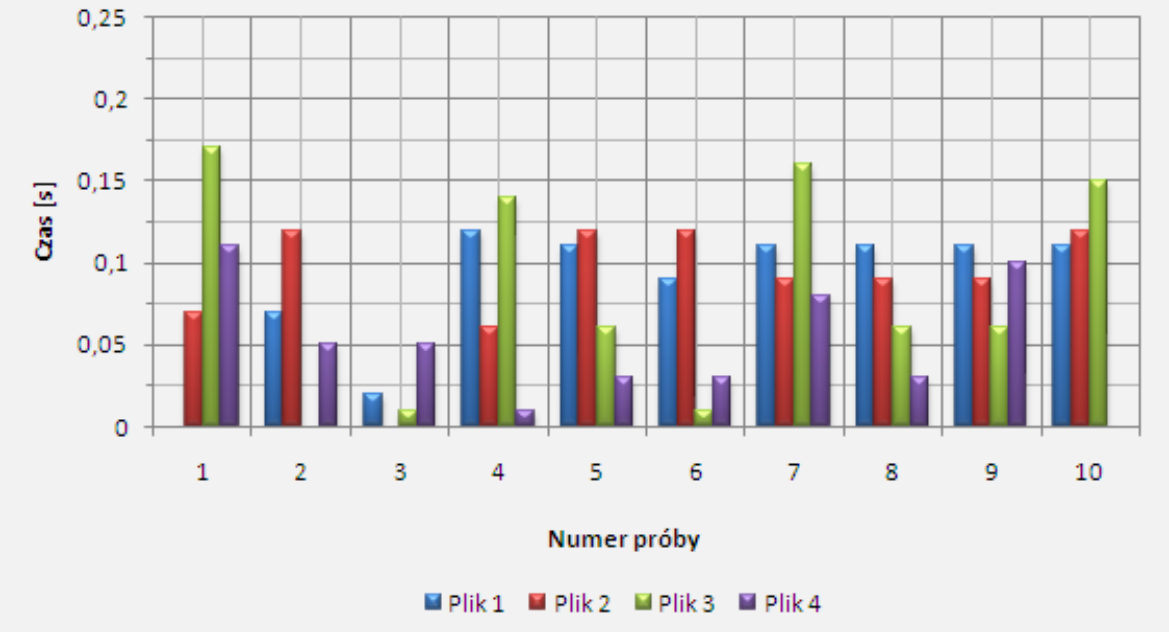
Celem tego testu jest sprawdzenie stałości czasu analizy pojedynczego dokumentu. Na potrzeby testu zostały stworzone 4 dokumenty o różnej liczbie wierszy:

Tabela 1

Wyniki testu powtarzalności czasu analizy dla czterech rodzajów plików

Lp.	Plik 1 [s]	Plik 2 [s]	Plik 3 [s]	Plik 4 [s]
1	0,39	2,23	0,53	5,39
2	0,46	2,28	0,36	5,33
3	0,41	2,16	0,37	5,33
4	0,51	2,22	0,50	5,29
5	0,50	2,28	0,42	5,31
6	0,48	2,28	0,37	5,31
7	0,50	2,25	0,52	5,36
8	0,50	2,25	0,42	5,31
9	0,50	2,25	0,42	5,38
10	0,50	2,28	0,51	5,28
Max t	0,51	2,28	0,53	5,39
Min t	0,39	2,16	0,36	5,28
Δ t	0,12	0,12	0,17	0,11

- Plik 1. Plik bez atrybutów zawierający ok. 30 wierszy.
- Plik 2. Plik bez atrybutów zawierający ok. 1000 wierszy.
- Plik 3. Plik z atrybutami zawierający ok. 30 wierszy.
- Plik 4. Plik z atrybutami zawierający ok. 1000 wierszy.



Rys. 2. Wyniki testu powtarzalności czasu analizy dla czterech rodzajów plików
Fig. 2. Results of test of analysis time repeatability for four data files

Przekształcając dane z tabeli 1, otrzymujemy wykres zależności $f(t) = t - \min(t)$, przedstawiający wartość odchyleń czasu od wartości minimalnej (rys. 2).

Na podstawie wykresu oraz danych z tabeli można zauważyć, że niezależnie od liczby wierszy w pliku oraz od tego, czy dokument posiada atrybuty czy nie, czas odchyleń od wartości minimalnej utrzymuje się na poziomie 0.13 s. Czas ten jest bardzo niewielki i może być spowodowany niekoniecznie błędnym zaprojektowaniem algorytmów parsujących, lecz dostępem i przetwarzaniem tablic przechowywanych w pamięci lub obciążeniem systemu, spowodowanym przez wykonywanie procesów systemowych lub aplikacyjnych. Czas ten jest w pełni akceptowalny i może nie być uznany za wadę systemu.

