**WordNet**

Следуя принципам построения словарей WordNet и EuroWordNet, сформировавших- ся в современной компьютерной лингвистике, RussNet имеет следующие структурные харак- теристики: • состоит из четырех взаимосвязанных файлов для основных частей речи: существи- тельных, глаголов, прилагательных и наречий; • основными единицами структуры RussNet, на которых задаются семантические от- ношения, являются синонимические ряды (синсеты); • синсеты связаны между собой различными семантическими отношениями: парадиг- матическими и синтагматическими.

Синонимические отношения, связывающие слова по общности значений, являются базовыми для всех словарей типа WordNet. В рамках проектов WordNet и EuroWordNet си- нонимия определяется через понятие взаимозаменяемости. «Два слова (выражения) считают- ся синонимами, если существует хотя бы один контекст С, в котором замена одного слова другим не приводит к изменению истинностного значения» (Miller et al., 1990). Однако, на практике использовать этот критерий оказывается довольно сложно: взаимозаменяемость в контексте не всегда связана с общностью значений. Для слов любой части речи можно по- добрать такой контекст, в котором будут взаимозаменяемы слова, семантически не связан- ные друг с другом. Например, метаязыковое употребление слов: «Идти – это глагол. Мол- чать – это глагол» или контекстуальная синонимия: «во всем была сладкая и горькая грусть» (И. Бунин). Вместе с тем некоторые синонимы не являются взаимозаменяемыми в контексте из-за особенностей синтаксической (начать издавать, приступить к изданию, приняться за издание) или же лексической сочетаемости (актер играл; певец, танцор испол- нял; актер представлял [устар.]) 1 .

Разделы

Синонимия

Гипонимия

Антонимия

**HypergraphBD**

*«HyperGraphDB — это расширяемая, портативная, распределенная, встраиваемая система общего назначения со свободным (open-source) механизмом хранения данных. Эта система разработана специально для проектов использующих возможности искусственного интеллекта и семантического вэба и может использоваться как встраиваемая, объектно-ориентированная база данных для проектов любого масштаба.»*

* Для работы необходима Java5+
* Есть реализации и для [Windows](http://hypergraphdb.googlecode.com/files/hypergraphdb-1.0.zip), [\*nix](http://hypergraphdb.googlecode.com/files/hypergraphdb-1.0.tar.gz) и [mac](http://hypergraphdb.googlecode.com/files/hypergraphdb-1.0-macos.tar.gz) платформ
* API только для Java (и языки на платформе Java)
* [Написан на Java](http://code.google.com/p/hypergraphdb/)
* Способ запросов — [Java или P2P](http://www.kobrix.com/wikishow?page=IntroQuerying&project=hypergraphdb)
* P2P репликация и P2P фреймворк для распределенных вычислений и данных, а так же репликации данных
* Согласованность (concurrency) основана на механизме **STM** — Software Transactional Memory (механизм аналогичный транзакциям в базе данных — подробнее на [en:Wikipedia](http://en.wikipedia.org/wiki/Software_transactional_memory))
* Лицензия LGPL

В это базе данных как уже стало ясно из названия используются *гиперграфы*. Гиперграф это расширение обычной концепции графа, позволяющей ребру графа иметь более 2-х вершин. HyperGraphDB расширяет эту концепцию еще больше, позволяя ребру указывать на другое ребро, и позволяет любому ребру или вершине иметь произвольное значение в виде полезной нагрузки.

Основная единица хранения в HyperGraphDB называется *atom*. Каждый atom имеет произвольное значение и может указывать любое количество других atom. Типы данных управляются единой расширяемой системой встроенной в структуру гиперграфов.

Текущая реализация полностью построена на основе Java. Возможность автоматического преобразования типов данных Java в типы HyperGraphDB позволяют использовать HyperGraphDB как объектно-ориентированную базу данных для большинства бизнес приложений. С++ реализация неоднократно обдумывалась, но не была начата из-за нехватки разработчиков.

HyperGraphDB is primarily what its carefully chosen name implies: a database for storing hypergraphs. While it falls into the general family of graph databases, it is hard to categorize HyperGraphDB as yet another database because much of its design evolves around providing the means to manage structure-rich information with arbitrary layers of complexity. For instance, a relational as well as an object-oriented style of data management can be emulated. As a graph database, HyperGraphDB doesn't impose any constraints and offers much more generality than all other graph databases we've come across. The design is minimalistic at its core and the end-goal is to evolve a set of concepts and practices, combining structure and interpretation in such a way as to allow future software to meet the complexities of the real-world better that now.

