A Hypergraph Data Model for Building Multilingual Dictionary Applications

Louis Lecailliez
NLP Centre, Masaryk University
Botanická 68a, Brno, Czech Republic
louis.lecailliez@outlook.fr

Mathieu Mangeot
Université Savoie Mont Blanc, LIG
38000 Grenoble, France
mathieu.mangeot@imag.fr

Abstract

A non-negligible part of learners of an East Asian language have an interest for another tongue from East Asia that may share some common areal features such as the use of Chinese characters as well as limited word morphology. It makes sense to build for this niche a dictionary application that provides multiple languages in one bundle and allows easy navigation between them and in the lexicon. This paper describes a data model and a generic dictionary application architecture that addresses this use case.

The task of merging lexical resources with vastly differing micro-structures is complex. Even more so when updating it to include new data types or languages after release. In this regard, lexical networks are appealing: they solve the problem by exploding the micro-structure into data nodes and explicitly linking them with edges that can be discovered and traversed automatically. One of these approaches, The Linked Data, is gaining traction in lexicography. It is however plagued with issues within the Resource Description Framework (RDF) that backs it up. Most notably, the lack of three-valent relationships makes it harder than it should to handle the Chinese writing system.

We therefore came up with a simple and consistent hypergraph data model whose main features are: hyperlinks (links of arity greater than two), a flat type system (non-ontological lexical network) and annotations for both node and link instances. We propose a generic software architecture based on this model and illustrate it with a working mobile application. The user interface is constructed from independent components, allowing the display of complex data while increasing further its updatability and maintainability. We use data from the Revised Mandarin Chinese Dictionary of the Ministry of Education of Taiwan and augmented it with open-data Japanese readings to feed the prototype.

1. Introduction

It is quite common for students majoring in East Asian Studies to have an interest for another East Asian language: some universities actually provide double major degrees to meet this demand. A lot of vocabularies of Sinitic and Sinoxenic (Hashimoto, 1973) languages come from a common ancestry or were borrowed from each other, so they are related in meaning and pronunciations. It then makes sense to present learners with similar words of other languages he or she is learning while browsing an entry in a dictionary.

On the application developer side, the task of integrating different dictionaries for a given language is not easy. Creating a multilingual dictionary is even harder: each one has its own microstructure that doesn't necessarily play well with others. Moreover, the transposition from paper dictionaries to electronic ones led to digital resources (XML or databases) which retain a structure calqued on what is displayed to the end user (Polguère, 2012). Hence, the abstraction level needed to easily merge and use these resources is not provided.

Graph-based Data Models

Some electronic dictionary systems such as the one described by Mangeot *et al.* (2001) exhibit a latent graph structure while still being mostly based on dictionaries structured in a traditional fashion. Nonetheless, the groundbreaking work that made researchers start to rethink the modeling of dictionaries themselves is WordNet (Miller, 1995). Other lexical networks of two flavors were created: ontological and non-ontological (Polguère, 2014a). Amongst the former are the Semantic Web and the Linked Data frameworks which are used in past and ongoing research projects (Declerck, 2015). However, because the Semantic Web relies on Internet connectivity to reach its full potential, it is by definition not suitable for offline dictionaries, even if its technology may be used in an embedded way. In addition, theses frameworks are based on RDF which have their own share of problems.

This situation motivates us to create a data model that tries to capitalize on the essence of what makes the Semantic Web a potentially powerful technology — its underlying graph model — while not being limited in our implementation by the inherent complexity of RDF-based frameworks like Lemon (McCrae et al., 2011) in which support of East-Asian languages is lacking (Lecailliez, 2017).

2. Issues with Existing Data Models & Frameworks

Issues with the Relation Model

The major paradigm for organizing software data is the relational model. It is generally used in conjunction with a software that is layered in three main parts: the data layer which communicates with the database system to retrieve or store data, the business layer which contains the core logic of the software and the presentation layer that displays content to the end-user. Dictionary software are typically implemented in this way.

The main issue with the three-layered software on top of a relational database approach is that each modification of the schema requires a database migration, an update of the business logic and changes in the presentation layer. That is, every layer of the application is impacted.

Thus, we need a model which allows a dictionary to change its micro-structure frequently with the least possible impact to any other software layer. The model described in section 3 has the following properties: a change in schema would (1) not affect the business layer. Changes in data access (2) and presentation (3) layers made to support new data types are made by adding self-contained components; existing code does not require modifications.

