A Virtual Knowledge Graph Based Approach for Object-Centric Event Logs Extraction

Jing Xiong[1 [0000−0002−3604−9645]], Guohui Xiao[2,3,4 [0000−0002−5115−4769]], Tahir Emre Kalayci[5 [0000−0001−6228−1221]], Marco Montali[1 [0000−0002−8021−3430]], Zhenzhen Gu[1 [0000−0002−7346−6093]], and Diego Calvanese[1,4,6 [0000−0001−5174−9693]]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
[2] University of Bergen, Bergen, Norway
[3] University of Oslo, Oslo, Norway
[4] Ontopic S.R.L., Bolzano, Italy
[5] Virtual Vehicle Research GmbH, Graz, Austria
[6] Umeå University, Umeå, Sweden

Abstract. Process mining is a family of techniques that support the analysis of operational processes based on event logs. Among the existing event log formats, the IEEE standard eXtensible Event Stream (XES) is the most widely adopted. In XES, each event must be related to a single case object, which may lead to convergence and divergence problems. To solve such issues, object-centric approaches become promising, where objects are the central notion and one event may refer to multiple objects. In particular, the *Object-Centric Event Logs* (OCEL) standard has been proposed recently. However, the crucial problem of extracting OCEL logs from external sources is still largely unexplored. In this paper, we try to fill this gap by leveraging the Virtual Knowledge Graph (VKG) approach to access data in relational databases. We have implemented this approach in the OnProm system, extending it to support both XES and OCEL standards. We have carried out an experiment with OnProm over the Dolibarr system. The evaluation results confirm that OnProm can effectively extract OCEL logs and the performance is scalable.

Keywords: Process mining · Object-centric event logs · Virtual knowledge graphs · Ontology-based data access.

1 Introduction

Process mining [1] is a family of techniques relating the fields of data science and process management to support the analysis of operational processes based on event logs. To perform process mining, normally the algorithms and tools expect that the event logs follow certain standards. However, in reality, most IT systems in companies and organizations do not directly produce such logs, and the relevant information is spread in legacy systems, e.g., relational databases. Event log extraction from legacy systems is a key enabler for process mining [8,7,6].

There have been several proposals for the representation of event logs, e.g., eXtensible Event Stream (XES) [16], JSON Support for XES (JXES) [15], Open

SQL Log Exchange (OpenSLEX) [14], and eXtensible Object-Centric (XOC) [12], where XES is the most adopted one, being the IEEE standard for interoperability in event logs [11]. In XES (and other similar proposals), each event is related to a single case object, which leads to problems with convergence and divergence [2], as later explained in Section 2.2. To solve these issues, object-centric approaches become promising, where objects are the central notion, and one event may refer to multiple objects. In particular, along this direction, the *Object-Centric Event Logs* (OCEL) standard [10] has been proposed recently.

To the best of our knowledge, the crucial problem of extracting OCEL logs from external sources is still largely unexplored. The only exception is [3], where OCEL logs are extracted by identifying the so-called master and relevant tables in the underlying database and building a *Graph of Relations (GoR)*. Though promising, this approach might be difficult to adopt when the underlying tables are complex and the GoR is hard to model because it does not separate the storage level (i.e., the database) from the concept level (i.e., domain knowledge about events).

In this work, we try to fill this gap by leveraging the OnProm framework [5,7] for extracting event logs from legacy information systems. OnProm v1 was already relying on the technology of Virtual Knowledge Graphs (VKG) [17] to expose databases as Knowledge Graphs that conform to a conceptual model, and to query this conceptual model and eventually generate logs by using ontology and mapping-based query processing. Using VKG, a SPARQL query q expressed over the virtual view is translated into a query $\mathcal{Q}$ that can be directly executed on a relational database $\mathcal{D}$, and the answer is simply the RDF graph following the standard SPARQL semantics. The workflow of OnProm v1 for extraction of event logs in XES format from relational databases consists of three steps: *conceptual modeling*, *event data annotations*, and *automatic event log extraction* [7]. OnProm came with a toolchain to process the conceptual model and to automatically extract XES event logs by relying on the VKG system Ontop [18].

