Chapter 5
Automation in Graph-Based Data Integration and Mapping

Marcel Friedrichs

Abstract Data integration plays a vital role in scientific research. In biomedical research, the OMICS fields have shown the need for increasingly larger datasets, like proteomics, pharmacogenomics, and even newer fields like foodomics. In 2019 Nucleic Acids Research counted 1637 databases, accounting only for a fraction of all data sources available online. Data integration efforts need to process large amounts of heterogeneous data from different file formats ranging from simple files to complex relational databases and increasingly graph databases. Aside from data formats, availability is another obstacle. Whether files are available for direct download, need a user account, or are available only through an application programming interface (API). Keeping data sources up-to-date is important to make use of the latest discoveries in the respective fields, retrieve error corrections, and potentially mitigate issues with other data sources referencing newly added entities. Finally, all data sources provide information on certain entities and in most cases make use of specific identification systems. In the best case, data sources provide cross-references to other data sources. In order to generate robust mappings between all required data sources, identifiers of good quality need to be selected forming new connections between the entities. All of these vital steps and issues of data integration and mapping benefit from automation and are in most parts able to be fully automated. Workflow systems and integration tools are capable of automating different elements of the aforementioned steps and require varying levels of computer science skills. This chapter describes these issues, and the potential of the fully automated, graph-based data integration and mapping tool BioDWH2 is explored.

Keywords Data warehouse · Data integration · Graph database · Software tools

M. Friedrichs (✉)
Faculty of Technology, Bioinformatics/Medical Informatics Department, Bielefeld University, Bielefeld, Germany
e-mail: mfriedrichs@techfak.uni-bielefeld.de

M. Chen, R. Hofestädt (eds.), *Integrative Bioinformatics*,
https://doi.org/10.1007/978-981-16-6795-4_5

5.1 Introduction

Data integration plays a vital role in scientific research analyses. Advancements in biomedical research gave rise to the OMICS fields starting with genomics, transcriptomics, and proteomics. The list of new OMICS fields has increased dramatically with additions such as pharmacogenomics, foodomics, and antibody-omics. Each of these fields requires its own data, experiments, and new databases increasing the overall complexity of available data sources and effort needed to keep information up-to-date. In 2018 Imker conducted a survey of published databases in the Nucleic Acids Research (NAR) database issues and concluded that 1700 databases were covered in 25 years (Imker, 2018). The 2021 NAR database issue added 90 new resources and with updates and removals now count 1641 databases (Rigden and Fernández, 2020). The Online Bioinformatics Resources Collection (OBRC) contained 1542 bioinformatics databases and other resources (Chen et al., 2007) which has grown to 2417 as of July 2021. These numbers only represent the resources added to common registries resulting in a likely larger number of databases available online.

For the use-case of medical information systems, multiple OMICS levels are relevant in drug therapy safety (Kapoor et al., 2016; Qian et al., 2019). Where previously the main focus of analyses were interaction networks of drugs, diseases, and side effects, the growing opportunities of molecular information in the clinical context (Krier et al., 2016; Sanderson et al., 2019) add new OMICS fields in the form of genes, variants, pathways, RNA regulation, and many more. Examples would be the "PharmGKB" (Whirl-Carrillo et al., 2012), "DrugCentral" (Avram et al., 2020), "DrugBank" (Wishart, 2006), and "OMIM" (Online mendelian inheritance in man, 2021) databases. The integration and mapping of this information could provide an in-depth understanding of individual patient cases and reduce adverse drug reactions toward personalized medicine.

This growing complexity increases lead time of research projects as users need to analyze data sources with heterogeneous file formats, availability, and information schemata. Much of these issues benefit from integration pipelines and tools which are easy to use and take care of data warehousing and mapping tasks. With the growing adoption of graph databases and formats (Fabregat et al., 2018; Hassani-Pak et al., 2016; Shoshi et al., 2018; Yoon et al., 2017), the transformation of heterogeneous data sources into a common graph data structure is beneficial in representing complex and highly connected biological information. While data warehousing solutions provide users with all data sources in a single database, the information is still loosely coupled. Most data sources provide identification systems or external references for their data. However, changes in referenced data sources are not immediately propagated and might lead to loss of information, and data sources need to be constantly updated. Finally, automated mapping techniques are important in building tightly coupled relationships between data sources in a data warehouse. While these mapped relationships may never cover all available

information, they build a starting ground for research analyses and enable the discovery of new and potentially meaningful information.