Issues with RDF

While RDF data forms a graph (Powers, 2003), it is not a general-purpose graph modelling tool. The two central concepts of RDF are nodes and triples. Nodes exist in three flavors: resource, literal and blank. A triple (subject, property, object) denotes a "property" link between two nodes. The first major issue is that literals, which contains the actual data useful to a human being cannot be the subject of a relationship. The second issue is related to blank nodes, which are used to aggregate data from more than two nodes.

The Lemon Cookbook (McCrae *et al.*, 2010) gives a good example of these issues. In the Figure 1, a blank node is figured by square brackets and literals are double quoted. It encodes that some abstract resource has the written representation ⟨日本語⟩ and two "transliterations" ⟨にほんご⟩ and ⟨nihongo⟩ written in higarana and latin script.

```
:nihongo lemon:canonicalForm [
  lemon:writtenRep "日本語"@ja-Jpan ;
    isocat:transliteration "にほんご"@ja-Hira ;
    isocat:transliteration "nihongo"@ja-Latn ] .
isocat:transliteration rdfs:subPropertyOf lemon:representation .
```

Figure 1: Example 15 from the Lemon Cookbook

Because the literals cannot be the source of a link, the node ⟨日本語⟩ cannot be linked to entities representing each of its three constituent Chinese characters. For that, another node linking multiple resources and literal nodes is required. This makes creating and expanding the graph difficult: each modification requires the creation of multiples intermediate nodes and properties. This also changes radically the structure of the RDF molecule (Ding *et al.*, 2005) under modification. Additionally, this generates many layers of indirections that have to be handled by client applications consuming the graph. The creation of n-ary relationships also requires additional nodes and properties (W3C, 2006) because every relation in RDF is binary.

RDF does not have annotations (Lopes *et al.*, 2010), which is problematic to encode some common cases encountered in lexicography (*cf. infra*). Graph databases such as Neo4J (Robinson *et al.*, 2013) provide a similar mechanism. Finally, reification — an edge used as source or target of another link — is notoriously difficult and controversial (Trame *et al.*, 2013).

3. The Graph Model

The data model we propose is based on the notion of hypergraph (Lecailliez, 2016). A similar system is described by Williams (2000) under the name associative model of data (AMD). Despite their resemblances, this model is not directly built on AMD. Identical core concepts in both models are the use of only two kinds of entities (item/node and links) and the ability of a relation to link other relations in addition to nodes. There are however key differences in the model presented here such as the arity of links (always three in the AMD but unconstrained here) and the absence of relationship between types.

Type System

The graph is made of two kinds of objects: vertices and edges. Each object instance is associated with a type, which is an aggregate of multiple key-value pairs. There are no predefined types of vertex or edge. The six defined properties are: Name, Identifier, Description, Object Kind, is_oriented and is_direct_content.

The **name** of a type is destined to human lexicographers and developers; it should be short yet informative. This property is never used by software to test the equality of two types. The **identifier** property is used to that end. A type identifier could be shared between projects. The **description** is where potentially long explanations are made to describe how a node content must be written and what semantic an edge carries. It may address concerns of lexicographers such as the transcription system to use, as well as developer problematics like the encoding of multimedia content. The **object kind** is either edge or vertex.

The last two properties have only meaning depending on whether the type describes a node or an edge. A vertex type with a **is_direct_content** set to true indicate the node's value can be displayed to the end user of a dictionary application without any further processing.

Node Instances & Atomicity

A node instance aggregates the following values: a type, a language tag and a list of indexed strings. In addition, the internal value of a node is stored in the *Content* property; it cannot be referenced directly from another node of the graph, nor can it contain a reference to a graph object. Nodes are hence atomic values in respect to the graph. This is an important property because (1) it allows a node to be safely removed from the graph without leaving dandling pointers and (2) any link between data must be explicitly declared with a relationship instance; this enables its automatic discovery and processing. This is contrary to the (Polguère, 2014b) model which explicitly makes use of non-atomic lexical nodes.

Atomicity allows complex content to be stored in vertices. For example, information about vowel devocalization in a Japanese word or multimedia content such as an image can be added this way. Most of the time however, simple lexicographic data can be stored as a string which can be displayed directly to the dictionary user.

