On the Way to Temporal OBDA Systems

Diego Calvanese[1,2], Cem Okulmus[2], Magdalena Ortiz[2] and Mantas Šimkus[2]

[1]Free University of Bozen-Bolzano, Italy
[2]Umeå University, Sweden

Abstract
Extending the OBDA approach – where multiple data sources are exposed to users via a unified conceptual schema based on description logics – to also cover temporal reasoning has been a long standing goal, with many proposals over the last decades. To the best of our knowledge, these have yet to yield results in the form of systems or prototypes. As part of our ongoing work towards practical applicability, we identify here a number of key problems, which we believe have not been addressed suitably by previous works. Among these is the ability to deal with heterogeneous representations of time, the ability to deal with temporal inconsistencies, either due to missing value samples or conflicting values for a given time point and finally we also seek a suitable query language, where we in particular want compositionality – the ability to use the output of queries to form new temporal views on the data. We present here our initial ideas on how to meet these challenges.

Keywords
Ontology-based data access, temporal database, description logic

1. Introduction

Ontology-based data access (OBDA) describes the method of enriching relational databases with semantical reasoning tools developed in the area of Description Logics (DLs). Specifically OBDA allows one to create mappings from various data sources to an ontology, and extend the data via concept and role inclusions, and thus create a “virtual knowledge graph” (VKG) over which queries can be answered. This VKG need not be materialised, as the query can be rewritten to incorporate the richer semantics from the ontology, and this rewritten query can then be run on existing commercial RDBMs. At this point, OBDA is increasingly used in practice, with both open-source and proprietary systems available.

While temporal databases have been the focus of research for a long time [1], there has been a wider adoption in the industry in recent years. We note the introduction of many commercial systems with a specific focus on temporal data, such as InfluxDB[1], Prometheus[2], TimescaleDB[3],

AMW'23: 15th Alberto Mendelzon International Workshop on Foundations of Data Management, May 22–26, 2023, Santiago de Chile, CL

calvanese@inf.unibz.it (D. Calvanese); okulmus@cs.umu.se (C. Okulmus); magortiz@cs.umu.se (M. Ortiz); simkus@cs.umu.se (M. Šimkus)
0000-0001-5174-9693 (D. Calvanese); 0000-0002-7742-0439 (C. Okulmus); 0000-0002-2344-9658 (M. Ortiz); 0000-0003-0632-0294 (M. Šimkus)

[1]www.influxdata.com
[2]prometheus.io
[3]www.timescale.com

and many others. In addition to the use of temporal data, the literature shows clearly that there is a strong interest in the industry to have query languages that capture complex temporal events in a succinct and intuitive way [2].

There have been decades of work on temporal data and ontologies, both for the more fundamental question of understanding the complexity of temporal DLs [3, 4, 5] and even leading to very promising proposals to extend existing OBDA systems with temporal reasoning [2]. Unfortunately, none of these works or proposals have yet led to prototypes of OBDA systems with rich temporal reasoning, let alone complete systems. Unsatisfied with this state of affairs, we want to identify key challenges that we believe have yet to be addressed or met by the research on temporal OBDA, and we are convinced that our work in overcoming such challenges will lead to working prototypes and, hopefully, ultimately pave the way toward practical systems.

We will highlight in this paper the following challenges that we have identified, and present our initial ideas on how to tackle them.

- Finding ways of dealing with heterogeneous temporal data representations uniformly.
- Exploring the possibility of temporal inconsistencies arising both in the input data and as the result of complex queries, and finding solutions that still enable reliable query answering.
- Finding a composable temporal query language, with suitable complexity in the form of, ideally, FO rewritability. We note that expressivity of this query language can be extended via a complex ontology language, to be used by experts, in order to allow users to refer to complex temporal events without the need to understand complicated temporal logics.

In the remainder of the paper we will give more insights into these challenges, and present on-going work on how we are planning to meet them.

Related Work. Bridging temporal reasoning and DLs has been an area of focus for many years [3, 4, 5, 6, 7]. These works focused on initial exploration of the complexity landscape, and in particular the boundaries of decidability. Understandably, mappings to real data sources are not explored, and most papers assume time to be simply represented by the integers and an ordering on them, for example $\langle \mathbb{Z}, < \rangle$, either purely point-based or including intervals too. The work on finding suitable ontology languages, which can encode complex temporal events, and allowing their use in much simpler user queries has led to promising results, such as the work of Kontchakov et al. [8], which features a fragment of the interval logic $\mathcal{HS}$ in the form of an extension of Datalog. The proposal by Kalaycı et al. [2] is very close to what we aim to ultimately realise with our work. Their proposal supports complex temporal events at the ontology level and an extension of SPARQL to connect validity periods to facts.

