OSTIS-2013
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

РЕАЛИЗАЦИЯ ХРАНИЛИЩА УНИФИЦИРОВАННЫХ СЕМАНТИЧЕСКИХ СЕТЕЙ

Корончик Д. Н.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

denis.koronchik@gmail.com

В работе описана одна из возможных реализаций хранилища унифицированных семантических сетей. Приводится описание принципов хранения элементов семантической сети и связей между этими элементами. Описанное хранилище может быть использовано для работы различных интеллектуальных систем в среде интернет.

Ключевые слова: графовые базы данных, унифицированные семантические сети, хранение семантических сетей.

Введение

Процесс проектирования интеллектуальной системы, с помощью технологии OSTIS **[OSTIS, 2012]**, сводится к двум этапам: разработке логико-семантической модели (sc-модели) проектируемой системы и разработке интерпретатора этой модели. Может существовать достаточно большое количество интерпретаторов sc-моделей (в том числе и для одной платформы), они могут отличаться по функционалу, быстродействию и другим критериям. Все они входят в библиотеку компонентов и могут быть использованы многократно. В настоящее время ведутся работы по разработке интерпретатора sc-моделей, который ориентирован на работу в среде Интернет. Он состоит из следующих подсистем:

- программная реализация sc-хранилища (ориентирована на хранение sc-графов **[Голенков, 2012]**);
- программная реализация языка SCP (язык программирования ориентированный на обработку sc-графов **[Голенков и др, 2001]**);
- программная реализация компонентов ядра внешних интерфейсов (набор программных средств ориентированных на взаимодействие системы с внешней средой **[Корончик, 2012]**).

В данной работе более детально будут рассмотрены принципы, которые лежат в основе программной реализации sc-хранилища, так как оно является базовым компонентом всего интерпретатора и взаимодействие всех подсистем осуществляется именно с его помощью.

Под sc-хранилищем понимается информационная подсистема, предназначенная для хранения sc-графов (тексты, записанные с помощью SC-кода, любой sc-граф представляет собой семантическую сеть). Основными функция sc-хранилища являются:

- организация хранения sc-графов;
- предоставление программного интерфейса для добавления, удаления и извлечения хранимой информации (ассоциативный доступ);
- поддержка функций администрирования хранимой информации (распределение прав доступа);
- поддержка асинхронного выполнения запросов через программный интерфейс.

В настоящее время разрабатывается большое количество программных систем, которые также решают проблему хранения информации в виде графов. Это так называемые графовые базы данных **[graph-database, 2012]**. Наиболее популярные среди них Neo4j **[Neo4j, 2012]**, HyperGraphDB **[HyperGraphDB, 2012]**, BigData **[bigdata, 2012]**. Конечно можно использовать их для хранения sc-графов, но так как производительность sc-хранилища является важнейшим фактором влияющим на производительность системы в целом (так как все подсистемы взаимодействуют между собой через него), то указанные выше системы не достаточно эффективны, так как они изначально не учитывают специфику хранимого sc-графа.

1. Описание sc-хранилища

Так как sc-хранилище является частью интерпретатора sc-моделей ориентированного на работу в среде Интернет, то к нему предъявляются следующие требования:

- высокая производительность – минимизация времени затрачиваемого на выполнение операций добавления, удаления и доступа к хранимой информации;
- минимальные затраты памяти и дискового пространства для хранения sc-текстов;
- масштабируемость – возможность простого добавления вычислительных мощностей при увеличении нагрузки.

За хранение sc-графов в рамках sc-хранилища отвечает программная реализация sc-памяти. Она имеет сегментную адресацию, а адрес хранимого в ней sc-элемента будем называть sc-адресом. Использование сегментной адресации позволяет быстро получать доступ к sc-элементу по его адресу, а также минимизировать проблемы с ограничениями оперативной памяти (сегменты можно сохранять на диск и загружать их по мере надобности). От количества и размера сегментов зависит максимальное количество sc-элементов, которые можно хранить в sc-памяти.

В настоящее время максимальное количество сегментов и максимальное количество элементов, которые можно хранить в одном сегменте, равно 2^{16}. Это позволяет хранить в памяти до 2^{32} элементов. Таким образом, sc-адрес имеет размер 4 байта и состоит из двух частей: номер сегмента (2 байта), смещение в сегменте (2 байта).

