Datenbank Spektrum (2017) 17:113–129
DOI 10.1007/s13222-017-0253-x

SCHWERPUNKTBEITRAG

The *Hydra.PowerGraph* System

Building Digital Archives with Directed and Typed Hypergraphs

Holger Meyer[1] · Alf-Christian Schering[1] · Andreas Heuer[1]

Received: 23 February 2017 / Accepted: 16 May 2017 / Published online: 21 June 2017

Abstract Directed hypergraphs are known from graph theory [11] and are well understood within their own domain [7–9, 22, 23]. This paper provides an overview on the expressiveness of directed and typed hypergraphs as a modeling paradigm not only for the content of digital libraries and archives but a variety of applications. Furthermore, hypergraphs are sufficiently expressive to provide an implementation logic for conceptual models like CIDOC/CRM [18] in the context of museum-related systems and digital archives.

The directed hypergraph model supports typed nodes and individual flexible sets of attributes on a per node type basis. This allows for efficient mapping on object-relational database structures. It also features a flexible, semi-structured type system for hyperedges. The graph model is accompanied by a set of well defined graph operations forming an algebra and a descriptive hypergraph query language GrafL. This language supports typed, structure and value based queries as well as fundamental graph algorithms.

The suitability of such a hypergraph-based model is illustrated with a large digital ethnological archive system, which is developed in the *WossiDiA* project [43, 52, 53].

Keywords Graph databases · Directed hypergraphs · Dynamic type checking · Digital humanities · Digital archive systems

This work was part-financed by the German Research Foundation (DFG) under contract INST-16963/2-1, SCHM-1855/1-2 and the Federal Office of Civil Protection and Disaster Assistance (BBK) under contract III.1 122/04.

✉ Holger Meyer
hm@IEEE.org

[1] Database Research Group, University of Rostock, 18051 Rostock, Deutschland

1 Introduction

Richard Wossidlo was a German folklorist in Mecklenburg (1859–1939). During his ethnological field research, he wrote all facts on small paper slips. Fig. 1 shows him in front of the box shelf with his nearly two million field research notes.

These small notes contain a lot of well-defined references to scholars, narrators, contributors, places, villages, landscape, points and periods in time, literature, research papers, etc.

Richard Wossidlo not only collected all information on every day life in Mecklenburg, he also recorded all variations of narrative materials, and from whom he learned things, where and when. This was done not only by himself but also for all narrators and contributors he was corresponding with. He also put remarks on related research on such paper slips. As a result, he ended up with a so-called convolute of such notes on small paper slips enclosed in an envelope and then organized by different topics in the boxes. Fig. 2 shows such a box with some convolutes.

A main outcome of Richard Wossidlo's research was the edition of the "Mecklenburger Wörterbuch", an ethnological and linguistic encyclopedia for the Mecklenburg landscape.

To support the folklorists and ethnologists in their research, all the field research notes and related documents collected by Richard Wossidlo have been digitized, exposed on 35mm film for long-term preservation, and a digital archive system for the community was built.

A main effort of the Wossidlo Digital Archive (*WossiDiA*) project was the digitization process [53]. The main steps are shown in Fig. 4. The *WossiDiA* system contains not only all these different, digitized documents. It also al-

Fig. 1 Richard Wossidlo in front of his box shelf with field research notes. (© Karl Eschenburg)

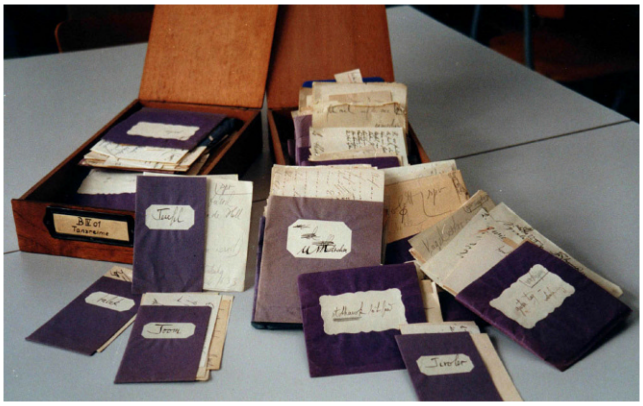
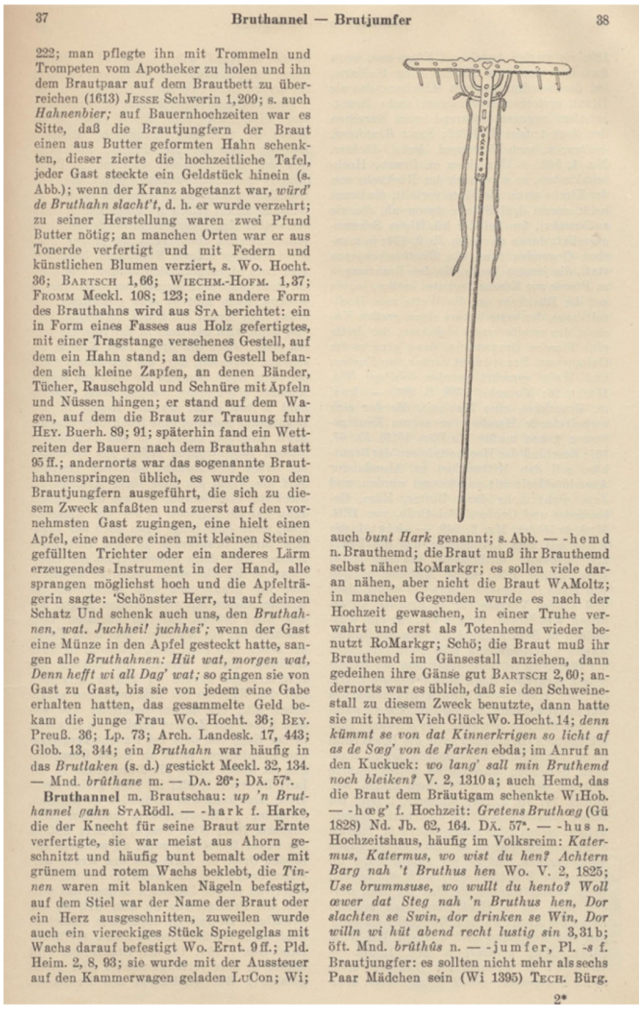


Fig. 2 Field research notes in envelopes and boxes from Wossidlo's archive



37 **Bruthannel — Brutjumfer** 38

222; man pflegte ihn mit Trommeln und Trompeten vom Apotheker zu holen und ihn dem Brautpaar auf dem Brautbett zu überreichen (1613) Jesse Schwerin 1,209; s. auch *Hahnenbier;* auf Bauernhochzeiten war es Sitte, daß die Brautjungfern der Braut einen aus Butter geformten Hahn schenkten, dieser zierte die hochzeitliche Tafel, jeder Gast steckte ein Geldstück hinein (s. Abb.); wenn der Kranz abgetanzt war, *würd' de Bruthahn slacht't,* d. h. er wurde verzehrt; zu seiner Herstellung waren zwei Pfund Butter nötig; an manchen Orten war er aus Tonerde verfertigt und mit Federn und künstlichen Blumen verziert, s. Wo. Hocht. 36; Bartsch 1,66; Wiechm.-Hofm. 1,37; Fromm Meckl. 108; 123; eine andere Form des Brauthahns wird aus Sta berichtet: ein in Form eines Fasses aus Holz gefertigtes, mit einer Tragstange versehenes Gestell, auf dem ein Hahn stand; an dem Gestell befanden sich kleine Zapfen, an denen Bänder, Tücher, Rauschgold und Schnüre mit Äpfeln und Nüssen hingen; er stand auf dem Wagen, auf dem die Braut zur Trauung fuhr Hey. Buerh. 89; 91; späterhin fand ein Wettreiten der Bauern nach dem Brauthahn statt 95 ff.; andernorts war das sogenannte Brauthahnenspringen üblich, es wurde von den Brautjungfern ausgeführt, die sich zu diesem Zweck anfaßten und zuerst auf den vornehmsten Gast zugingen, eine hielt einen Apfel, eine andere einen mit kleinen Steinen gefüllten Trichter oder ein anderes Lärm erzeugendes Instrument in der Hand, alle sprangen möglichst hoch und die Apfelträgerin sagte: 'Schönster Herr, tu auf deinen Schatz Und schenk auch uns, den *Bruthahnen, wat. Juchhei! juchhei';* wenn der Gast eine Münze in den Apfel gesteckt hatte, sangen alle *Bruthahnen: Hüt wat, morgen wat, Denn hefft wi all Dag' wat;* so gingen sie von Gast zu Gast, bis sie von jedem eine Gabe erhalten hatten, das gesammelte Geld bekam die junge Frau Wo. Hocht. 36; Bey. Preuß. 36; Lp. 73; Arch. Landesk. 17, 443; Glob. 13, 344; ein *Bruthahn* war häufig in das *Brutlaken* (s. d.) gestickt Meckl. 32, 134. — Mnd. *brûthane* m. — Da. 26^a; DÄ. 57^a.

Bruthannel m. Brautschau: *up 'n Bruthannel gahn* StaRödl. — -hark f. Harke, die der Knecht für seine Braut zur Ernte verfertigte, sie war meist aus Ahorn geschnitzt und häufig bunt bemalt oder mit grünem und rotem Wachs beklebt, die *Tinnen* waren mit blanken Nägeln befestigt, auf dem Stiel war der Name der Braut oder ein Herz ausgeschnitten, zuweilen wurde auch ein viereckiges Stück Spiegelglas mit Wachs darauf befestigt Wo. Ernt. 9 ff.; Pld. Heim. 2, 8, 93; sie wurde mit der Aussteuer auf den Kammerwagen geladen LuCon; Wi; auch *bunt Hark* genannt; s. Abb. — -hemd n. Brauthemd; die Braut muß ihr Brauthemd selbst nähen RoMarkgr; es sollen viele daran nähen, aber nicht die Braut WaMoltz; in manchen Gegenden wurde es nach der Hochzeit gewaschen, in einer Truhe verwahrt und erst als Totenhemd wieder benutzt RoMarkgr; Schö; die Braut muß ihr Brauthemd im Gänsestall anziehen, dann gedeihen ihre Gänse gut Bartsch 2,60; andernorts war es üblich, daß sie den Schweinestall zu diesem Zweck benutzte, dann hatte sie mit ihrem Vieh Glück Wo. Hocht. 14; *denn kümmt se von dat Kinnerkrigen so licht af as de Sœg' von de Farken* ebda; im Anruf an den Kuckuck: *wo lang' sall min Bruthemd noch bleiken?* V. 2, 1310 a; auch Hemd, das die Braut dem Bräutigam schenkte WiHob. — -hœg' f. Hochzeit: *Gretens Bruthœg* (Gü 1828) Nd. Jb. 62, 164. DÄ. 57^a. — -hus n. Hochzeitshaus, häufig im Volksreim: *Katermus, Katermus, wo wist du hen? Achtern Barg nah 't Bruthus hen* Wo. V. 2, 1825; *Use brummsuse, wo wullt du hento? Woll œwer dat Steg nah 'n Bruthus hen, Dor slachten se Swin, dor drinken se Win, Dor willn wi hüt abend recht lustig sin* 3,31 b; öft. Mnd. *brûthûs* n. — -jumfer, Pl. -s f. Brautjungfer: es sollten nicht mehr als sechs Paar Mädchen sein (Wi 1395) Tech. Bürg.

