Scaling and Querying a Semantically Rich, Electronic Healthcare Graph

Hayden Freedman[a], Mark A. Miller[a], Heather Williams[a], Christian J. Stoeckert Jr.[ab]
[a]*Institute for Biomedical Informatics, Perelman School of Medicine, University of Pennsylvania, 3700 Hamilton Walk, Philadelphia, PA, 19104, United States*
[b]*Department of Genetics, Perelman School of Medicine, University of Pennsylvania, 415 Curie Boulevard, Philadelphia, PA, 19104, United States*

Abstract. The Open Biomedical Ontologies Foundry (OBO) provides a set of ontologies that can be used to create a semantically rich clinical data model. However, concerns exist regarding the scalability of a pipeline to load clinical data into such a model using Resource Description Framework (RDF). In order to investigate this further, we describe our development of a pipeline for transforming clinical patient data to conform with a model designed using OBO Foundry ontologies, and discuss the runtime and disk space requirements. In order to determine the efficacy of our approach at various throughput levels, we used the synthetic patient data generation service Synthea to generate four patient cohort models of different sizes, the largest containing information about one million patients, and ran each through the pipeline. We also discuss the development of an exemplar question to simulate the type of request we might receive from researchers, and its implementation in the SPARQL query language. We report the results of executing the exemplar question query against each patient cohort model, and conclude that our approach produces a knowledge base which can be generated and queried roughly linearly, and handle requests against large clinical data sets in reasonable time.

Keywords. clinical data, biomedical ontologies, OBO Foundry, semantic richness, common data model, Resource Description Framework

1. Introduction

In recent years, ontologies have become popular for enriching and standardizing clinical data.[1] One particular advantage of this approach is that, while the relationships between data elements may be unclear to users of semantically shallow schemas, they become explicit when the same knowledge is manifest as instances of a well-designed ontology.[2] Specifically, members of the Open Biomedical and Biological Ontologies Foundry community (OBO)[3] develop ontologies following guidelines that foster consistency in the ways that biomedical data from heterogeneous sources can be represented and queried.[4]

Additional benefits can be reaped by using well-designed ontologies represented with the Resource Description Framework (RDF) in a triplestore, a type of graph database. This technique facilitates searches involving relationships between classes in ways which may be harder to implement in relational systems. Specifically, searches which involve traversing a hierarchy of concepts some variable number of steps away from a starting node can take advantage of the convenient syntax of the Property Paths

feature in SPARQL, the query language for RDF. It has been shown that the equivalent of the Property Paths feature can be implemented for relational databases using recursive SQL; however, the syntax is more complex and may involve highly nested clauses that pose a challenge for relational optimizers.[5]

In their paper submitted to the ICBO 2019 conference proceedings, Miller and Stoeckert discuss their pipeline for populating a RDF repository with synthetic Electronic Health Record (EHR) data, using a clinical data model constructed with classes from OBO Foundry ontologies. They incorporated information about roughly 1,000 synthetic patients, and implemented an exemplar question in order to compare the experience of querying both the original relational database and the RDF repository for the same information. Their results demonstrated the viability and advantages of modeling clinical data in a triplestore using a semantically rich model.[6]

However, there are potential challenges involved with scaling such a pipeline to hundreds of thousands or millions of patients. In an effort to determine how well a similar pipeline would scale in terms of time and space requirements for creating, loading, transforming, and querying large quantities of clinical data, we developed synthetic patient cohort models of 1,000, 10,000, 100,000, and 1,000,000 patients, and ran each of them through a pipeline similar to the one described in the original paper.

In this paper, we will first describe our methods for building the synthetic data pipeline with the goal of enabling reproducibility by others, including an analysis of the time and space requirements for each step. Then, we will discuss the development of an exemplar question that incorporates traversals of ontological hierarchies to retrieve information about diagnoses and medications. Finally, we will present the results and time performance of our query against each of the four patient cohort models of various sizes.