2. Challenges for Practical Temporal OBDA

Here we highlight the key challenges that we still see as unaddressed in the existing literature and which any practical ontology-based system that allows for rich temporal reasoning on real-world temporal databases must meet.

Heterogeneous temporal data representations. One can identify in the literature different views at the data level, both *point-based* (often also called *time-series*) and *interval-based relations.* When looking at schemas involving temporal data, it becomes apparent that even the same database will combine time-series and interval-based views. As such, one needs to identify ways to jointly represent both and have ways to safely translate one representation into another. By "safely" we refer to the ability to identify cases of temporal inconsistency, discussed next.

Inconsistency and temporal data. In temporal databases, where facts are enriched via validity periods, we define *temporal consistency* to refer to the fact that we have a consistent assignment of values in the domain to non-temporal attributes for any time-instant that the temporal data is defined over. We believe that the case of temporal inconsistency will occur quite often in practice, and systems will need robust and transparent ways to manage it. Furthermore, we believe that this differs from the case of inconsistency in the non-temporal case. After all, temporal inconsistency only refers to cases where for a given time-point we have too many choices on values for non-temporal attributes (ambiguity) or none at all (gaps in the data). Temporal inconsistency due to ambiguity can be introduced naturally just by allowing general interval-based temporal relations. Managing temporal inconsistency becomes even more crucial when one also considers the third challenge, namely an expressive temporal query language, with the ability to use queries within other queries. With the ability to create new temporal relations (or views) comes the possibility that these new relations themselves could be temporally inconsistent. Thus, dealing with inconsistency becomes a necessity. In addition to inconsistency due to ambiguity, another issue is "gaps" in the temporal data, due to a low temporal resolution, for example. This too will need to be addressed in settings where queries are expected to return useful answers for any time point, regardless of the temporal resolution of any specific data source.

An expressive, composable temporal query language. Just as with standard relational databases and SQL, we would expect temporal query languages to be able to produce new temporal relations, either point-based or interval-based, out of existing temporal data. In addition, we believe it is necessary to be able to express predicates between time-intervals, as detecting complex patterns on time is necessary for the kind of event detection that is of practical interest in industry. An option that we also need to consider in the OBDA setting is the distinction between ontology and query level. As complex and expressive languages on time might be hard for non-expert users to master, one can delegate this task to the ontology engineer via a complex ontology language, which would then introduce new facts to signify temporal events. Users could then make use of the complex temporal machinery without the need to define it themselves.

3. Discussion on Ongoing Work & Outlook

We present for each of the challenges a proposal or give some further details.

For the data model, our current idea is to extend the relational setting and explicitly support both time-series relations and interval-based relations. We use $attr(R)$ and $pk(R)$ to respectively denote the set of all attributes and the primary key attributes of a relation R. We use

Table 1
Temporal relation “Project”, and the result of a temporal rule applied to it.

Project			$extend(\mathbf{time}_{ext}, budget)$	
time	*budget*		**time**	*budget*
(0, 15)	150		(0, 25)	150
(16, 30)	300		(16, 40)	300

capital letters to identify attributes in the schema and lowercase letters for values inside a tuple. We assume that the type **time** is realised as time-stamps.

Definition 1 (Time-series Relation). A *time-series relation* R_{TS} is a relation that has the following property: there must exist exactly one attribute $T \in attr(R_{\mathrm{TS}})$ of type **time** such that $T \in pk(R_{\mathrm{TS}})$.

Definition 2 (Time-interval Relation). A *time-interval relation* R_{I} is a relation that satisfies the following properties:

- There are $T_1, T_2 \in attr(R_{\mathrm{I}})$ of type **time** such that $T_1, T_2 \in pk(R_{\mathrm{I}})$. We assume w.l.o.g. that T_1, T_2 are the first two attributes of R_{I}.
- For every tuple $(t_1, t_2, x_3, \ldots, x_n) \in R_{\mathrm{I}}$, we have that $t_1 \leq t_2$.

