


Models and Query Languages for Temporal Property Graph Databases

Valeria Soliani^{1,2}(✉) 

¹ Hasselt University, Hasselt, Belgium
`valeria.soliani@uhasselt.be`

² Instituto Tecnológico de Buenos Aires, Buenos Aires, Argentina
`vsoliani@itba.edu.ar`

Abstract. Although property graphs are increasingly being studied by the research community, most authors do not consider the evolution of such graphs over time. However, this is needed to capture a wide range of real-world situations, where changes normally occur. In this work, we propose a temporal model and a high level query language for property graphs and analyse the real-world cases where they can be useful, with focus on transportation networks (like road and river networks) equipped with sensors that measure different variables over time. Many kinds of interesting paths arise in this scenario. To efficiently compute these paths, also path indexing techniques must be studied.

Keywords: Property graphs · Temporal graphs · Sensor networks

1 Problem Statement and Motivation

Property graphs are graphs whose nodes and edges are annotated with (property, value) pairs [3]. They are widely used for modeling and analyzing different kinds of networks. The property graph data model underlies most graph databases in the marketplace.¹ Typically, the graphs used in practice do not change over time. However, in real-world problems, time is present in most applications and graphs are not the exception. Many changes may occur in a property graph as the world they represent evolves over time: edges, nodes and properties can be added and/or deleted, property values can be updated, to mention the most relevant ones. Social networks are clear examples of this statement: If u and v are vertices modeling persons, and an edge represents a relationship between u and v telling that u follows v or u isFriendOf v , these relationships may change over time or even the persons may unregister from the network. Accounting for these changes would allow queries like “Who were friends of Mary while she was working at Hasselt University”, that otherwise could not be answered. Another example are graphs representing road networks, where vertices model locations and edges represent roads or highways that exist at different periods of time and

¹ For example, <http://www.neo4j.com>, <http://janusgraph.org/>.

whose properties (like road condition) also vary over time. This allows queries like “Compute the time that we saved going from Buenos Aires to Pinamar after the construction of Highway Number 11.”

Our work builds on the hypothesis that keeping the history of changes in graphs is relevant in many interesting real-world applications and that this has been largely overlooked in property graph database modeling. We study how temporal databases concepts can be applied to graph databases in order to model, store, and query temporal property graphs.

In Sect. 2, we review related work. Section 3 presents our approach to the problem and the main results obtained so far. Section 4 discusses ongoing work and open problems. We conclude in Sect. 5.

2 Related Work

Literature on temporal graphs is mostly oriented to address certain path problems in homogeneous graphs (graphs whose nodes are all of the same kind), not tackling property graphs. This is the case of [13, 14], where only edges are timestamped with the initial validity time of the relationship and the duration of the relationship, and there is only one kind of relationship in the graph. Wu et al. [13] study temporal paths and introduce the notions of earliest-arrival path, latest-departure path, fastest path, and shortest path. Along the same lines, Chronograph [5] is a temporal model for graphs that enables temporal traversals such as: temporal breadth-first search, temporal depth-first search, and temporal single source shortest path.

A first approach to the problem of temporal property graphs modeling was presented by Campos et al. [6]. Over this work we build the model we introduce in Sect. 3. Further, the notion of Continuous path used in our work was initially introduced in [12], where a model and index for temporal XML documents was proposed. Pokorny et al. [11] index graph patterns in Neo4j, using a structure stored in the same database as the graph, an approach we follow in our work for indexing temporal paths. Another approach we build on is the temporal database index proposed by Elmasri et al. [8] to process temporal selections and temporal join operations. In [10], an index structure is designed for temporal attributes with various frequency and rates of changes.

3 Our Approach and First Results

The first result of our work is a temporal graph data model, called *TGraph*, that accounts for changes in nodes and relationships in property graphs [7]. Together with this model, we proposed and implemented a high-level query language, T-GQL, that not only considers temporal operators to query a TGraph, but also deals with temporal path structures, the actual first-class citizens in our model. We introduce the model and query language next.

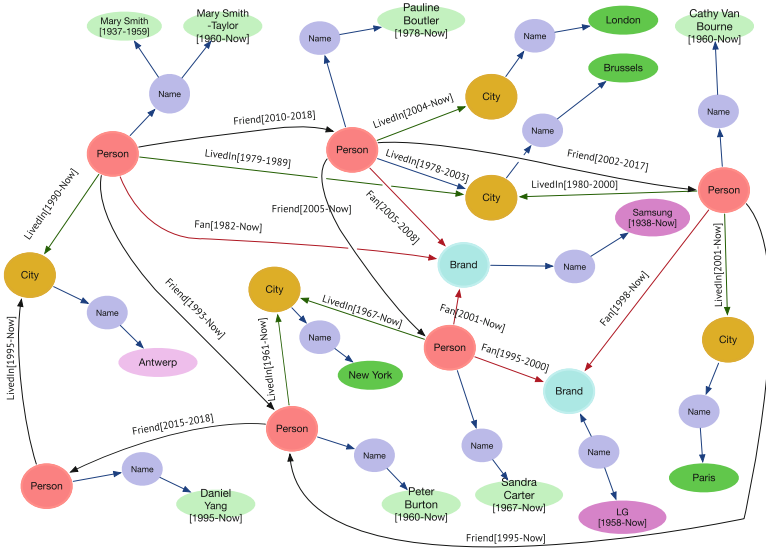


Fig. 1. A temporal property graph.

