Proceedings of the Federated Conference on Computer Science and Information Systems pp. 173–177
DOI: 10.15439/2018F103
ISSN 2300-5963 ACSIS, Vol. 15

Named Property Graphs

Dominik Tomaszuk
Institute of Informatics, University of Bialystok
ul. Ciołkowskiego 1M, 15-245 Białystok, Poland
Email: d.tomaszuk@uwb.edu.pl

Łukasz Szeremeta
Institute of Informatics, University of Bialystok
ul. Ciołkowskiego 1M, 15-245 Białystok, Poland
Email: l.szeremeta@uwb.edu.pl

***Abstract*—The amount of information that is stored and processed by computer systems is constantly increasing. The relational model is still popular. Unfortunately, despite its simplicity, it has many disadvantages, which more often exclude it from large-scale applications. The property graph model seems to be a good alternative for describing real world data with its relationships. Therefore, property graph based databases become more and more popular every day. In this paper we introduce Named Property Graph model that allows to group graphs into separate units and describe information about them. We also present Cypher$_n$ query language that supports our proposal, mapping algorithms, use cases with the chemical data, and SDFEater that is our tool for processing data. Presented solutions are fully backward compatible with existing databases.**

I. Introduction and Motivation

THE amount of information that is stored and processed by computer systems is constantly increasing. The way they are stored becomes more and more important. The relational model is still popular. Unfortunately, despite its simplicity, it has many disadvantages, which more often exclude it from large-scale applications. Today, with much more attention is also looking at alternatives, e.g. graph and document databases [1], [2], [3], [4]. The graph data model is often very well suited for describing real relationships. It can be successfully used for example, on social networks. Presenting of the relationships between users, as well as the relationship of their posts, seems much more natural in this model. Users are represented by vertices, and the edges describe relationships between them. Using property graphs [5], we can additionally add some information about each person, and even that, from when they are friends.

Importantly, the property graph data model can be simply mapped back to other data models [5], [6], [7], [8]. This means that the presentation of data, for example in tabular and semi-structured form, is also possible.

The property graphs model [5], unlike the Resource Description Framework (RDF) [9], does not support named graphs [10]. In this article, we present our solution that allows to group property graphs, and give them a name and properties. This solution allows to describe graphs (context, provenance information, graph hashes and graph signatures or other metadata).

A similar approach was presented in [11], [12]. The authors propose graph collections that are logical partitions of a graph called logical graphs. These graphs are subsets of shared sets of vertices and edges. Unfortunately, logical graphs may have common vertices and edges, so it in not possible to unambiguously hash and sign these graphs.

Another approach was presented in [13] and [14]. In the first paper, Levene et al. present hypergraph which is a generalization of graphs where the concept of edge is extended to hyperedge, which relates an arbitrary set of nodes. In the second paper, the authors propose hypernodes that are directed graphs whose nodes can themselves be graphs, allowing nesting of graphs. Unfortunately, these both solutions do not support the property graph model, which is used most widely in databases.

In [15] the authors present the GOOD data model with directed labeled graph and object in database as nodes. In this model, nodes can be printable or not. In addition, two edge types are distinguished – functional and non-functional. Functional edges allow to define functional relationships between objects. GOOD data model supports edge and node labeling. Another data model was presented in [16]. The LDM data model is based on labeled directed multigraph which means graph that can have one and more edge between pair of nodes. All edges have one specific type: Basic, Product, Power or Union. LDM supports only node labels, edge labels are not supported. Unfortunately, GOOD and LDM data models do not support properties.

The GRAD data model presented in [17] is based on property graphs and extends property graphs by specific semantics. Ghrab et al. define different types of nodes, eg. entity, attribute, and literal node. Authors also introduced four types of entity edges such as association, generalization, aggregation, and composition edges. GRAD supports hypernodes which are represented as subgraphs. Unfortunately, authors do not provide any algorithms for mapping from their proposal into existing databases and query languages. Furthermore, hypernodes in GRAD do not support properties that can be used in storing metadata about subgraphs.