In [mathematics](https://en.wikipedia.org/wiki/Mathematics), a **hypergraph** is a generalization of a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) in which an [edge](https://en.wikipedia.org/wiki/Graph_theory) can connect any number of [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)). Formally, a hypergraph H is a pair H = (X,E) where X is a set of elements called *nodes* or *vertices*, and E is a set of non-empty subsets of X called **hyperedges** or **edges**. Therefore, E is a subset of \mathcal{P}(X) \setminus\{\emptyset\}, where \mathcal{P}(X) is the [power set](https://en.wikipedia.org/wiki/Power_set) of X.

While graph edges are pairs of nodes, hyperedges are arbitrary sets of nodes, and can therefore contain an arbitrary number of nodes. However, it is often desirable to study hypergraphs where all hyperedges have the same cardinality; a *k*-**uniform hypergraph** is a hypergraph such that all its hyperedges have size *k*. (In other words, one such hypergraph is a collection of sets, each such set a hyperedge connecting *k* nodes.) So a 2-uniform hypergraph is a graph, a 3-uniform hypergraph is a collection of unordered triples, and so on.

* [The mathematical definition of a hypergraph](http://en.wikipedia.org/wiki/Hypergraph) is an extension to the standard graph concept that allows an edge to point to more than two nodes. HyperGraphDB extends this even further by allowing edges to point to other edges as well and making every node or edge carry an arbitrary value as payload.

<http://www.kobrix.com/hgdb.jsp>

**Semantic Web** projects are an obvious domain of application of HyperGraphDB. The so called "conceptual graphs" or RDF graphs and even the more advanced modeling practices utilizing higher-order relationships have a straightforward and natural expression within the HyperGraphDB framework.

**Networks research** can benefit from the capacity of HyperGraphDB to store very large, distributed graphs and have pattern mining, computationally intensive algorithms operate on them.

| **Component** | **Description** |
| --- | --- |
| [JSON](http://www.hypergraphdb.org/?page=Json&project=hypergraphdb) | Implementation of [JSON](http://json.org/) storage as a hypergraph - JSON structures as graphs rather than blobs as commonly implemented in so called "document-oriented" database. |
| [WordNet](http://www.hypergraphdb.org/?page=WordNet&project=hypergraphdb) | Representation of the lexical WordNet database from Princeton. |
| [TopicMaps](http://www.hypergraphdb.org/?page=TopicMaps&project=hypergraphdb) | Implementation of the Topic Maps 1.0 standard. |
| [RDF via Sail](http://www.hypergraphdb.org/?page=SesameSail&project=hypergraphdb) | Implementation of the RDF standard using the [openRDF.org](http://www.openrdf.org/) Sesame framework. |
| [OWL 2.0](http://www.hypergraphdb.org/?page=Home&project=owl) | Full implementation of the OWL 2.0 standard with distributed versioning and a [Protege](http://protege.stanford.edu/" \t "_blank) plugin. |
| [Protege Plugin](http://www.hypergraphdb.org/?page=ProtegePlugin&project=hypergraphdb) | Integration of HyperGraphDB backed ontology storage into the popular [Protege](http://protege.stanford.edu/" \t "_blank) ontology editor. Full GUI support for the distributed version control management with pull, push, commit, revision graph, history and everything you would expect from a modern version control system. |
| [TuProlog](http://www.hypergraphdb.org/?page=TuProlog&project=hypergraphdb) | Integration with the TuProlog interpreter for reasoning over hypergraphs through Prolog. |
| [XmlSchema](http://www.hypergraphdb.org/?page=XmlSchema&project=hypergraphdb) | Implementation of the XML Schema standard within the HyperGraphDB type system.***[incomplete]*** |
| [Feedforward Neural Nets](http://www.hypergraphdb.org/?page=NeuralNets&project=hypergraphdb) | Implementation for a feed-forward 3-layer neural net as a hypergraph |
| [Distributed Dataflow](http://www.hypergraphdb.org/?page=DataFlow&project=hypergraphdb) | Flow-based programming in Java based on a HyperGraphDB representation and using its P2P framework.***[unreleased]*** |

All those components are implemented as [HyperGraphDB Applications](http://www.hypergraphdb.org/?page=HyperGraphDBApplications&project=hypergraphdb)

**Hypergraph and wordnet**

The HyperGraphDB Wordnet application (called henceforth HGWN) is essentially a representation of all the information from [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) within HyperGraphDB. This includes support for:

Querying the DB for a given word.