This chapter describes the problems and solutions of data integration and information mapping and closes with a possible solution using the open-source BioDWH2 tools.

5.2 Data Integration and Mapping

Different data integration approaches have been developed in the past decades. As with many architectural problems, each comes with their own set of advantages and disadvantages (Schwinn and Schelp, 2005). The approaches differ in a multitude of aspects, such as heterogeneity, distribution, access, redundancy, technology, and more. This section will first look at federated databases, data warehouses, and data lakes under the aforementioned aspects. Afterwards, the role of mapping strategies for these approaches is explored.

5.2.1 Federated Database System

Arguably the simplest approach to implement is federated database systems (FDBS). A FDBS consists of multiple, independent component databases which are directly accessed by the FDBS. There are no restrictions on the location or technology of the component databases. The only exception is that the FDBS needs access via any means of local or remote communication. The access may be restricted using credentials which need to be stored in the federated database management system. FDBS can be divided into loosely and tightly coupled systems.

Loosely coupled FDBS give the user direct access to the component database schemas. The advantage is a minimal overhead in administration of the database system and new schema additions of the component databases are directly available to the user. However, the users need to understand and process the schema and heterogeneity of the component databases themselves which may result in redundant work.

Tightly coupled FDBS mitigate this problem by introducing schema transformations and views on the component databases. Heterogeneous information from different component databases can be normalized and provided to the user for direct use. Additionally, selecting and filtering the raw data into qualitative subsets is possible by providing schema views. This increases the administrative overhead of the FDBS as changes in the components need to be updated. If a user needs specific information from a component, the transformations and views may need to be changed by the FDBS administrator. The main benefit is that these transformations need to be done only once and not for each user.

Using component databases directly has the obvious advantage that no data has to be stored locally by the FDBS. New information is directly available and no update strategy, except for schema changes, needs to be employed. In the early days of FDBS one of the downsides was performance as sending queries and results via the internet was slow. With the increasing internet speed worldwide this problem is less relevant today. Another issue is availability. A FDBS is not protected against component databases being unavailable due to maintenance, outages, and more. Finally, all queries are sent directly to component databases outside of the FDBS control. Sensible information such as patient data may be sent in the queries and therefore need to adhere to privacy and security regulations, which may be complicated in a FDBS setting.

Federation regained popularity in recent years in the field of plant breeding with the BrAPI (Breeding API) project (Selby et al., 2019). Researchers worldwide can provide plant breeding data via a standardized application programming interface (API) which then can be used in a federated system. An advantage of the API standard is the reduced need for schema transformation on the FDBS side.

5.2.2 Data Warehouse

Data warehouses (DWH) are in contrast to FDBS central databases of integrated data. Heterogeneous data sources are parsed and all the information is stored in one central database. If necessary, the information is transformed to match the used database system or the central database schema. In case the data warehouse is created for a specific purpose, the data may also be filtered or further processed. This process is often referred to as ETL (extract, transform, load). Figure 5.1 visualizes this process.

The integration in a central database has the advantage of independence from third parties and network connections to component databases. Outages will affect either all or no data in the central database and the data warehouse administration can implement preventative measures. The central integration comes at a cost.

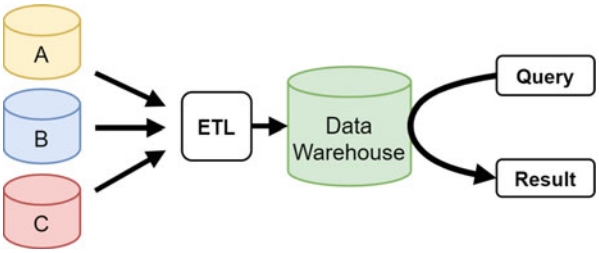

Fig. 5.1 Heterogeneous data sources A, B, and C are integrated into a central data warehouse by means of an ETL (extract, transform, load) process. Queries are performed directly on the data warehouse which has a single schema for all data

Hardware for data storage needs to be available and data sources need to be integrated on a semi-regular basis when updates are available. Data sources and their data formats need to be understood and suitable integration pipelines developed. Mapping the data source schema to the central database schema is comparable to the tightly coupled FDBS approach and changes to the source schemas need to be updated as well. Privacy and security aspects are easier with data warehouses because sensible information can stay inside a controlled network environment like a hospital for example.