Relationships

A link indicates that two or more graph objects are linked by a given relationship denoted by its type. The model currently defines three kinds of edges: non-oriented edge, oriented edge and hyper-edge. The difference between oriented and non-oriented links is semantic: an oriented instance of a relationship means it has meaning in only one direction. If the relationship can be interpreted backwards, the *Description* field should explain the associated semantic. In the prototype, a single class implement the three kinds of edges.

Hyper-edge — a link between more than two objects — is mainly motivated to enable representation of information of dictionaries where Chinese characters are present. In these languages the minimal bilingual entry is made of a word of the source language written in two forms and a word or expression of the target language. One of the graphical source forms contains Chinese characters while the other is written in a script not ambiguous about its pronunciation. For example, a minimal triplet entry for the word "birthday" in Japanese is (誕生日, たんじょうび, birthday) or (誕生日, tanjōbi, birthday) if we use the Latin script to transliterate the word.

Annotations

Not every situation benefits from being modelled as a node or an edge. Typically, information such as part of speech would have a huge number of linked nodes in a real dictionary. Because such nodes change the topology of the graph, they don't play nicely with an automatic display system such as the one presented in the next section. Annotations solve this issue by providing an alternative way to associate data with graph objects.

An annotation is a triple of strings (namespace, key, value) that can be associated to any object of the graph. It offers great flexibility of modeling but less automatic processing capabilities. The namespace allows multiple keys with the same name, for example two *freq* properties indicating the frequency of a word that were computed using different corpora.

4. Application Architecture

Using a graph doesn't provide much benefit by itself if not exploited by software. In fact, it creates problematics such as how to display the graph to the end user. We propose in this section an application architecture that uses a graph and displays it in a traditional fashion.

The Figure 2.a illustrates the architecture of a mobile application. The application package is made of three parts, one of them being mandatory. The required part (2) is composed of (a) the graph model implementation, (b) the page composer which dynamically generates dictionary pages and displays components for (c) vertexes and (d) relationships. A file (e) or code section declares the associations between graph types and display components.

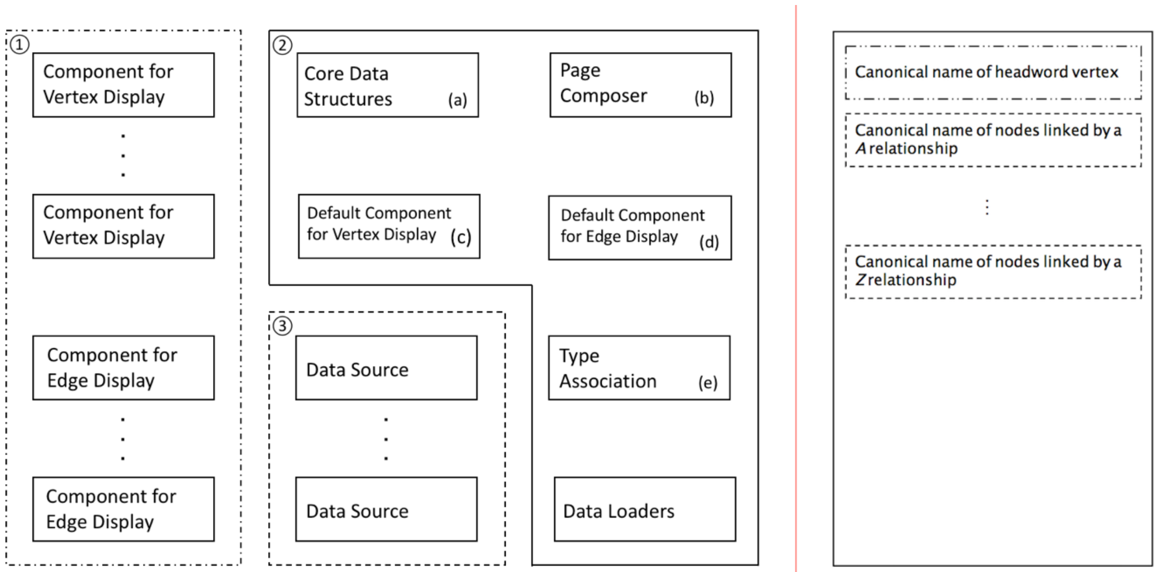

①
Component for Vertex Display
Component for Vertex Display
Component for Edge Display
Component for Edge Display
②
Core Data Structures (a)
Page Composer (b)
Default Component for Vertex Display (c)
Default Component for Edge Display (d)
③
Data Source
Data Source
Type Association (e)
Data Loaders
Canonical name of headword vertex
Canonical name of nodes linked by a A relationship
Canonical name of nodes linked by a Z relationship

Figure 2: a. Application Architecture; b. Page Generated by the Page Composer

Types for which no custom display component (1) association is declared are handled by the default (c) (d) components. This method allows modular development and incremental additions to the dictionary. User interface is not declared in a single hard-to-maintain file. Finally, data files are bundled (3) with the application or can be retrieved from the network.