2*

Fig. 3 Excerpt from lexeme "Austharke" in the Mecklenburg Encyclopedia

lows for managing, linking, navigating, and querying these different, highly interconnected documents.

For the folklorist or ethnologist it is of immense interest, how the encyclopedia entries, like the one shown in Fig. 3, are derived from the set of these small notes.[1] A real challenge is to collect, aggregate and summarize all the information scattered over a network of notes. Therefore, right from the beginning in the development of the *WossiDiA* system, a graph based approach was used [52] in addition to a semi-structured XML document model [2].

Collecting all the related information from the paper slip network is done by applying graph based operations like content, structure and type based graph filtering combined with hyperedge contraction, aggregation and summarizing. The resulting graph can then be mapped to a logical document similar to published lexeme. This document is a set of hyperedges with a certain content model (cf. Sect. 3.2).

The *WossiDiA* system was the main motivation and source of inspiration which led to the development of *Hydra.PowerGraph*. *Hydra.PowerGraph* consists of a hypergraph database system *PowerGraph* and a digital archive framework *Hydra*.

- *PowerGraph* is build as an extension of the object-relational database system PostgreSQL. It uses PostgreSQL's spatio-temporal extensions and object-relationl features like user defined types and inheritance.
- *Hydra* (a HYpergraph of Documents in a Relational Archive) contains building blocks for digitization workflows, supports spatio-temporal contexts, GIS-based presentation of query results and has a set of frequently used object types for persons, places, events and others requested by digital archive developers.

[1] The scenario in question here is about the exchange of gifts and wooing. A daytaler or farmhand is presenting a decorated rake, the "Austharke" (Lower German language for dowry rake), to his bride-to-be. The information about this rural custom is spread over several field notes with the "Austharke" in the centre.

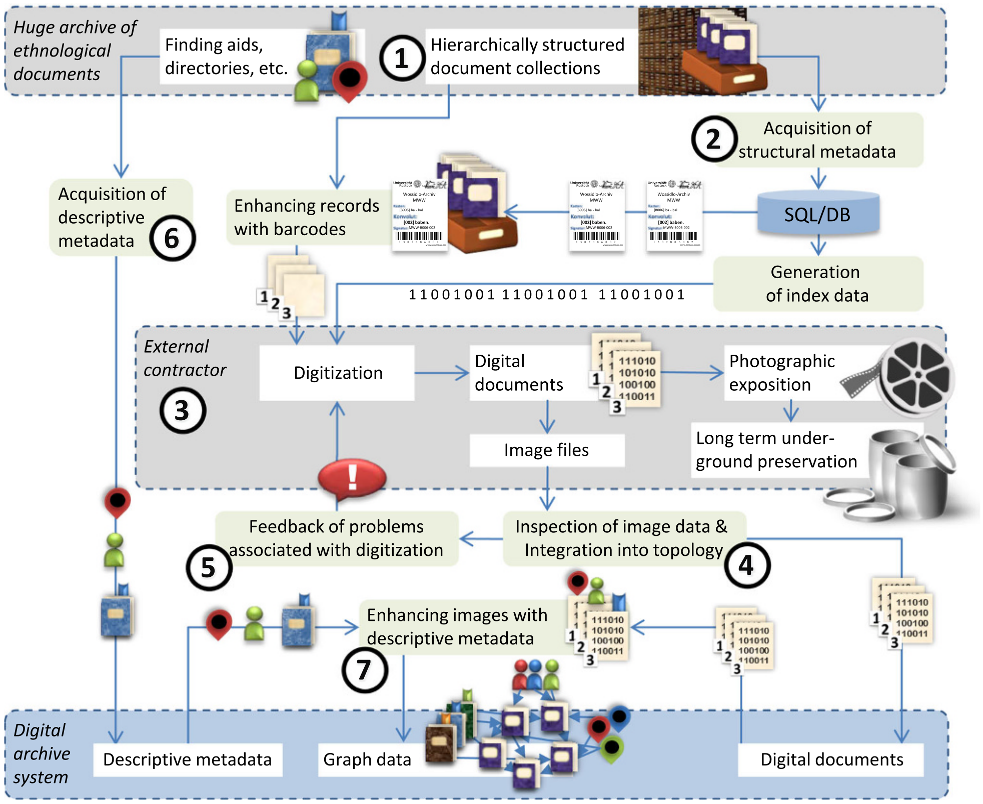

Huge archive of ethnological documents
Finding aids, directories, etc.
1
Hierarchically structured document collections
2
Acquisition of structural metadata
SQL/DB
Generation of index data
Enhancing records with barcodes
Acquisition of descriptive metadata
6
11001001 11001001 11001001
External contractor
3
Digitization
Digital documents
Photographic exposition
Image files
Long term under-ground preservation
Feedback of problems associated with digitization
5
Inspection of image data & Integration into topology
4
Enhancing images with descriptive metadata
7
Digital archive system
Descriptive metadata
Graph data
Digital documents

Fig. 4 *WossiDiA* digitization workflow

The remainder of this article is organized as follows: First of all, we are motivating why a hypergraph based model is preferred over a more structured, constrained database or a simple graph database model in building a large archive system for data from ethnological field research. The introduction will briefly indicate *WossiDiA* as our main *PowerGraph* usage scenario, for which the hypergraph system will be utilized prototypically.

Then, we outline the fundamentals of such a directed hypergraph model in a semi-formal way in Sect. 3. It is followed by a basic description of the architecture and the query processing techniques implemented on top of an object-relational database system in Sect. 4. Sect. 5 illustrates how directed hypergraphs were used as a modelling paradigm in the *WossiDiA* system and presents some statistics of the graph data.

Related work is discussed in Sect. 6. The article is summed up by a presentation of open questions and further research and forthcoming projects in Sect. 7.

2 Motivation – Hypergraphs for the Study of Complex Cultural Phenomena

Before we will motivate directed, typed hypergraphs as a modelling paradigm for digital archives, we give a short introduction to CIDOC/CRM.

2.1 CIDOC/CRM

CIDOC/CRM [18] is a conceptual model primarily designed for the exchange of historical information in the course of archives, museums, and libraries. The conceptual model is meant for the abstract description of objects (called *domain entities*) and their relationships (*properties*) to other entities (called *range*) and maybe vice versa. The model supports some basic types like strings, date, timestamp, and coordinates of places, which are mandatory for the spatio-temporal aspects of historical information.

The small example in Fig. 5 depicts the CIDOC/CRM representation for the exchange of gifts, an excerpt from the Wossidlo archive (cf. Fig. 6 for the hypergraph representation in the *WossiDiA* system). The rectangles represent the entity types with their instance's values below. Properties are shown as labeled arcs directed from the domain entity to their range.

CIDOC/CRM concepts also include some kind of inheritance for building specialization hierarchies on entity and property types and is self-describing and extensible by entities having type properties ([P2] *has type*). This allows for substituting entities of a sub-type where its supertype is allowed in a property, e. g. substituting [E22] *Man-Made Object* if [E19] *Physical Object* is requested as the domain entity or range of a property. CIDOC/CRM comes along with hundreds of these pre-defined entities and prop-