Каждая ячейка sc-памяти хранит информацию об одном sc-элементе и имеет структуру, представленную в таблице 1.

Таблица 1 – Структура ячейки sc-памяти

Поле	**Размер**
Тип sc-элемента	2 байта
Временная метка создания	4 байта
Временная метка удаления	4 байта
sc-адрес первого коннектора в списке входящих в элемент коннекторов (для которых описываемый sc-элемент является вторым компонентом)	4 байта
sc-адрес первого коннектора в списке выходящих из элемента коннекторов (для которых описываемый sc-элемент является первым компонентом)	4 байта
Количество элементов в списке выходящих коннекторов	4 байта
Количество элементов в списке входящих коннекторов	4 байта
Дополнительные данные об элементе	33 байта

Как видно из таблицы 1, на хранение одного sc-элемента требуется 59 байт. Рассмотрим поля ячейки более подробно:

- **тип sc-элемента**. Каждому биту этого поля соответствует некоторый тип. Принадлежность к тому или иному типу sc-элемента указывается наличием 1 в соответствующем бите поля. Если же значение поля равно нулю, то ячейка считается пустой. Полный список типов приведен в таблице 2.

Таблица 2 – Таблица битов кодирующих тип sc-элементов

Бит	**Кодируемый тип или признак**
1	sc-узел
2	sc-ссылка
3	sc-ребро
4	sc-дуга общего вида
5	sc-дуга принадлежности
6	sc-константа
7	sc-переменная
8	позитивная sc-дуга / узел, обозначающий n-арную связку
9	негативная sc-дуга / узел, обозначающий структуру
10	нечеткая sc-дуга / узел, обозначающий ролевое отношение
11	нестационарная sc-дуга / узел, обозначающий бинарное, неролевое отношение
12	стационарная sc-дуга / узел, обозначающий понятие, не являющееся отношением
13	узел, обозначающий абстрактный объект
14	узел обозначающий материальный объект

Из таблицы видно, что начиная с восьмого, последующие биты могут кодировать различные признаки. Интерпретация кодируемого признака зависит лишь от того является ли хранимый элемент sc-дугой (первый признак) или же sc-узлом (второй признак). Работа с типами sc-элементов при такой организации их хранения сводится к простым побитовым "*и*" и "*или*".

- **временная метка создания/удаления sc-элемента**. Данные поля используются для обеспечения асинхронного доступа к содержимому sc-памяти. Наличие в этих полях значения 0, означает, что элемент еще не был создан или же еще не удален;
- **sc-адрес первой входящего/выходящего коннектора.** Каждый sc-элемент имеет два списка инцидентности: по входящим и выходящим дугам. В этих полях хранятся sc-адреса первых sc-коннекторов для каждого из списков инцидентности;
- **количество элементов в списке входящих/выходящих коннекторов.** Эти поля используются для оптимизации доступа к элементам. К примеру, если нам надо найти sc-дугу выходящую из элемента *А* в и входящую в элемент *Б*, то нам необходимо просмотреть либо список выходящих из элемента *А* или же список входящих в элемент *Б* sc-коннекторов. Когда длина обоих списков известна, то проход лучше осуществить по наименьшему из них.

Хранение sc-ссылок

SC-ссылки – это sc-элемент, который обозначает либо определенный файл, который можно просматривать или в котором закодирована в определенном формате некоторая внешняя, инородная для SC-кода информационная конструкция, либо некоторую компьютерную систему, с которой можно взаимодействовать **[Голенков, 2012]**. Хранение содержимого sc-ссылки (путь к файлу, путь к ресурсу, адрес

информационной системы, содержимое файла и т. д.) храниться на файловой системе. Важным является то, что множество различных sc-ссылок могут ссылаться на разные файлы, которые имеют одинаковое содержимое. Было бы разумно не хранить содержимое одинаковых файлов дважды. Для этого при создании sc-ссылки и указании файла на который она ссылается, вычисляется hash-сумма содержимого с помощью алгоритма SHA256. В результате получается строка из 32 символов, которая и хранится в поле дополнительных данных для sc-ссылок. Само же содержимое копируется в файл (создаваемый sc-памятью рядом с сохраняемыми сегментами), путь к которому строится на основании hash-суммы. Рядом с этим файлом создается файл, в котором хранятся sc-адреса всех ссылок ссылающихся на файлы с изоморфным содержимым. Таким образом, для того чтобы найти все sc-ссылки ссылающиеся на файлы с указанным содержимым, нам необходимо лишь вычислить hash-сумму искомого образца и проверить наличие файла по пути вычисляемому из hash-суммы и если он существует, то вернуть список хранящихся sc-адресов.