2. Methods

Our pipeline utilizes Synthea[7] to generate the synthetic clinical data, the ETL-Synthea service[8] to transform the data into Observational Medical Outcomes Partnership (OMOP)[9] standards, the Carnival application[10] to generate concise, directly-mapped triples[11] from the OMOP database, and the Semantic Engine[12] to transform those triples into the semantically rich model. Each of these steps is described in more detail below.

Step 1: Generating 1,000,000 patient synthetic dataset with Synthea

Synthea is an open source, synthetic patient generation service. One of its goals is to enable research and development with clinical data that would otherwise be impossible if working with real patient data. The service provides realistic data on patients and their associated health records. The Apache-licensed source code can be downloaded from the Synthea Github Repository and run using the command line.

After cloning the Synthea codebase from GitHub, we modified the ‘exporter.csv.export` parameter in the Synthea properties file to `true` so that the program would output the synthetic patient data in CSV format. We then ran the service with the following command:

```
./run_synthea -p 1000000 -o false Pennsylvania Philadelphia
```

The "-p" flag designates the requested population size for the dataset to be generated. The "-o" flag is not listed on the main Synthea documentation page, but it ensures that dead patients will be counted towards the total patient count. If this flag were set to "true" or not explicitly set, the resulting dataset would include one million living patients, and some additional number of dead ones. Specifying a city, in this case Philadelphia, instructs Synthea to generate data representative of people from that city.

Synthea formats the output as a set of fourteen CSV files, each representative of a component of a typical EHR.

Step 2: Loading native Synthea data and converting to OMOP Common Data Model

Once Synthea had generated the set of fourteen CSV files, we configured a PostgreSQL instance and installed Observational Health Data Science and Informatics (OHDSI)'s open source tool ETL-Synthea as an R package to load the data into PostgreSQL. After connecting our PostgreSQL database to the R environment, the script was able to load each CSV file into its own eponymously named table in the PostgreSQL database.

The next step in the ETL-Synthea script was to populate OMOP Common Data Model (CDM) tables based on the native Synthea tables within the PostgreSQL database. OMOP is an OHDSI initiative to harmonize heterogeneous clinical coding formats and allow for the use of standardized analytics tools. Our use of the OMOP format in this case was mainly as a staging area for the generation of RDF triples. However, we anticipate that maintaining a large synthetic dataset in OMOP format will provide utility for future performance testing of our applications.

The ETL-Synthea documentation includes a detailed diagram of how Synthea tables are mapped to OMOP tables.[13] The final result of the pipeline is a PostgreSQL database with two schemas. Schema "native" contains the data in Synthea format, and schema "cdm_synthea10" contains the data in OMOP format.

Step 3: Creation of concise RDF triples from OMOP database

The Carnival application was developed by our collaborators at the University of Pennsylvania as a clinical data aggregation service. It utilizes an in-memory Neo4j property graph database to store data from heterogeneous sources in a core model. Although its use of an in-memory database prevents it from storing large quantities of data, we chose to use this service because it already included facilities to transform data from an OMOP-formatted database into concise RDF triples.

In order to overcome memory issues, we used a batched approach to process chunks of data from the PostgreSQL OMOP database and export them to an instance of Ontotext GraphDB[14] as RDF triples. After each batch was processed, the Carnival graph was cleared. For this experiment, we ran Carnival four times, creating concise RDF models of patient cohorts of 1,000, 10,000, 100,000, and 1,000,000 patients, which were each stored in their own GraphDB repository. Carnival automatically partitions the data into named graphs containing a designated number of entities, so that future processing of the data can also occur in manageable chunks. For this instantiation we set the maximum number of entities per named graph to 100,000.

Step 4: Transformation of concise RDF triples to semantically rich model

Once the four concise cohort models were loaded as Carnival output into GraphDB repositories, our final step was running the Semantic Engine against each of those repositories. As previously reported[12], we developed the Semantic Engine in order to enrich a concise RDF data set by applying a semantically rich ontological model. A user

of the Semantic Engine who wishes to apply it to a new data source must design a new custom configuration that specifies the shape of their incoming RDF data as well as the relevant relationships in the semantically enriched target model. In our case, we had previously created a Semantic Engine configuration to read concise RDF output from Carnival's triples generation service.

