STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Robert MARCJAN, Jakub WYROSTEK
Akademia Górniczo-Hutnicza, Katedra Informatyki

PRZETWARZANIE DOKUMENTÓW XML NA PODSTAWIE MODELU QUASI-RELACYJNEGO I JĘZYKA SQLxD

Streszczenie. Autorzy proponują zastosowanie modelu quasi-relacyjnego jako podstawy przetwarzania danych pochodzących z dokumentów XML. Dane XML są transformowane do zbiorów relacyjnych według intuicyjnych reguł, wykorzystujących składnię i strukturę źródłowego dokumentu. Elementem rozwiązania jest język zapytań nazwany SQLxD, opierający się na składni popularnego SQL. Element ten stanowi narzędzie do transformacji danych oraz dalszego ich przetwarzania.

Słowa kluczowe: XML, relacyjny model danych, SQL, zapytanie

PROCESSING XML DOCUMENTS ON THE BASIS OF QUASI-RELATIONAL MODEL AND SQLxD LANGUAGE

Summary. The authors propose to use the quasi-relational model as a basis to the XML data processing. The XML data is transformed to the relational data sets according to the intuitive rules that make use of the syntax and structure of the source document. Part of the solution is the query language called SQLxD, which is based on the popular SQL syntax. SQLxD is a tool for both data transformation and its further processing.

Keywords: XML, relational data model, SQL, query

1. Wprowadzenie

Dokumenty XML zapewniają sposób organizacji informacji, szeroko wykorzystywany we współczesnej informatyce także na styku aplikacji i systemów zarządzania baz danych. Mnogość dostępnych technik i narzędzi, służących do przetwarzania dokumentów XML, świadczy z kolei o tym, że zapewnienie wygodnej obsługi tego popularnego formatu jest zadaniem niełatwym. Niniejszy artykuł proponuje rozważenie modelu quasi-relacyjnego jako

podstawy przetwarzania dokumentów XML. Dane XML są transformowane do modelu relacyjnego wg intuicyjnych reguł, wykorzystujących składnię i strukturę źródłowego dokumentu. Elementem rozwiązania jest, opierający się na składni SQL, język zapytań nazwany SQLxD. Język ten stanowi narzędzie służące do uruchamiania procesu transformacji XML do modelu relacyjnego oraz dalszego przetwarzania tych danych w rachunku zbiorów i predykatów.

2. Dokument XML jako nośnik informacji

XML został opracowany w latach 1996-1998 przez grupę specjalistów W3C jako podtyp szerszego języka znaczników SGML i jako standard jest rozwijany do dziś [1,2]. Jako nośnik danych XML pozwala na niespotykane wcześniej bogactwo opisu informacji, wykorzystujące nie tylko cechy pojedynczych jednostek opisu, ale również – poprzez hierarchiczną budowę – wzajemne relacje tych jednostek. Z praktycznego punktu widzenia XML miał mieć takie cechy, jak: przenośność w obrębie sieci Internet, wsparcie dla różnego rodzaju aplikacji czy sposób zapisu, czytelny dla człowieka [3].

Dokument XML, szczególnie taki ze skomplikowaną i rozbudowaną strukturą, jest jednak niemal zawsze bezużyteczny bez oprogramowania, które potrafi dane zawarte w dokumencie wczytać i przetwarzać [4]. Prosta, a jednocześnie wielofunkcyjna, możliwie wydajna w przetwarzaniu technika wydaje się zatem koniecznością.

Od czasu wprowadzenia XML próbowano wielu rozwiązań tego zagadnienia. Format danych XML, mimo iż potężny w wymowie jest niestety niespecjalnie wygodny w zakresie „łatwego przetwarzania i manipulowania danymi", szczególnie jeśli porównamy go z danymi o charakterze relacyjnym wraz z językiem SQL czy też nawet z rozwiązaniami takimi, jak np. proste pliki tekstowe i odpowiedni zasób wyrażeń regularnych (rozwiązania te nie sprawdzają się w bezkontekstowej gramatyce XML i jego hierarchicznej naturze).