Хранение sc-коннекторов

Сложностью при хранении sc-коннекторов является то, что нельзя предугадать, сколько их будет связано с элементом. Поэтому хранить отдельно для каждого элемента списки инцидентности за пределами сегментов не является разумным. Было решено использовать следующее свойство коннектора: *каждый коннектор всегда имеет начальный и конечный элемент*. Другими словами каждая sc-дуга входит в два списка смежности: выходящих и входящих дуг. На основании этого было решено хранить списки смежности прямо в сегментах, используя поле дополнительных данных для sc-коннекторов (таблица 3).

Таблица 3 – Структура ячейки с дополнительными данными для sc-коннекторов

Поле	**Размер**
sc-адрес элемента являющегося первым компонентом sc-коннектора	4 байта
sc-адрес элемента являющегося вторым элементом sc-коннектора	4 байта
sc-адрес следующего коннектора в списке выходящих коннекторов	4 байта
sc-адрес следующего коннектора в списке входящих коннекторов	4 байта

Как видно из таблицы, каждый sc-коннектор хранит адрес следующего за ним в списке входящих и выходящих sc-коннекторов. Пример хранения списков инцидентности sc-коннекторов представлен на рисунке 1, где справа представлен хранимый граф. Каждый элемент на рисунке пронумерован от 1 до 7, где 1,4,2 и 3-й элементы – это sc-узлы, а 7, 6 и 5-й – sc-дуги. Ячейки, в которых хранятся элементы, изображены в виде небольших таблиц, с номером в первой строке. Под номером изображаются поля ячеек (не все поля элемента, а лишь необходимые для хранения списков инцидентности) и их значения. Поясним эти поля:

- **first_out_arc** – sc-адрес первого коннектора в списке выходящих из указанного элемента коннекторов;
- **first_in_arc** – sc-адрес первого коннектора в списке входящих в указанный элемент коннекторов;
- **begin_addr** – sc-адрес элемента являющегося первым компонентом sc-коннектора;
- **end_addr** – sc-адрес элемента являющегося вторым компонентом sc-коннектора;
- **next_out_arc** – sc-адрес следующего выходящего в некоторый sc-элемент коннектора, (в указанном списке все sc-коннекторы имеют одинаковый первый компонент. Другими словами это список выходящих из некоторого sc-элемента дуг);
- **next_in_arc** – sc-адрес следующего входящего в некоторый sc-элемент, коннектора (в указанном списке все sc-коннекторы имеют одинаковый второй компонент. Другими словами это список дуг, входящих в некоторый sc-элемент)

Стрелками показаны переходы для списков инцидентности по выходящим sc-коннекторам для элементов 1 и 4. Аналогичным образом хранятся списки инцидентности для выходящих sc-коннекторов (дуг).

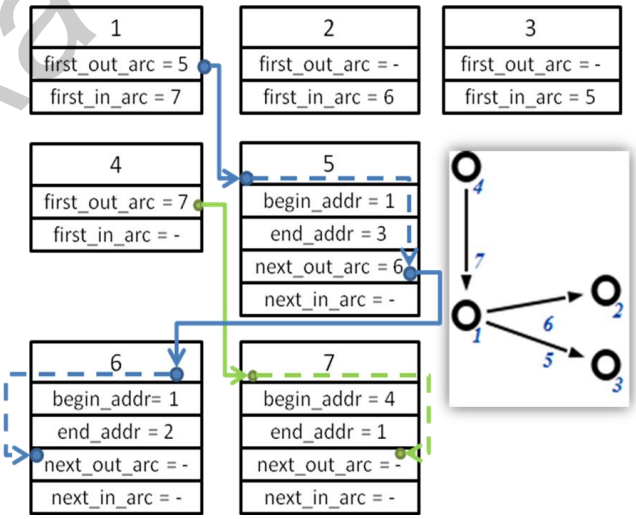

Рисунок 1 – Пример хранения списков инцидентности

Описанный выше метод хранения списков инцидентности позволяет значительно повысить плотность хранимой информации, а также обеспечить фиксированный размер сегментов. Очевидно, что при таком подходе всегда можно спрогнозировать необходимый объем памяти *M* необходимый для хранения того или иного sc-графа. Он всегда будет вычисляться по формуле 1.