We present here OnProm v2, which we have modularized so that it becomes easier to extend, and in which we have implemented OCEL-specific features to extract OCEL logs. We have carried out an experiment with OnProm over the Dolibarr Enterprise Resource Planning (ERP) & Customer Relationship Management (CRM) system. The evaluation results confirm that OnProm can effectively extract OCEL logs. The code of OnProm and the data for reproducing the experiment can be found on GitHub `https://github.com/onprom/onprom`.

2 Event Log Standards: XES and OCEL

A variety of event log standards have emerged in the literature. In this paper, we are mostly interested in the XES and OCEL standards.

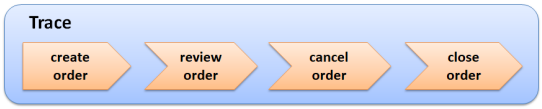
2.1 XES Standard

The *eXtensible Event Stream* (XES) is an XML-based standard for event logs. It aims to provide an open format for event log data exchange between tools and applications. Since it first appeared in 2009, it has quickly become the de-facto

standard in Process Mining and eventually became an official IEEE standard in 2016 [4]. The main elements of the XES standard are *Log*, *Trace*, *Event*, *Attribute*, *Extension*, and *Classifier*. We emphasize the following points:

- Log is the root component in XES.
- The Trace element is directly contained in the Log root element. Each trace belongs to a log, and each log may contain many traces.
- Each event belongs to a trace, and each trace usually contains many events.
- All information in an event log is stored in attributes. Both traces and events may contain an arbitrary number of attributes.

Example 1: Consider the order management process in an ERP system, and suppose there is an instance of order cancellation. Taking the order as a *case*, there is a trace containing *events* such as *create order*, *review order*, *cancel order*, and *close order*, as shown below.



◁

2.2 OCEL Standard

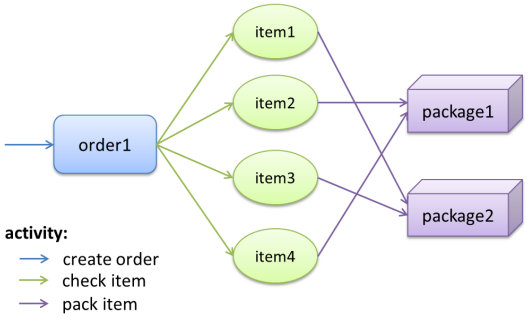
The purpose of the *Object-Centric Event Logs* (OCEL) standard is to provide a general standard to interchange event data with multiple case notions. Its goal is to exchange data between information systems and process mining analysis tools. It has been proposed recently as the mainstream format for storing object-centric event logs [10].

The main elements of the OCEL standard are *Log*, *Object*, *Event*, and *Element*. The main difference between XES and OCEL lies in the usage of *Case* in XES and *Object* in OCEL. Recall that XES requires a single case to describe events. In contrast, in OCEL the relationship between objects and events is many-to-many. This gives OCEL several advantages [10] compared with existing standards:

- It can handle application scenarios involving multiple cases, thus making up for the deficiencies of XES.
- Each event log contains a list of objects, and the properties of the objects are written only once in the log (and not replicated for each event).
- In comparison to tabular formats, the information is strongly typed.
- It supports lists and maps of elements, while most existing formats (such as XOC, tabular formats, and OpenSLEX) do not properly support them.

One main motivation for OCEL is to support multiple case notions. Using a traditional event log standard like XES may lead to problems of convergence (when an event is related to different cases and occurs repetitively) and divergence (when it is hard to separate events with a single case) [10]. We show these in the following example.

Example 2: Considering again an ERP system as in Example 1, when a valid order has been confirmed and payment has been completed, the goods are about to be delivered. Usually the items in the same order may come from different warehouses or suppliers, and may be packaged into different packages for delivery, as shown below.



Suppose that we want to use XES to model this event log. If *item* is regarded as a case and *create order* is regarded as an activity, then *create order* will be repeated because there are multiple items (e.g., item1, item2, ...), even if there is only one order *order1*. This is the *convergence* problem.

If *order* is regarded as a case and *pack item* and *check item* are regarded as activities, then in the same *order* case, there are multiple *pack item* events that should be executed after the *check item* events. However, we cannot distinguish different items in an *order*, and the order between the two activities may be disrupted. This is the *divergence* problem.