Między innymi z tego powodu zaprojektowanie łatwego w użyciu narzędzia zapytań do dokumentów XML nie jest łatwym zadaniem do wykonania. Z tego wynika istnienie tak wielu narzędzi, które próbują sprostać tym oczekiwaniom, począwszy od niskopoziomowych parserów iteracyjnych jak SAX [5], przez narzędzia zapytań XPath i XQuery [6] czy też dostarczana w ramach produktu Microsoft SQL Server metoda OPENXML() [7], a skończywszy na wysokopoziomowych narzędziach mapowania na klasy w rodzaju LINQ to XML [8]. Każde z tych narzędzi zapewnia wiele funkcjonalności. Niestety, najczęściej ignoruje przy tym potrzebę wprowadzenia innych. Z tego powodu powstała koncepcja przetwarzania dokumentu XML za pomocą transformacji do sprawdzonego już w informatyce modelu relacyjnego.

3. Przetwarzanie danych XML na podstawie modelu quasi-relacyjnego

3.1. Podejście relacyjne

Model relacyjny został przedstawiony po raz pierwszy w 1970 roku przez Edgara F. Codda w [9] i doczekał się kilku implementacji, z których najpopularniejszą (chociaż tylko przybliżoną) są bazy danych SQL. Tą nazwą określa się całą rodzinę produktów baz danych korzystających z SQL, jako języka do definiowania i manipulacji danymi.

Model relacyjny i język SQL są de facto standardami w głównym nurcie technologii informatycznych w zakresie przechowywania i przetwarzania danych, docenionymi przez tysiące programistów na całym świecie. Składnia SQL jest naturalna i łatwa do nauczenia, a jednocześnie język pozostaje potężnym narzędziem do konstruowania czasem bardzo skomplikowanych zapytań.

3.2. Język SQLxD

SQLxD [10] oznacza *SQL for XML Data*, czyli SQL dla danych XML. Mówiąc bardziej szczegółowo: SQLxD jest językiem zapytań do dokumentów XML, którego składnia ma w maksymalnie możliwym stopniu przypominać tę znaną z ANSI-SQL. SQLxD jest jednocześnie silnikiem wykonywania tych zapytań, który „w locie" przekształca węzły XML do modelu relacyjnego i odpytuje powstały model tak, jakby była to zwykła, tradycyjna baza danych. Co więcej, przekształcenie struktury XML w model „płaskiej" tabeli nie powoduje utraty istoty dokumentu XML – hierarchicznej relacji między węzłami. SQLxD wykorzystuje te relacje udostępniając nową funkcjonalność, nazwaną złączeniem naturalnym (NATURAL JOIN). W koncepcji rozwiązania możliwe jest również odpytywanie o obiekty XML modelu DOM oraz tworzenie nowych dokumentów XML na podstawie istniejących.

3.3. Transformacja danych XML do modelu relacyjnego

Podstawowym zadaniem silnika SQLxD jest pozwolenie użytkownikowi na mapowanie hierarchicznych danych dokumentu XML na pewien model relacyjny. Model ten tutaj jest w znaczeniu czysto wirtualnym – jest on tworzony na żądanie, w trakcie wykonywania zapytania. Dla konstrukcji tego modelu przyjęto, że w pewnym sensie wszystkie węzły XML o tej samej nazwie i położeniu w hierarchii dokumentów będą opisywały podobne encje, a zatem będą w jakiś sposób dzieliły wspólny schemat węzłów-dzieci oraz atrybutów. To założenie stanowi podstawę projektu języka i jednocześnie podstawową zasadę transformacji do modelu relacyjnego. Oczekiwaniem wobec języka jest w takim razie umożliwienie użytkownikowi wskazania, które węzły będą stanowić podstawę relacji (tabeli), którą chce stworzyć. Funkcjonalność jest

zapewniona przez proste adresowanie w postaci `root.node.node.node`... Możliwe jest również używanie masek w postaci `?` (dokładnie jeden węzeł o dowolnej nazwie) lub `*` (zero lub więcej węzłów o dowolnej nazwie). Bardziej rozbudowane adresowanie z filtrowaniem (jak np. w XPath) nie jest konieczne – zadania te przejmie znany z SQL rachunek zbiorów i predykatów.