The Semantic Engine executes transformations by running dynamically generated SPARQL statements against a set of named graphs. In our case, each of these named graphs contained information about one of the following: patient demographics (gender identity, race, centrally registered identifier, etc.), measurements (height, weight, BMI, blood pressure, etc.), diagnoses (disease or disorder code, versioned source terminology, description string), or medications (medication code, versioned source terminology, description string).

The Semantic Engine is capable of launching queries that run in parallel and process data in multiple named graphs simultaneously. To understand the performance advantages of using the Semantic Engine's parallel processing capabilities, we ran the Semantic Engine twice against each cohort model, once with parallel processing disabled and once with parallel processing enabled (see end of *Section 3.1).*

Step 5: Import and Process Ontologies

The final step in preparing our four patient cohort models was importing additional ontologies from outside sources and processing them for ease of querying. **Table 1** shows the ontologies we included in each cohort model repository, the domains they describe, and how we obtained them in RDF.

After the ontologies were imported, we used SPARQL queries which implemented the Property Path feature to execute a transitive subclass materialization update in each ontology to explicitly state all subclass relationships, regardless of depth in the subclass hierarchy. For example, if **someOntology:classC** is a subclass of **someOntology:classB**, which is itself a subclass of **someOntology:classA**, after running our update we will see explicitly that **someOntology:classC** is a subclass of **someOntology:classA**. The motivation behind materializing these relationships was to pre-process some of the work involved in answering the exemplar question, rather than having to do the same work repeatedly at query time. GraphDB's RDFS+ reasoning service would also have been a reasonable choice to apply these transitive relationships.

Figure 1 shows a visual representation of each of the steps described in this section, demonstrating the flow of data through the pipeline from Synthea CSV files to semantically rich RDF triples.

Table 1: Terminologies included in patient cohort model repositories

Terminology Name	Domain Described	Method of Obtainment in RDF
Mondo Disease Ontology (Mondo)[15]	diseases	downloaded from Monarch Initiative GitHub
Snomed Clinical Terms (SNOMED)[16]	clinical terms	built from UMLS terminologies download
Chemical Entities of Biological Interest (ChEBI)[17]	molecular entities of biological interest and their roles	downloaded from European Bioinformatics Institute

Drug Ontology (DrOn)[18]	clinical drugs	downloaded from BioPortal
RxNORM[19]	clinical drugs	downloaded from BioPortal

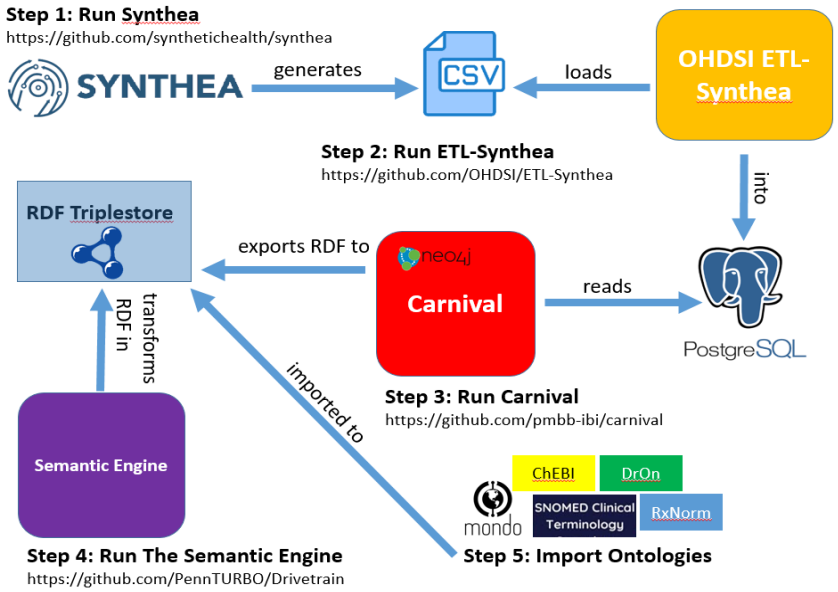

Figure 1: Diagram showing each step of the pipeline for creating our synthetic patient graph, referencing the steps declared in the **Methods** section of this paper

3. Performance

3.1 Creating, Loading, and Transforming

Table 2 shows a comparison of the concise RDF data to the semantically rich RDF data in terms of time and space requirements, and number of triples outputted. The concise RDF dataset grows roughly fourfold on disk when it is transformed to conform with the semantically rich model.