With these two representations of time, we already get a number of issues that one needs to address in any working implementation of temporal OBDA systems. The first issue is that of clearly defining when one can safely transform one representation of time into the other one. A second issue (or rather feature we want to have) is the ability to define for a data source methods of making the data denser, for example by means of interpolation on numerical data points to extend the temporal relation.

For the problem of dealing with temporal inconsistency due to ambiguity, we only give here a simple example showing how inconsistency might be introduced even by the ontology language or at the query level, in addition to the possibility of being already present in the temporal database itself. For the purpose of this example, we pick the ontology language proposed by Brandt et al [6], Datalog for sensor log data, or short *DslD*. We omit a detailed introduction of it here, and refer interested readers to the original paper. They propose a number of temporal operators in DslD, including ones that can manipulate intervals, such as $lshift_x$ and $rshift_x$, which extend the left (resp., right) boundary of the interval by x units. We note that DslD is based on a metric view of time with a fixed time unit, such as seconds. Brandt et al. show the need for such operators to formulate rules to detect complex temporal events, informed by the needs of industrial partners. However, the expressive power of DslD introduces the possibility of temporal inconsistency arising from the execution of rules, as demonstrated in the following example.

Example 1. Let us assume that our schema contains a relation “Project”, which indicates the available budget for our project at a given time interval. We further assume we are given a concrete database with the relation “Project” containing the tuples as shown in Table 1. Furthermore, consider the following rule in DslD:

$$extend(\mathbf{time}_{ext}, budget) \leftarrow \mathbf{time}_{ext} \text{ is } rshift_{10}(\mathbf{time}), \text{Project}(\mathbf{time}, budget).$$

This rule would simply extend every existing interval period, while retaining the budget value for that interval. In Table 1, we also show the tuples obtained after applying this rule. We can see that the resulting relation introduces ambiguity in the form of overlaps between time intervals of tuples that have different values for the non-temporal attributes.

Outlook. We plan to continue tackling the challenges we sketched out in this short paper, and ideally realise a first prototype implementation once we have a clearer picture of temporal ontology-based data access. The design of a suitable combination of query language and ontology language, while retaining FO reducibility, is in particular a crucial goal. It requires a balance in order to ensure accessibility for non-expert users at the query level, while also maintaining high expressivity overall.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. It was also partially supported by the Austrian Science Fund (FWF) projects P30360 and P30873, by the Vienna Business Agency's project CoRec, by the Italian Basic Research (PRIN) project HOPE, and by the Province of Bolzano through the project D2G2.

References

[1] R. T. Snodgrass, I. Ahn, Temporal databases, Computer 19 (1986) 35–42.
[2] E. G. Kalaycı, S. Brandt, D. Calvanese, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Ontology-based access to temporal data with Ontop: A framework proposal, Int. J. Appl. Math. Comput. Sci. 29 (2019) 17–30.
[3] A. Artale, R. Kontchakov, V. Ryzhikov, M. Zakharyaschev, Tractable interval temporal propositional and description logics, in: Proc. AAAI 2015, 2015, pp. 1417–1423.
[4] V. Gutiérrez-Basulto, J. C. Jung, R. Kontchakov, Temporalized EL ontologies for accessing temporal data: Complexity of atomic queries, in: Proc. IJCAI 2016, 2016, pp. 1102–1108.
[5] S. Brandt, E. G. Kalaycı, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Querying log data with Metric Temporal Logic, J. Artif. Intell. Res. 62 (2018) 829–877.
[6] S. Brandt, D. Calvanese, E. G. Kalaycı, R. Kontchakov, B. Mörzinger, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Two-dimensional rule language for querying sensor log data: A framework and use cases, in: Proc. TIME 2019, volume 147 of *LIPIcs*, 2019, pp. 7:1–7:15.
[7] S. Klarman, T. Meyer, Querying temporal databases via OWL 2 QL, in: Proc. RR 2014, volume 8741 of *LNCS*, 2014, pp. 92–107.
[8] R. Kontchakov, L. Pandolfo, L. Pulina, V. Ryzhikov, M. Zakharyaschev, Temporal and spatial OBDA with many-dimensional Halpern-Shoham logic, in: S. Kambhampati (Ed.), Proc. IJCAI 2016, 2016, pp. 1160–1166.