5.2.3 Data Lake

A relative new approach is the so-called data lakes (Khine and Wang, 2018). Originating from the field of big data and machine learning, data lakes differ from data warehouses in several key aspects. First, all data from any data source is dumped as-is or with as little transformation as possible into the data lake. This can be structured data such as relational databases, documents such as PDFs, or even raw data such as sensor readouts. The principal idea is that the use of the data is unknown beforehand or may change in the future. Therefore, all data are equally important and should neither be modified, nor filtered. Queries are then performed on the data lake and the heterogeneous information transformed during query execution. Figure 5.2 visualizes this process.

For big data applications using machine learning (ML) algorithms this approach is of great interest, because many modern ML algorithms extract features automatically from heterogeneous and large amounts of data without prior knowledge. However, when writing traditional queries for data analysis, data lakes may impose an even greater burden on the user, similar to loosely coupled FDBS. While the idea of collecting all data possible and having them ready anytime is daunting, this has several downsides. First, even as storage space is getting cheaper over time, data lakes will require a lot of space because all the information is stored. Secondly, different data require different storage solutions. Data lakes often consist of a multitude of subsystems including relational, graph, and document databases

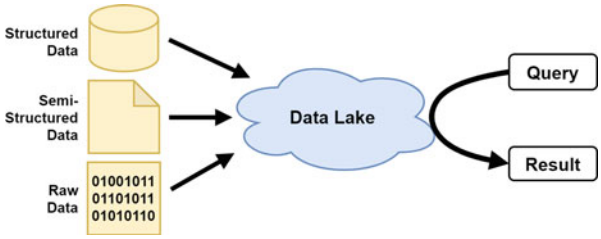

Fig. 5.2 Data lakes consist of structured, semi-structured, and raw data. Queries are performed directly on the data lake and information are transformed in the query processing

as well as key-value stores. The administrative overhead in maintaining all of these systems is larger than a singular database system. Lastly, queries need to handle all types of heterogeneous data. For example SQL queries are tailored to relational databases, but are not well suited for graph database. Query plan optimization is a complicated task for data lakes in order for queries to execute in a reasonable time frame.

5.2.4 Data Mapping

Data integration and analysis require some form of data mapping to connect entities from heterogeneous data sources. In the case of integration, FDBS and data warehouses can use mappings for schema transformation and linking or merging entities together. Data lakes store data as-is and therefore mapping entities are shifted to query execution of subsequent data analyses. Mapping helps in connecting entities and relationships between data sources in order to gain a new data quality. New insights can be generated if mapping connects previously disconnected information clusters.

Mapping can be performed on a variety of information. This includes names, synonyms, identification systems, or more specific entity properties. For example chemical structures can be represented as IUPAC International Chemical Identifier (InChI) identifiers. These InChI ids can then be used to map similar structures. Name and synonym mappings in general are more error-prone than other methods. Depending on the context names may be used for different entities or the words of the name are ordered or cased differently than in other data sources. Additionally, different languages may further complicate the mapping process.

One of the most common mapping methods are identification systems. Almost all data sources define their own identifiers for entities and sometimes even relationships. Examples are the DrugBank identifier “DB00122” for the chemical Choline or the HGNC id “HGNC:5962” for the gene IL10. Databases can provide cross-references to other databases using these identifiers making them especially suited for mapping between data sources. However, not all identification systems should be used to merge entities as being the same. Depending on the scope of the database or identification system information may be represented as a singular entity where other databases provide more granular entities of the same kind. A selection of suitable identification systems can therefore drastically improve the mapping result.

Multiple strategies exist on how mapped entities should be handled. Entities can either be merged together destructively into a singular entity or relationships between these entities can be created non-destructively marking them as mapped. Here, we will explore a hybrid solution by introducing a mapping layer for entities and relationships using only identification systems. The example uses terminology of graph structures but can be transferred to other systems as well.