Page Generation

Dictionary pages are generated on the fly by the page composer for a given “headword vertex”. The head vertex content is displayed at the top of the page (Figure 2.b) by the default vertex presenter or a custom interface component. All neighboring nodes are grouped by relation type and each group is displayed by an instance of the default edge presenter or a custom component.

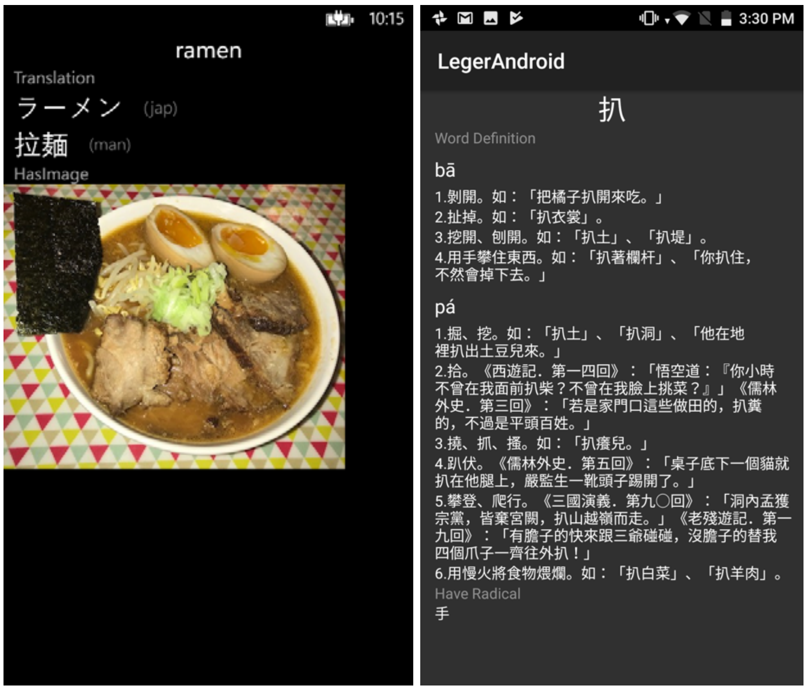
The behavior is the same with any number of nodes and relationship types. New types can be added to the graph at runtime, it does not affect nor break the page composer. New data is handled by the default display component if their content can be displayed directly.

These are the two key points that allow the statement (1) made in section 2 concerning the immutable business logic of the application.

In addition to displaying vertex content, the default relationship presenter implements a navigation behavior: a click listener is added to each text data displayed, which changes the current headword with the target node associated with the data being clicked. By changing the current vertex head, a new page is generated and displayed. The user can navigate through the whole graph using this mechanism.

Custom Display Component

Finally, the composer can make use of self-contained display components that interpret a node content to produce a rich formatting. The formatting can include colors, multimedia elements and custom click behavior. Figure 3 shows two pages that use different components for data display instead of the generic one.



10:15
ramen
Translation
ラーメン (jap)
拉麵 (man)
HasImage

3:30 PM
LegerAndroid
扒
Word Definition
bā
1.剝開。如：「把橘子扒開來吃。」
2.扯掉。如：「扒衣裳」。
3.挖開、刨開。如：「扒土」、「扒堤」。
4.用手攀住東西。如：「扒著欄杆」、「你扒住，不然會掉下去。」
pá
1.掘、挖。如：「扒土」、「扒洞」、「他在地裡扒出土豆兒來。」
2.拾。《西遊記．第一四回》：「悟空道：『你小時不曾在我面前扒柴？不曾在我臉上挑菜？』」《儒林外史．第三回》：「若是家門口這些做田的，扒糞的，不過是平頭百姓。」
3.撓、抓、搔。如：「扒癢兒。」
4.趴伏。《儒林外史．第五回》：「桌子底下一個貓就扒在他腿上，嚴監生一靴頭子踢開了。」
5.攀登、爬行。《三國演義．第九〇回》：「洞內孟獲宗黨，皆棄宮闕，扒山越嶺而走。」《老殘遊記．第一九回》：「有膽子的快來跟三爺碰碰，沒膽子的替我四個爪子一齊往外扒！」
6.用慢火將食物煨爛。如：「扒白菜」、「扒羊肉」。
Have Radical
手