The paper is constructed as follows. In Section II we formalize Named Property Graph data model and propose how to map our approach into Property Graphs. Section III is devoted to a use case that presents our proposal in a molecular entities scenario. In Section IV we present tested datasets and our experiments. The paper ends with conclusions.

II. Named Property Graph

This section describes the Named Property Graph (NPG) model and shows how to map our proposal to property graphs

(Subsection II-A). Then, we discuss the possible use of our proposal. In Subsection II-B we present Cypher$_n$ that is a query language for NPGs. Then, we show a mapping algorithm that transforms our proposal into openCypher [18].

Our following proposal allows to group graphs into separate units and describe information about them. The ability to express metainformation about graphs can be required for:

- access control – ability to add additional metadata for precise access control,
- information usage control – ability to add additional metadata about authorship, license, and policy to graph in order to limit information usage,
- data syndication – ability to store and update original information,
- graph singing – allows to apply good practices, where all singing data is kept in a different graph,
- aggregation or encapsulation of graph elements – provide a wider view of the graph as it enables a higher level design and analysis,
- expressing propositional attitudes – such as trust and temporal metrics.

Potential drawbacks of Named Property Graph model is data redundancy. However, there are mechanisms for removing redundant vertices [19], [20].

A. Named Property Graph Model

According to [5], we provide a formal definition below.

Definition 1 (Property Graph): A *Property Graph* is a tuple $PG = \langle V, E, S, P, h_e, t_e, l_v, l_e, p_v, p_e \rangle$, where:

1) V is a non-empty set of vertices,
2) E is a multiset of edges, which are elements of $V \times V$,
3) S is a non-empty set of character strings,
4) P is a Cartesian product $S \times S$, where each member has a form $p = \langle k, v \rangle$,
5) $h_e : E \to V$ is a function that yields the source of each edge (head),
6) $t_e : E \to V$ is a function that yields the target of each edge (tail),
7) $l_v : V \to S$ is a function mapping each vertex to a label,
8) $l_e : E \to S$ is a function mapping each edge to a label,
9) $p_v : V \to 2^P$ is a function that assigns vertices to their multiple properties, and
10) $p_e : E \to 2^P$ is a function that assigns edges to their multiple properties.

We propose a general and simple variation on PG model, called Named Property Graphs. A named property graph is a property graph which is assigned a name (label), and properties. A name should be in the form of a string, and properties should be in the form of a set of key-value.

Definition 2 (Named Property Graph): A *Named Property Graph* is a tuple $NPG = \langle PG, N, l_n, p_n \rangle$, where:

1) PG is a Property Graph (see Definition 1),
2) N is a non-empty set of named nodes,
3) $l_n : N \to S$ is a function mapping each named node to a label, and
4) $p_n : N \to 2^P$ is a function that assigns named nodes to their multiple properties.

Algorithm 1: Mapping Named Property Graph into Property Graph

input : Named Property Graph NPG
output: Property Graph PG

```
n ← getName(NPG) ;
foreach g ∈ NPG do
    v ← getVertex(g) ;
    PG ← addVertex(v) ;
    PG ← addEdge(n, "related", v) ;
return PG;
```

Named nodes, unlike vertices, cannot connect to each other using edges. The set of all PG and NPG graphs is the Named Property Graph Database that allows to group graphs into separate units and describe information about them.

Definition 3 (Named Property Graph Database): A *Named Property Graph database* consists of a (possible empty) set of Named Property Graphs (with distinct labels) and a set of Property Graphs.

In order for our solution to work on current databases, we show the transformation of our NPG into PG in Algorithm 1. The algorithm adds one additional vertex, with the same label as the name of the PG graph, and leaves the properties assigned to it. The next step is to assign an edge to the label related to each of the vertices.

