12th International Workshop on Innovative Simulation for Healthcare
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0371 © 2023 The Authors.
doi: 10.46354/i3m.2023.iwish.001

Enhancing Interoperability of HL7 Resources Using Namespaces in Graph Databases

Andreas Pointner[1,*], Christoph Praschl[1] and Oliver Krauss[1]

[1]University of Applied Sciences Upper Austria Campus Hagenberg, Softwarepark 11, Hagenberg i. M., 4232, Austria

[*]Andreas Pointner. Email address: andreas.pointner@fh-hagenberg.at

Abstract

The adoption of the FHIR (Fast Healthcare Interoperability Resources) standard has led to an exponential growth of modular healthcare data that needs to be managed efficiently. Graph databases such as Neo4j offer an effective way to store and query this data, but can become complex when dealing with FHIR resources that contain numerous extensions. We explore the use of namespaces in Neo4j graph databases to manage FHIR resources and compare it with the existing tool, CyFHIR. We demonstrate that by embedding extensions using the namespace concept, the complexity of the graph can be significantly reduced. Furthermore, we evaluate our approach on a generated dataset and show that the use of namespaces in Neo4j outperforms CyFHIR conventional methods for storing FHIR resources in graph databases. Our findings suggest that the use of namespaces can be a valuable addition to Neo4j graph databases for managing complex FHIR resources.

Keywords: Namespaces; Graph-Databases; HL7 FHIR

1. Introduction

Managing health data is a challenging task, that many organizations and developers around the world face. In order to have a standardized and exchangeable data format the HL7® organization developed a standard to manage such resources, called the Fast Healthcare Interoperability Resources (FHIR®)[1]. For example, they defined a patient resource (HL7 FHIR, 2023b), which allows exchanging patient data in a standardized format.

The FHIR standard is developed via an 80/20 approach, meaning that 80% of use cases that are generic are covered by the standard. This is in some cases too generic, or too restrictive, and thus the remaining 20% of specialized cases can be modeled via profiles and extensions.

For example, there may be country-specific differences in patient data. FHIR offers a profiling method to create a CountryXPatient from the basis of the generic Patient resource. These profiles allow restrictions, for example, to require a field to be set, or to have a specific set of values that can be used in a field. Extensions can be used to add new fields to the resource that may not be covered in the base standard. The profiles are defined as structure definitions and are available for different countries, often defined by the local HL7 affiliate (HL7 FHIR, 2023a).

Having country-specific adaptions to resources enables having a more fine-granular definition covering actual real-world needs. However, this brings the issue that multiple patient profiles (e.g. FrancePatient and GermanyPatient), may be incompatible with each other, due to different requirements in the profile, such as one profile removing a field, and another profile requiring it to be set.

1 HL7®, FHIR® and the FHIR® logo are the registered trademarks of Health Level Seven International and their use does not constitute an endorsement by HL7®.

This would require storing data from different contexts in the same domain object. This task can be achieved by assigning a context to the various data elements. XML namespaces are a prominent option in order to put elements into a specific context. This ensures that the same concept (e.g. field) is covered in different namespaces and incompatibilities are prevented.

While common implementations of a FHIR server, like HAPI (hapifhir, 2023a), are relying on relational databases, there are scenarios, where it may help to represent the data in a graph database. Such databases are especially helpful if many relations between single nodes and entities need to be stored and queried. Similar to the mentioned profiles in FHIR, respectively namespaces in XML, Pointner et al. (2022) showed a concept for storing data with different expressions using a Neo4j graph database.

The goal of our work is to demonstrate the use of namespaces in graph databases, as shown by Pointner et al. (2022), and apply it in the context of HL7 FHIR resources. Using this concept, it is possible to retrieve different views for specific profiles only containing single namespaces, more easily than with conventional methods.