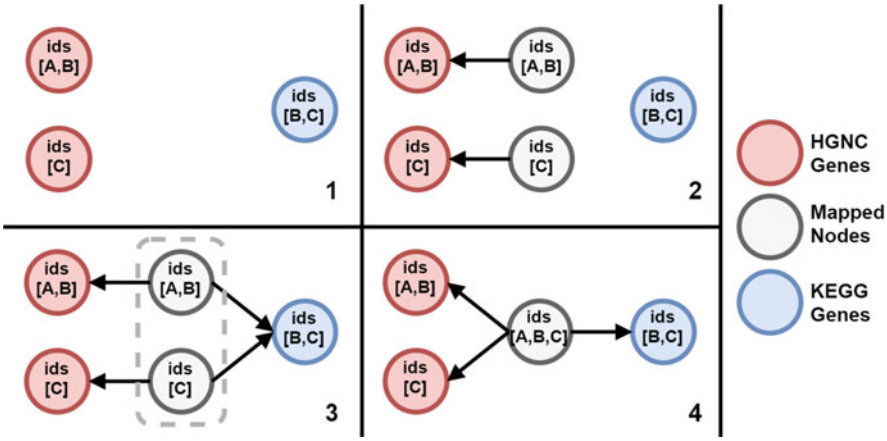

Fig. 5.3 Node mapping example for Gene nodes from two data sources. (1) Mapping operates on a single graph with all data sources merged. (2) Nodes of the first data source are mapped. As no identifiers overlap, two mapping nodes are created and connected to the source nodes. (3) Nodes from the second data source are mapped. This results in an identifier overlap between two mapping nodes. (4) The result is a single mapping node as the two mapping nodes are merged

Nodes of interest are mapped into the mapping layer as visualized in Fig. 5.3. This process takes each individual node and generates a node mapping description. Identifiers from suitable identification systems as well as names and synonyms are collected in the mapping description. If mapping nodes with overlapping identifiers exist, they are collected and collapsed into a singular mapping node. Identifiers and names are merged using standard sets. If none is matched, a new mapping node is created from the mapping description. Finally, an edge is introduced from the source node to the respective mapping node. This process is repeated for all nodes building up the basis for the mapping layer.

Mapping of direct relationships (edges) or more complex relationship paths across multiple nodes is handled similar to the node mapping. For each data source, edge paths of interest need to be defined. A path is comprised of a series of node labels which are connected by an edge label and edge direction. The edge direction can be either forward, backward, or bidirectional and is important to prevent paths going backward where needed. The first and last node labels of the path are required to be used in the node mapping process before, so that their mapped nodes already exist. These edge paths can then be mapped as an edge in the mapping layer between the two mapping nodes. A trivial path of length one being mapped is visualized in Fig. 5.4.

However, meaningful relationships between nodes may involve a more complex path of edges. As paths get longer, the time a mapping process takes will increase accordingly as all possible paths are searched for using depth-first search starting from all existing nodes with the first node label. A path example of length three is visualized in Fig. 5.5.

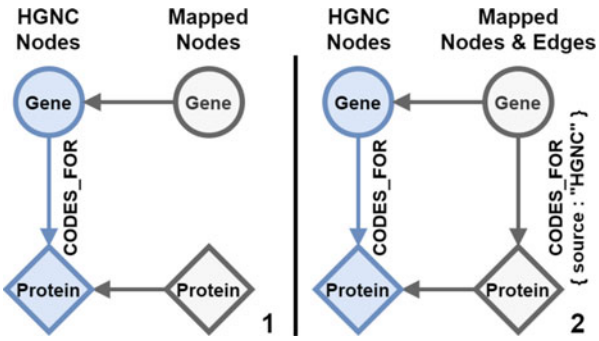

Fig. 5.4 Trivial edge mapping between two mapped data source nodes. (1) The HGNC data source provides two nodes Gene and Protein in blue which are connected with a CODES_FOR edge. Both are connected to their respective mapping node in grey. (2) A new edge with the mapped CODES_FOR label is created between the mapping nodes