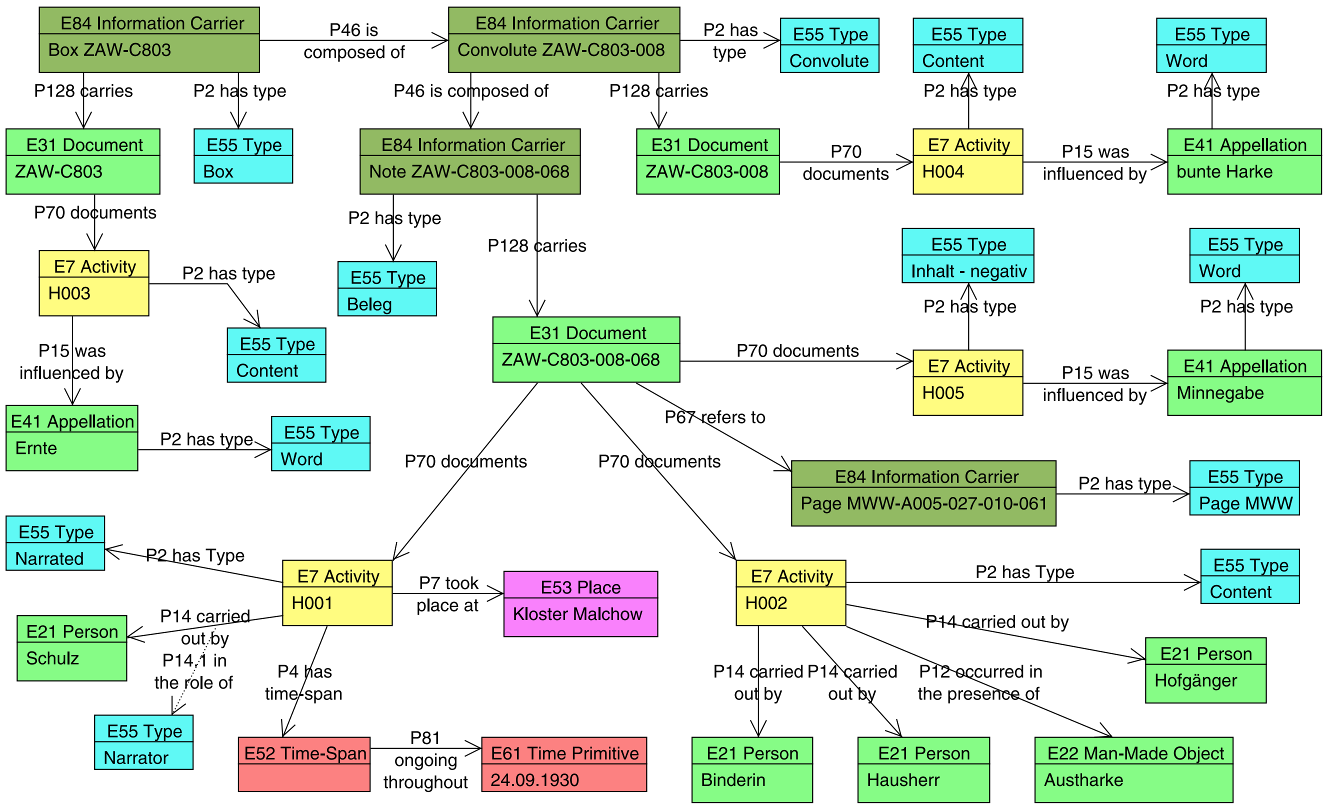


Fig. 5 The "Austharke" and exchange of gifts scenario expressed with CIDOC/CRM

erties. Nevertheless, since the main design goal was data exchange, not querying and retrieval, it only defines structures and does not contain any query language facilities.

Obviously, the hypergraph model is much more concise and compact than the conceptual CIDOC/CRM. Nevertheless, CIDOC/CRM is just a conceptual model never ment for storing or querying data. One always has to map it onto a logical model like the relational one or a graph data model.

Now, we will argue why a hypergraph model is more concise and compact not only in modelling digital archive data but also usable as an implementation or logical model.

2.2 The hypergraph model

Hypergraphs are an unusually apt model for complex cultural phenomena such as folklore [44, 55]. In this graph model, which departs significantly from standard single mode graphs, multiple classes of nodes are connected across a series of potential relationships. In "normal" graphs, a single edge can connect any two nodes. In contrast, hypergraphs generalize this idea by introducing *hyperedges* which can connect any arbitrary set of nodes. Consequently, a hypergraph can be seen as a set of hyperedges with each hyperedge consisting of a set of nodes [11]. Thus, hypergraphs can express everything that a normal graph can but also allowing the connection of all directly related nodes with a single hyperedge. In this manner, many-to-many relationships between several objects/nodes, usually represented by strongly connected components in a normal graph, are represented as a single hyperedge in a hypergraph. Because of this core characteristics, hypergraphs are well suited to model complex heterogeneous data sets, with many types of objects and their multiple relationships. Hypergraphs are not only a commonly accepted formalism in database theory [9, 19] but are also used as a formal basis in several semantic data models [41].

2.3 The *WossiDiA* hypergraph model

In the Wossidlo Digital Archive *WossiDiA* (cf. Fig. 1) hypergraph model, nodes can represent stories, storytellers, places, collectors, named entities, keywords, story actants. The relationships across and between these entities are captured as hyperedges. A hypergraph representation for the aforementioned example (exchange of gifts) is given in Fig. 6.

While these structures have certain characteristics that make them somewhat harder to work with than in single mode graphs, advances in tensor decomposition and network theory have made hypergraph models a powerful representation of complex data that does not unnecessarily reduce or compress the high dimensionality of these data.

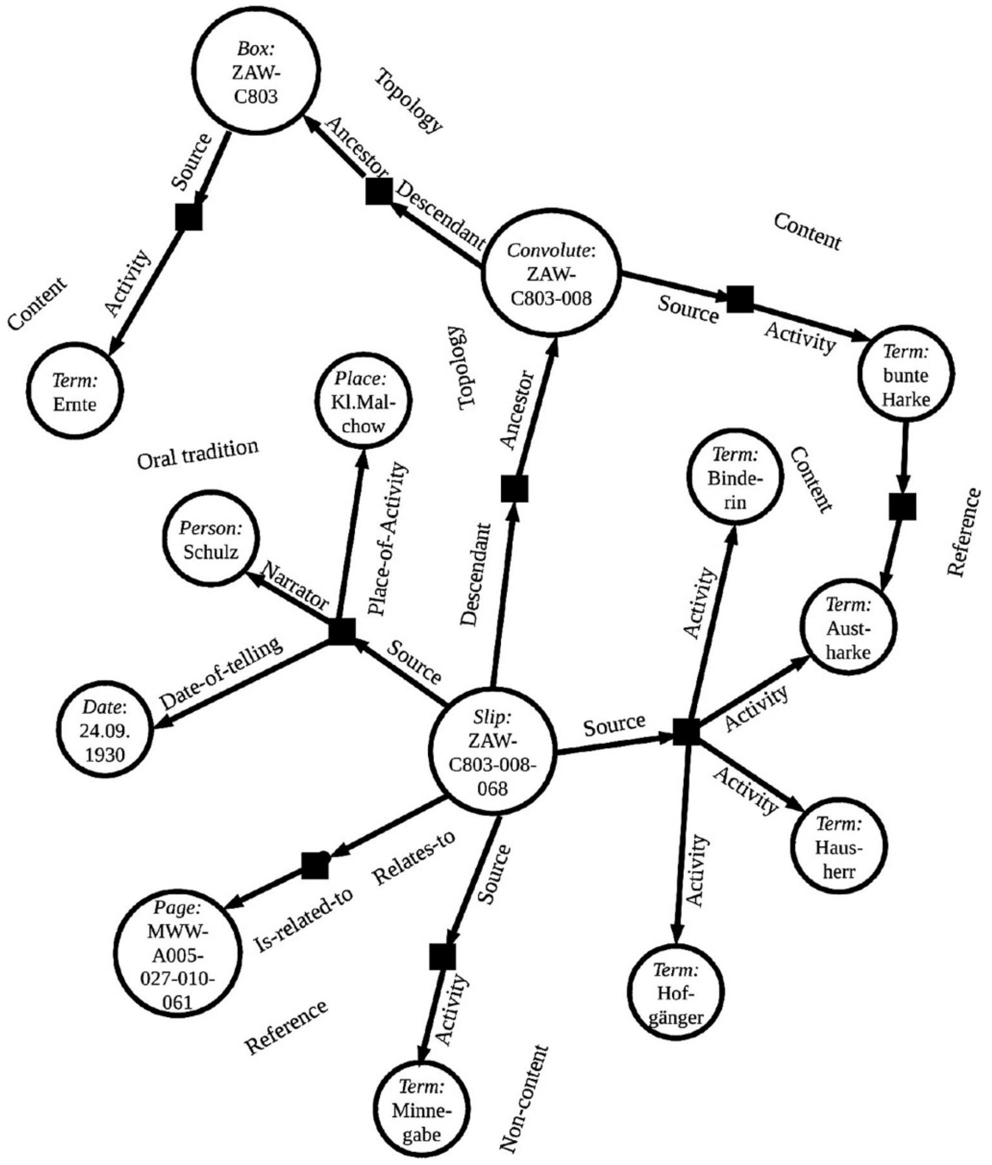


Fig. 6 Excerpt from the *WossiDiA* system: topology and activities connected with topic "Gabentausch" (*German* for gift exchange) modelled with hyperedges

A central observation from network science for the application of hypergraphs to heterogeneous cultural data is that local phenomena are repeated within a huge set of similar subgraph with slightly different peculiarities.

Another powerful feature of the hypergraph model for folklore is that standard relations can be modeled efficiently, while avoiding the limiting factors of standard relational databases[2]. For any arbitrary query, one wants to be able to construct a summary document representing the information needs of that query, such as: What is the network of contributors in the Wossidlo archive and how does that network depend on context (location, time, topics, social background, etc) over time? Since no single document exists to answer this question, the construction of such a summary can grow rapidly in complexity. In graph theory, this summary operator is related to the concept of graph minors [13] which can be constructed from a graph by applying sequences of edge removals, edge contraction and deleting singular nodes. Importantly, the hypergraph model supports both content and structure based queries through typed nodes and hyperedges and provides adequate operators for querying both.

2.4 Directed and typed hypergraphs

The *WossiDiA* hypergraph model (cf. Sect. 3) consists of directed and typed hypergraphs. Direction adds more semantics to the hyperedges, where a directed hyperedge can be seen as a mapping of a node set to another node set, which is similar to functional dependencies, like one-to-

[2] Missing adequacy, i. e., most concepts of graph and semi-structured data models are not supported in querying such data relational systems using SQL.

many or many-to-many relationships in more common relational database models.

A major addition to the *WossiDiA* hypergraph model [43] is the typing of nodes and hyperedges, which groups similar objects (nodes within the hypergraph) with similar properties, e. g., persons, places, datetime, tags, etc., and stores them more efficiently in specialized node containers. Query performance is improved by search paths and optimized type-specific operators. For example, top-k-queries return the k best spatial objects stored as nodes of type `place` which are near a certain place. The *WossiDiA* hypergraph system also uses typed hyperedges resembling the content models of semi-structured data in XML. Regarding query evaluation, there are two modes, typed and untyped querying. The first mode restricts graph processing to certain node and hyperedge types, e. g., if one is only interested in information about the places a story was recorded at.

2.5 Querying hypergraphs

Since each hypergraph can be represented by a bipartite graph (cf. Sect. 3.1), operators and algorithms for normal graphs can be exploited for hypergraph processing. In addition, there are numerous specialized hypergraph algorithms, such as the k-shortest hyperpath algorithm [25] which take advantage of the hyperedge concept for fast search. Additional algorithms include shortest paths computation, node connectivity testing, cycle detection using the specialized Graham reduction, sub-graph matching, node and hyperedge contraction, and some forms of graph products. These operators are used as building blocks for more descriptive graph query languages like GraphQL [33] or SPARQL [49].