The OCEL standard is a good solution to these problems, since they can be easily solved by treating *order*, *item*, and *package* as objects, and then each event can be related to different objects. In this way, the properties of the objects are written only once in the event log and not replicated for each event. ◁

3 The OnProm v2 Framework

We describe now the OnProm approach for event log extraction. OnProm v1, which supports only the XES standard, has been discussed extensively in [5,7]. We describe here the revised version v2, which has a better modularized architecture and supports also OCEL. The architecture of OnProm v2 is shown in Figure 1. We first briefly introduce the basic components of the framework.

To extract from a legacy *information system* $\mathcal{I}$, event logs that conform to an event log standard X, OnProm works as follows:

(*A*) *Creating a VKG specification.* The user designs a domain ontology $\mathcal{T}$ using the *UML Editor* of OnProm. Then they create a VKG mapping $\mathcal{M}$ (using, e.g., the Ontop plugin for Protégé [18]) to declare how the instances of classes and properties in $\mathcal{T}$ are populated from $\mathcal{I}$. This step is only concerned with modeling the domain of interest and is agnostic to the event log standard.

(*B*) *Annotating the domain ontology with the event ontology.* OnProm assumes that for the event log standard X, a specific event ontology $\mathcal{E}_X$ is available. The *Annotation Editor* of OnProm imports $\mathcal{E}_X$, and allows the user to create annotations $\mathcal{L}_X$ over the classes in $\mathcal{T}$ that are based on the classes of $\mathcal{E}_X$.

(*C*) *Extracting the event log.* OnProm assumes that for the standard X also a set of SPARQL queries for extracting the log information is defined. The *Log Extractor* of OnProm relies on a conceptual schema transformation approach [6] and query reformulation of Ontop, using $\mathcal{L}_X$, $\mathcal{T}$, $\mathcal{M}$, and $\mathcal{R}$. It

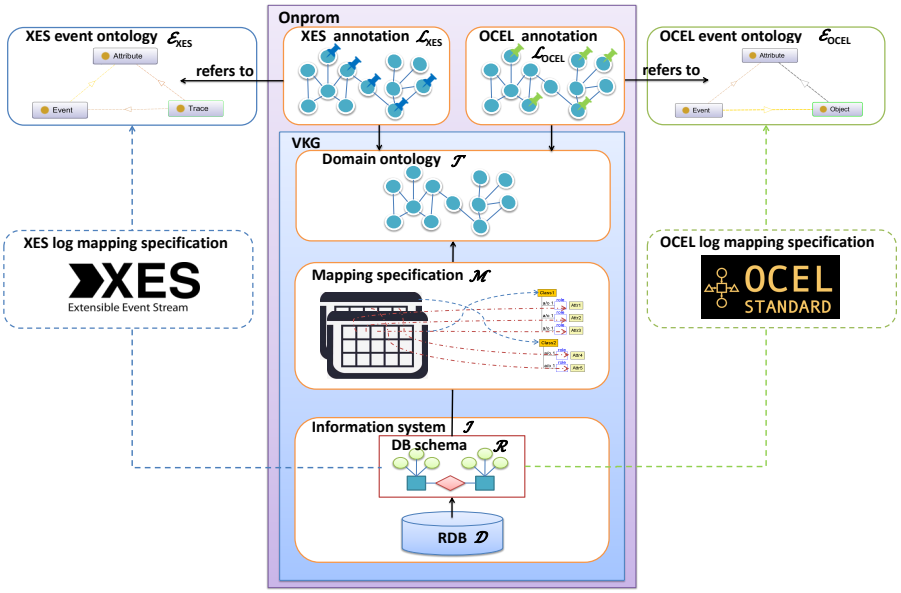


Fig. 1: OnProm event log extraction framework.

internally translates these SPARQL queries to SQL queries over $\mathcal{I}$, and evaluates the generated SQL queries to construct corresponding Java objects and serialize them into log files compliant with X.

In this work, we have first modularized the system, by separating the above steps into different software components, so as to make it more extensible. Then we have introduced OCEL-specific features in Steps (B) and (C). Hence, OnProm v2 is now able to extract OCEL logs from relational databases.