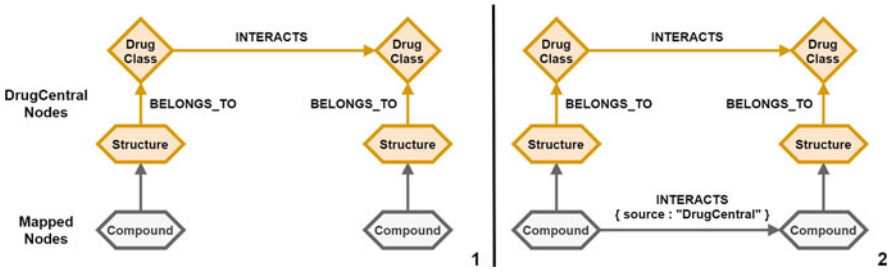

Fig. 5.5 Path mapping of four data source nodes and three edges. (1) Two Structure nodes in orange from the same data source are both associated with a respective DrugClass node. These two DrugClass nodes are linked with an INTERACTS edge. Both Structures are connected to their respective mapping node in grey. The path of length three is matched and provided to the path mapping. (2) A new edge with the mapped INTERACTS label is created between the mapping nodes

5.3 BioDWH2

As shown before, a multitude of problems and techniques exist in the field of data integration and mapping. The BioDWH2 tool presented here solves multiple of these issues while being as easy and automated as possible (Friedrichs, 2021). BioDWH2 is implemented as a modular open-source Java program that is easily extensible with new data source modules. For BioDWH2 to be run, an existing installation of the Java Runtime Environment (JRE) 8 is required. The goal is the automation of data warehouse creation for research projects. A data warehouse solution was chosen for its simplicity in user accessibility and better privacy and security control.

Where data warehouses usually filter data for specific purposes, BioDWH2 uses the unbiased approach of data lakes by integrating all information from each data source where possible. This allows for generated databases to be usable as broadly as possible. Graph database structures were chosen for their high flexibility in large amounts of data and relationships.

5.3.1 BioDWH2 Workspace

As more and more heterogeneous data sources are required for a certain task, the amount of files to be handled gets increasingly complex. Therefore, a fixed schema of managing source, as well as intermediate files in a folder structure is crucial. BioDWH2 takes care of this task by introducing the concept of workspaces. Workspaces allow users to create as many physically separate data warehouse projects as needed. A strict folder structure simplifies research data management. With all sources and intermediate processing steps in a central location, workspaces are easy to compress, backup, and transfer if necessary. The workspace provides a sub-folder structure for each data source containing the source files and metadata information stored in a JSON file. Metadata include the current source version stored, file names, and flags whether the updater, parser, and exporter of the data source finished successfully.

5.3.2 Architecture

BioDWH2 is developed using a modular architecture and the factory method pattern. This allows for new data source modules to be added and maintained without modification of the core project. An architectural overview is visualized in Fig. 5.6.

Every modular architecture needs a core project containing the abstract base classes for the implementing modules. The BioDWH2-Core component provides these base classes as well as a graph data structure and many additional utilities. Networking utilities for example help in communication with HTTP and FTP requests. Dependencies for popular file format libraries, as well as custom implemented file format parsers help data source modules load common formats and simplify the implementation process. These include Open Biological and Biomedical Ontology (OBO), CSV, structure-data file (SDF), and many more.

Data source modules are slim java modules with the BioDWH2-Core as a dependency. They implement the abstract ETL classes of the core for their respective data source. This includes an updater, parser, graph exporter, and mapping describer. This ensures a streamlined implementation process for new modules and reduces the maintenance effort.