B. Cypher$_n$ Query Language

Named Property Graphs need a query language. We propose Cypher$_n$ that supports our proposal. It is a simple extension of openCypher [18]. OpenCypher a high-level declarative graph query language with an ongoing standardization work. We add `FROM` clause which specify name of property graph. Listing 1 presents Cypher$_n$.

```
MATCH (n)
FROM (m)
RETURN n, m
```

Listing 1. Cypher$_n$ example

The extension of openCypher grammar[1] that defines our query language consists of one production, which adds a FROM clause. A fragment of grammar is presented in Listing 2 in EBNF. The key fragment of grammar is shown in Fig. 1 in the form of a railroad diagram.

```
From = ((F,R,O,M), SP, NodePattern);
```

Listing 2. Extension of openCypher grammar

In order to ensure interoperability with solutions that already exist, we present Algorithm 2 that transforms our Cypher$_n$ into openCypher. The algorithm gets the name of the graph from the `FROM` clause and modifies the `MATCH` clauses so that the

[1]https://www.opencypher.org/resources

Algorithm 2: Mapping Cypher$_n$ into openCypher

input : Cypher$_n$ query CN
output: openCypher query C

$f \leftarrow$ getFormClause(CN) ;
if $f \notin \emptyset$ **then**
  $P \leftarrow$ getMatchPattern(CN) ;
  foreach $p \in P$ **do**
    $C \leftarrow$ addRelationship(p, f) ;
  $C \leftarrow$ cloneWhereAndReturnClause(CN) ;
  return C;
else
  return CR ◁ `From clause is optional` ;
return PG;

relation selects all the paths connecting the special vertex with the metadata with other vertices. The algorithm allows some graphs to be unnamed, thanks to which we retain backward compatibility with existing solutions. Listing 3 shows how openCypher query is generated from a Cypher$_n$ query given in Listing 1.

```
MATCH (n)<-[:RELATED]-(m)
RETURN n, m
```

Listing 3. OpenCypher after transformation

III. Use Case: Molecular Entity Representation

Our approach may have many practical applications. One of them is the molecular entity representation. In the standard property graphs, it is possible to describe individual atoms and their relationships. Using our solution, it is also possible to describe the entire molecule as shown in Fig. 2. In this particular case, we have compound "dioxygen" with the properties describing it. All atoms and chemical bonds are also additionally described with properties.

We have developed chemical data parser called SDFEater, available on GitHub[2] under MIT license. Our cross-platform parser is written in Java and works from the command line. It reads molecules, atoms, and bonds data from the file, and then places it in the appropriate program structures. Moreover, the program can add additional atoms data from periodic table, and tries to match the hyperlinks to the database identifiers placed in the input file. The parser accepts Structure-data file (SDF), which is part of Chemical Table file (CT File) [21] family. CT File is the collection of text formats describing

[2]https://github.com/lszeremeta/SDFEater

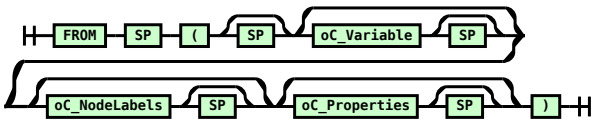

Fig. 1. Cypher$_n$ railroad diagram

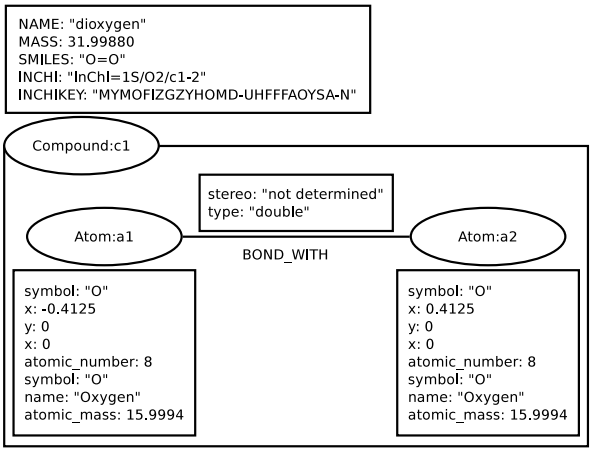

Fig. 2. Example of a named property graph

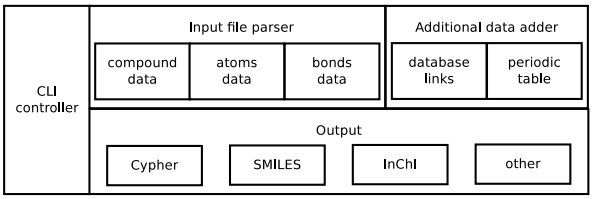

Fig. 3. Parser architecture

chemical data. Among them there is Molfile that contains information about atoms and bonds which is stored in the tabular form. SDF, in addition to Molfile, may also contain additional information about the whole molecule such as description, Simplified Molecular Input Line Entry Specification (SMILES), International Chemical Identifier (InChI), mass, and others in the key-value form. Parser supports four output formats. One of them is the output to the openCypher query language. This allows to easily import data into the Neo4J[3] graph database.