Below we detail these steps and provide a case study with the Dolibarr system. This example also serves as the base of the experiments in the next section. Dolibarr [9] is a popular open source ERP & CRM system. It uses a relational database as backend, and we consider a subset of the tables that are related to the *Sale Orders*. We model it as an information system $\mathcal{I} = \langle \mathcal{R}, \mathcal{D} \rangle$, where the schema $\mathcal{R}$ consists of 9 tables, related to product, customer, order, item, invoice, payment, shipment, etc., and the data $\mathcal{D}$ includes instances of the tables, a sample of which is shown in Table 1. Note that the table name is not immediately understandable (`llx_commande` is a table about orders).

Table 1: Table `llx_commande`.

rowid	fk_soc	ref	date_creation	date_valid	total_ttc
38	3	C07001-0010	2017-02-16 00:05:01	2021-02-16 00:05:01	200.00
40	10	C07001-0011	2017-02-16 00:05:10	2021-02-16 00:05:10	1210.00

3.1 Creating a VKG Specification

In this step, we define a domain ontology $\mathcal{T}$ and a mapping $\mathcal{M}$ for creating a VKG from $\mathcal{I} = \langle \mathcal{R}, \mathcal{D} \rangle$. This is to provide a more understandable knowledge graph view of the underlying data in terms of the domain.

The domain ontology $\mathcal{T}$ is a high-level abstraction of the business logic concerned with the domain of interest. The UML editor in OnProm uses UML class diagrams as a concrete language for ontology building and provides their logic-based formal encoding according to the OWL 2 QL ontology language [13]. In this case study, the domain ontology about *Sale Orders* is constructed using the UML editor as shown in Figure 2.

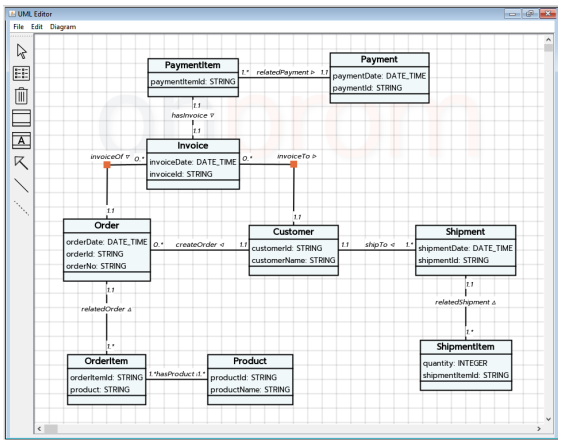


Fig. 2: Domain ontology in the UML editor.

In a VKG system, the domain ontology $\mathcal{T}$ is connected to the information system $\mathcal{I}$ through a declarative specification $\mathcal{M}$, called *mapping*. More specifically, $\mathcal{M}$ establishes a link between $\mathcal{I}$ and $\mathcal{T}$. The mapping $\mathcal{M}$ is a collection of mapping assertions, each of which consists of a SQL statement (called Source) over $\mathcal{I}$ and a triple template at the data concept schema level (called Target) over $\mathcal{T}$. For example, the following mapping assertion constructs instances of the `Order` class in $\mathcal{T}$, with their creation date, from a SQL query over the `llx_commande` table:

```
SELECT rowid, date_creation  FROM llx_commande                              (Source)
⇝ :Order/{rowid} a :Order; :createDate {date_creation}^^xsd:dateTime.       (Target)
```

By instantiating `rowid` and `date_creation` with the values from the first row of `llx_commande` in Table 1, this mapping assertion would produce two triples:
`:Order/38 a :Order. :Order/38 :createDate "2017-02-16T00:05:01"^^xsd:dateTime.`

The mapping can be edited using the Ontop plugin for Protégé. Figure 3 shows the mapping for the running example.