Rozważmy następujący dokument XML:

```
<?xml version="1.0" encoding="utf-8"?>
<people>
   <person id="34">
      <firstname>John</firstname>
      <lastname>Toster</lastname>
      <age>32</age>
      <address>
         <city>Atlanta</city>
         <street>Plain Valley</street>
         <house type="apartment">
            <building>34</building>
            <flat>12</flat>
         </house>
      </address>
      <interests>
         <interest>dogs</interest>
         <interest>cats</interest>
         <interest>parrots</interest>
      </interests>
   </person>
</people>
```

Użytkownik zleca utworzenie relacji (tabeli) na podstawie powyższego dokumentu poprzez wskazanie interesujących go węzłów dokumentu. Na przykład wyrażenie `people.person.interests.interest` wskaże trzy elementy dokumentu o nazwie „interest", leżące w odpowiednim miejscu hierarchi. To wyrażenie stanowi podstawę konstrukcji klauzuli FROM, drugim elementem konstrukcji jest alias, nadający utworzonej relacji tymczasową nazwę w modelu relacyjnym.

Węzły dokumentu wskazane przez wyrażenie stanowić będą odpowiedniki wierszy tabeli w modelu relacyjnym. Odpowiednikami zaś kolumn są wszystkie „bezpośrednie i pierwsze" węzły-dzieci danego węzła – włączając w to atrybuty, węzły tekstowe i węzły potomne. „Bezpośrednie i pierwsze" oznacza tutaj tyle, że jeżeli istnieją dwa węzły dzieci tego samego typu lub z tą samą nazwą, to tylko pierwszy z nich w kolejności dokumentu XML jest traktowany jako kolumna. Pozostałe węzły są na tym etapie przetwarzania ignorowane.

Oczywiście zestaw kolumn jest rozszerzany rekurencyjnie na „bezpośrednie i pierwsze" węzły-dzieci tych już zawartych w zbiorze kolumn. Pozwala to adresować dowolne węzły potomne za pomocą adresowania z kropkami: column.child.child.child.

3.3.1. Przykładowe zapytanie

Aby uczynić powyższe wprowadzenie nieco jaśniejszym, na przykładowym dokumencie wykonamy pierwsze zapytanie w języku SQLxD. Zapytanie to pobierze kilka informacji o pojedynczej osobie (tu dla prostoty dokumentu jest tylko jedna):

```
SELECT
   person.#id,
   person.firstname,
   person.lastname,
   person.age,
   person.interests.interest AS firstinterest
FROM
   people.person AS person
```

Otrzymany rezultat powinien wyglądać tak:

```
person.#id person.firstname person.lastname person.age firstinterest
========== ================ =============== ========== ==================
34         John             Toster            32       dogs
```

Należy zwrócić uwagę, że zgodnie z tym, co zostało wcześniej powiedziane (zasada „bezpośredni i pierwszy"), tylko „dogs" zostało zwrócone jako wartość kolumny adresowanej frazą `person.interests.interest`. Jakkolwiek w tym momencie może się to wydawać ograniczeniem, to jest ono usuwane poprzez przedstawione poniżej złączenia naturalne, które idealnie wpasowują się w założenia relacyjnej transformacji dokumentu.

3.3.2. Rozróżnianie: węzeł jako wiersz vs. węzeł jako kolumna

Wcześniejsze przykłady mogą być nieco enigmatyczne dla rozróżnienia, kiedy właściwie węzeł XML jest tabelą, wierszem, a kiedy kolumną. Klucz do zrozumienia SQLxD leży właśnie tutaj: węzeł dokumentu jest tym, czym programista chce, aby ten węzeł był w kontekście konkretnego zapytania. Następny przykład pokaże dokument badany przez dwa różne zapytania na dwóch rożnych poziomach szczegółowości:

```
<?xml version="1.0" encoding="utf-8"?>
<document>
   <child>
      <column1>Text data</column1>
      <column2>Other text data</column2>
      <column3>Additional text data</column3>
   </child>
   <child>
      <column1>Second child data</column1>
      <column2>More text data</column2>
      <column3>Another piece of text</column3>
   </child>
</document>
```

Odpytamy najpierw o węzły poziomu „document" za pomocą następującego zapytania:

```
SELECT
   document.child.column1,
   document.child.column2
FROM
   document AS document
```

To zapytanie traktuje główny węzeł dokumentu o nazwie „document" jako punkt utworzenia wiersza tabeli – skutkuje to wirtualną tabelą o pojedyczym wierszu. Ponieważ tylko

pierwszy w kolejności węzeł-dziecko o nazwie „child” jest dostępny jako węzeł-kolumna, następujące dane są zwracane jako rezultat zapytania:

```
document.child.column1   document.child.column2
======================   ======================
Text data                Other text data
```