2.6 Requirements leading to *PowerGraph* concepts

In the end, requirements from the project setting led to the development of *Hydra.PowerGraph*. The following list shows a mapping of those requirements to *PowerGraph* concepts:

- The system should support various appropriate metadata representations, e. g.:
  - structural metadata describing the archive topologies, e. g. slip, envelope, box, shelf (will be mapped to different node and edge types in *PowerGraph*),
  - administrative and technical metadata on digital artefacts (node types),
  - descriptive metadata for persons, places, events, activities (node types)
- The ethnological thesaurus or system of concepts classifying the field research notes (edge types: part-of, is-related-to, ...)
- Setting everything into the right historical context by spatial data, timespans with granule (node and egde types)
- Representing fuzzy data and uncertainty (special edge types)
- Relating all contributors and observers from the field research network (linking, edge types)
- Mashup with other systems like OPACs, document repositories, GOV and GND databases (node types)

Next, we will see to what extent a hypergraph based data model can cope with these requirements.

3 A Data Model Based on Directed Typed Hypergraphs

There are several reasons for using directed hypergraphs as a logical data model for the large amount of highly interconnected[3] data in the *WossiDiA* digital archive.

Nodes are used for modeling entities or objects of the application domain, whereas hyperedges represent relationships among a set of nodes. The hyperedge arcs (or links) represent different roles an object (node) can play in a certain relationship. Thus, nodes represent the ground truth or the static information particles while hyperedges are more on the dynamic and evolving side of information representation.

In the *PowerGraph* system, the directed hypergraph model is typed: nodes and hyperedges are associated with a type. Nodes, arcs/roles and hyperedges can have attributes, which are typed, too.

3.1 Introducing Directed Hypergraphs

3.1.1 Undirected Hypergraphs

Following Claude Berge [11] a hypergraph is a family $(E_1, ..., E_m)$ of non-empty subsets of nodes or vertices V, i. e. $G = (V, E)$ with $E \subseteq P(V) \setminus \emptyset$ (cf. Fig. 8a)[4]. The incidence graph is a bipartite graph with hyperedges mapped to node partitions and incidences mapped to edges (Fig. 8b). Undirected hypergraphs have applications in database schema design, social network (peer-groups) analysis, protein complex networks, chemical reaction and many other life science problems. Fig. 7 shows a simple typed but undirected hypergraph from the *WossiDiA* system.

[3] The archive consists of about two million paper slips containing field research notes, which in turn contain up to twenty million references or links to other field notes, in fact represented as a sparsely populated graph.

[4] $P(V)$ is the powerset, the set of all subsets, of V.

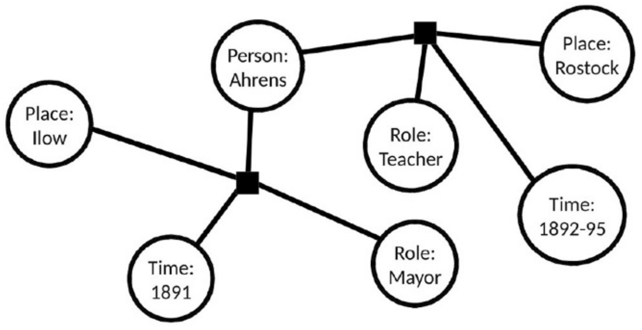


Fig. 7 Activities involving person, place, timespan, and role modelled as hyperedges

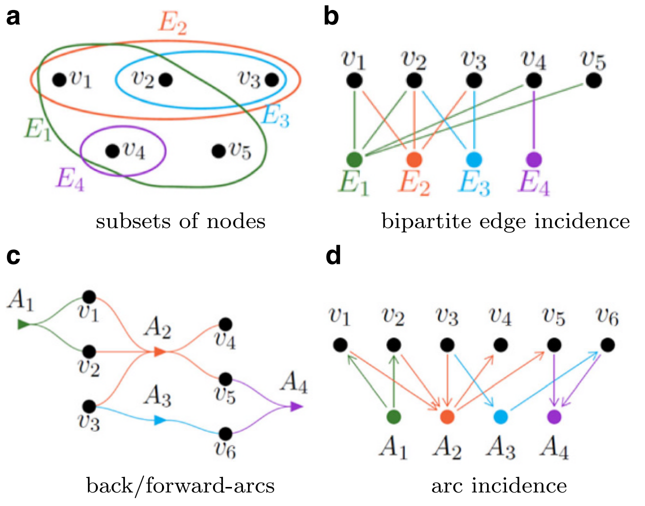


Fig. 8 Different representations of hypergraphs. **a** subsets of nodes, **b** bipartite edge incidence, **c** back/forward-arcs, **d** arc incidence

3.1.2 *Directed Hypergraphs*

Directed hypergraphs (Ausiello et al. [7–9]; Gallo et al. [22, 23]) are a generalization of directed graphs (*digraphs*) and they can model binary relations among subsets of a given set. The basic idea in the generalization of digraphs is the subdivision into source and destination, which is trivial in common digraphs since both consist of exactly one element each.

A directed hypergraph then can be defined as $G = (V, A)$ with a set of vertices V and the set of hyperarcs (hyperedges) A, where a hyperarc is a pair (T, H), T and H being subsets of vertices V. T is called the b-arc (backward-arc or tail) and H the f-arc (forward-arc or head) of the hyperarc. Similar to undirected hypergraphs, a directed hypergraph can be visualized using hyperarcs (Fig. 8c) or based on arc incidence (Fig. 8d).

Additionally, Gallo postulates T and H to be disjoint, i. e., $T \cap H = \varnothing$ but allows both to be empty sets. In contrast, Ausiello defines a hyperarc to be a pair (T, h)

Table 1 Comparison of different directed hypergraph definitions

Approach	B-arc	F-arc	Empty	Disjoint
Gallo	$T \subseteq V$	$H \subseteq V$	+	+
Ausiello	$T \subseteq V$	$h \in V$	–	+/–
Power Graph	$T \subseteq V \times R$	$H \subseteq V \times R$	+	–

with $h \in V$, $T \neq \varnothing$, and does not exclude $h \in T$. The destination set is restricted to one vertex. Gallo's definition [23] is more general since it allows for m:n-relationships whereas Ausiello [9] restricts to m:1-relationships. In this sense, Ausiello is a special case of Gallo.

3.1.3 *Directed Hypergraphs in* PowerGraph

The definition of a hyperedge[5] follows Gallo but with an essential extension: we introduce the concept of roles in a hyperedge, to allow a vertex or node to participate in a hyperedge multiple times as long as the role differs. With the set of roles R, we define the hypergraph G as a pair (V, A) with a set of vertices or nodes V and a set of hyperedges $A = (T, H)$ with $T \subseteq V \times R$ and $H \subseteq V \times R$, i. e., incidence is defined not only on nodes but (role, node) pairs. Furthermore, T and H are not disjoint. There can be nodes which are element of both, T and H, i. e., if $(v, r) \in T \cap H$ the incidence is bi-directional.

Table 1 compares our definition of a directed hypergraph with that of Gallo and Ausiello.

3.1.4 *Directed and Typed Hypergraphs in* PowerGraph

The type system of the directed hypergraph model is defined by sets of node types Γ^V, edge types Γ^A, roles (link types) R and a set of mappings $\{\alpha^V, \alpha^A, \rho^V, \rho^A, \delta\}$ with: $\alpha^V : V \longmapsto \Gamma^V$, $\alpha^A : A \longmapsto \Gamma^A$, $\rho^V : R \longmapsto \Gamma^V$, $\rho^A : R \longmapsto \Gamma^A$, and $\delta : R \longmapsto \{-1, 0, 1\}$. δ defines the direction for b-arcs as -1, f-arcs as 1 and as bi-directional if 0.

Following integrity constraints must hold for the types of a hyperedge:

- $\forall (v, r) \in T \cup H : \rho^V(r) = \alpha^V(v)$
- $\forall (v, r) \in T \cup H : \rho^A(r) = \alpha^A(A)$
- $\forall (v, r) \in T : \delta(r) \in \{-1{,}0\}$
- $\forall (v, r) \in H : \delta(r) \in \{0{,}1\}$

Aside from this more formal definition, we will now describe how typed and directed hypergraphs in *PowerGraph* can be used for data modeling based on examples from the *WossiDiA* system.

[5] Gallo calls them hyperarcs.

3.2 Data Modeling Using Directed Hypergraphs

Existing graph database systems didn't care much about node and edge types to allow for flexibility in modeling semi-structured data of various formats. In *Hydra.PowerGraph* we make use of typed information. By introducing node and hyperedge types, special data structures and associated operations can be exploited in the database system. As a result, the user gains efficiency and effectiveness during query evaluation and execution. He benefits from a rich set of type-specific operations in query formulation.

Since the *WossiDiA* graph database is built on top of an object-relational database system, PostgreSQL, operations on certain node types like point in time, time-span and temporal relationships (e. g., before, after, within, concurrently) can be used. Operations on spatial data (point, line segments, polygons derived from GPS-coordinates) and spatial relationship querying (e. g., contains, overlaps, touches, crosses, intersects) can be seamlessly integrated into the data model.

These advantages come at the cost of defining and implementing application specific types before using the system for storing and retrieving. From the user supplied data type definition, explicit data structures (in fact database tables) for each node type can be created separately. The content of the hyperedges created by the user is checked against the hyperedge type. On the one hand, the system can assist in creating hyperedge information by suggesting relevant (allowed) node types participating in a certain hyperedge. On the other hand, the system can ensure that all aspects (nodes) are inserted into the database, no data is omitted.

When querying the archive the user doesn't have to know about node and hyperedge types, but he can make use of certain query operators provided by *PowerGraph*.

3.2.1 Basic Node Types for Storing Domain Specific Data

Nodes in the hypergraph model represent entities, events, material or immaterial objects in the application domain. The basic node type has just an identity. All other node types are derived from this basic node type and must be specified by the user. Nevertheless, the *Hydra.PowerGraph* system comes with a set of built-in node types commonly used in historic information systems namely the Person, Place, and Word node types. The definition of the person node type is exemplified below:

```
CREATE NODETYPE person (
   firstname STRING,
   lastname STRING,
   birthdate TIMESPAN,
   appellation STRING
);
```

All node types inherit attributes of their super node types and add an individual set of attributes. Node types are identified by their unique name. The attributes are also typed and make use of the data types supplied by relational database systems like numbers, strings, date, timestamp, etc., but can benefit from extensions like PostGIS for spatial data, too. The *PowerGraph* system takes care of creating separate database tables for each node type and handles query evaluation using the extended node type-specific attribute set.