The remainder of this work outlines an overview of related work on FHIR in graph databases research in section 2. Section 3 describes our approach to obtain store FHIR resources in graph databases using namespaces. It further states the research questions we seek to answer, as well as the experiment. The results of our experiment are presented in section 4. Finally, section 5 concludes the paper with some final remarks and possible future work.

2. State of the art

We compare existing approaches concerning the use of HL7 FHIR in graph databases with our work.

CyFHIR(Optum, 2023) is a Neo4j plugin targeting to act as a bridge between FHIR HL7 resources and Neo4j. The process proposed by the authors is automatically mapping all the data from an exchange file such as JSON or XML to Neo4j graph nodes. Additionally, they provide a custom API based on the APOC (neo4j, 2023) extension framework to query FHIR Resources that meet the HL7 standard. Due to their mapping approach, some information such as extensions is stored in additional nodes instead within the actual resource, leading to an increased number of branches in the graph. In contrast, our work is focussing on showing a concept of how HL7 FHIR extensions are stored together with the base information in one node using the concept of namespaces.

Fette et al. (2019) developed an approach for executing Clinical Quality Language (CQL) queries inside the Neo4j database. In order to test their approach, they stored HL7 FHIR resources inside Neo4j. They are using a CSV representation of their data as exchange format. To do so, they are exporting all primitive type fields as columns in the CSV, and all relations and complex types are exported in separate CSV files. These files are then used to import the data into Neo4j, by creating a node for each row of the CSV file. While their work primarily focuses on the CQL and how to use it in Neo4j, our work focuses on mapping extensions and resources from different contexts into custom namespaces.

Kharwar (2022) showed an example on how to use the graph database Neo4j on FHIR data. In their example, they showed the strength that a query language like Cypher offers when traversing graph-like structures. However, the authors didn't focus on the used data model for persisting HL7 FHIR resources as a graph. In our approach, we want to show the advantages of such a data model and the utilization of namespaces to provide even more querying options.

The primary difference of these approaches, compared to our work is, that they are creating a new node for every different type and extensions that is contained in the data. In contrast, our work embeds extensions nodes into the main node, which highly reduces the complexity of the graph.

3. Materials and Methods

The used data is synthetically generated using Synthea (syntheticheath, 2023) and hosted using a HAPI FHIR R4 Server. This data set is based on the HL7 R4 base domain model (hapifhir, 2023b) and contains different extensions including additional `Patient` information from the Synthea generation process such as `disabilityAdjustedLifeYears` and also FHIR extensions including a patient's `mothersMaidenName`. Additionally, `Addresses` are extended by coordinates from the World Geodetic System of 1984 (Kumar, 1988) with `longitude` and `latitude` information. An example of a patient with such extensions is shown in Listing 3. Pointner et al. (2023) have shown in a recent publication, how to make `AuditEvents` useable in the context of process mining applications in health care by defining custom extensions. For this reason, the artificially generated data set is also extended by such events including two attributes (`timestamp` and `action`) and a reference to the affected patient, as shown in Listing 5. Like this, the classes of the base model are extended by additional attributes but also by relationships. An excerpt of this data set including the most important domain objects is shown in Figure 1 using the Enhanced Entity Relationship (EER) model as defined by Teorey et al. (1986) with Chen Notation (Chen, 1976).

FHIR offers a strong extension mechanism, which is built upon the concept of having an additional element inside the main element. This *extension* element holds a list of extensions that were made to the actual object. Such an extension always contains a URL, which uniquely identifies the extensions, and an extension value. The value can be either another complex object, like a coding or a reference, or it may be a simple object like a string or a decimal value.

