КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 004.318

А.В. ПАЛАГИН, Н.Г. ПЕТРЕНКО, А.О. СЕВРУК

ОБ ОДНОМ ПОДХОДЕ К ФОРМАЛИЗОВАННОМУ ПРЕДСТАВЛЕНИЮ ОНТОЛОГИИ ТЕКСТОВОГО ДОКУМЕНТА

При анализе лингвистических корпусов средствами интеллектуальных информационных систем целесообразно хранить обработанный документ в виде его онтологии, что позволит включить его в общую базу данных для последующего анализа, построения схемы зависимостей между документами, упрощения анализа больших объемов информации и повышения релевантности поиска информации при запросе пользователя системы. Данная задача является комплексной. В настоящей работе освещена ее важная часть по представлению онтологий в реляционной системе управления базами данных (РСУБД) и правила выборки части онтологии из БД в соответствии с определенными критериями, для использования в прикладных задачах пользователей.

Для анализа информации на больших массивах данных целесообразно использовать знание-ориентированную поисковую систему (ЗнПС) [1], задачу которой можно представить состоящей из трех фрагментов:

- построение онтологии отдельного текстового документа;
- интеграция построенной онтологии в онтологию предметной области (ОПО);
- последующий поиск документов по запросу пользователя и развитие самой системы.

Следует отметить, что работы, связанные с автоматическим анализом текстов, требуют определенного набора лингвистических средств анализа, основу которых составляют машинные словари, программы морфологи-

A.V. Palagin, N.G. Petrenko
A.O. Sevruk

ABOUT ONE APPROACH TO FORMALIZED REPREZENTATION ONTOLOGY OF THE TEXT DOCUMENT

The ontology in the form of a binary tree, its storage in RDBMS and samples from the database, in accordance with the branches of the binary tree, what solve the issues with organization of database for the storage the building ontologies and the next this ontologies processing in accordance with certain criteria user is considered.

Рассмотрено представление онтологии в виде бинарного дерева, его хранение в РСУБД и проведение выборок из базы данных в соответствии с ветвями бинарного дерева, что позволяет решить вопросы организации БД для хранения построенных онтологий и последующей обработки данных онтологий в соответствии с определенными критериями пользователя.

ческого, синтаксического и семантического анализа, выделения терминологической лексики и т.д. Для ЗнПС указанные средства предоставляет языково-онтологическая информационная система (ЯОИС) [2]. Кроме лингвистических ресурсов, для анализа естественно-языковых текстов необходимы базы данных внешней информации, которая должна исходить от экспертов предметной области. В данном случае инструментом взаимосвязи между экспертами предметной области и программными моделями анализа естественно-языкового текста (ЕЯТ) является ОПО.

Онтология предметной области – специализированная база знаний некоторого домена, представляющая ориентированный граф, узлами которого являются фреймы с именем соответствующего концепта, а дугами – соответствующие концептуальные отношения между концептами.

Будем опираться на общеизвестные концептуальные положения построения онтологии, которые выступают как требования к ее структурной организации [3, 4].

1. Понятия в онтологии отражают физические или логические объекты и отношения в рассматриваемой предметной области, формируют таксономическую структуру, т. е. естественную иерархию классов и их элементов (подклассов).

2. Классы представляют понятия предметной области (а не термины, обозначающие эти понятия), поэтому синонимы одного и того же понятия не представляют различные классы.

3. Классы, которые являются прямыми подклассами одного и того же класса в иерархии (кроме тех, что находятся в корне), должны располагаться на одном уровне.

4. Класс может быть подклассом нескольких классов.

5. Если понятия с разными значениями слота* становятся ограничениями для различных слотов в других классах, то для их разделения следует создать новый класс. В противном случае разделение представляется в значении слота.

6. Если в предметной области разграничение понятий представляется важным, и объекты с другими значениями разграничения относятся к другим типам объектов, то для такого разграничения целесообразно создать новый класс.

7. Для формализации представления иерархии классов целесообразно использовать типовые отношения.

Приведем рекомендации, которых следует придерживаться при построении онтологии предметной области [5]:

- процедура перехода между соседними уровнями должна быть соразмерной. Это означает, что объединение объемов или соединение значений концептов нижнего уровня должно составить концепт верхнего уровня;

- процедура перехода должна осуществляться по одному основанию, т. е. характеристика, выбираемая в качестве основания деления, в ходе деления не должна подменяться другой характеристикой;

* Использовано определение фреймового представления знаний.