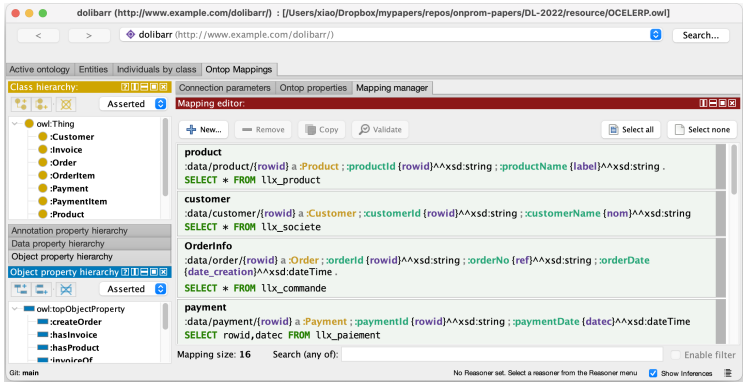


Fig. 3: Mapping in the Ontop plugin for Protégé.

3.2 Annotating the Domain Ontology with the Event Ontology

In this step, we establish the connection between the VKG and the event ontology. This is achieved by annotating the classes in the domain ontology using the elements from the event ontology.

An event ontology $\mathcal{E}$ is a conceptual event schema, which describes the key concepts and relationships in an event log standard. For the OCEL standard, we have created an ontology $\mathcal{E}_{\text{OCEL}}$, whose main elements are shown in Figure 4.

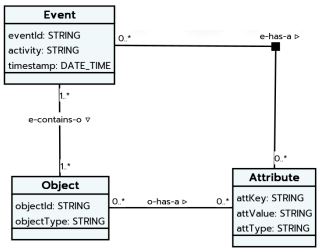


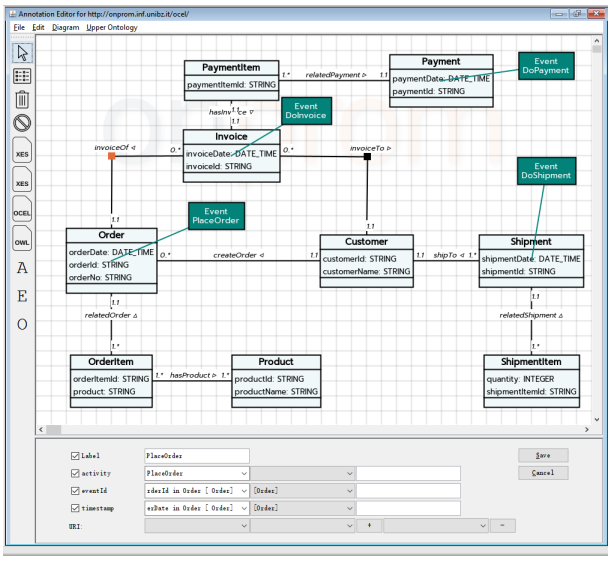
Fig. 4: OCEL event ontology.

In this ontology, the classes *Event* and *Object* are connected by the many-to-many relation *e-contains-o*. One event may contain multiple objects, and an object may be contained in multiple events. Events and objects can be related to attributes through the relations *e-has-a* and *o-has-a*, respectively. An attribute has a name (*attKey*), a type (*attType*), and a value (*attValue*).

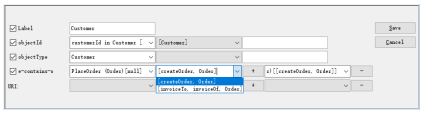
Now, using the *Annotation Editor* in the OnProm tool chain, we can annotate the classes in $\mathcal{T}$ using the elements from $\mathcal{E}$ to produce an annotation $\mathcal{L}$. For OCEL, there are three kinds of annotations:

– The *event annotation* specifies which concepts in $\mathcal{T}$ are OCEL events. Each event represents an execution record of an underlying business process and contains mandatory (e.g., id, activity, timestamp, and relevant objects) and optional elements (e.g., event attributes). A screenshot of such example annotation is shown in Figure 5(a), where we annotate the `Order` class with an *Event* and specify its properties label, activity, eventId, and timestamp.

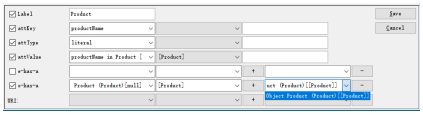
– The *object annotation* specifies which concepts in $\mathcal{T}$ are OCEL objects. An OCEL object contains mandatory elements (e.g., id and type) and optional elements (e.g., price and weight). A screenshot of an example object annotation is shown in Figure 5(b).
– The *attribute annotation* specifies the attributes attached to the events/objects. Both an event and an object may contain multiple attributes, and each attribute annotation consists of an *attKey*, an *attType*, and an *attValue*. A screenshot of an (event) attribute annotation using the *Annotation Editor* tool is shown in Figure 5(c).