Figure 3: Use of Custom Components in (a) a Windows Phone and (b) an Android application

In Figure 3.a, the first component displays the content of nodes reached by the *Translation* relationship. It appends the language of the node in gray and between parentheses after the data itself. Another one is used to display an image related to the node reached by a *HasImage* link. In Figure 3.b data is taken from the (MoE, 2015) dictionary. The *Word Definition* relationship uses a custom UI component while the *Have Radical* does not.

Default and custom display components are the way the statement (3) of section 2 is held true. The final user interface is broken down in various and totally independent software classes. Additional components can be added without any need for changes to existing ones when a new data type is added to the dictionary. If this type is just a string that can be directly displayed, the creation of a component is no longer needed.

Prototype & Data

Distinct implementations of the application architecture were made for Android and Windows Phone. Each of them relies on a C# library that implements the graph model presented in section 3. The library is also used in other lexicography related programs.

We tested the model with various data sets, most notably data extracted from the Revised Mandarin Chinese Dictionary, an unofficial Japanese vocabulary list for the highest Japanese Language Proficiency Test level, and the Jibiki Project (2015), (Mangeot, 2016). A graph of Chinese and Nôm character decomposition was built with data from the Kanji Database Project (2015).

Known Display Issues

The graph connectivity can have a bad effect on display. Some nodes are connected to a huge number of vertices, so their display can be overwhelming for the user. There is no obvious and automated fix for this case: the problem must be tackled during the creation of the dictionary. Node pruning can be achieved by different algorithms that take various information in account, but it should be done statically if it uses intensive computations. The best solution in term of quality of entries is to manually annotate the most interesting entries, display a limited number of linked vertices and provide a button to load more at the user's will.

5. Conclusion and Future Research

The present paper has introduced a lightweight hypergraph model that alleviates the difficulties that existing frameworks based on RDF impose for modeling graphs. Three issues are specifically addressed: first, the need for an annotation system. Secondly, the model allows relation of n-arity, that is particularly useful when dealing with Chinese characters. Thirdly, it provides a built-in reification mechanism.

Moreover, it lays the way to create dictionary applications that can be updated easily to support additional languages. In particular, the prototype we built on the described abstract application architecture is robust to changes in the dictionary data: new types — which encode part of the dictionary micro-structure — can be added or deleted without breaking the application. It is capable of discovering and displaying string data without further modification. Elaborate display of data can be added to the software with new components; no change is required to any existing user interface files.

Finally, an edge that may recursively link nodes or relationships allows expressing complex lexicography situations. It could allow for example the user to navigate to the meaning of a word it has in the context from a higher level lexicographic construct such as a Chinese proverb. Future work will be done to provide a demonstration of this capability.

One issue though is the lack of a constraint system for relationships which could hinder the discovering capabilities of a client processing the graph (for example a program building a SQL database from a graph file). This point may be addressed in a future revision of the model.

6. Acknowledgement

This paper was partially written with the support of the MSMT10925/2017-64-002 scholarship grant from the Czech Ministry of Education, Youth and Sports.

7. References

Declerck, T., Wand-Vogt, E. et Mörth, K. (2015). Towards a pan european lexicography by means of linked (open) data. In Kosem, I., Jakubícek, M., Kallas, J. et Krek, S., editors: *Proceedings of eLex 2015. Biennial Conference on Electronic Lexicography (eLex-2015), electronic lexicography in the 21st century: Linking lexical data in the digital age*. Trojina, Institute for Applied Slovene Studies/ Lexical Computing Ltd., Trojina, Institute for Applied Slovene Studies, Ljubljana.