3.2.2 The Hyperedge Content Model (HECM)

The idea of not only having user defined node types but also introducing hyperedge types is to enforce restrictions on nodes participating in a hyperedge. In fact, by defining hyperedge types, the user effectively manages the meaningful relationships nodes (or objects, entities) can be part of.

With hyperedges, two aspects are controlled: (1) which nodes with certain types can participate, and how they participate (direction), (2) how the overall hyperedge type is constituted by the participating nodes and a postulated content model.

A hyperedge content model resembles the idea of XML content models or semi-structured data in general. The main difference is in dropping the document order property essential to semi-structured data. There is no ordering of nodes within a hyperedge on the type level. Nevertheless, an application can use link attributes to implement order if necessary.

The basic elements of the content model are the participating incoming and outgoing nodes and their node type and role name. The content model is built from the set of (role, node)-pairs by allowing arbitrary combinations of them using sets (`','`) or unions (`'|'`), quantifiers (`'?'`, `'*'`, `'+'`, exact cardinalities: `min-occurs` and `max-occurs`) and grouping using `'('` and `')'`. Formally, the content model is defined by a context free grammar.

The following depicts an example content model for activities as used in *WossiDiA*. An activity encompasses date, location, actors, observers, and a description of the activity, it can be defined in *GrafL* DDL like:

```
CREATE HYPEREDGE TYPE activity ...
    MODEL (
 start:date IN, end:date IN,
 location:place? IN, actor:person* IN,
 action:descr OUT, observer:person? OUT
);
```

3.2.3 *Types in Transit – Type Checking Hyperedges*

If we initially create a hyperedge with

```
CREATE HYPEREDGE activity (
   start:date("2016/10/31"),
   action:descr("Halloween"),
);
```

it is an incomplete hyperedge since the mandatory `end:date` is still missing. After adding this `end:date` link it is already a complete hyperedge, i. e., the current type matches the hyperedge content model. Nevertheless, it is not a closed hyperedge since we can still add some `actor`, `place`, or an `observer` link.

To cope with incomplete types, we introduce a so called *remainder type*. Whereas the content model of a hyperedge is the same for all instances of the same hyperedge type, the remainder type represents the nodes which can still be added until the hyperedge is closed. The current type of a hyperedge is one of the following states:

(1) `incomplete` if not all required links are present, or usually the hyperedge is an empty set of (link, node)-pairs,
(2) `complete` if the current type matches the content-model but the remainder type is not empty, i. e., if one can still add some nodes, and
(3) `closed` if nodes can not be added anymore.

As a consequence, there are some content models which will be permanently open regardless how many links we add to the hyperedge. Only content models not containing any link (or link group) with quantifier `'*'` and `'+'` can be closed. Closed hyperedge types are easier to be stored, indexed, type checked and retrieved. Open hyperedge types are more flexible and better suited for semi-structured data but need more effort in management and dynamic type checking.

Hyperedge type checking takes place every time we add or delete a link from a hyperedge. Before adding the first link to or after deleting the last link from the current hyperedge, the type state will be `incomplete`. If the current type tree[6] is equivalent to the content model tree of the related hyperedge type, type checking succeeds and state will be `complete`. To hyperedges in state `complete` links can still be added until their remainder tree gets empty and in turn the hyperedge type gets `closed`. To hyperedges in state `closed` no links can be added, only links can be deleted from this hyperedge.

[6] The data structure used for the remainder type is an and-or-tree [14] with grouping nodes and quantifier annotations as edge labels, and used also for the current type and the content model.

The basic algorithm for adding links will delete the first occurrence of the link from the tree if there is no quantifier or `'?'` annotated. Links with `'*'` or `'+'` will remain in the tree. After deleting a link from the remainder tree a check for empty groups will eliminate the whole subtree. If the remainder tree becomes empty the type state will be set to `closed`. Deleting a link will add the last occurrence of that link from the content model to the remainder tree.

Since the content model is expressed in a regular language which is a subset of the original UNIX regular expressions [58], it can be checked in $O(n)$ time for n links in the current hyperedge by compiling both the remainder type and the content model into a deterministic finite automaton [40]. The implementation of the dynamic type checking mechanism as implemented in *WossiDiA* is described in [60].

3.3 Graph Querying

Currently, *PowerGraph* supports a descriptive query language *GrafL* based on a set of graph algebra operators. These operators belong to three basic classes:

- content and structure predicates
- type-checking operations
- graph operations

The query language encompasses both content and structure querying, which select nodes and hyperedges based on predicates over their attribute set. Additionally, the query language allows for hyperedge matching by predicates formed over role and/or node types. Simple graph operations allow for testing the connectivity of two nodes, querying the k-neighborhood of certain nodes or implement hyperedge contraction as a union of all (role, node) pairs of two hyperedges.

Complex graph operations can be implemented using these basic operations, e. g., a synopsis operator will be evaluated as a combination of node and edge contractions and structure retrieval. A k-shortest path operator uses a special hypergraph algorithm [24].

The query language *GrafL* is similar to XQuery in querying node, hyperedge content and structure. The graph algorithms are encapsulated into functions. Where XQuery operates on sequences of XML fragments, *GrafL* is evaluated on node and hyperedge sets and usually returning hyperedge sets. To demonstrate the querying features of *GrafL*, we present some examples from the *WossiDiA* application domain.

Queries which retrieve nodes or edges by their content or structure use variables. These are bound to the node or hyperedge sets or some subtypes of them. First, this simple query returns all hyperedges of type `Narrative`:

```
LET $e := Edge::Narrative
RETURN $e
```

The following content-based query returns all nodes Person born in "Rostock":

```
LET $p := Node::Person,
    $o := Node::Place,
    $e := Edge::Vital-stats
WHERE $e.person = $p
   AND $e.birthplace = $o
   AND $o.name = "Rostock"
RETURN $p
```

The following structure-based query returns the node type of those nodes having an appellation and coordinates attribute:

```
LET $p := Node
WHERE $p.appellation AND $p.coordinates
RETURN $p.type
```

Graph algorithms which are currently supported in *PowerGraph* include node-connectivity, shortest-paths, and k-neighbourhood. This query returns the shortest path (as a list of hyperedges) connecting persons "Ahrens" and "Helms":

```
LET $p1 := Node::Person,
    $p2 := Node::Person
WHERE $p1.last-name = "Ahrens"
  AND $p2.last-name = "Helms"
RETURN shortest-path($p1, $p2)
```

The k-neighbourhood operator returns k = 10 nearest neighbours nodes of type Person of person "Richard Wossidlo":

```
LET $p = Node::Person
WHERE $p.fist-name = "Richard"
  AND $p.last-name = "Wossidlo"
RETURN k-nearest
   -neighbour($p, Node::Person, 10)
```

The last query constructs a new set of hyperedges from existing node sets. For each person nodes $p all places nodes $o are collected from hyperedges $e representing the place of living.

```
FOR $p IN Node::person
LET $e := Edge::livingplace,
    $o := node-set($e.place)
WHERE $e.person = $p
RETURN EDGETYPE allplaces
  MODEL (who:person, where:place+)
  VALUES(who:$p, where:$o)
```

For more details of the graph data definition, manipulation and querying features of *GrafL* can be found in [46].

4 Architecture and Implementation of the Hypergraph Database System

4.1 Object-Relational Storage Structures for Hypergraphs

Using a model, in our case a hypergraph model, always raises several questions around implementation issues, beginning from "how can hypergraph data be represented in the digital archive system", through "how to store data persistently" to "how to query and evaluate those data efficiently". Those problems are quite complex; answering these questions in detail is way beyond the limitations and goals of this article. Nevertheless at least one of these aspects will be presented here briefly: storage of hypergraph data. As mentioned before, we use the object-relational database system PostgreSQL as the backend for the graph database. The relational storage structure is shown in Fig. 9.

The backend database is divided into two parts, a catalog schema and a data schema. This helps to distinguish and organize type and model information on the one hand from the actual data on the other hand.

Besides the nodes in the hypergraph, which do not differ fundamentally from nodes in graphs, and therefore are easy to understand, the most important and structurally most relevant constituents of hypergraphs are their hyperedges and the connectors between hyperedges and participating nodes, which we call arcs (without loss of generality). These data are represented on the right side of the figure.

To accommodate further information about edges and arcs, the model allows for attributes to be attached to both of them (EdgeAttr, LinkAttr), on the far right in the figure. Since we have to deal with a single graph only, there is no need to store graph properties, such as ID and type at all. Relationships between attributes and edges/arcs, as well as the relationship between arcs and edge are guaranteed by referential integrity.

All hypergraph constituents, not only edges and arcs, even their attributes are typed, as shown in the catalog (left) part in the figure. Attribute types are always flat, however edge and node types are part of an inheritance hierarchy (therefore they have parent attributes). This makes the definition of hyperedge content models much easier. Hyperedge content models, described in more detail in Sect. 3.2, are important properties of hyperedge types. They define the