(a) Event annotation over `Order`.



(b) Object annotation over `Customer`.



(c) Attribute annotation over `Product`.

Fig. 5: Annotation samples.

3.3 Extracting the Event Log

Once the annotation is concluded, OnProm will compute a new VKG specification $\mathcal{P}'$ with a new mapping $\mathcal{M}'$ and the event ontology $\mathcal{E}$, so that it exposes the information system $\mathcal{I}$ as an VKG using the vocabulary from the event standard. For example, among others, OnProm produces a new mapping assertion in $\mathcal{M}'$ from the Dolibarr database to the *Event* classes in $\mathcal{E}_{\text{OCEL}}$.

```
SELECT v1.‘rowid‘, v1.‘date_creation‘                                    (Source)
FROM ‘llx_commande‘ v1 WHERE v1.‘date_creation‘ IS NOT NULL
⇝ :PlaceOrder/{rowid} a ocel:Event ;                                    (Target)
   :timestamp {date_creation}^^xsd:dateTime .
```

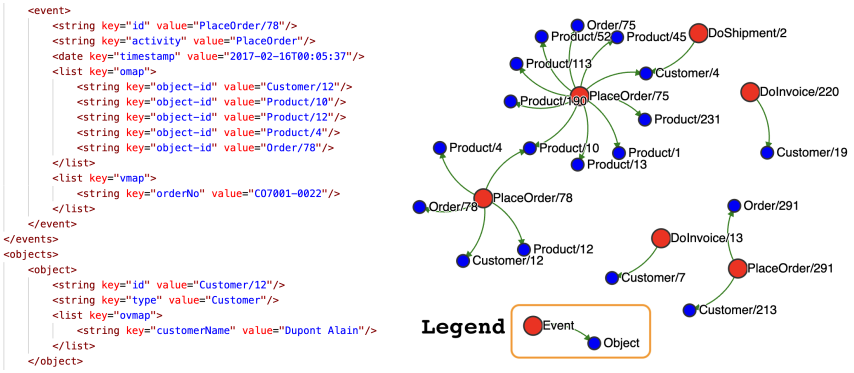
Now all the information in an OCEL log can be obtained by issuing several predefined SPARQL queries over $\mathcal{P}'$. For example, the following query extracts all OCEL events and their attributes:

```
PREFIX ocel: <http://onprom.inf.unibz.it/ocel/>
SELECT DISTINCT * WHERE { ?event a ocel:Event .
    OPTIONAL { ?event ocel:e-has-a ?att. ?att a ocel:Attribute ;
      ocel:attType ?attType; ocel:attKey ?attKey; ocel:attValue ?attValue. }}
```

The following query extracts the relations between OCEL events and objects through the property *e-contains-o*:

```
SELECT DISTINCT * WHERE { ?event a ocel:Event ; ocel:e-contains-o ?object }
```

To evaluate these SPARQL queries, OnProm uses Ontop, which translates them to SQL queries over the database. In this way, extracting OCEL event logs boils down to evaluating some automatically generated (normally complex) SQL queries over the database directly. Finally, OnProm just needs to serialize the query results as logs in the XML or JSON format according to OCEL. Figure 6(a) shows a fragment in XML-OCEL and Figure 6(b) shows its visualization.



(a) OCEL XML serialization.

(b) OCEL graph.

Fig. 6: A fragment of the extracted OCEL log from the Dolibarr ERP system.

4 Experiments

We have conducted an evaluation of OnProm based on the scenario of Dolibarr. The experiments have been carried out using a machine with an Intel Core i7 2.0 GHz processor, 16 GB of RAM, Dolibarr v14, and MySQL v8. In order to test the scalability, we have generated 8 database instances of difference sizes from 2K to 1M. The size of a database is the number of rows.

Table 2: Extraction details of OCEL log elements.