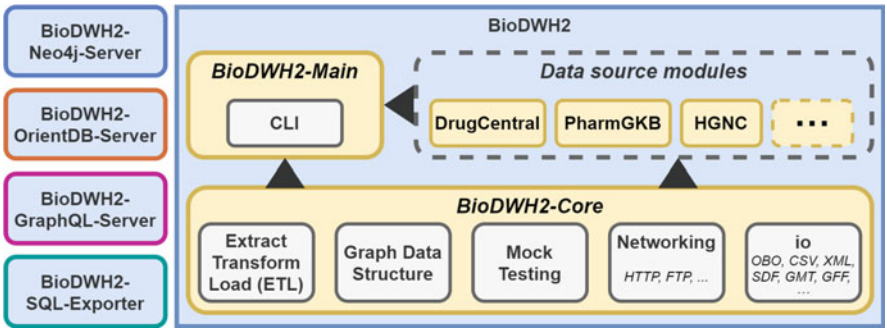

Fig. 5.6 BioDWH2 is built using a modular architecture. The core provides the general program flow and utilities. Data source modules are built on top of the core and implement the abstract ETL process. The main module brings the core and all data source modules together for execution. Additional server and exporter tools complement BioDWH2 for access and analysis needs. These include a Neo4j-, GraphQL-, and OrientDB-Server as well as an SQL exporter

The third component of the architecture is BioDWH2-Main. This java module references the core and all data source modules in the BioDWH2 project. Additional third-party data source modules are included as jar files using the java runtime classpath. The main component provides a simple command-line interface (CLI) as the primary interaction point for the end-users. All tasks such as creating and updating workspaces are performed using this CLI.

5.3.3 Program Flow

BioDWH2 follows the data warehouse paradigm as outlined in Sect. 5.2.2 with the addition of a subsequent mapping step. This results in a strictly defined program flow. Every BioDWH2 project will follow the steps as visualized in Fig. 5.7. As projects are created as workspaces, the creation of a workspace and configuration of used data sources is always the first step. Subsequently, the status of a workspace can be queried or the integration process started as often as necessary.

The integration process itself is split into five tasks and can be repeated whenever a new version of a data source or data source module has become available. As data source modules need to load their respective raw data files, each respective updater implementation checks for the newest version online and downloads them to the workspace if necessary. Once downloaded, the raw data files need to be parsed and exported into the BioDWH2 internal graph data structure by each data source module. The graph data structure is a simple file-based directed property graph model comprised of nodes and edges. Custom unique and non-unique index structures for edge and node properties enable fast queries for existing data. Nodes hereby represent entities such as genes or proteins. Edges represent entity relationships such as a gene codes for a specific protein.

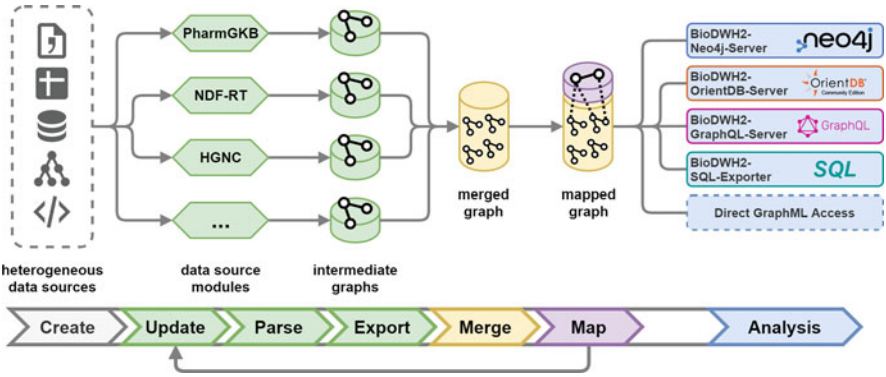

Fig. 5.7 Complete overview of the BioDWH2 data flow. Heterogeneous data sources are updated, parsed, and exported via the data source modules. The resulting intermediate graphs are then merged and mapped into one graph. This graph may then be accessed for analysis using different platforms

The internal graph data structure is stored in each data sources directory. Additionally, each graph is also exported in Graph markup language (GraphML) format (Brandes et al., 2013) for easier access. GraphML was chosen for its simple structure and widespread adoption and interoperability. As the data sourceś graph schema may not be known by the user beforehand, a meta graph visualization and statistic is generated for each graph. The number of nodes and edges per label are exported in tabular format to a text file. The visualization is generated as a static image and interactive HTML page.