- компоненты делимого концепта должны исключать друг друга. Это означает, что их объемы не должны содержать общих элементов или их значения не должны иметь общих частей;

- процедура перехода должна быть последовательной. Это означает, что от родового понятия необходимо переходить к видовым понятиям одного и того же уровня, а от целого к частям, частям частей и т.д.

Одной из задач анализа ЕЯТ является построение его онтологии в виде формализованных структур данных и правил их обработки. При этом каждая построенная онтология принимает вид [4]: $O = < X, R, F >$, где X – конечное множество концептов (терминов, понятий, квантов знаний) заданной предметной области; $X \neq Ø$; R – конечное множество отношений между концептами; F – множество функций интерпретации, заданных на концептах и/или отношениях.

Такая модель представления онтологии дает возможность анализировать знания в примитивах нижнего уровня и организовывать последние в структуры высшего уровня. В ЕЯТ основной единицей является слово, а основной структурой – предложение. Для классификации и организации знаний используются такие структуры высшего уровня как параграфы, разделы, главы и тома.

При анализе текста средствами ЗнПС и ЯОИС происходит его разбиение на отдельные логически связанные блоки, средним звеном которого является абзац, а низшим – предложение. Анализируя каждое предложение определенного абзаца, можно построить модель данного предложения в виде фрейма. Такое представление удобно тем, что строго декларативная информация во фрейме может быть оттранслирована в логику первого порядка [6], что позволит проводить дальнейший анализ при интеграции представлений отдельных предложений в более общие структуры. Обработанная текстовая информация, а также ее представление в виде фреймов заносятся в базу данных и используются для обновления и изменения начальной ОПО. Машина логического вывода, выполняющая данные функции, реализована как программная надстройка над базой данных ЗнПС.

Для наполнения баз данных, составления правил для машины логического вывода и обмена данными между системами ЗнПС и ЯОИС в последнее время широко применяется язык представления знаний KIF (Knowledge Interchange Format), который можно использовать как расширение синтаксиса языка программирования Lisp.

Приведем пример представления простого предложения в виде Lisp-программы с поддержкой представления данных в виде KIF. Для этого используем условное предложение: «*ЗнПС включает в себя базу данных и лингвистический процессор*».

Сначала построим строчную запись концептуального графа (КГ) [6] для данного предложения в следующем виде:

Object:ЗнПС <----- Включает в себя ----->Attr -----> база данных
-----> лингвистический процессор

Концептуальные графы - хорошо читаемый формальный язык, который с успехом используется программистами, системными аналитиками и другими профессионалами. Так как концептуальные графы были первоначально разработаны как семантическое и графическое представление высказываний естественного языка, они могут помочь сформировать мост между машинными и естественными языками. Переведем данный КГ в KIF:

```
1(unless (find-package "[KiF]" )
2  (defpackage [KiF]
3  ( :use :common-lisp)))
4(in-package [KiF])
5(defvar ЗнПС )
6(export '(ЗнПС – знание-ориентированная поисковая сис-
тема))
7(setq ЗнПС
8  `(   :name "ЗнПС"
9       :attr  (list  '(база  данных)  '(лингвистический
процессор))))
10(getf [KiF]:ЗнПС:name) ; -> [kif]::ЗнПС
11(getf  [KiF]:ЗнПС:attr)  ;  ->  [kif]::(база  данных)
(лингвистический процессор)
12(setf (getf [KiF]:ЗнПС:name) '(ЗнПС#1))
13(getf [KiF]:ЗнПС:name) ; -> ЗнПС#1
```

Данный листинг демонстрирует возможность представления предложения в KIF-формате с возможностью последующей его обработки средствами языка программирования Lisp, который предоставляет развитый набор средств для анализа полученных KIF-структур и их интеграции между собой [6].