Size	Attributes	Objects	Events	Relations	XML size (KB)
2K	751	751	1000	1227	419
10K	3750	3750	5000	6225	2094
50K	18749	18749	24999	31222	10531
100K	37448	37448	50000	62470	21093
250K	93753	93789	125000	156218	52986
500K	187502	187538	250000	312468	106112
750K	281250	281286	374999	468712	159234
1M	374999	374999	499997	624943	212682

Performance evaluation. The running times are reported in Figure 7. First, we notice that our approach scales well. The overall running time scales linearly with respect to the size of the database. In the biggest dataset of 1M rows, it takes less than 12 minutes to extract the event log. We also computed the divi-

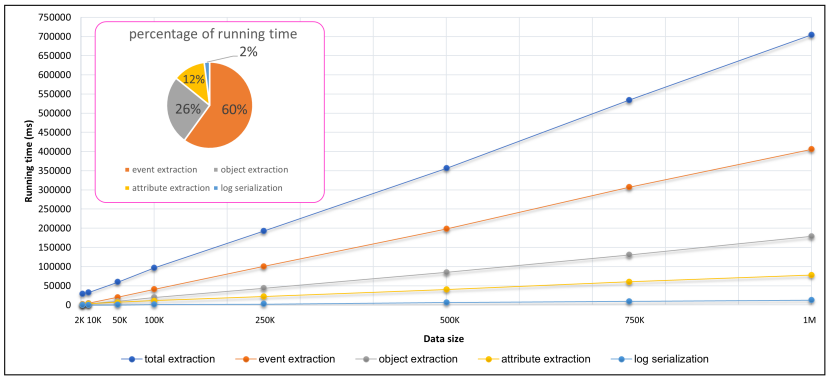


Fig. 7: Running times of the experiment.

sion of the running time over the subtasks of log extraction. The upper left corner of Figure 7 shows the proportion of the running time for each OCEL element. We observe that most of the time (98%) has been spent on the event, object, and attribute extraction, whose main tasks are to evaluate SPARQL queries and create corresponding Java objects in memory. The time for log serialization is almost negligible (2%). We note that since extracting these logs corresponds to evaluating the same SQL queries over databases of different sizes, it is actually not surprising to observe this linear behavior when the database tables are properly indexed.

We also report in Table 2 the number of OCEL elements extracted for each database size. At the size of 1M, OnProm extracts an OCEL event log with 374999 objects, 499997 events, 374999 attributes and 624943 relations, and it takes 207 MB to serialize the whole log in XML-OCEL.

Conformance test. The OCEL standard comes also with a Python library and one of the main functionalities is the validation of JSON-OCEL and XML-OCEL.

The library reports that the log obtained by our method is compliant with the OCEL standard.

5 Conclusions

In this work, we have presented how to extract OCEL logs using the revised version of the OnProm framework. OnProm uses an annotation-based interface for users to specify the relationship between a domain ontology and an event ontology. Then OnProm leverages the VKG system Ontop to expose the underlying sources as a Knowledge Graph using the vocabulary from the OCEL event ontology. Thus, extracting OCEL logs is reduced to evaluating a fixed set of SPARQL queries. Our experiments confirmed that the extraction is efficient and that the extracted logs are compliant with the standard. OnProm provides a flexible framework for users to choose XES or OCEL according to their needs. In the non-many-to-many business, the results are similar, but OCEL has higher extraction efficiency because it does not need to manage events in one case. In modeling many-to-many relations, OCEL has greater advantages because it is actually a graph structure.

There are several directions for future work. First of all, we would like to carry out a user-study to let more users try out our toolkit, and confirm that it is indeed easy-to-use. We are also interested in extracting logs from other sources beyond relational databases, e.g., from graph databases. Finally, the modularity of the approach makes it relatively straightforward to support other standards, and we will study this possibility.

Acknowledgements This research has been supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Italian PRIN project HOPE, by the EU H2020 project INODE (grant n. 863410), by the Free University of Bozen-Bolzano through the projects GeoVKG (CRC 2019) and SMART-APP (ID 2020), and by the Norwegian Research Council via the SIRIUS Centre for Research Based Innovation (grant n. 237898).