Table 2: Comparison of time and space required for first four steps from the **Methods** section for the 1,000,000 synthetic patient cohort. The disk space metric measures the data in an uncompressed format and does not include indexes or imported ontologies.

Step	Time Consumed by Step	Disk Space Consumed by Data	Number of Triples in Outputted Dataset
Carnival creation of concise RDF triples	45 hours 29 minutes 28 seconds	89.68 GB	903,373,439
Transformation to semantically rich model by	20 hours 33 minutes 0 seconds	348.61 GB	3,522,298,786

Semantic Engine (parallel processing enabled)			

The large growth from concise to semantically rich RDF triples is expected and a necessary component of representing a semantically rich model. In order to accurately represent the semantic context of the data, we instantiate classes even if they are not directly mapped to data elements, rather than only creating one-to-one mappings between data elements and ontology terms. This provides advantages in terms of comprehending the context of what each data element is about, but also takes up more disk space than a concise model. The semantic enrichment is implemented in a uniform and standardized way based on the configurations provided to the Semantic Engine. The expansion ratio of roughly 4:1 between concise and semantically rich RDF datasets should be consistent even as the size of the dataset changes, as long as the same Semantic Engine configuration is used. **Figure 2** provides a visual reference for how an example concise RDF dataset might be semantically enriched by the Semantic Engine.

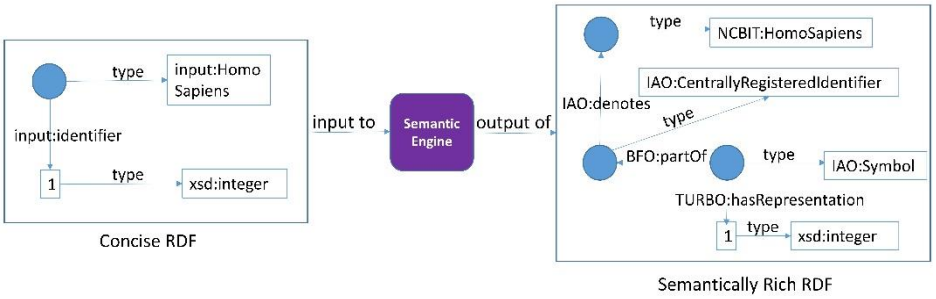

Figure 2: Visual examples of a concise RDF dataset that is input to the Semantic Engine, and a semantically rich RDF dataset that is the output of the Semantic Engine based on the input.

We have completed a comparison of Semantic Engine completion time for various cohort model sizes with and without query parallel processing enabled. When this feature is enabled, up to four RDF named graphs will be pre-processed simultaneously. Although GraphDB allows only a single transaction to be committed at a time, the simultaneous pre-processing leads to time consumption savings, which become more significant as the size of the dataset increases. For the million patient dataset, it cut the time to about half.

The imported ontologies shown in **Table 1** along with the transitive subclass materializations require about 5 gigabytes of additional disc space per patient repository, on top of the space requirements for the patient data shown in **Table 2**. As these ontologies are static resources, their footprint in terms of storage space is not dependent on the size of the patient cohort. These ontologies are included in each patient repository, which leads to some reproduction of the same data across multiple repositories. However, storing these ontologies in a single, shared location and using a federated SPARQL query to access it from the patient repositories caused a steep performance degradation of our exemplar question query.