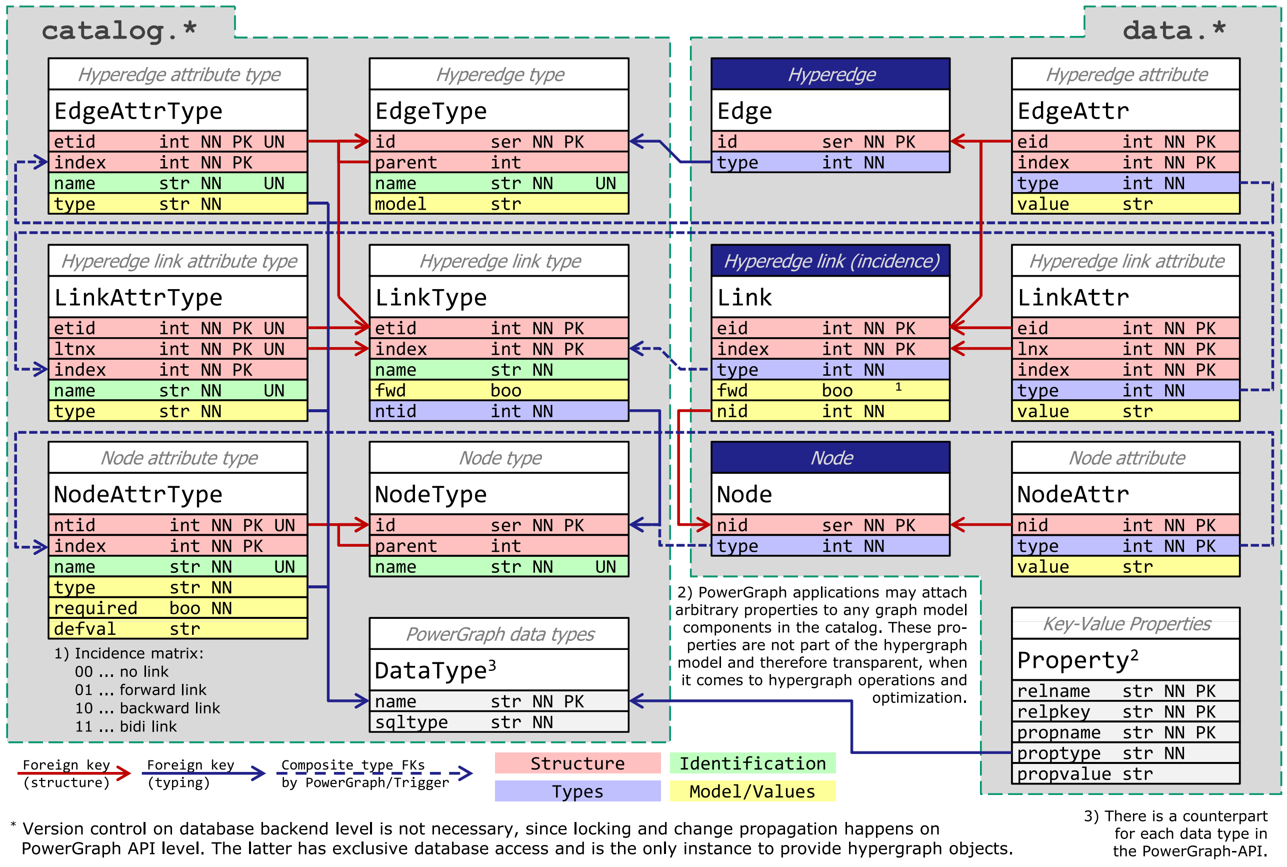


Fig. 9 Relational storage structure of the directed hypergraph model

way nodes participate in hyperedges they are associated with, hence defining the semantics of the hyperedge.

Inheritance of node types also implies inheritance on node storage, which means they all inherit from a base node which resides in the node table. As can be seen in the figure, node attributes are separated from node identities, which makes the system very flexible and independent from DDL operations and database system restrictions on inheritance, when it comes to modeling inherited nodes. However, this does not necessarily represent the implementation in the database. An alternative would hold tables on a per-node-type basis and hence make use of the inheritance capability of PostgreSQL tables and individual specific attribute type support[7]. A tradeoff between those two storage options would be the split-attributes-by-type approach, which separates attributes of individual types from node identities in their respective own specific data type tables, retaining the database system's type-specific capabilities, while providing a very flexible storage for attributes of inherited nodes. The decision on how to handle inherited nodes' storage has not yet been made and is still subject to research.

[7] Another reason for using PostgreSQL are the PostGIS extension for spatial data and the range type supporting temporal operators [3].

When running a *PowerGraph* instance, the whole catalog will be loaded into memory, which is inevitable for all hypergraph operations processed in *PowerGraph*. Catalog/model changes in the system are made persistent in the backend database immediately, to ensure ACID properties, using the respective backend database system capabilities.

For the sake of high efficiency, parts of the data schema are also cached into main memory. This does not include node, edge, or arc attributes, as they may include large amounts of data, especially nodes. In any case, it is advisable to cache node identities, edge identities, and incidence information, including arc directions, to provide for efficient graph operations. Initial strategies and ideas on how to do that, are described in the following section.

4.2 Linking Cache – Main Memory Graph Representation

The linking cache is a main memory structure taking care of loading graph data and writing back updated parts to the backend database. It exploits two main data structures: (1) a so called plex for graph, node, and hyperedge access and (2) an union-find index for fast node connectivity testing.

4.2.1 *The plex implementation of a hypergraph*

The plex[8] is a simple but efficient interface to a variety of graph or tree structured datasets. It consists of four basic sets:

- containers, set of plexes where the current plex occurs
- contents, set of plexes contained in the current one
- sources, set of plexes the current plex originates from
- destinations, set of plexes the current one is pointing to

Table 2 shows how graphs, hyperedges, and nodes can be represented in the same plex structure:

Operations on the plex structure have to be valid under two general, global integrity constraints

$$plex_i \in plex_j.containers \Leftrightarrow plex_j \in plex_i.contents$$
$$plex_i \in plex_j.sources \Leftrightarrow plex_j \in plex_i.destinations$$

and some validity rules specific for graphs, hyperedges, or nodes. These specific constraints are shown together with a sketch of the Java classes using generics for implementing the plex structure.

```
class plex<T1,T2,T3> {
   set<T1> containers;
   set<T2> contents;
   set<T3> sources;
   set<T3> destinations;
};
  class graph extends plex
  <null,edge,null> {
     //set<edge> contents;
```

$this \in edge.containers$

```
};
class edge extends plex
  <graph,node,node> {
     //set<graph> containers;
```

$this \in graph.contents$

```
     //set<node> sources;
```

$this \in node.destinations$

```
     //set<node> destinations;
```

$this \in node.sources$

```
};
  class node extends plex
  <edge,null,edge> {
    //set<edge> containers;
```

$this \in edge.contents$

```
     //set<edge> sources;
```

$this \in edge.destinations$

```
     //set<edge> destinations;
```

$this \in edge.sources$

```
};
```

[8] We could not track where the idea of the plex structure originally came from, but at least David Matuszek is an online source for it: https://www.cis.upenn.edu/~matuszek/.

Table 2 How graphs, hyperedges, and nodes can be represented in the same plex structure

Set	Graph	Edge	Node
containers	∅	`set<graph>`	`set<edge>`
contents	`set<edge>`	∅	∅
sources	∅	`set<node>`	`set<edge>`
destinations	∅	`set<node>`	`set<edge>`

4.2.2 *The Union-Find Data Structure for Node Connectivity*

This data structure [10] is used in addition to the plex structure for graph operations which depend on fast node connectivity testing. Using path compression, it performs well not only on connectivity testing ($O(lgN)$), but also on the union operation connecting two nodes ($O(lgN)$) or deleting a connection ($O(1)$). For secondary storage, there are also efficient data structures for testing connectivity (cf. [59]).

5 The *PowerGraph* Application Context

The relationships between the *PowerGraph* system, the *Hydra.PowerGraph* extension, and the *WossiDiA* digital archive are shown in Fig. 10 and explained in the following subsections.

5.1 *Hydra.PowerGraph*

In this paper, we mainly discuss the graph database system *PowerGraph* which is part of the *Hydra* framework for digital archive systems. *Hydra* adds a rich set of pre-

Table 3 *WossiDiA* topology hypergraph quantity structure

Archive topology nodes and hyperedge counts:						
Corpus	Lev.0	Lev.1	Lev.2	Lev.3	#nodes	#edges
				Lev.4		
zaw	1,112	29,728	920,749	981,064	1,932,654	951,590
mww	108	3,157	20,471	26,062	433,461	49,799
				383,662		
mwt	127	753	1,208	394,813	396,902	2,089
bkw	1,653	8,979	12,963	49,434	73,030	23,596
ztw	50	819	26,788	28,307	55,965	27,658
fna	16	956	1,687	5,088	7,748	2,660
pwa	1	6	140	2,366	2,514	148
lit	1	2	299	1,086	1,389	303
ibw	1	1	306		309	3
och	2	2	20	223	248	25
Topology (incl. corpus and archive nodes/hyperedges):					2,904,221	1,057,872
Metadata					257,468	339,832
TOTAL					3,161,689	1,397,704

defined node and hyperedge types, like the one discussed above. It is accompanied by a flexible workflow engine and digitization and publishing modules [54].

5.2 *WossiDiA* on Top of *Hydra.PowerGraph*

WossiDiA is a project about digitization of ethnological documents, written and collected in the late 19th and the early 20th century by Richard Wossidlo, see Fig. 1. Fig. 2 shows a sample of Wossidlo's hand-written documents.

More detailed information about the collection and the project can be found in [43] and [52]. In order to illustrate the type of documents, the complexity of the digitization workflow, and the need of a sophisticated way to describe digital archive topology data with metadata information, connected in a hypergraph type of way, we have included Fig. 4. It gives a general overview about the workflow and depicts how archival contents are acquired and amended in a way that generates the high degree of interconnectivity in the digital archive. The digitization process is outlined in [53].

Table 3 gives an overview of the current hypergraph data quantity structure of the Digital Wossidlo Archive's *PowerGraph* instance. The upper part of the table shows the number of nodes and hyperedges in the archive topology, which is a hierarchical structure, consisting of the corpora of the Wossidlo Archive (e. g.: main box shelf with field research notes (ZAW), correspondences with contributors (BKW), alphabetical dictionary source documents (MWW), etc.). The numbers of nodes on the individual levels of the different corpora are listed explicitly to explain the quantity of hyperedges, which are used to hold together the corpora's topologies. Level 0 represents the highest directory unit, directly below the respective corpus's root node. Level 3 or 4 represents the actual digital image document nodes. The lower part of the table displays the overall node and hyperedge counts across all corpora. They are juxtaposed with the total numbers of all descriptive metadata nodes and all

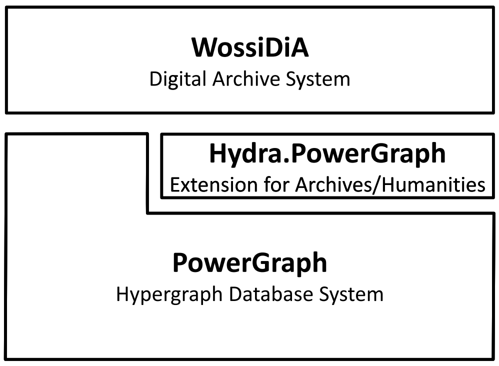


Fig. 10 *WossiDiA* as *Hydra.PowerGraph* application

Table 4 *WossiDiA* hyperedge types (selection of)

Hyperedge type	# edges	Avg. nodes/edge
[1019] Content	118,912	2.132
[70] Narrated	25,943	2.592
[100] Thesaurus	17,517	2.012
[5] Gen. reference	1,969	2.953
[31] Wossi region	122	22.574
[10] Folder	118	76.195
[1007] Fuzzy person	49	10.735
[1016] Seminar	28	4.607

descriptive hyperedges. The latter are used to link metadata nodes with topology nodes.