The goal of our work is to bring HL7 FHIR resources

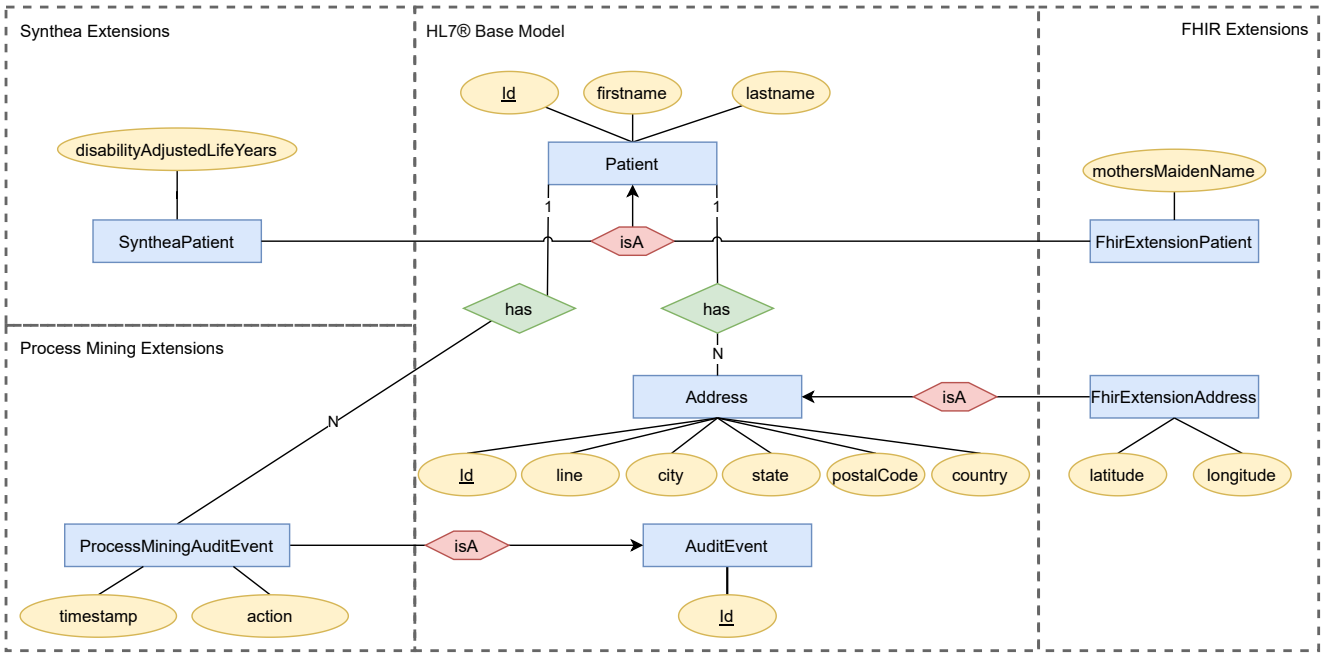

Figure 1. Subset of the HL7 domain model with different extension namespaces including Synthea, FHIR and an extension package intended for process mining purposes.

into a graph database and add such extensions using the namespace concept proposed by Pointner et al. (2022). Conventional mapping methods, such as proposed by Optum (2023) are automatically creating a node for every complex object seen in the list. Using our approach it should be possible, to have much more information embedded in the node itself, which in turn should make it easier for querying the information and increasing memory efficiency. While in theory, it would be possible to apply this concept to complex objects as well, by just embedding the data as JSON structure, we are only applying it to simple types such as strings and decimals.

Because the used implementation of namespaces in Neo4j requires adding specific Java-Annotations to the domain classes, we currently cannot create a fully automated mapping. Instead, we are creating custom Java domain classes and mapping the data accordingly. We are using custom namespace names, which are inspired by the extension URL names. For the evaluation of the proposed methodology, synthetic data created with Synthea is loaded from a HAPI FHIR, transformed to a domain model containing namespaces for the available extensions and uploaded to a Neo4j database using the object-to-graph mapper implementation proposed by Pointner et al. (2021). This illustrative example is shown in Figure 2.