Querying for specific semantical or lexical relationship of a given word

Morphological analysys - e.g. finding base form of a given word or collocation

HGWN does not support the so called "polysemy count"(количество разных значений слова) - number of uses of a given word in a specific sense as obtained by analyzing several sample texts. This feature is obsolete in [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) too and is left unchanged since 2003.

**Representation Details**

The basis of the representation is the [Word class](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/data/Word.html) which represents a single token (or a lemma). Each word is stored as an atom in HyperGraphDB. Words are then grouped into so called *synsets* or "synonym sets" which represent senses. Thus a sense of a word is exemplified as a set of synonyms. In HyperGraphDB, those are represented as links of type [SynsetLink](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/data/SynsetLink.html" \t "_blank) of which there's one variety for each part-of-speech. Such links have variable arity depending on the number of synonyms in a synset. Thus, noun senses are represented as instances of NounSynsetLink. Those are HyperGraphDB links with arbitrary arity >= 1. The original [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) ID is stored as the id property of synsets and the sense definition (a dictionary style definition in plain english) is stored as the gloss property.

**Exceptions**

For each part of speech (noun, adjective, verb, adverb), [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) contains a list of exceptional forms of the root lexeme. For nouns, this is usually the irregular plural forms, for adjectives and adverbs irregular superlatives, for verbs irregular conjugations. This is useful in morphological analysis where the normal rules for inflexion do not apply. Exceptions are represented in HGWN as links of type [ExcLink](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/data/ExcLink.html" \t "_blank), specifically its four sub types, one for each part of speech. The first target of an exception link is the root word and the other targets are its exceptional forms. The links have an arity >= 2. Both root and exceptional forms are stored as lexemes in HyperGraphDB (i.e. as instances of the type Word).

**Semantic Tools**

HGWN offers some extras for working with lexical semantics, and in particular for implementing WSD (word sense disambiguation) algorithms.

First, there's a utility class [WNGraph](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/WNGraph.html" \t "_blank) which offers an API for working with [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) in a underlying HyperGraphDB instance. In addition, the [SemTools](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/SemTools.html" \t "_blank)offers ready-made implementation of several word sense similarity measured commonly found in the computational linguistics literature.

Finally, the [WNStat](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/WNStat.html" \t "_blank) is simple, generic representation of some global information about the [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) graph that is computationally intensive to calculate. Some of the predefined such WNStats' are the depths of the noun and verb synset hierarchies in [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb).

**Loading [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) in a HyperGraphDB**

The JWNL (Java [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) Library) - <http://sourceforge.net/projects/jwordnet> is used to load the [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) database files. It is an open-source Java API for accessing [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) which provides API-level access to [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) data. The configuration files needed by that library are maintained in[org.hypergraph.app.wordnet.configuration](http://www.kobrix.com/javadocs/hgapps/wordnet/org/hypergraphdb/app/wordnet/configuration). They are only needed when Wordnet is initially loaded into HyperGraphDB. The JWNL library is also needed only for loading the data. Once the data is in the HyperGraphDB instance, the library can be removed from Java's classpath.

Here is a sample code to load [WordNet](http://www.kobrix.com/wikishow?page=WordNet&project=hypergraphdb) data into a HGDB instance. The code disables transactions for faster processing. So if the data already in your HGDB instance is important to you, make a backup copy beforehand.

|  |  |
| --- | --- |
|  | import org.hypergraphdb.\*;  import org.hypergraphdb.app.wordnet.\*;  String wordnetDB = "d:/data/graphs/wn"; // the location of your HGDB  // Disable transactions  HGConfiguration config = new HGConfiguration();  config.setTransactional(false);  // Open/Create the HGDB instance  HyperGraph graph = HGEnvironment.get(wordnetDB, config);  long startTime = System.currentTimeMillis();  HGWordNetLoader loader = new HGWordNetLoader();  // Modify the following line to point to your WordNet installation  loader.setDictionaryLocation("C:/tools/wordnet/dict");  // This should take a while, usually under 10 minutes but maybe  // more if you have an older machine  loader.loadWordNet(graph);  long endTime = System.currentTimeMillis();  System.out.println("Completed in " + (endTime - startTime) / 1000 + " seconds."); |