Наряду с задачей построения онтологии текстового документа возникает необходимость поиска определенной ветви онтологии в базе данных. При представлении онтологии в виде KIF-структур вместе с дополнительными программами-процедурами на языке Lisp возникают затруднения в использовании традиционных реляционных БД. Возможно, наиболее рациональным подходом может оказаться организация отдельной БД для хранения только «образа» онтологии: $O = < X,R >$, т. е. усечения онтологии до содержания исключительно названий объектов и отношений между ними, которые определяются посредством ОПО. В данном случае онтологию можно представить в виде дерева с произвольным ветвлением, которое легко можно преобразовать в двоичное дерево. Такое представление упростит проведение выборки определенной ветви дерева. При использовании данной БД возможно использование языка доступа к базам данных SQL для выбора интересующей ветви онтологии.

Двоичные деревья поиска позволяют выполнять следующие операции с динамическими множествами [7]: Search (Поиск), Minimum (Минимум), Maximum (Максимум), Predecessor (Предыдущий), Successor (Следующий), Insert (Вста-

вить), Delete (Удалить). Время выполнения основных операций пропорционально «высоте» дерева. Если все уровни в двоичном дереве имеют максимальное число вершин, то его «высота» (и, соответственно, время выполнения операций) пропорциональна логарифму числа вершин.

В двоичном дереве поиска каждая вершина может иметь (или не иметь) «левого и правого ребенка»; каждая вершина, кроме корня, имеет «родителя». При представлении с использованием указателей, помимо ключа `key` и дополнительных данных, также хранятся и указатели `left, right, p` ("левый ребенок", "правый ребенок", "родитель"). Если "ребенка" (или "родителя" – для корня) нет, соответствующее поле содержит значение `NIL`.

Ключи в двоичном дереве поиска хранятся с соблюдением свойства упорядоченности: пусть `x` – произвольная вершина двоичного дерева поиска. Если вершина `y` находится в левом поддереве вершины `x`, то `key[y] ≤ key[x]`. Если у находится в правом поддереве `x`, то `key[y] ≥ key[x]`.

Процедуру поиска можно представить алгоритмом вида

```
Tree-Serch(x,k)
1 while x ≠ NIL and k ≠ key[x]
2   do if k < key[x]
3        then x <- left[x]
4        else x <- right[x]
5 return x
```

Предложим демонстрационный пример представления упрощенной онтологии некоторого домена предметной области "Операционные системы" (рис. 1), в виде бинарного дерева и осуществления процедуры выборки определенной ветви данной онтологии.

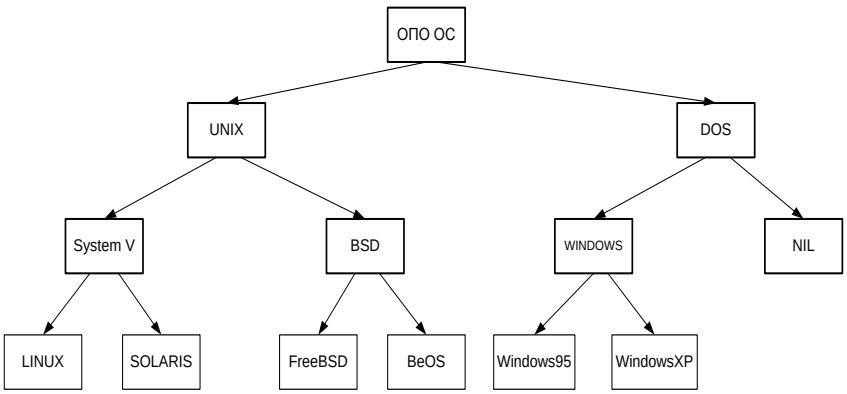

РИС. 1. Онтология некоторого домена предметной области «операционные системы»

Составим таблицу в СУБД mySQL для хранения элементов онтологии:

```
mysql> create table base (
    -> id           int,
    -> name_hub     varchar(15),
    -> parent  varchar(15),
    -> child_1      varchar(15),
    -> child_2      varchar(15),
    -> key     numb  int);
```