A few examples for specific hyperedges types, which are either prominent in *WossiDiA* or make heavy use of hyperedges, are shown in Table 4. The left column indicates the edge types, which themselves represent the meaning of facts behind the association of participating nodes. The middle column gives a hint on how many of these hyperedges have been created and the right column shows the average fan-out of the edge-node relationship.

For example, the hyperedge type *Wossi region*, which puts places into a regional context with a more prominent place nearby, has only a small number of edges but a huge fan-out of connected nodes. The same is true for *fuzzy person* and *seminar* networks. On the other hand, we have lots of edges describing document content (*content*) and information origin (*narrated*). Unlike the networks, they naturally don't have a huge fan-out; it is still greater than two, proving the need of hypergraphs over graphs with edges only allowing for a pair of two nodes. Hyperedge type *thesaurus* represents the so-called "ethnological canon" which relates terms mostly hierarchically, but features a few very important cross references, being the average +0.012 nodes on top of the regular pair.

6 Related Work

The CIDOC/CRM [18] is a conceptual model primarily designed for the exchange of historical information in the course of archives, museums, and libraries. It has to be mapped to a logical data model to be used for storage and retrieval. This has be done by Thalheim et al. for the relational model in [37]. More details on mapping the CIDOC/CRM to *Hydra.PowerGraph*'s directed and typed hypergraph model can be found in [38]. This approach was also used as a conceptual model in the Lagomar project which dealt with the inventarisation of the maritime cultural heritage of lagoon regions and employed CIDOC/CRM for data storage [42, 45]. In Lagomar, CIDOC/CRM was mapped to an RDF-triple-store (Jena Framework[9]).

Claude Berge [11] established the hypergraph theory. Directed hypergraphs were first introduced by Ausiello et al. [7–9] and Gallo et al. [22, 23]) as a generalization of directed graphs.

Angles and Guttierez summarize and compare graph based data models [4]. But the majority of recently developed graph database techniques and approaches does not focus on hypergraphs and does not put too much effort into complex type concepts for graph edges or hyperedges. However, there are some approaches dealing with hypergraphs, such as [16] and [20]. Some use labeled edges such as [16, 17, 29, 34, 35, 51, 61], etc. Although labeled edges can be counted as a name based type system, *PowerGraph* uses a structure based type system.

A very small number of approaches deals with both hypergraphs and typed edges, e. g. [16]. But even in this case, edge typing means no more than simply using labeled edges. When it comes to widely spread graph database and graph processing systems, such as neo4j and Apache Giraph, the native support of hypergraphs is limited. Building hypergraphs in neo4j means using special node types acting as hyperedges (bipartite graphs), and sophisticated edge types need to be implemented explicitly. As a consequence, query evaluation and optimization do not benefit from node clustering and typed edges.

In contrast to the above-mentioned approaches and systems, *PowerGraph* provides a complex hypergraph type system, which is based on hyperedge content models. It features the notion of applying restrictions on nodes participating in hyperedge relationships, as described in Sect. 3.2. Using this, the user can define complex constraints in a way similar to content models known from semi-structured data and documents.

With respect to the hypergraph model and edge types, *PowerGraph* uses a query language resembling XQuery. This approach is different to current graph languages like SPARQL since it mixes graph processing techniques with querying semi-structured data and documents. Angles et al. [5] survey the foundations of graph languages like SPARQL, Cypher and Gremlin. Since RDF-triples can be represented by bipartite graphs as well as hypergraphs can, there should be a way to adopt a SPARQL-like language for broader use in the *PowerGraph* system. Some researchers [31] argue that hypergraphs may better represent RDF graphs. RDF graphs allow for several representations, for example bipartite graphs [31], labeled directed graphs [28, 32], and hypergraphs [30].

As for managing XML data with relational technologies [26], there are situations where managing graph data with (object-)relational databases can perform as good as a native graph database. Gubichev [27] at least gives some evidences for simple graph matching scenarios.

The main memory structure of *PowerGraph*, the linking cache, relies on efficient main memory graph representations. Succinct data structures are a basic building block, [21, 36, 47] show more complex discrete data structures such as trees and graphs that can be built using them. Some of the tasks for which they have used include Web graphs [15], XPath indexing [6], partial sums [50], and short read alignment [39].

[9] https://jena.apache.org/

7 Conclusion

Primarily developed for the *WossiDiA* digital archive, the directed hypergraph model is versatile enough to build a foundation for not only this special archive structure but more general applications. For the realm of digital libraries, archives, and museums, we have shown that our model is well suited as implementation logic for CIDOC/CRM, a conceptual model well accepted in those domains.

The directed hypergraph model is not only a set of concepts for defining structures but is accompanied with a set of operators forming a graph query language and graph algebra. Such a graph algebra is also a formal basis for implementing efficient algorithms of algebra operators and developing an optimization framework. Nevertheless, a more descriptive query language like GraphQL, SPARQL or Cypher should be adapted to hypergraphs and implemented on top of such an algebra. *GrafL* is a first try on a hypergraph query language and borrows much from XQuery. The semantics and the optimization is still an open topic in the project and needs further research. Other open questions include more complex graph operators for graph abstraction and summarizing as well as customized index structures for frequent graph pattern matching.

Graph matching will be also a focus of a forthcoming project. It will provide intelligent search and analysis across three of the world's largest machine actionable folklore collections (Dutch Folktale Database from Meertens Instituut, Amsterdam, the Danish Folktale Database from UCLA, and the Mecklenburger Folklore Database *WossiDiA* from University of Rostock) presenting the opportunity for large scale data-driven research into traditional folk expressive culture. By facilitating search, discovery and analysis across all three collections, the *Hydra.PowerGraph* infrastructure will provide researchers an unprecedented opportunity to discover patterns both within and across the target corpora. The proposed research, focusing on storytellers, legends and the dispersion of (historical) beliefs in magic, witchcraft, hauntings and supernatural beings seeks to reveal what ordinary people believed, and how storytelling traditions and story repertoires differed in and across these three areas [1, 56, 57].

Acknowledgements The authors would like to thank Christoph Schmitt, Reinhard Kerb, and Stefanie Janssen of the European Ethnology and Wossidlo Archive as well as Timothy R. Tangherlini from University of California in Los Angeles, and Theo Meder from Meertens Instituut, Amsterdam. Also, we would like to thank all the students involved in discussing, implementing, testing, and using the*WossiDiA*system, namely David Vendt, Roland Kiesendahl, Rasha Mukbil, Md. Janghir Alam, Martin Lichtwark, and Steffen Sachse.

References

1. Abello J, Broadwell P, Tangherlini TR (2012) Computational folkloristics. Commun ACM 55(7):60–70. doi:10.1145/2209249.2209267
2. Abiteboul S (1997) Querying semi-structured data. Database Theory ICDT 97:1–18
3. Alam MJ (2016) Spatio-Temporal Operations in Digital Archive Systems. Masters thesis, University of Rostock, Germany
4. Angles R, Gutiérrez C (2008) Survey of graph database models. ACM Comput Surv 40(1):1–39
5. Angles R, Arenas M, Barceló P, Hogan A, Reutter JL, Vrgoc D (2016) Foundations of modern graph query languages. Comput Res Repos. arXiv:1610.06264
6. Arroyuelo D, Claude F, Maneth S, Mäkinen V, Navarro G, Nguyen K, Sirén J, Välimäki N (2010) Fast in-memory xpath search using compressed indexes. In: Li F, Moro MM, Ghandeharizadeh S, Haritsa JR, Weikum G, Carey MJ, Casati F, Chang EY, Manolescu I, Mehrotra S, Dayal U, Tsotras VJ (eds) Proceedings of the 26th International Conference on Data Engineering, ICDE 2010 Long Beach CA, March 1-6, 2010. IEEE Computer Society, Washington, DC, pp 417–428
7. Ausiello G (1988) Directed Hypergraphs: Data Structures and Applications. In: Dauchet M, Nivat M (eds) Proceedings CAAP '88, 13th Colloquium on Trees in Algebra and Programming, Nancy, March 21-24, 1998. Lecture Notes in Computer Science, vol 299. Springer, Berlin Heidelberg, pp 295–303
8. Ausiello G, Laura L (2017) Directed hypergraphs: introduction and fundamental algorithms – a survey. Theor Comput Sci 658:293–306. doi:10.1016/j.tcs.2016.03.016
9. Ausiello G, D'Atri A, Saccà D (1986) Minimal representation of directed hypergraphs. SIAM J Comput 15(2):418–431
10. Ben-Amram A, Yoffe S (2011) A simple and efficient Union-Find-Delete algorithm. Theor Comput Sci 412(4):487–492. doi:10.1016/j.tcs.2010.11.005
11. Berge C (1989) Hypergraphs – combinatorics of finite sets, 1st edn. North Holland, Amsterdam
12. Boncz PA, Larriba-Pey J (eds) (2016) Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24, 2016. ACM, New York. doi:10.1145/2960414
13. Brandstädt A, Le VB, Spinrad JP (1999) Graph classes: a survey. Society for Industrial and Applied Mathematics, Philadelphia
14. Chauvin B, Flajolet P, Gardy D, Gittenberger B (2004) And/Or Trees Revisited. Comb Probab Comput 13(4-5):475–497. doi:10.1017/S0963548304006273
15. Claude F, Navarro G (2010) Fast and compact web graph representations. ACM Trans Web 4(4):1–31. doi:10.1145/1841909.1841913
16. Das M, Simitsis A, Wilkinson K (2016) A hybrid solution for mixed workloads on dynamic graphs. In: Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems - GRADES ´16. ACM, New York. doi:10.1145/2960414
17. Dave A, Jindal A, Li LE, Xin R, Gonzalez J, Zaharia M (2016) Graphframes: an integrated API for mixing graph and relational queries. In: GRADES ´16 Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems. ACM, New York. doi:10.1145/2960414
18. Doerr M, Ore CE, Stead S (2007) The CIDOC conceptual reference model – a new standard for knowledge sharing. In: Grundy JC, Hartmann S, Laender AHF, Maciaszek LA, Roddick JF (eds) ER (Tutorials, Posters, Panels & Industrial Contributions). CRPIT, vol 83. Australian Computer Society, Sydney, pp 51–56
19. Fagin R (1983) Degrees of Acyclicity for Hypergraphs and relational database schemes. J Assoc Comput Mach 30(3):514–550. doi:10.1145/2402.322390