Jeżeli z kolei węzły adresowane frazą „document.child” mają stanowić punkt utworzenia wierszy tabeli, tworzona jest dwuwierszowa tabela z węzłami „column1”, „column2”, „column3” jako kolumnami odpowiednich wierszy:

```
SELECT
   child.column1,
   child.column2
FROM
   document.child AS child

child.column1       child.column2
=================   ===============
Text data           Other text data
Second child data   More text data
```

3.3.3. Złączenia naturalne

W dokumentach XML podawana jest istotna informacja zakodowana „między węzłami”. Ta informacja nie jest przedstawiona jawnie poprzez wartości atrybutów czy tekst, ale jest wyrażona poprzez relację rodzic-dziecko, które występuje między węzłami dokumentu. SQLxD, w przeciwieństwie do innych technik przetwarzania XML, intensywnie wykorzystuje tę informację. Wirtualna relacja pomiędzy wirtualnymi tabelami jest tworzona poprzez wyrażenie naturalnego złączenia. Mimo iż nie istnieje stricte wyrażona relacja klucza głównego/obcego, złączenie naturalne jest wykonywane na podstawie hierarchii dokumentu, sprowadzając w ten sposób drzewiastą strukturę z powrotem do struktury tablicy-tablic-obiektów – tak naturalną w wielu współcześnie używanych językach programowania.

Niech kolejny dokument posłuży za przykład do tego zagadnienia:

```
<?xml version="1.0" encoding="utf-8"?>
<document>
   <child id="1">
      <info>Text data</info>
      <info>Other text data</info>
      <info>Additional text data</info>
   </child>
   <child id="2">
      <info>Second child data</info>
      <info>More text data</info>
      <info>Another piece of text</info>
   </child>
</document>
```

Składnia złączenia naturalnego jest prosta, nie wymaga żadnych operatorów logicznych ani precyzyjnego mapowania danych-na-kolumny. To świadczy o sile ekspresji SQLxD. Wiersz

tabeli nadrzędnej staje się podstawą wirtualnego poddokumentu. Ten dokument jest w dalszej kolejności odpytywany o swoje węzły w zwykły sposób. Rozważmy następujące zapytanie:

```
SELECT
   child.#id,
   child.info,
   info.#
FROM
   document.child AS child
      NATURAL JOIN child.info AS info
```

```
child.#id  child.info         info.#
=========  =================  ====================
1          Text data          Text data
1          Text data          Other text data
1          Text data          Additional text data
2          Second child data  Second child data
2          Second child data  More text data
2          Second child data  Another piece of text
```

Pokazano tutaj wyraźnie jak kolumna jednej tabeli może być wierszem innej tabeli. Wiązanie między dwiema tabelami jest wykonywane, dając w konsekwencji rezultat podobny do tego, dobrze znanego ze środowisk bazodanowych, opartych na SQL.

Aby dobitniej pokazać możliwości złączeń naturalnych, wróćmy na chwilę do dokumentu „people", przedstawionego w sekcji 3.3 na i informacji o pozostałych zainteresowaniach danej osoby, które wówczas gubiliśmy. Teraz takie przepisanie zapytania, aby ta dodatkowa informacja wracała na swoje miejsce powinno być łatwe. Należy przy tym zauważyć, że zwracany rezultat pozostaje w zgodności z modelem relacyjnym. Każdy zwrócony wiersz reprezentuje pojedyncze zainteresowanie osoby.