The parser architecture is shown in Fig. 3. SDFEater has 4 main modules – CLI controller, input file parser, additional data adder, and output. The first one is responsible for operating of the command line, writing help, and running of the appropriate parts of the code depending on the selected options. In the second, the data from the source file is parsed and saved to the appropriate program structures. In the next module, additional data not included in the input file is added. Depending on the options chosen by the user, the program tries to replace the IDs listed in the compound properties with full database URL. It can also add additional information about atoms by searching the built-in data from the periodic table of chemical elements. Finally, there is the output module, where the program prints data in the appropriate format depending on the option chosen by the user.

[3]https://neo4j.com/

TABLE I
TEST QUERIES

Name	Query	Description
$\mathbf{Q}_1$	MATCH (c:Compound) RETURN c.CASNumber	Selects CAS number from Compound data
$\mathbf{Q}_2$	MATCH ()−[r:BOND_WITH]−>() RETURN r	Displays all information about atoms bonds
$\mathbf{Q}_3$	MATCH (a1:Atom)−[r:BOND_WITH]−>(a2:Atom) WHERE r.type = 'double' RETURN a2	Chooses second atom with "double" bond type

IV. EXPERIMENTS AND EVALUATION

In this section we evaluate the creating, loading, and querying openCypher based on our Cypher$_n$, presented in Section III. We performed Cypher$_n$ generation tests, importing data to the Neo4j graph database, as well as query tests.

The loading and query tests were performed using the cypher-shell command-line tool[4].

All experiments were executed on a laptop with quad-core AMD A6-6310 APU with AMD Radeon R4 Graphics @ 2.4 GHz (4 cores, 4 threads), single channel 2x4GB RAM (clock speed: 800 MHz, available: 6.77 GB) and 5400 RPM HDD with reading speed rated at about 95 MB/sec[5]. The system used was Ubuntu 16.04.4 LTS with Oracle Java 1.8.0_171 and Neo4j 3.3.5 graph database (community server edition).

We prepared two SDF datasets based on ChEBI [22] (subset of ChEBI complete 3-star dataset[6]) and DrugBank [23] (subset of DrugBank open structures[7]).

In the first step, we measured the performance of generating Cypher$_n$ based on the prepared SDF subsets. We used our SDFEater which is publicly available on GitHub under MIT license. The generated data in openCypher has been published on Figshare [24] under Creative Commons (CC BY 4.0) license. In total, we created 8 datasets in openCypher (2 SDF subsets and 4 variants) marked as $\mathrm{DB}_{card}^{parm}$ and $\mathrm{CB}_{card}^{parm}$, where $parm$ represents one of four variants and $card$ is the number of compounds. For example, CB_{4000}^{r} means `r` variant of ChEBI subset with 4000 compounds. Similarly, DB_{7000}^{p} is `p` variant of DrugBank subset with 7000 compounds. We distinguish the following variants of sets:

- **r** – standard SDF to openCypher,
- **p** – standard SDF with added additional periodic table data to atoms,
- **u** – standard SDF with `-u` parser option enabled (try to generate URLs to other databases instead of IDs),
- **up** – standard SDF with added additional periodic table data to atoms and `-u` parser option enabled,

The **r** variant contains only data present in SDF files. In other cases, we add extra data that are not present in the SDF datasets.

Table II shows openCypher generation times for all discussed openCypher datasets. In the case of DrugBank, a huge increase in execution time is noticeable when additional data is added to atoms. In the case of the ChEBI subset, this is visible for CB_{4000}^{u} and CB_{4000}^{up}. We also provide SDF to PubChem JSON (pcj) [25] conversion times using OpenBabel 2.4.1[8]. Comparing execution times, we can see that our parser is much faster in processing DrugBank and slower in generating ChEBI Cypher$_n$ dataset. In the case of the DB_{7000}^{r}, the Cypher$_n$ is generated almost 11.5 times faster than PubChem JSON.