After the update, parse, and export steps for each data source the resulting intermediate graphs are collected and merged into a single graph. To distinguish nodes and edges from each data source, their labels are prefixed with their respective data source moduleś ID. This supports the user in writing distinct queries during analysis as well as the mapping process in associating nodes with data sources. The merged graph represents the first data warehouse stage of BioDWH2 containing all requested data sources. As described before, a meta graph and associated statistics are generated and the graph is exported in GraphML format.

The final step of the integration process is the generation of the mapping layer. This meta-layer creates new nodes and edges from common entities and relationships as provided by the data source modules. The mapping itself is based on the description in Sect. 5.2.4. Each data source module provides an implementation of a "MappingDescriber." This describer is able to tell the core mapping process which node labels and edge paths in the data sourceś graph are of interest. Each of these nodes and paths are then queried and presented to the describer individually. Where applicable, the describer then provides a mapping description which is used to create the meta-layer nodes and edges. If multiple entities from different data sources were mapped to the same meta-node, these data sources are now interconnected.

This implementation allows for an automated mapping of data warehouses with any number of sources and only limited by the descriptions provided by the data source modules.

5.3.4 Database Access

The BioDWH2 tool covers the whole integration and mapping process, but provides no analysis capabilities. Every graph in the process is exported to the workspace in GraphML format. These files could be used directly; however, this may not be feasible especially for large graphs. To provide users with easy-to-use analysis capabilities multiple complementary tools are available. As every user might have personal preferences, license restrictions, or technological restrictions, the following database systems were selected as choices and more may be added in the future. Each tool uses the mapped graph databases from a workspace to either provide the data directly, or export a platform specific database. The BioDWH2-Neo4j-Server allows for the creation of a Neo4j graph database as well as running a Neo4j server and browser embedded in the tool itself. No setup of a Neo4j server is needed and queries can be run using the Cypher query language directly in the user's web browser. This allows for a frictionless usage of BioDWH2 for users already familiar with the Neo4j ecosystem. Analogously the BioDWH2-OrientDB-Server tool creates an OrientDB graph database (https://www.orientdb.org) and provides an embedded OrientDB server and browser. GraphQL (https://graphql.org) despite the name is primarily a query language for APIs. However, it is possible to define a schema definition for property graphs such as the BioDWH2 graph data structure. The BioDWH2-GraphQL-Server is currently in development, to provide a GraphQL endpoint for analysis queries, which directly operate on the workspace database. Finally, if users may want to use their graph database on common web servers for which only SQL databases are available, the BioDWH2-SQL-Exporter can be used to transform a workspace graph into a relational SQL database dump. A complete overview of the data flow is visualized in Fig. 5.7 with access to the data using the aforementioned tools.

5.4 Summary

The integration and mapping of heterogeneous data sources is an important first step for scientific data analyses. A multitude of integration paradigms and common problems create a learning curve for researches new in the data integration field. This can delay research projects and shift attention away from subsequent data analyses. Therefore, the automation of integration and mapping tasks is important in reducing this barrier and bringing research projects to analyses faster.

The BioDWH2 suite of tools intends to help users with these issues. As every user has different needs or approaches to data integration and analyses, distinct workflow steps allow for more use-cases and reach a broader audience. For newly started research projects, the final mapping layer might be a good starting point in interconnecting data sources of interest and getting an overview of the data. However, it is always possible to use the merged graph of all data sources or even individual data source graphs directly if those are more fitting for a project. In being as broadly usable as possible and supporting multiple platforms and tools for analysis, BioDWH2 can help in reducing time and effort needed for research projects and prevent common data integration mistakes for inexperienced users.

5.5 Availability

The BioDWH2 tools are free to use and available at https://github.com/BioDWH2. BioDWH2 is developed to be usable out of the box without any prerequisites except the Java Runtime Environment (JRE) version 8.