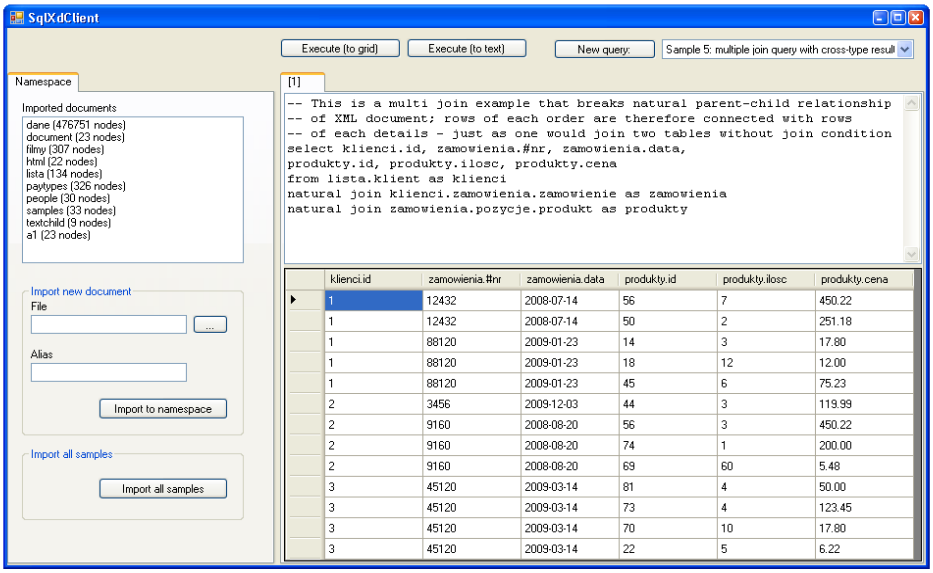
```
SELECT
   person.#id,
   person.firstname,
   interest
FROM
   people.person AS person
      NATURAL JOIN person.interests.interest AS interest
```

```
person.#id   person.firstname   interest
==========   ================   ========
34           John               dogs
34           John               cats
34           John               parrots
```

Przedstawione powyżej przykłady stanowią oczywiście zaledwie zalążek funkcjonalności SQLxD. Należy jednak podkreślić, że jednocześnie przedstawiono wszystko, co programiście niezbędne jest do niemal tak szybkiej pracy z dokumentami XML, jak z klasycznymi bazami danych. Reszta funkcjonalności języka jest bowiem przeniesiona bezpośrednio z języka rachunku zbiorów i predykatów, jakim jest SQL: złączenia wewnętrzne i zewnętrzne, wyrażenia na kolumnach, filtrowanie, sortowanie czy agregacja danych korzystają z danych relacyjnych, skonstruowanych przez instrukcje SQLxD.

4. Prototyp narzędzia do przetwarzania zapytań SQLxD

W ramach prac nad językiem SQLxD przygotowany został prototyp silnika przetwarzania zapytań. Prototyp ten ma stanowić tzw. dowód poprawności (ang. *proof of concept*), przedstawionej w artykule koncepcji transformacji danych z modelu hierarchicznego XML do relacyjnego modelu danych. Z punktu widzenia architektury prototyp został podzielony na moduły: SQLxDLib – dynamicznie dołączaną biblioteką klas, eksponującą interfejs służący do wykonywania zapytań oraz SQLxDClient – prostą aplikacją okienkową, korzystającą z biblioteki SQLxDLib. Aplikacja pozwala zaimportować dokumenty XML do przestrzeni nazw, wykonywać zapytania i obserwować ich rezultaty. Przykład ekranu aplikacji SQLxDClient został pokazany na rys. 1. Do aplikacji dołączono kilka demonstracyjnych plików XML i opisanych przykładów zapytań.



Rys. 1. Okno aplikacji SQLxDClient
Fig. 1. Screenshot of SQLxDClient application

Istnieje oczywiście możliwość korzystania wyłącznie z biblioteki SQLxDLib. Przykładowy kod C# korzystający z biblioteki mógłby wyglądać następująco:

```
using System;
using System.Collections.Generic;
using SqlXdLib;
using SqlXdLib.Queries;

class Program  {
   static void Main(string[] args)  {
      Engine engine = new Engine();
      engine.Import("zamowienia.xml", "zamowienia");
      Query query = engine.Parse(
         "SELECT klienci.id FROM zamowienia.klient as klienci");
      List<object[]> result = engine.Execute(query);
   }
}
```

5. Podsumowanie

W artykule przedstawiono nową, autorską koncepcji sposobu przetwarzania dokumentów XML, które w zamierzeniu ma być łatwiejsze w użytkowaniu od istniejących metod. Na koncepcję tę składają się model transformacji danych hierarchicznych do modelu relacyjnego oraz projekt autorskiego języka SQLxD – narzędzia do wykonywania takiego przekształcenia.

Budowa dokumentów XML, chociaż funkcjonalna, często sprawia wiele trudności w zakresie jej przetwarzania. Pomimo istnienia wielu metod i narzędzi nie jest ono wygodne ani szybkie. Z tego powodu powstała koncepcja prostej metody transformacji danych XML do modelu relacyjnego. W metodzie przyjęto założenie, że cały proces transformacji powinien sprowadzić się do wskazania węzłów dokumentu o podobnym znaczeniu *semantycznym* oraz utworzenia na podstawie tych węzłów *relacji*. Kolumnami relacji stają się atrybuty, dane tekstowe, elementy potomne – czyli wszystko to, co niesie ze sobą informacje opisujące wyjściowy, wskazany węzeł.