4. Results and Discussion

The presented approach using the namespace concept of Pointner et al. (2022) is capable of storing FHIR resources within the Neo4j graph database, merging the resource's base information and its extensions within one node. A node containing one patient's data stored in Neo4j is shown in Figure 3, together with nodes pointing to the patient. Furthermore, the JSON representation of a `Patient` and an `AuditEvent` is shown in Listing 4 and Listing 6 respectively. An underscore is used as a separator between the namespace and the actual key. That can be seen in both the labels and the properties of a single node. As an example in the *AuditEvents* a namespace `pm` is used. Thus, the representing domain object in java `PMAuditEvent` is prefixed with `pm_` as well as the two properties `action` and *timestamp*.

We compare our approach and the data mapped by CyFHIR. CyFHIR automatically creates a new node for every complex element inside a node, as well as a new node for each extension inside the graph. In contrast to that, our approach makes use of the namespace concept for extensions of simple types. This allows us to highly reduce the number of nodes needed to represent the data. For a comparison, Figure 4 shows the patient graph created with the use of namespaces, whereas Figure 5 shows the graph created by CyFHIR. Compared to CyFHIR which produces a graph of 161 nodes, we have a less complex graph with only 20 nodes. This comparison may be distorted, as CyFHIR creates separate nodes for the name of the patient which is included in the `Patient` node. When not counting these nodes the comparison is still 140 nodes from CyFHIR to 20 nodes from our approach.

To reproduce this comparison, the queries shown in Listing 1 and Listing 2 can be used. Listing 1 collects all nodes except `AuditEvents`, which only results in nodes that

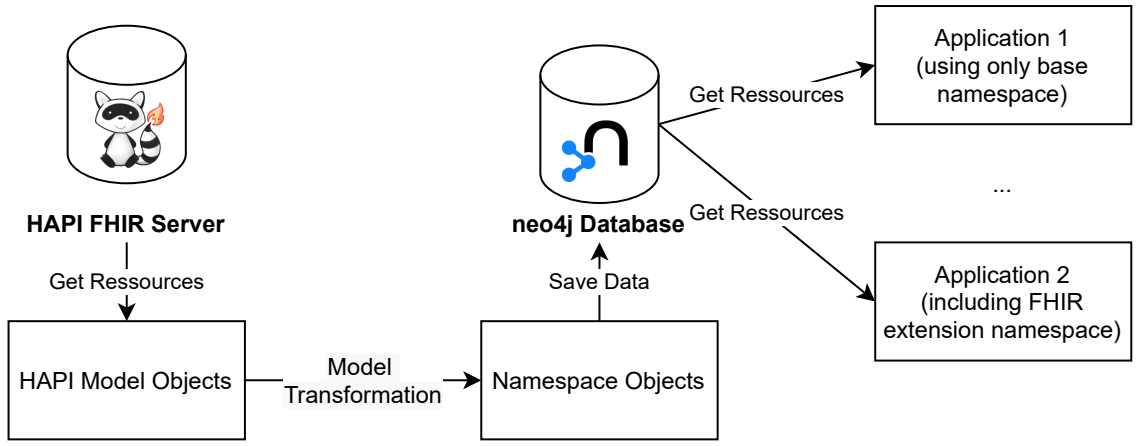

Figure 2. Process used to evaluate the approach of saving FHIR resources to a Neo4j database with respect to extension namespaces.