$$M = N * S; \quad (1)$$

где N – количество sc-элементов в графе, S – размер ячейки (в байтах) необходимый для хранения одного sc-элемента

Заключение

Описанные выше приемы позволили реализовать первую версию sc-хранилища, которая имеет следующие характеристики по быстродействию:

- размер sc-элемента 64 байта, таким образом, размер сегмента 4 Мб. Размер полного sc-хранилища 256 Гб (использованы все ячейки);
- скорость заполнения sc-хранилища sc-узлами - 5116188.644108 узлов/сек;
- скорость заполнения sc-хранилища sc-дугами - 1833769.525978 sc-дуг/сек;
- скорость загрузки сегментов с диска в память – 305.56 сегмент/сек;
- скорость выгрузки сегментов из памяти на диск – 12 сегмент/сек;

Полученные данные отражают работоспособность sc-хранилища в однопоточном варианте исполнения, в качестве тестовой машины использовался ноутбук с 4 Гб оперативной памяти, процессором Intel Core i3 2.27 ГГц. Тесты проводились на операционной системе Ubuntu 12.04 32-bit.

Библиографический список

[Голенков и др, 2001] Представление и обработка знаний в графодинамических ассоциативных машинах /В. В. Голенков, [и др]; – Мн. : БГУИР, 2001

[Голенков, 2012] Графодинамические модели параллельной обработки знаний / В. В. Голенков, Н. А. Гулякина // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2012

[Корончик, 2012] Семантические модели мультимодальных пользовательских интерфейсов и семантическая технология их проектирования / Д. Н. Корончик // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» - Минск, 2012

[bigdata, 2012] bigdata [Электронный ресурс]. – 2012. – Режим доступа: http://www.bigdata.com/bigdata/blog/ - Дата доступа: 21.11.2012.

[graph-database, 2012] graph-database [Электронный ресурс]. – 2012. – Режим доступа: http://www.graph-database.org - Дата доступа: 20.11.2012.

[HyperGraphDB, 2012] HyperGraphDB – A Graph Database [Электронный ресурс]. – 2012. – Режим доступа: http://www.hypergraphdb.org – Дата доступа: 21.11.2012

[neo4j, 2012] Neo4j [Электронный ресурс]. – 2012. – Режим доступа: http://www.neo4j.org – Дата доступа: 20.11.2012.

[OSTIS, 2012] Открытая семантическая технология проектирования интеллектуальных систем [Электронный ресурс]. – 2012. - Режим доступа: http://ostis.net. – Дата доступа: 10.11.2012

IMPLEMENTATION OF STORAGE FOR UNIFIED SEMANTIC NETWORKS

Koronchik D. N

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

denis.koronchik@gmail.com

This article describes storage for unified semantic networks. It contains description of principles, that used to store semantic networks oriented for processing.

Introduction

There are several systems developed today to handle the problem of storing large amounts of graph data. But for each type of data and set operations different systems differ in suitability. But all of them not efficient to store unified semantic networks that used to store information in OSTIS project.

Main Part

Development of intelligent system with OSTIS technology consists of two parts: development of unified logical model of system, development of interpreter for the last one. Storage of unified semantic networks (sc-storage) is a main part of model interpreter.

SC- storage is a software system that provides storage of sc-graphs (unified semantic networks described with SC-code). It used segment addressing. Address of each element of sc-graph (sc-element) consists of two parts: segment number, offset in segment. Each segment – is a fixed size array of cells. Cells used to store data just for one sc-element and have a fixed size – 59 bytes. There are three types of stored sc-elements: sc-node, sc-link, sc-connector. Each cell has common data, that suitable for all element types and 33 bytes to store type specific information. For example, last 33 bytes store first and second, components for sc-connectors. For sc-links this 33 bytes contains hash sum of data calculate by SHA256 algorithm.

Conclusion

Described principles were implemented in sc-storage prototype. The last one was implemented with C programming language. Each sc-element in sc-storage has 64 bytes size. So full size of data that can be stored in implemented sc-storage is 256Gb or (2^{32} sc-element).