3.2 Querying for Exemplar Question

For this project, we added two additional clauses about patient diagnosis and medication history to the original exemplar question. These clauses both require hierarchical traversals of imported ontologies. Specifically, we wrote a SPARQL query to count the number of patients who:

- are African-American males born between 1960 and 1980
- have an average systolic blood pressure within the normal range of 110 and 130
- have been diagnosed with a form of hypertensive disorder
- have been prescribed a hypoglycemic agent

Our goal in creating the above exemplar question was to include fields of interest to researchers (demographics, assays, medications, diagnoses) without prioritizing that the actual ranges or classes make clinical sense or will be useful for any particular research study. Additionally, once a query template is created, modifications to the ranges and classes can easily be made. The representativeness of this query for the purpose of research information retrieval was assessed to be adequate based on the fields included in past requests from researchers.

3.2.1 Diagnosis Traversal

Hypertensive disorder is a disease entity represented in the Mondo Disease Ontology by code **MONDO:0005044**. It would be trivial to search for all subclasses of hypertensive disorder within Mondo. However, Synthea diagnoses mention diseases and disorders within the SNOMED ontology. Mondo includes links from its own classes to SNOMED classes, each of which is described using one or more of these predicates:

- **http://www.w3.org/2002/07/owl#equivalentClass**
- **http://www.w3.org/2004/02/skos/core#exactMatch**
- **http://www.w3.org/2004/02/skos/core#closeMatch**
- **http://www.w3.org/2004/02/skos/core#broadMatch**
- **http://www.w3.org/2004/02/skos/core#narrowMatch**
- **http://www.geneontology.org/formats/oboInOwl#hasDbXRef**

It is not safe to assume that diagnoses from other data sources will always contain references to SNOMED, and therefore some alterations may need to be made to the query in order to traverse to other terminologies. For example, it is common for hospital diagnosis data to reference ICD codes, which are helpful for billing. We have established a pattern to discover ICD codes from a given Mondo term by looking for direct mappings from Mondo to ICD as well as paths from Mondo through SNOMED to ICD. Data sources referencing other terminologies would require additional labor for pathway discovery before this type of search could be implemented.

3.2.2 Medication Traversal

A hypoglycemic agent is a drug role represented in the Chemical Entities of Biological Interest (ChEBI) ontology by code **CHEBI:35526**. However, Synthea prescriptions mention specific drugs and compounds from the RxNorm terminology, not from ChEBI. RxNorm does not include drug roles, and there are no direct references in ChEBI to RxNorm or vice versa. The DrON ontology does include references to ChEBI and to RxNorm, although we found that a significant number of the DrON mappings to RxNorm point to inactive terms. Bioportal[20] includes mappings from DrON to RxNorm which we found to be useful and mostly accurate.

In order to map DrON classes to RxNorm classes, we used the Bioportal API to discover mappings between the two and materialized the relationships in our graph. We could then start the medications component of our exemplar question query by finding DrON terms related to a given ChEBI term or any of its direct or indirect subclasses. We then traverse all direct and indirect subclasses of each DrON term discovered, and search for BioPortal mappings to RxNorm from each of the DrON terms. We then have a set of RxNorm terms to be matched against the synthetic patient instance data.

3.2.3 Results

Table 3 shows the results of running our exemplar question against each of the four synthetic patient repositories after transformation to the semantically rich model. The query was implemented with SPARQL and ran using the graphical web-based interface of GraphDB. These trials were executed on an instance of GraphDB Standard Edition 9.1.1, running on a server with the Centos 6.9 operating system and 64 GB of RAM. We ran the query against each patient repository three times, and took an average of the results. The repositories were not repopulated between trials, so the query ran against the same synthetic data each time. The "Time for Query Completion" column was populated using the value reported by GraphDB after rendering the results. We can see that the query completion times generally scale linearly with respect to the cohort model size, with a slight performance degradation when run on the largest repository.

Table 3: Number of patients found and query runtimes for running the exemplar question with SPARQL against each of the four synthetic patient repositories after transformation to the semantically rich model. We observed that the query completion time is roughly linear with respect to the cohort model size.