References

1. van der Aalst, W.M.P.: Process Mining – Data Science in Action. Springer, 2nd edn. (2016)
2. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: Proc. of the 17th Int. Conf. on Software Engineering and Formal Methods (SEFM). LNCS, vol. 11724, pp. 3–25. Springer (2019). https://doi.org/10.1007/978-3-030-30446-1_1
3. Berti, A., Park, G., Rafiei, M., van der Aalst, W.M.P.: An event data extraction approach from SAP ERP for process mining. CoRR Technical Report arXiv:2110.03467, arXiv.org e-Print archive (2021), https://arxiv.org/abs/2110.03467
4. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A.: OBDA for log extraction in process mining. In: Reasoning Web: Semantic Interoperability on the Web – 13th Int. Summer School Tutorial Lectures (RW), LNCS, vol. 10370, pp. 292–345. Springer (2017). https://doi.org/10.1007/978-3-319-61033-7_9

5. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A.: The onprom toolchain for extracting business process logs using ontology-based data access. In: Proc. of the BPM Demo Track and BPM Dissertation Award (BPM-D&DA). CEUR Workshop Proceedings, vol. 1920. CEUR-WS.org (2017), http://ceur-ws.org/Vol-1920/BPM_2017_paper_207.pdf
6. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A., van der Aalst, W.M.P.: Conceptual schema transformation in ontology-based data access. In: Proc. of the 21st Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW). LNCS, vol. 11313, pp. 50–67. Springer (2018). https://doi.org/10.1007/978-3-030-03667-6_4
7. Calvanese, D., Kalayci, T.E., Montali, M., Tinella, S.: Ontology-based data access for extracting event logs from legacy data: The onprom tool and methodology. In: Proc. of the 20th Int. Conf. on Business Information Systems (BIS). LNBIP, vol. 288, pp. 220–236. Springer (2017). https://doi.org/10.1007/978-3-319-59336-416
8. Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P.: Ontology-driven extraction of event logs from relational databases. In: Proc. of the 11th Int. Workshop on Business Process Intelligence (BPI). LNBIP, vol. 256, pp. 140–153. Springer (2016). https://doi.org/10.1007/978-3-319-42887-1_12
9. Dolibarr Open Source ERP CRM: Web suite for business. https://www.dolibarr.org/ (2021)
10. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: A standard for object-centric event logs. In: ADBIS 2021 Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS. CCIS, vol. 1450, pp. 169–175. Springer (2021)
11. IEEE Computer Society: 1849-2016 - IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams (2016). https://doi.org/10.1109/IEEESTD.2016.7740858
12. Li, G., López de Murillas, E.G., de Carvalho, R.M., van der Aalst, W.M.P.: Extracting object-centric event logs to support process mining on databases. In: Mendling, J., Mouratidis, H. (eds.) Proc. of CAiSE Forum. LNBIP, vol. 317, pp. 182–199. Springer (2018). https://doi.org/10.1007/978-3-319-92901-9_16
13. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), http://www.w3.org/TR/owl2-profiles/
14. de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.P.: Connecting databases with process mining: A meta model and toolset. Software and System Modeling **18**(2), 1209–1247 (2019). https://doi.org/10.1007/s10270-018-0664-7
15. Shankara Narayana, M.B., Khalifa, H., van der Aalst, W.M.P.: JXES: JSON support for the XES event log standard. CoRR Technical Report arXiv:2009.06363, arXiv.org e-Print archive (2020), https://arxiv.org/abs/2009.06363
16. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Information Systems Evolution: Selected Extended Papers of CAiSE Forum 2010. LNBIP, vol. 72, pp. 60–75. Springer (2010). https://doi.org/10.1007/978-3-642-17722-4_5
17. Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual Knowledge Graphs: An overview of systems and use cases. Data Intelligence **1**(3), 201–223 (2019). https://doi.org/10.1162/dint_a_00011
18. Xiao, G., Lanti, D., Kontchakov, R., Komla-Ebri, S., Güzel-Kalayci, E., Ding, L., Corman, J., Cogrel, B., Calvanese, D., Botoeva, E.: The virtual knowledge graph system Ontop. In: Proc. of the 19th Int. Semantic Web Conf. (ISWC). LNCS, vol. 12507, pp. 259–277. Springer (2020). https://doi.org/10.1007/978-3-030-62466-8_17