Ding, L., Finin, T., Joshi, A., Peng, Y., Da Silva, P. P., McGuinness, D. L. (2005). Tracking RDF Graph Provenance using RDF Molecules (revision 2). Technical report *TR-CS-05-06*, April 2005. Accessible at: https://ebiquity.umbc.edu/_file_directory_/papers/178.pdf

Hashimoto, J. M. (1973). Current Developments in Sino-Vietnamese Studies. *Journal of Chinese Linguistics*, 1-26. Accessible at: http://www.jstor.org/stabe/23752818

Jibiki Project. (2015). Accessed at: http://jibiki.fr. (20 May 2017).

Kanji Datase Project. (2015). 字形ＩＤＳデータ. [jikei IDS dēta] Accessed at: http://kanji-database.sourceforge.net/ids/ids.html (20 May 2017).

Lopes, N., Zimmermann, A., Hogan, A., Lukácsy, G., Polleres, A., Straccia, U., & Decker, S. (2010, June). RDF needs annotations. In *W3C Workshop on RDF Next Steps*, Stanford, Palo Alto, CA, USA.

Lecailliez, L. (2016). Pour une modélisation de dictionnaires de japonais sous forme de graphe. [Towards Graph Modeling of Japanese dictionaries] Master thesis, Paris Diderot. Accessible at: https://louis.lecailliez.net/dl/memoire_m2_jap_Lecailliez.pdf.

Lecailliez, L. (2017). Preliminary Thoughts on Issues of Modeling Japanese Dictionaries Using the OntoLex Model. In Horák, A.; Rychlý, P.; and Rambousek, A., editor(s), *Proceedings of the Eleventh Workshop on Recent Advances in Slavonic Natural Languages Processing*, RASLAN 2017, pages 11-19, 2017. Tribun EU. Accessible at: https://nlp.fi.muni.cz/raslan/raslan17.pdf.

Mangeot, M. (2016) Collaborative Construction of a Good Quality, Broad Coverage and Copyright Free Japanese-French Dictionary. *International Journal of Lexicography 2016*; doi:10.1093/ijl/ecw035; 35 p.

Mangeot, M., Sérasset, G. (2001). Papillon Lexical Database Project: Monolingual Dictionaries and Interlingual Links. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*. November 27-30, 2001, pages 119–125, Tokyo, France.

McCrae, J., Aguado-de-Cea, G., Buitelaar, P., Cimiano, P., Declerck, T., Pérez, A. G., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., Wunner, T. (2010). The lemon cookbook. Technical report, Monnet Project (June 2012), www.lemon-model.net.

McCrae, J., Spohr, D., Cimiano, P. (2011). Linking Lexical Resources and Ontologies on the Semantic Web with Lemon. In: Antoniou G. et al. (eds) *The Semantic Web: Research and Applications*. ESWC 2011. Lecture Notes in Computer Science, vol 6643. Springer, Berlin, Heidelberg.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38, No. 11: 39-41.

MoE. (2015). Revised Mandarin Chinese Dictionary of the Ministry of Education of Taiwan. [教育部重編國語辭典修訂本]
Accessed at: http://resources.publicense. moe.edu.tw/dict_reviseddict_download.html

Polguère, A. (2012). Lexicographie des dictionnaires virtuels. In Apresjan, Y., Boguslavsky, I., L'Homme, M.-C., Iomdin, L., Milićević, J., Polguère, A. et Wanner, L., eds:

Meanings, Texts, and Other Exciting Things. A Festschrift to Commemorate the 80th Anniversary of Professor Igor Alexandrovič Mel'čuk.

Polguère, A. (2014a). From Writing Dictionaries to Weaving Lexical Networks. *International Journal of Lexicography*, 27(4):396–418.

Polguère, A. (2014b). Principes de modélisation systémique des réseaux lexicaux. Principes de modélisation systémique des réseaux lexicaux. In *TALN* 2014 (pp. 79-90), Marseille.

Powers, S. (2003). *Practical RDF*. Sebastopol: O'Reilly & Associates.

Robinson, I., Webber, J., Eifrem, E. (2013). *Graph databases*. O'Reilly Media, Inc.

Trame, J., Keßler, C., Kuhn, W. (2013). Linked data and time–modeling researcher life lines by events. In *International Conference on Spatial Information Theory* (pp. 205-223). Springer, Cham.

W3C. (2006) Accessed at: https://www.w3.org/TR/swbp-n-aryRelations/. (20 May 2017)

Williams, S. (2000). *The associative model of data*. Second Edition. Lazy Software.