In the next step, we tested importing data to Neo4j graph database. Mapping Named Property Graph into Property Graph was based on Algorithm 1. Table III shows that for both sets, the importing time grows significantly only in the case of the variant with additional data from the periodic table.

In the last step, we prepare 3 openCypher queries based on Algorithm 2. The prepared queries are representative and checks all features. These queries are presented in Table I.

Table IV shows the execution times for discussed queries. The execution times do not change significantly even in the case of different openCypher datasets variants. The exception to this are subsets with added additional periodic table data (DB_{7000}^{p}, CB_{4000}^{p}, DB_{7000}^{up}, and CB_{4000}^{up}). This is noticeable only for `Q3`.

TABLE II
GENERATION TIMES

$\mathbf{DB}_{7000}^{r}$	$\mathbf{DB}_{7000}^{u}$	$\mathbf{DB}_{7000}^{p}$	$\mathbf{DB}_{7000}^{up}$	$\mathbf{DB}_{7000}$ (pcj)
18.898 s	18.374 s	167.633 s	168.163 s	215.54 s
$\mathbf{CB}_{4000}^{r}$	$\mathbf{CB}_{4000}^{u}$	$\mathbf{CB}_{4000}^{p}$	$\mathbf{CB}_{4000}^{up}$	$\mathbf{CB}_{4000}$ (pcj)
891.308 s	4123.369 s	965.62 s	4218.856 s	111.67 s

TABLE III
IMPORTING TO NEO4J

$\mathbf{DB}_{7000}^{r}$	$\mathbf{DB}_{7000}^{u}$	$\mathbf{DB}_{7000}^{p}$	$\mathbf{DB}_{7000}^{up}$
1951.68 s	1860.018 s	4313.933 s	4665.185 s
$\mathbf{CB}_{4000}^{r}$	$\mathbf{CB}_{4000}^{u}$	$\mathbf{CB}_{4000}^{p}$	$\mathbf{CB}_{4000}^{up}$
1308.953 s	1547.778 s	2622.072 s	2834.983 s

TABLE IV
QUERIES EXECUTION TIMES

	$\mathbf{DB}_{7000}^{r}$	$\mathbf{DB}_{7000}^{u}$	$\mathbf{DB}_{7000}^{p}$	$\mathbf{DB}_{7000}^{up}$
Q_1	3.586 s	3.616 s	3.659 s	3.636 s
Q_2	7.478 s	7.314 s	7.654 s	7.951 s
Q_3	6.188 s	6.208 s	8.894 s	8.909 s
	$\mathbf{CB}_{4000}^{r}$	$\mathbf{CB}_{4000}^{u}$	$\mathbf{CB}_{4000}^{p}$	$\mathbf{CB}_{4000}^{up}$
Q_1	3.336 s	3.336 s	3.354 s	3.319 s
Q_2	6.37 s	6.581 s	6.29 s	6.426 s
Q_3	5.223 s	5.402 s	6.339 s	6.387 s

[4]https://neo4j.com/docs/operations-manual/current/tools/cypher-shell/
[5]tested using `hdparm -t`
[6]https://www.ebi.ac.uk/chebi/downloadsForward.do
[7]https://www.drugbank.ca/releases/latest#open-data
[8]http://openbabel.org/wiki/Main_Page

V. Conclusions

The property graph model is increasingly used in databases. We present a Named Property Graph model which allows to group graphs into separate units and describe information about them. Named Property Graphs provide a high-value and incremental change to the property graph model. We also introduce Cypher$_n$ query language that supports our proposal. In the paper we also present mapping algorithms, use cases with the chemical data, and SDFEater that is our tool for processing data. Our proposal can be easily applied to existing databases. The results of the experiments show the good potential of the presented solutions.

The future work will focus on providing support for temporal, uncertainty, and trust metrics. Another challenge is to find a relationship between our solution and hypernodes. We would also like to focus on developing methods for transforming our solution into other query languages.