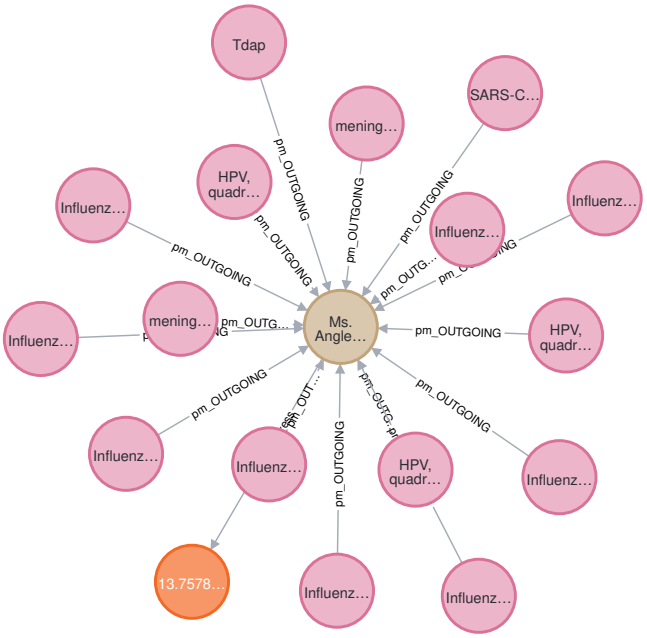

Figure 3. Subgraph stored in the Neo4j database. Shown is a single `Patient` (brown) with all connected *AuditEvents* (purple) and the address (orange) of the patient.

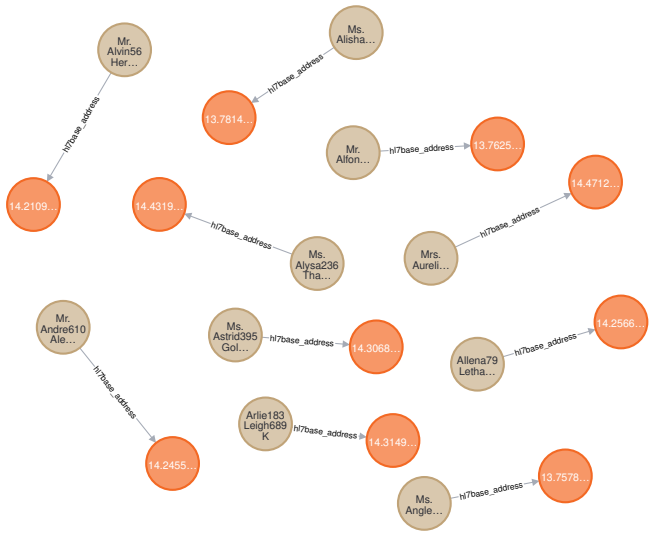

Figure 4. Graph of the patients stored with our namespace concept. It contains the patient resource with its address node.

are related to a `Patient`. CyFHIR is creating additional meta nodes, namely *entry* and *request*. As we are comping the number of nodes concerning FHIR resources in this work, it is necessary to filter out these nodes. In order to remove this information, we are selecting all *resources* nodes representing a *Patient* using Cypher. Starting from such nodes we are matching all elements that have outgoing nodes of any depth starting from the selected patients. In order to not falsely take the name into account, we are filtering those nodes out as well. This query is shown in Listing 2.

The replication package containing the source code and datasets used in this study are open-sourced and can be found at anonymized.

Listing 1. Query for our namespace approach used to match all patient and their connection, excluding AuditEvents.

```
MATCH (x)
WHERE NOT (x:hl7base_AuditEvent)
RETURN x
```

Listing 2. Query for CyFHIR to match all patients and their connection, excluding the meta information such as request and entry nodes and name nodes.

```
MATCH (x:resource)-[*..]->(y)
WHERE NOT (y:name)
RETURN x, y
```