To założenie było punktem wyjścia do zaproponowania SQLxD – języka o składni maksymalnie zbliżonej do SQL – czyli narzędzia najczęściej używanego do przetwarzania danych w bazach relacyjnych. Niegasnąca popularność, duża liczba korzystających z niego programistów, wysoka czytelność składni oraz możliwości ekspresji sprawiają, że SQL może stanowić dobrą podstawę do konstrukcji nowego języka przetwarzania dokumentów XML. Jeżeli bowiem dane hierarchiczne można prostą metodą sprowadzić do modelu relacyjnego – nic nie stoi na przeszkodzie, aby dane te przetwarzać dalej tak, jak zwykłe dane relacyjne i za pomocą sprawdzonych rozwiązań. W ten sposób instrukcje złączeń, filtrowania, przekształcania czy sortowana danych zachowują formę deklaratywną. Prostota zasad transformacji sprawia, że może być ona wykonywana w sposób *dynamiczny*, bez jakiejkolwiek komplikacji składni języka zapytań. Jednocześnie zaproponowany rodzaj złączenia, nazwany *naturalnym*, za pomocą prostych instrukcji pozwala w pełni wykorzystać zalety wielokrotnie zagnieżdżonej struktury danych przenoszonych przez dokument XML. Wydaje się, że opracowana metoda spełnia dostatecznie dobrze zadania, które są stawiane przed podobnymi narzędziami. Zapewnia bowiem możliwość łatwego i szybkiego przetwarzania dokumentów XML, przy pełnym utrzymaniu informacji przenoszonych zarówno przez dane, jak i przez ich strukturę. Jest zatem na pewno ciekawą alternatywą dla funkcjonujących współcześnie rozwiązań.

BIBLIOGRAFIA

1. Bert B.: The XML data model. W3C, 1997.
2. Pardede E.: Open and Novel Issues in XML Database Applications: Future Directions and Advanced Technologies. Information Science Reference, 2009.

3. Harold E. R., Means W. S.: XML in a Nutshell, O'Reilly, 2004.
4. Melton J., Buxton S.: Quering XML. Morgan Kaufmann, 2006.
5. Brownell D.: SAX2. O'Reilly, 2002.
6. Walmsley P.: XQuery. O'Reilly, 2007.
7. Pal S., Fussell M., Dolobowsky I.: XML Support in Microsoft SQL Server, Microsoft Corporation, 2005.
8. Calvert C., Kulkarni D.: Essential LINQ. Addison-Wesley Professional, 2009.
9. Codd E. F.: A Relational Model of Data for Large Shared Data Banks. IBM, 1970.
10. Wyrostek J.: Przetwarzanie dokumentów XML w oparciu o model quasi-relacyjny wraz z projektem języka zapytań. Praca dyplomowa pod kierunkiem R. Marcjana; Katedra Informatyki AGH; Kraków, 2010

Recenzenci: Dr inż. Michał Kozielski
Prof. dr hab. inż. Mieczysław Muraszkiewicz

Wpłynęło do Redakcji 31 stycznia 2011 r.

Abstract

The XML documents provide the organization of information broadly used in the modern computer science, application development and data management systems. The diversity of available techniques and tools used for the XML document querying and processing, shows that providing the convenient service to that popular format is not an easy task. The authors propose to use the quasi-relational model as a basis to XML data processing. The XML data is transferred to the relational model according to the intuitive rules that make use of syntax and the structure of the source document. Part of the solution is the query language called SQLxD, which is based on the popular SQL syntax. SQLxD is a tool for both data transformation and its further processing. It provides the simplicity and efficiency in XML data processing, maintaining the transmission of information both by the data and their structure.

Adresy

Robert MARCJAN: Akademia Górniczo-Hutnicza, Katedra Informatyki,
al. Mickiewicza 30, 30-059 Kraków, Polska, marcjan@agh.edu.pl.
Jakub WYROSTEK: Akademia Górniczo-Hutnicza, Katedra Informatyki,
al. Mickiewicza 30, 30-059 Kraków, Polska, kuba@wyrostek.pl.