Acknowledgments

This publication has received financial support from the Polish Ministry of Science and Higher Education under subsidy for maintaining the research potential of the Faculty of Mathematics and Informatics, University of Bialystok.

References

[1] J. Webber, "A programmatic introduction to Neo4J," in *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, ser. SPLASH '12. New York, NY, USA: ACM, 2012. doi: 10.1145/2384716.2384777. ISBN 978-1-4503-1563-0 pp. 217–218. [Online]. Available: http://dx.doi.org/10.1145/2384716.2384777

[2] C. Tesoriero, *Getting started with OrientDB*. Packt Publishing Ltd, 2013. ISBN 978-1782169956

[3] L. Dohmen, "Algorithms for large networks in the NoSQL database ArangoDB," Bachelor's Thesis, RWTH Aachen University, Aachen, 2012.

[4] K. Chodorow, *MongoDB: The definitive guide: powerful and scalable data storage*. O'Reilly Media, Inc., 2013. ISBN 978-1449344689

[5] D. Tomaszuk, "RDF data in property graph model," in *Metadata and Semantics Research: 10th International Conference, MTSR 2016, Göttingen, Germany, November 22-25, 2016, Proceedings*. Springer, 2016. doi: 10.1007/978-3-319-49157-8_9 pp. 104–115. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-49157-8_9

[6] R. De Virgilio, A. Maccioni, and R. Torlone, "Converting relational to graph databases," in *First International Workshop on Graph Data Management Experiences and Systems*, ser. GRADES '13. ACM, 2013. doi: 10.1145/2484425.2484426. ISBN 978-1-4503-2188-4 pp. 1:1–1:6. [Online]. Available: http://dx.doi.org/10.1145/2484425.2484426

[7] R. De Virgilio, A. Maccioni, and R. Torlone, "R2G: A tool for migrating relations to graphs," in *Proceeding of the 17th International Conference on Extending Database Technology (EDBT 2014)*, 2014, pp. 640–643.

[8] S. Lee, B. H. Park, S. H. Lim, and M. Shankar, "Table2Graph: A scalable graph construction from relational tables using Map-Reduce," in *2015 IEEE First International Conference on Big Data Computing Service and Applications*, March 2015. doi: 10.1109/BigDataService.2015.52 pp. 294–301. [Online]. Available: http://dx.doi.org/10.1109/BigDataService.2015.52

[9] G. Schreiber and Y. Raimond, "RDF 1.1 Primer," W3C, W3C Note, 2014. [Online]. Available: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

[10] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler, "Named graphs, provenance and trust," in *Proceedings of the 14th International Conference on World Wide Web*. ACM, 2005. doi: 10.1145/1060745.1060835 pp. 613–622. [Online]. Available: http://dx.doi.org/10.1145/1060745.1060835

[11] M. Junghanns, A. Petermann, N. Teichmann, K. Gómez, and E. Rahm, "Analyzing extended property graphs with Apache Flink," in *Proceedings of the 1st ACM SIGMOD Workshop on Network Data Analytics*, ser. NDA '16. New York, NY, USA: ACM, 2016. doi: 10.1145/2980523.2980527. ISBN 978-1-4503-4513-2 pp. 3:1–3:8. [Online]. Available: http://dx.doi.org/10.1145/2980523.2980527

[12] M. Junghanns, P. André, and R. Erhard, "Distributed grouping of property graphs with GRADOOP," in *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, B. Mitschang, N. Daniela, L. Frank, S. Harald, H. Melanie, T. Jens, H. Theo, K. Oliver, and W. Matthias, Eds. Gesellschaft für Informatik, Bonn, 2017, pp. 103–122.