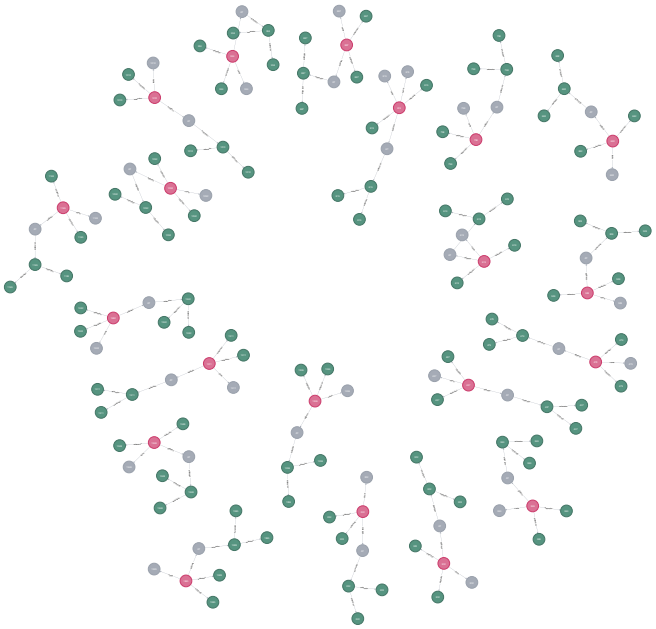

Figure 5. Graph of the patients as they are created with CyFHIR. It contains the patient, extensions, address, and name node.

5. Conclusions

We showed the utilization of the namespace concept in graph databases, enabling a reduction of the complexity of FHIR resource graphs. We achieve this by using namespaces to embed extensions directly into the corresponding nodes. In comparison to CyFHIR we produce a graph that is only 1/7 as large (140 nodes CyFHIR vs. 20 nodes via our namespace approach), drastically reducing the graph complexity and corresponding query performance. In addition, our namespace concept, similar to XML, prevents the issue of FHIR profiles being incompatible with each other within a single resource.

The main limitation of our work currently is, that it is necessary to create a mapping between the FHIR resources and the custom elements, that are used for the namespace concept, requiring developer effort.

In the future, we plan to overcome this limitation by implementing an automated mapping between namespace elements and FHIR extensions. This can be achieved because FHIR profiles are machine-readable and can be used for exactly such a purpose. This concept could be further enhanced by adding some sort of extension concept directly into the Neo4j namespace concept.

Currently, we only consider the storage of data according to profiles and ignore validation of FHIR resources, e.g. if they are conformant to a profile, and upholding all its constraints. This could be achievable by mapping FHIR profile constraints to Neo4j Cypher queries.

6. Funding

This project is financed by research subsidies granted by the government of Upper Austria. It is part of the Process Intelligence and Conformance Auditing (PICA) project in cooperation with the campus for business and management of the University of Applied Sciences Upper Austria in Steyr.

References

Chen, P. P. S. (1976). The entity relationship model toward a unified view of data. *ACM transactions on database systems (TODS)*, 1(1):9–36.

Fette, G., Kaspar, M., Liman, L., Ertl, M., Krebs, J., and Puppe, F. (2019). Implementation of a hl7-cql engine using the graph database neo4j. *Studies in health technology and informatics*, 267:46–51.

hapifhir (2023a). hapi-fhir. https://github.com/hapifhir/hapi-fhir. [Online; accessed 3. Apr. 2023].

hapifhir (2023b). hapi-fhir-structures-r4 6.2.2. https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-structures-r4/6.2.2. [Online; accessed 3. Apr. 2023].

HL7 FHIR (2023a). Hl7 core profile austrian patient - fhir v4.0.1. https://fhir.hl7.at/r4-core-ballot_2021_01/StructureDefinition-austrian-patient.html. [Online; accessed 3. Apr. 2023].

HL7 FHIR (2023b). Patient - FHIR v4.0.1. https://hl7.org/fhir/R4/patient.html. [Online; accessed 3. Apr. 2023].

Kharwar, R. (2022). Getting FHIR'ed up with a Graph Database(neo4j). *Medium*.

Kumar, M. (1988). World geodetic system 1984: A modern and accurate global reference frame. *Marine Geodesy*, 12(2):117–126.

neo4j (2023). Awesome Procedures On Cypher (APOC) - Neo4j Labs. [Online; accessed 21. Jun. 2023].

Optum (2023). CyFHIR. https://github.com/Optum/CyFHIR. [Online; accessed 3. Apr. 2023].