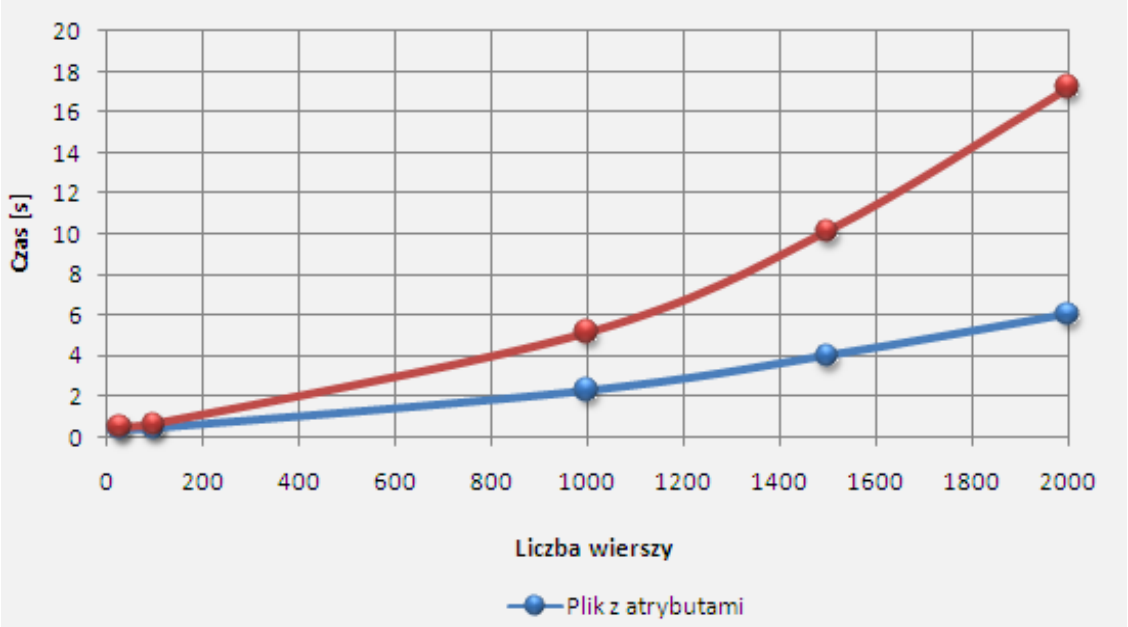
3.2. Testowanie czasu analizy

Celem tego testu jest sprawdzenie czasu analizy dokumentu w zależności od analizowanej liczby wierszy dla pliku z atrybutami oraz dla pliku bez atrybutów. Tabela 2 zawiera wyniki tego testu.

Tabela 2

Wyniki testu zależności czasu analizy od liczby rekordów w pliku XML.

Typ pliku	Liczba wierszy	Czas [s]
Plik z atrybutami	30	0,43
	100	0,46
	1000	2,32
	1500	3,99
	2000	6,05
Plik bez atrybutów	30	0,53
	100	0,62
	1000	5,15
	1500	10,13
	2000	17,23



Rys. 3. Zależności czasu analizy od liczby rekordów w pliku XML
Fig. 3. Dependency of analysis time from numbers o records in XML file

Po analizie powyższych danych można zauważyć, że plik z atrybutami jest wykonywany w krótszym czasie niż plik bez atrybutów. Ponadto, można zauważyć, że funkcja czasu wykonania analizy od ilości analizowanych wierszy dla pliku zawierającego atrybuty jest bliska liniowej, natomiast dla pliku niezawierającego atrybutów jest bliska funkcji wykładniczej. Dzieje się tak dlatego, gdyż dla pierwszej grupy plików w pamięci jest przechowywana mniejsza liczba atrybutów encji, spowodowana przeznaczeniem samych atrybutów (atrybuty stosuje się dla wartości stałych, identycznych dla każdego elementu encji), natomiast w drugiej grupie plików atrybutów tych będzie przechowywanych znacznie więcej. Przykładowo, dla pliku zawierającego 1000 wierszy, gdzie w co 5 wierszu znajduje się jeden atrybut, mamy ok. 1200 atrybutów przechowywanych w pamięci, natomiast w pliku, w którym atrybuty nie

są stosowane, będzie ich się znajdować ok. 2000. Różnica ta tłumaczy nam charakter przebiegu obu funkcji.

4. Podsumowanie i wnioski

Wynikiem przedstawionej w artykule pracy jest bardzo wydajny parser plików XML, pozwalający automatycznie przekształcać strukturę zapisaną w dokumentach XML do struktury relacyjnej bazy danych. Zastosowanie tego rozwiązania może być bardzo szerokie, jednak dedykowane będzie do rozwiązań baz relacyjnych pracujących w architekturze klient-serwer systemów rozproszonych działających w globalnej sieci Internet. Aktualnie prowadzone są prace nad bardzo wydajnym algorytmem, pozwalającym na eksport obiektów bazy danych do struktur XML. Planowanym dalszym krokiem rozwiązania jest budowa serwera plików XML, pozwalającego na jeszcze lepsze dopasowanie narzędzia do zastosowań migracji i integracji danych ze źródeł heterogenicznych.