Заполним таблицу согласно ОПО ОС, после чего она примет вид

```
mysql> select * from base;
```

```
+------+-------------+-------------+-----------+-----------+---------+
| id   | name_hub    | parent      | child_1   | child_2   | keynumb |
+------+-------------+-------------+-----------+-----------+---------+
|    1 | ОПО ОС      | NULL        | UNIX      | DOS       |      10 |
|    2 | UNIX        | ОПО ОС      | System V  | BSD       |       5 |
|    3 | DOS         | ОПО ОС      | Windows   | NULL      |      15 |
|    4 | System V    | UNIX        | Linux     | Solaris   |       2 |
|    5 | BSD         | UNIX        | FreeBSD   | BeOS      |       7 |
|    6 | Windows     | DOS         | Windows95 | WindowsNT |      12 |
|    7 | Linux       | System V    | NULL      | NULL      |       1 |
|    8 | Solaris     | System V    | NULL      | NULL      |       3 |
|    9 | FreeBSD     | BSD         | NULL      | NULL      |       6 |
|   10 | BeOS        | BSD         | NULL      | NULL      |       8 |
|   11 | Windows 95  | Windows     | NULL      | NULL      |      11 |
|   12 | Windows NT  | Windows     | NULL      | NULL      |      14 |
+------+-------------+-------------+-----------+-----------+---------+
```

Поле keynumb позволяет представить данную таблицу в виде бинарного дерева. Выборка отдельной ветви при данной организации является тривиальной операцией, требующей одного прохода по базе данных. Например, выделим ветвь с названием узла 'UNIX':

```
mysql> select *
    -> from base
    -> where keynumb >= 1
    -> and keynumb < 10
    -> order by id;
```

```
+------+----------+------------+----------+---------+---------+
| id   | name_hub | parent     | child_1  | child_2 | keynumb |
+------+----------+------------+----------+---------+---------+
|    2 | UNIX     | ОПО ОС     | System V | BSD     |       5 |
|    4 | System V | UNIX       | Linux    | Solaris |       2 |
|    5 | BSD      | UNIX       | FreeBSD  | BeOS    |       7 |
|    7 | Linux    | System V   | NULL     | NULL    |       1 |
|    8 | Solaris  | System V   | NULL     | NULL    |       3 |
|    9 | FreeBSD  | BSD        | NULL     | NULL    |       6 |
|   10 | BeOS     | BSD        | NULL     | NULL    |       8 |
```

Данную выборку из таблицы можно представить в виде ветви бинарного дерева (рис. 2). Расположение и зависимости элементов друг от друга содержатся в поле keynumb:

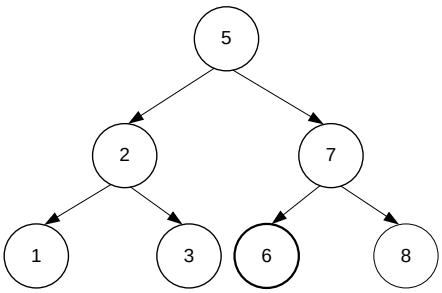

РИС. 2. Бинарное дерево для представления ветви “UNIX”

Заключение. В данной статье рассмотрены подход к формализованному представлению онтологии текстового документа, а также возможность представления текстовой информации при помощи языка обмена знаниями KIF. Данная задача является комплексной, и в настоящей работе сделан акцент на обработке онтологий с помощью РСУБД. Показан пример представления онтологии в виде бинарного дерева, его хранение в РСУБД и проведение выборок в соответствии с ветвями бинарного дерева, что позволяет решить вопросы с организацией БД для хранения построенных онтологий в приложениях пользователя для последующей обработки данных онтологий в соответствии с заданными критериями пользователя.

1. *Севрук О.О., Петренко М.Г.* Знання-орієнтована пошукова система на основі мовно-онтологічної картини світу // Тези доп. XIII міжнар. конф. з автоматичного управління “Автоматика-2006”, Вінниця, 25 – 28 вересня, 2006. – С. 413.
2. *Палагін О.В., Петренко М.Г.* Модель категоріального рівня мовно-онтологічної картини світу // Математичні машини і системи. – 2006. – № 3. – С. 91 – 104.
3. *Палагин А.В., Яковлев Ю.С.* Построение онтологии предметной области «Интеллектуальные информационные системы» // УСиМ. – 2005. – № 6. – С. 18 – 27.
4. *Палагин А.В., Яковлев Ю.С.* Системная интеграция средств компьютерной техники. – Винница: «УНІВЕРСУМ-Вінниця», 2005. – 680 с.
5. *Ивлев Ю.В.* Логика: Учебник для вузов. – М.: Логос, 1997. – 272 с.
6. *Sowa John F.* Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, ©2000. 513 p.
7. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. – М.: МСНМО, 1999. – 960 с.

Получено 09.04.2007