Patient Cohort Model Size	Qualifying Patients Found	Time for Query Completion
1,000	2	**Average: 1.1 seconds**
10,000	11	**Average: 9.1 seconds**
100,000	153	**Average: 95 seconds**
1,000,000	1,760	**Average: 1,360 seconds**

4. Discussion

4.1 Limitations

Additional optimizations could improve the time or space performance of some steps of the pipeline. One limitation was Carnival's use of an embedded graph database with access to limited amounts of memory. Carnival could generate the concise RDF triples more quickly if the application were hooked up to a dedicated Neo4j property graph database server with significant allocated memory, which would allow for a greater number of patients to be processed at a time and reduce the amount of Neo4j cleanup operations required.

Another improvement could reduce redundancy between the four patient repositories and save storage space by storing the imported ontologies with transitive subclass relationships materialized in a separate repository and using federated queries launched from the patient repositories to access them. Federated queries are a type of SPARQL query that traverse multiple RDF repositories, with some performance degradation compared to traversing a single repository. We attempted to answer our exemplar question using federated queries but found that the query time to completion was not tolerable even on the smaller repositories. Finding a way to launch efficient federated queries in GraphDB would avoid duplicating the ontology data between each of the four repositories, and mean that transitive subclass materializations, mappings between ontologies, and any other changes would only have to be implemented once.

4.2 Using Real Data

Although we used synthetic data for the purposes of this paper, we anticipate that our system will be useful for the storage and retrieval of real patient data as well. We have previously generated a repository containing data about 51,031 real patients in the Penn Medicine Biobank, including information about loss of function mutation predictions about specific genes. This data originated in Penn Data Store, a large clinical data warehouse that is part of the University of Pennsylvania Health System. Similarly to the pipeline described in this paper, we were able to use Carnival to generate concise RDF triples about these patients and the Semantic Engine to transform the data into the semantically rich model. In this case we did not import external ontologies and execute medication and diagnosis related searches, but this exercise did provide a proof-of-concept that our pipeline can be modified to work with various data sources and types of data.

A potential complication of using real data is that it may be frequently changed at the source. For example, the result of a laboratory test assay could be updated in the clinical data warehouse from which our system pulls data. Since our system captures a snapshot of the relevant data at a given point in time, such a change would not be immediately reflected in the RDF output. If the output is expected to include recent updates to the source data, setting up software infrastructure to perform automatic rebuilds of the semantically rich electronic healthcare graph could be a reasonable option.

5. Conclusion

Previous work has shown low-throughput semantic instantiation and querying of clinical data from relational sources using a toy dataset of roughly 1,000 patients and their

associated data. In addition to improving the representation of clinical data relative to relational models, we now show that generating and querying these models scales in a roughly linear fashion. Based on the previous work, we created a new pipeline and used it to instantiate a more practically sized dataset of one million synthetic patients and their associated health record data. We report performance for each of the necessary steps so that others wishing to use these methods can anticipate the time and space requirements.

We discuss the development of an exemplar question to include hierarchy traversals of biomedically-oriented ontologies. Asking questions regarding patients who have taken any of a given class of drugs or received any of a given class of diagnoses using traditional relational database systems can be cumbersome. However, the hierarchy-based nature of ontologies allows these questions to be answered easily using SPARQL against the semantically rich electronic health care graph. We tracked the time to completion of our exemplar question query against each of our generated repositories, and present evidence that our exemplar question can be answered when run against clinical datasets of a practical size. Since the query completion time scales linearly based on the size of the data, it may not be performant to use this pipeline against significantly larger datasets.

Although we would prefer to make every component of our pipeline available as open source projects, one relevant Carnival module, which contains sensitive information about internal data structures, could not be made publically available. However, there are many other open-source tools available to convert data from relational to concise RDF format. We encourage those interested in our methods to provide feedback as we continue to develop and improve the pipeline.

References

[1] M. Ivanović, and Z. Budimac, An overview of ontologies and data resources in medical domains, *Expert Syst. Appl.* **41** (2014) 5158–5166.

[2] K. Munir, and M. Sheraz Anjum, The use of ontologies for effective knowledge modelling and information retrieval, *Applied Computing and Informatics*. **14** (2018) 116–126.

[3] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, C.J. Mungall, OBI Consortium, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R.H. Scheuermann, N. Shah, P.L. Whetzel, and S. Lewis, The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, *Nat. Biotechnol.* **25** (2007) 1251–1255.