BIBLIOGRAFIA

1. Rosa T., Jagoda Ł., Pelikant A.: Migracja danych pomiędzy XML, a relacyjnymi modelami danych. 1st Polish and International PD Forum – Conference on Computer Science 2005, Bronisławów 2005.
2. Bourret R., Bornhövd C., Buchmann A.: A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. 2000 IEEE.
3. Kotsakis E., Bohm K.: XML Schema Directory: A Data Structure for XML Data Processing. 2000 IEEE.
4. Fong J., Pang F., Bloor C.: Converting Relational Database into XML Document. 2001 IEEE.
5. Bohannon P., Freire J., Roy P., Simeon J.: From XML Schema to Relations: A Cost-Based Approach to XML Storage. 2002 IEEE, 18th International Conference on Data Engineering (ICDE.02).
6. Afonso de Sousa1 A., Luís Pereira J., Álvaro Carvalho J.: Querying XML Databases. 2002 IEEE, XXII International Conference of the Chilean Computer Science Society (SCCC'02).
7. Ishitani Y.: Document Transformation System from Papers to XML Data Based on Pivot XML Document Method. 2003 IEEE, Seventh International Conference on Document Analysis and Recognition (ICDAR'03).

8. Krishnamurthy R., Chakaravarthy V., Kaushik R., Naughton J.: Recursive XML Schemas, Recursive XML Queries, and Relational Storage: XML-to-SQL Query Translation. 2004 IEEE, 20th International Conference on Data Engineering (ICDE'04)

Recenzent: Dr inż. Aleksandra Werner

Wpłynęło do Redakcji 15 lutego 2009 r.

Abstract

The aim of this project was formation an instrument, mechanism for analysis and associate documents described using XML standard at databases schemes i.e.: tables, attributes and data included in theirs through application an efficient and safe algorithms data parsing, caused increase a performance and decrease analysis time. Storage and processing data found in changeable table variable take place in memory, thanks to performance of processing rise, as a result of skip reading operation and saving data to physical localizations, which are slower then operations on data in memory.

Working on solution, had been accepted a few additional assumptions:

- Document exposed to analysis can include attributes, processing instructions, CDATA sections and comments.
- Impinge each attribute described XML elements, on entity attributes and formation for every entity id name tag according to first normal form and name tags for strange keys, reference to overriding entity according to second normal form.
- Formation functions analyzing (analyzed) data, which task is return (returning) their type along with optimal length.
- Formation functions checking if generated attributes of entity exist physically in database.
- Formation the whole solution in the form of one object – PL/SQL Project packet.

XMLMS packet includes all procedures and functions needed to generation file with database structure created on the basis of XML document. Below there are detailed descriptions of individual their objects and operation algorithms along with examples. Fallowing objects are included in packet:

- GET_DATA_TYPE function – it accepts any sequence p_data_value and returns its type. NUMBER or VARCHAR2 could be a returned type.
- CALCULATE_ATTRIBUTE function – it accepts any sequence p_element included element and attribute definitions and returns the number of included in it attributes.

- PARSE_TABLE_OBJECT function – it analyzes transmitted element p_element, being a hypothetical line from file and it returns the table constructed on the basis of it, i.e. it returns the tag name as entity name, attributes with values as attribute names with ascribed them values.
- GET_PARENT_NAME function – it returns main element name from given element p_element.
- GET_TAB_COLL_EXISTS function – function checks if given table p_table_name or pair table and column p_column_name exist physically in database.
- PARSER procedure – procedure is responsible for execution all file analysis and returns as result, ready combination instruction DDL and DML in the form of file. As front parameter p_path, it adopts path to XML file and flags responsible for parser arrangement. These flags decide on generation individual instruction's groups. P_gen_prompt flag means generation comments, p_gen_table flag means generation table's definitions, p_gen_pk and p_gen fk flags mean generation main and strange key's definitions, p_gen_sequence means generation sequences, p_gen_trigger means generation of triggers, p_gen_tab_columns means generation instructions, which modify existing structures on base, and p_gen_insert means generation DML instructions, which put on data.

Result of presented in article research is a very efficient files XML parser, which permit automatically transform structure stored in XML documents to relational database's structure. Application of this solution could be a very wide, however it will be dedicated for solutions relational database's working at (in) architecture client – server systems working in global internet network. Next steep for this solution is building of file XML server, which allows to another, better fit instrument to applications of migration and integration data from heterogenic sources.

Adresy

Michał KWAPISZ: Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, ul. Stefanowskiego 18/22, 90-924 Łódź.
Tomasz SPALENIAK: Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, ul. Stefanowskiego 18/22, 90-924 Łódź.
Adam PELIKANT: Politechnika Łódzka, Instytut Mechatroniki i Systemów Informatycznych, ul. Stefanowskiego 18/22, 90-924 Łódź.