[13] M. Levene and A. Poulovassilis, "An object-oriented data model formalised through hypergraphs," *Data & Knowledge Engineering*, vol. 6, no. 3, pp. 205–224, 1991. doi: 10.1016/0169-023X(91)90005-I. [Online]. Available: http://dx.doi.org/10.1016/0169-023X(91)90005-I

[14] M. Levene and A. Poulovassilis, "The hypernode model and its associated query language," in *Information Technology, 1990. 'Next Decade in Information Technology', Proceedings of the 5th Jerusalem Conference on (Cat. No.90TH0326-9)*, Oct 1990. doi: 10.1109/JCIT.1990.128324 pp. 520–530. [Online]. Available: http://dx.doi.org/10.1109/JCIT.1990.128324

[15] M. Gyssens, J. Paredaens, and D. V. Gucht, "A graph-oriented object model for database end-user interfaces," *ACM SIGMOD Record*, vol. 19, no. 2, pp. 24–33, 1990. doi: 10.1145/93605.93616. [Online]. Available: http://dx.doi.org/10.1145/93605.93616

[16] G. M. Kuper and M. Y. Vardi, "The logical data model," *ACM Transactions on Database Systems (TODS)*, vol. 18, no. 3, pp. 379–413, 1993. doi: 10.1145/155271.155274. [Online]. Available: http://dx.doi.org/10.1145/155271.155274

[17] A. Ghrab, O. Romero, S. Skhiri, A. Vaisman, and E. Zimányi, "Grad: On graph database modeling," *arXiv preprint arXiv:1602.00503*, 2016.

[18] J. Marton, G. Szárnyas, and D. Varró, "Formalising openCypher graph queries in relational algebra," in *Advances in Databases and Information Systems*, M. Kirikova, K. Nørvåg, and G. A. Papadopoulos, Eds. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-66917-5_13. ISBN 978-3-319-66917-5 pp. 182–196. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-66917-5_13

[19] R. Zhou and E. A. Hansen, "Parallel Structured Duplicate Detection," in *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'07. AAAI Press, 2007. ISBN 978-1-57735-323-2 pp. 1217–1223.

[20] D. Tomaszuk and K. Pak, "Reducing vertices in property graphs," *PLOS ONE*, vol. 13, no. 2, pp. 1–25, 02 2018. doi: 10.1371/journal.pone.0191917. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0191917

[21] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer, "Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited," *Journal of Chemical Information and Computer Sciences*, vol. 32, no. 3, pp. 244–255, 1992. doi: 10.1021/ci00007a012. [Online]. Available: http://dx.doi.org/10.1021/ci00007a012

[22] J. Hastings, G. Owen, A. Dekker, M. Ennis, N. Kale, V. Muthukrishnan, S. Turner, N. Swainston, P. Mendes, and C. Steinbeck, "ChEBI in 2016: Improved services and an expanding collection of metabolites," *Nucleic Acids Research*, vol. 44, no. D1, pp. D1214–D1219, 2016. doi: 10.1093/nar/gkv1031. [Online]. Available: http://dx.doi.org/10.1093/nar/gkv1031

[23] D. S. Wishart, Y. D. Feunang, A. C. Guo, E. J. Lo, A. Marcu, J. R. Grant, T. Sajed, D. Johnson, C. Li, Z. Sayeeda, N. Assempour, I. Iynkkaran, Y. Liu, A. Maciejewski, N. Gale, A. Wilson, L. Chin, R. Cummings, D. Le, A. Pon, C. Knox, and M. Wilson, "DrugBank 5.0: a major update to the DrugBank database for 2018," *Nucleic Acids Research*, vol. 46, no. D1, pp. D1074–D1082, 2018. doi: 10.1093/nar/gkx1037. [Online]. Available: http://dx.doi.org/10.1093/nar/gkx1037

[24] Ł. Szeremeta and D. Tomaszuk, "SDFParser example Cypher outputs," 5 2018. doi: 10.6084/m9.figshare.6249962.v1. [Online]. Available: http://dx.doi.org/10.6084/m9.figshare.6249962.v1

[25] Y. Wang, S. H. Bryant, T. Cheng, J. Wang, A. Gindulyte, B. A. Shoemaker, P. A. Thiessen, S. He, and J. Zhang, "PubChem BioAssay: 2017 update," *Nucleic acids research*, vol. 45, no. D1, pp. D955–D963, 2016. doi: 10.1093/nar/gkw1118. [Online]. Available: http://dx.doi.org/10.1093/nar/gkw1118