Pointner, A., Krauss, O., Erhard, A., Schuler, A., and Helm, E. (2023). Multi-Perspective Process Mining Interfaces for HL7 AuditEvent Repositories: XES and OCEL. In *dHealth*. in press.

Pointner, A., Praschl, C., and Krauss, O. (2021). AIST Neo4j.

Pointner, A., Praschl, C., and Krauss, O. (2022). Towards modelling namespaces in graph databases. In *Proceedings of the European Modeling & Simulation Symposium, EMSS*. CAL-TEK srl.

synthetichealth (2023). Synthetic Patient Population Simulator. https://github.com/synthetichealth/synthea. [Online; accessed 3. Apr. 2023].

Teorey, T. J., Yang, D., and Fry, J. P. (1986). A logical design methodology for relational databases using the extended entity-relationship model. *ACM Comput. Surv.*, 18(2):197–222.

Appendix

Listing 3. JSON representation of a FHIR patient, with two extensions in the patient and the extension of the address.

```
{
  "resourceType": "Patient",
  "id": "222",
  "extension": [
    {
      "url": "http://hl7.org/fhir/
          StructureDefinition/patient-
          mothersMaidenName",
      "valueString": "Logan497 Emard19"
    },
    {
      "url": "http://synthetichealth.github
          .io/synthea/disability-adjusted-
          life-years",
      "valueDecimal": 0.30
    },
  ],
  "name": [
    {
      "use": "official",
      "family": "Mohr916",
      "given": ["Alfonzo975"]
    }
  ],
  "address": [
    {
      "extension": [
        {
          "url": "http://hl7.org/fhir/
              StructureDefinition/
              geolocation",
          "extension": [
            {
              "url": "latitude",
              "valueDecimal": 47.96
            },
            {
              "url": "longitude",
              "valueDecimal": 13.76
            }
          ]
        }
      ],
      "line": ["732 Jacobi Rest Suite 45"],
      "city": "Gmunden",
      "state": "Ober\u00c3\u00b6sterreich",
      "postalCode": "4810",
      "country": "AT"
    }
  ]
  ...
}
```

Listing 4. JSON representation of the Patient data stored inside the Neo4j database.

```
{
  "identity": 83,
  "labels": [
    "fhir.structuredefinition
        _FhirExtensionPatient",
    "hl7base_Patient",
    "synthea_SyntheaPatient"
  ],
  "properties": {
    "hl7base_name": "Ms. Anglea614 Dawn804
        Adams676",
    "fhir.structuredefinition
        _mothersMaidenName": "Edyth30
        Botsford977",
    "synthea_disabilityAdjustedLifeYears":
        0.0
  }
}
```

Listing 5. JSON representation of an AuditEvent with four extensions, one to reference the patient and the other three to enable process mining (Pointner et al., 2023).

```
{
  "resourceType": "AuditEvent",
  "id": "283",
  "extension": [
    {
      "url": "http://fhirr5.ext/
          occurredDateTime",
      "valueDateTime": "2013-06-06T00:16:47
          +02:00"
    },
    {
      "url": "http://fhirr5.ext/patient",
      "valueReference": {
        "reference": "Patient/222"
      }
    },
    {
      "url": "http://fhirr5.ext/encounter",
      "valueReference": {
        "reference": "Encounter/223"
      }
    },
    {
      "url": "http://fhirr5.ext/code",
      "valueCoding": {
        "system": "http://snomed.info/sct",
        "code": "53741008",
        "display": "Coronary Heart Disease"
      }
    }
  ]
}
```

Listing 6. JSON representation of the AuditEvent data stored inside the Neo4j database.

```
{
  "identity": 168,
  "labels": [
    "hl7base_AuditEvent",
    "pm_PMAuditEvent"
  ],
  "properties": {
    "pm_action": "HPV, quadrivalent",
    "pm_timestamp": "2014-12-27T13:57:41+01
        :00"
  }
}
```