[4] B. Smith, and W. Ceusters, Ontological realism: A methodology for coordinated evolution of scientific ontologies, *Appl. Ontol.* **5** (2010) 139–188.

[5] N. Yakovets, P. Godfrey, and J. Gryz, Evaluation of SPARQL Property Paths via Recursive SQL, *AMW*. **1087** (2013). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.2132&rep=rep1&type=pdf.

[6] Miller and Stoeckert. A Collaborative, Realism-Based, Electronic Healthcare Graph: Public Data, Common Data Models, and Practical Instantiation. ICBO 2019 Conference Proceedings. https://drive.google.com/file/d/1eYXTBl75Wx3XPMmCIOZba-8Cv0DIhlRq/view.

[7] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, and S. McLachlan, Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record, *J. Am. Med. Inform. Assoc.* **25** (2018) 230–238.

[8] Simulacra and Simulation: How simulated data can enable OHDSI application development, methods research, and user adoption – OHDSI, (n.d.). https://www.ohdsi.org/2019-us-symposium-showcase-9/ (accessed March 4, 2020).

[9] G. Hripcsak, J.D. Duke, N.H. Shah, C.G. Reich, V. Huser, M.J. Schuemie, M.A. Suchard, R.W. Park, I.C.K. Wong, P.R. Rijnbeek, J. van der Lei, N. Pratt, G.N. Norén, Y.-C. Li, P.E. Stang, D. Madigan, and P.B. Ryan, Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers, *Stud. Health Technol. Inform.* **216** (2015) 574–578.

[10] D. Birtwell, H. Williams, R. Pyeritz, S. Damrauer, and D.L. Mowery, Carnival: A Graph-Based Data Integration and Query Tool to Support Patient Cohort Generation for Clinical Research, *Stud. Health Technol. Inform.* **264** (2019) 35–39.
[11] J. Sequeda, F. Priyatna, and B. Villazón-Terrazas, Relational database to RDF mapping patterns, in: Proceedings of the 3rd International Conference on Ontology Patterns-Volume 929, CEUR-WS. org, 2012: pp. 97–108.
[12] H.G. Freedman, H. Williams, M.A. Miller, D. Birtwell, D.L. Mowery, and C.J. Stoeckert, A novel tool for standardizing clinical data in a realism-based common data model, *bioRxiv*. (2020) 2020.05.12.091223. doi:10.1101/2020.05.12.091223.
[13] *ETL-Synthea*. (n.d.). https://ohdsi.github.io/ETL-Synthea/ (accessed March 30, 2020).
[14] GraphDB - Semantic Web Standards, (n.d.). https://www.w3.org/2001/sw/wiki/GraphDB (accessed May 21, 2020).
[15] O.T. Wg, Mondo Disease Ontology, (n.d.). http://www.obofoundry.org/ontology/mondo.html (accessed May 18, 2020).
[16] S. El-Sappagh, F. Franda, F. Ali, and K.-S. Kwak, SNOMED CT standard ontology based on the ontology for general medical science, *BMC Med. Inform. Decis. Mak.* **18** (2018) 76.
[17] K. Degtyarenko, P. de Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcántara, M. Darsow, M. Guedj, and M. Ashburner, ChEBI: a database and ontology for chemical entities of biological interest, *Nucleic Acids Res.* **36** (2008) D344–50.
[18] J. Hanna, E. Joseph, M. Brochhausen, and W.R. Hogan, Building a drug ontology based on RxNorm and other sources, *J. Biomed. Semantics*. **4** (2013) 44.
[19] S.J. Nelson, K. Zeng, J. Kilbourne, T. Powell, and R. Moore, Normalized names for clinical drugs: RxNorm at 6 years, *J. Am. Med. Inform. Assoc.* **18** (2011) 441–448.
[20] P.L. Whetzel, N.F. Noy, N.H. Shah, P.R. Alexander, C. Nyulas, T. Tudorache, and M.A. Musen, BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications, Nucleic Acids Res. 39 (2011) W541–5.