20. Firth H, Missier P (2016) Workload-aware streaming graph partitioning. In: Palpanas T, Stefanidis K (eds) Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016, CEUR Workshop Proceedings, vol 1558. CEUR-WS.org, Bordeaux
21. Fotakis D, Pagh R, Sanders P, Spirakis P (2005) Space efficient hash tables with worst case constant access time. Theory Comput Syst 38:229–248. doi:10.1007/s00224-004-1195-x
22. Gallo G, Scutella MG (1998) Directed hypergraphs as a modelling paradigm. Riv Mat Sci Econ Soc 21(1-2):97–123
23. Gallo G, Longo G, Pallottino S, Nguyen S (1993) Directed hypergraphs and applications. Discrete Appl Math 42(2):177–201
24. Gao J, Zhao Q, Ren W, Swami A, Ramanathan R, Bar-Noy A (2012) Dynamic shortest path algorithms for hypergraphs. In: WiOpt. IEEE, Washington, pp 238–245
25. Gao J, Zhao Q, Ren W, Swami A, Ramanathan R, Bar-Noy A (2015) Dynamic shortest path algorithms for Hypergraphs. IEEE ACM Trans Netw 23(6):1805–1817. doi:10.1109/TNET.2014.2343914
26. Grust T, Rittinger J, Teubner J (2008) Pathfinder: Xquery off the relational shelf. IEEE Data Eng Bull 31(4):7–14
27. Gubichev A, Then M (2014) Graph pattern matching: do we have to reinvent the wheel? In: Proceedings of workshop on graph data management, experiences and systems. ACM, New York, pp 1–7
28. Gutierrez C, Hurtado CA, Mendelzon AO (2004) Foundations of semantic web databases. In: ACM Symposium on PODS. ACM, New York, pp 95–106
29. Haubenschild M, Then M, Hong S, Chafi H (2016) Asgraph: a mutable multi-versioned graph container with high analytical performance. In: Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems - GRADES ´16
30. Hayes J (2004) A Graph Model for RDF. Dipoma thesis, Technische Universität Darmstadt, Germany
31. Hayes J, Gutierrez C (2004) Bipartite graphs as intermediate model for RDF. In: Proceedings of the 3th Int. Semantic Web Conference (ISWC), number 3298 in LNCS. Springer, Berlin Heidelberg, pp 47–61
32. Hayes PJ (2004) Rdf semantics. W3C Recommendation. http://www.w3.org/TR/rdf-mt/. Accessed: 22 Febr 2017
33. He H, Singh AK (2008) Graphs-at-a-time: query language and access methods for graph databases. In: Wang JT (ed) Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD Vancouver BC, June 10-12, 2008. vol 2008. ACM, New York, pp 405–418
34. Hölsch J, Schmidt T, Grossniklaus M (2017) On the performance of analytical and pattern matching graph queries in neo4j and a relational database. CEUR-WS.org. http://ceur-ws.org/Vol-1810. Accessed: 22 Febr 2017
35. Georgia Institute of Technology Stinger. Tech. rep. http://www.stingergraph.com/
36. Jacobson G (1989) Space-efficient static trees and graphs. In: 30th Annual Symposium on Foundations of Computer Science, 30 Oct.-1 Nov. 1989, IEEE Computer Society, Washington, pp 549–554. doi:10.1109/SFCS.1989.63533
37. Jannaschk K, Rathje CA, Thalheim B, Förster F (2011) A generic database schema for CIDOC-CRM data management. Adv Databases Inf Syst 2(789):127–136
38. Kiesendahl R (2014) Konzeptuelle Modellierung historischer Daten in digitalen, historischen Informationssystemen. Bachelor thesis, University of Rostock, Germany
39. Kimura K, Koike A (2009) Localized Suffix Array and Its Application to Genome Mapping Problems for Paired-End Short Reads. In: Morishita S, Lee SY, Sakakibara Y (eds) Proceedings of the 20th International Conference on Genome Informatics. Genome Informatics Series, vol 23. Imperial College Press, London, 2009, pp 60–71
40. Laurikari V (2000) NFAs with tagged transitions, their conversion to deterministic automata and application to regular expressions. In: Seventh International Symposium on String Processing and Information Retrieval, SPIRE 2000, A Coruña, Spain, September 27–29, 2000. IEEE Computer Society, Washington, pp 181–187. doi:10.1109/SPIRE.2000.878194
41. Levene M, Poulovassilis A (1991) An object-oriented data model formalised through hypergraphs. Data Knowl Eng 6:205–234
42. Meyer H (2008) Inventarisation of historical maritime landscapes – an information science point of view. In: Meyer H, Springmann MJ, Wernicke H (eds) The Lagomar lagoons – unique maritime cultural landscapes in a scientific focus and in an interdisciplinary comparison. Steffen, Friedland, pp 21–33
43. Meyer H, Schering AC, Schmitt C (2014) WossiDiA – The Wossidlo Digital Archive. In: Meyer H, Schmitt C, Janssen S, Schering AC (eds) Corpora ethnographica online. Waxmann, Münster New York
44. Meyer H, Schmitt C (2015) Semantische, räumliche und zeitliche Vernetzung regionalethnologischer Archive. In: Bolenz E, Franken L, Hänel D (eds) Wenn das Erbe in die Wolke kommt – Digitalisierung und kulturelles Erbe. Klartext, Essen, pp 61–86
45. Meyer H, Springmann MJ, Wernicke H (eds) (2008) The Lagomar lagoons – unique maritime cultural landscapes in a scientific focus and in an interdisciplinary comparison. Steffen, Friedland
46. Meyer H, Mukbhil R, Schering AC (2017) The Hydra.PowerGraph Data Definition, Manipulation and Query Language GrafL. CS-01-17. University of Rostock, CS Department, Germany
47. Pagh R, Rodler FF (2004) Cuckoo hashing. J Algorithms 51(2): 122–144. doi:10.1016/j.jalgor.2003.12.002
48. Pitoura E, Maabout S, Koutrika G, Marian A, Tanca L, Manolescu I, Stefanidis K (eds) (2016) Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016. Bordeaux, March 15–16, 2016.
49. Prud'hommeaux E, Seaborne A (2013) SPARQL 1.1 Query Language for RDF. In: W3C Recommendation (Tech. rep.; 26 March 2013)
50. Raman R, Raman V, Satti SR (2007) Succinct indexable dictionaries with applications to encoding *k*-ary trees, prefix sums and multisets. ACM Trans Algorithms 3(4):43. doi:10.1145/1290672.1290680
51. van Rest O, Hong S, Kim J, Meng X, Chafi H (2016) PGQL: a property graph query language. In: Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems - GRADES ´16
52. Schering AC, Bruder I, Schmitt C, Meyer H, Heuer A (2007) Towards a digital archive for handwritten paper slips with ethnological contents. In: Goh DHL, Cao TH, Sølvberg I, Rasmussen EM (eds) ICADL. Lecture Notes in Computer Science, vol 4822. Springer, Berlin Heidelberg, pp 61–64
53. Schering AC, Bruder I, Jürgensmann S, Meyer H, Schmitt C (2011) From box to bin – semi-automatic digitization of a huge collection of ethnological documents. In: Xing C, Crestani F, Rauber A (eds) Digital Libraries: For Cultural Heritage, Knowledge Dissemination, and Future Creation. 13th International Conference on Asia-Pacific Digital Libraries, ICADL 2011, Beijing, China, October 24-27, 2011. Lecture Notes in Computer Science, vol 7008. Springer, Berlin Heidelberg, pp 168–171. doi:10.1007/978-3-642-24826-9
54. Schick S, Meyer H, Heuer A (2011) Flexible publication workflows using dynamic dispatch. In: Xing C, Crestani F, Rauber A (eds) Digital Libraries: For Cultural Heritage, Knowledge Dissemination, and Future Creation. 13th International Conference on Asia-Pacific Digital Libraries, ICADL 2011, Beijing, China, October 24-27, 2011. Lecture Notes in Computer Science, vol 7008. Springer, Berlin Heidelberg, pp 257–266. doi:10.1007/978-3-642-24826-9
55. Schmitt C (2014) Szenarien semantischer Vernetzung zwischen regionalethnographischen und dialektlexikographischen Korpora

im Online-Projekt WossiDiA. In: Bühler R, Bürkle R, Leonhardt N (eds) Sprachkultur – Regionalkultur. Neue Felder kulturwissenschaftlicher Dialektforschung. TVV-Verlag, Tübingen, pp 255–286
56. Tangherlini TR (2013) The folklore Macroscope. The Archer Taylor memorial lecture. West Folk 72(1):7–27
57. Tangherlini TR, Broadwell PM (2016) WitchHunter: GeoSemantic browsing in a large folklore corpus. J Am Folklore 129(511):14–40
58. Thompson K (1968) Regular expression search algorithm. Commun ACM 11(6):419–422
59. Trißl S, Leser U (2007) Fast and practical indexing and querying of very large graphs. In: Chan CY, Ooi BC, Zhou A (eds) Proceedings of the ACM SIGMOD International Conference on Management of Data Beijing, June 12-14, 2007. ACM, New York, pp 845–856
60. Vendt D (2013) Hochvernetzte Archivstrukturen und NoSQL-Systeme. Bachelor thesis, University of Rostock, Germany
61. Wang J, Ntarmos N, Triantafillou P (2016) Indexing query graphs to speedup graph query processing. In: Pitoura E, Maabout S, Koutrika G, Marian A, Tanca L, Manolescu I, Stefanidis K (eds) (2016) Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016. Bordeaux, March 15–16, 2016.
62. Xing C, Crestani F, Rauber A (eds) (2011) Digital Libraries: For Cultural Heritage, Knowledge Dissemination, and Future Creation. 13th International Conference on Asia-Pacific Digital Libraries, ICADL 2011, Beijing, China, October 24-27, 2011. Lecture Notes in Computer Science, vol 7008. Springer, Berlin Heidelberg. doi:10.1007/978-3-642-24826-9