References

Avram S, Bologa CG, Holmes J, Bocci G, Wilson TB, Nguyen DT, Curpan R, Halip L, Bora A, Yang JJ, Knockel J, Sirimulla S, Ursu O, Oprea TI (2020) DrugCentral 2021 supports drug discovery and repositioning. Nucleic Acids Res 49(D1): D1160–D1169

Brandes U, Eiglsperger M, Lerner J, Pich C (2013) Graph markup language (GraphML)

Chen YB, Chattopadhyay A, Bergen P, Gadd C, Tannery N (2007) The online bioinformatics resources collection at the university of Pittsburgh Health Sciences library system–a one-stop gateway to online bioinformatics databases and software tools. Nucleic Acids Res 35(Database):D780–D785

Fabregat A, Korninger F, Viteri G, Sidiropoulos K, Marin-Garcia P, Ping P, Wu G, Stein L, D'Eustachio P, Hermjakob H. (2018) Reactome graph database: efficient access to complex pathway data. PLoS Comput Biol 14(1):e1005968

Friedrichs M (2021) BioDWH2: an automated graph-based data warehouse and mapping tool. J Integr Bioinform 18(2):167–176

Hassani-Pak K, Castellote M, Esch M, Hindle M, Lysenko A, Taubert J, Rawlings C (2016) Developing integrated crop knowledge networks to advance candidate gene discovery. Appl Translat Genomics 11, 18–26

Imker HJ (2018) 25 years of molecular biology databases: a study of proliferation, impact, and maintenance. Front Res Metrics Anal 3:18

Kapoor R, Tan-Koi WC, Teo YY (2016) Role of pharmacogenetics in public health and clinical health care: a SWOT analysis. Eur. J. Hum. Genet. 24(12):1651–1657

Khine PP, Wang ZS (2018) Data lake: a new ideology in big data era. ITM Web Conf 17:03025

Krier JB, Kalia SS, Green RC (2016) Genomic sequencing in clinical practice: applications, challenges, and opportunities. Dialogues Clin Neurosci 18(3):299–312

Online Mendelian Inheritance in Man, OMIM® (2021) Mckusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD). https://omim.org. Accessed: 2021-01-24

Qian T, Zhu S, Hoshida Y (2019) Use of big data in drug development for precision medicine: an update. Expert Rev Precision Med Drug Dev 4(3):189–200

Rigden DJ, Fernández XM (2020) The 2021 nucleic acids research database issue and the online molecular biology database collection. Nucleic Acids Res 49(D1):D1–D9

Sanderson SC, Hill M, Patch C, Searle B, Lewis C, Chitty LS (2019) Delivering genome sequencing in clinical practice: an interview study with healthcare professionals involved in the 100,000 genomes project. BMJ Open 9(11):e029699

Schwinn A, Schelp J (2005) Design patterns for data integration. J Enterp Inf Manag 18(4):471–482

Selby P, Abbeloos R, Backlund JE, Salido MB, Bauchet G, Benites-Alfaro OE, Birkett C, Calaminos VC, Carceller P, Cornut G, Costa BV, Edwards JD, Finkers R, Gao SY, Ghaffar M, Glaser P, Guignon V, Hok P, Kilian A, KÖnig P, Lagare JEB, Lange M, Laporte MA, Larmande P, LeBauer DS, Lyon DA, Marshall DS, Matthews D, Milne I, Mistry N, Morales N, Mueller LA, Neveu P, Papoutsoglou E, Pearce B, Perez-Masias I, Pommier C, Ramírez-González RH, Rathore A, Raquel AM, Raubach S, Rife T, Robbins K, Rouard M, Sarma C, Scholz U, Sempéré G, Shaw PD, Simon R, Soldevilla N, Stephen G, Sun Q, Tovar C, Uszynski G, Maikel V (2019) BrAPI—an application programming interface for plant breeding applications. Bioinformatics 35(20):4147–4155

Shoshi A, Hofestädt R, Zolotareva O, Friedrichs M, Maier A, Ivanisenko VA, Dosenko VE, Bragina EY (2018) GenCoNet—a graph database for the analysis of comorbidities by gene networks. J Integr Bioinform 15(4):20180049

Whirl-Carrillo M, McDonagh EM, Hebert JM, Gong L, Sangkuhl K, Thorn CF, Altman RB, Klein TE (2012) Pharmacogenomics knowledge for personalized medicine. Clin Pharmacol Ther 92(4):414–417

Wishart DS (2006) DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res 34(90001):D668–D672

Yoon BH, Kim SK, Kim SY (2017) Use of graph database for the integration of heterogeneous biological data. Genomics Inform 15(1):19