Temporal Model. We mentioned that in property graphs, nodes and edges are labeled with a sequence of (property,value) pairs. All of them can evolve over time. Thus, to keep the history of the graph we need a data model that can deal with all of these kinds of changes. In our model, entities are represented as *Object* nodes. These nodes may have attributes that may be either static or which can change over time. The former are represented as classic properties of the Object node. The latter are represented as a different kind of node, called *Attribute* node, connected to the Object node. For example, Object nodes may represent a person as an entity, a static attribute of such person may be her date of birth and a temporal attribute may be her name, which can change when a person gets married. The values that this attribute can take are represented as *Value* nodes connected to the corresponding Attribute node. Figure 1 depicts an example of the TGraph model representing a social network. There are three kinds of Object nodes: **Person**, **Brand** and **City**. There are also three types of temporal relationships: **LivesIn**, **Fan**, and **Friend**. The first one is labeled with the periods when someone lived somewhere, the second one with the periods when a person was fan of a brand, and the last one, with the period when two people were friends, for example, Mary Smith-Taylor has been a friend of Peter Burton since 1993. The temporal Attribute node **Name** represents the name associated with a **Person** node. We see for example that “Mary Smith” became “Mary Smith-Taylor” in 1960. For clarity, if a node is valid throughout the complete history of the graph, the temporal labels are omitted.

Nodes and edges in TGraph must satisfy a set of temporal constraints that state how the nodes in the graph must be connected. An Object node can only be

connected to an Attribute node or to another Object node, Attribute nodes can only be connected to non-Attribute nodes and Value nodes can only be connected to Attribute nodes. Cardinality constraints state that Attribute nodes must be connected by exactly one edge to an Object node, and Value nodes must only be connected to one Attribute node through one edge. Also, for any pair of edges with the same name between the same pair of nodes, their temporal intervals must have empty intersection. Value nodes connected to the same Attribute node must have non-overlapping intervals.

Temporal Paths. Path computation is a key problem in graph databases. In temporal (property) graphs, this problem gets even more complex since time comes into play. Therefore, as mentioned above, temporal paths are first-class citizens of our model. We initially defined three kinds of temporal paths [7], based on their applicability to real-world problems: Continuous paths, Pairwise Continuous paths and Consecutive paths.

A *Continuous path (CP)* is a path valid continuously during a certain interval. That is, given a consecutive sequence of edges, there is a continuous path over the intersection of all of their intervals. For example, in a social network we may be interested in finding out chains of people that were friends during the same period. In many cases, a weaker condition over temporal paths suffices. The user may be interested in paths such that there is an intersection in the intervals of consecutive edges pairwise. These paths are called *Pairwise Continuous paths (PCP)*. *Consecutive paths (CSP)*, are useful for scheduling. In these cases, we require that consecutive edges do not overlap, for example, to leave a certain time for a train or flight connection. Thus, CSPs are sequences of edges such that the pairwise intersection between consecutive edges is empty. Different kinds of Consecutive paths can be defined, according to the use that is given to them. For example, in scheduling problems, the *earliest-arrival path (EAP)* is the path that can be completed in a given interval such that the ending time of the path is minimum; the *latest-departure path (LDP)* is the path that can be completed in a given interval such that the starting time of the path is maximum; the *fastest path (FTP)* is the path that can be completed in a given interval such that its duration is minimum; finally, the *shortest path (STP)* is the path that can be completed in a given interval such that its number of edges is minimal.

Query Language. TGraph comes equipped with a high-level SQL-like query language called T-GQL. The T-GQL language has a mixed flavour between SQL and Cypher [9], Neo4j’s high-level query language. The syntax of the language has the typical SELECT-MATCH-WHERE form. The SELECT clause performs a selection over variables defined in the MATCH clause. The MATCH clause may contain one or more path patterns and function calls. The result of the query is always a temporal graph (analogous to relational temporal databases theory), although the query may not mention temporal attributes. This can be modified by the **SNAPSHOT** operator, which allows retrieving the state of the graph at a certain point in time. T-GQL supports the three path semantics above, namely Continuous, Pairwise Continuous, and Consecutive paths,

implemented as functions included in a library of Neo4j plugins. Consider the query “Compute the friends of the friends of each person and the period when the relationship occurred through all the path.” For example, in Fig. 1, Pauline Boutler was a friend of Cathy Van Bourne between 2002 and 2017. Also, Mary Smith-Taylor was a friend of Pauline between 2010 and 2018. Thus, the path $(MarySmith - Taylor \xrightarrow{\text{Friend}} Pauline \xrightarrow{\text{Friend}} Cathy, [2010, 2017])$ will be in the answer since the whole path was valid in this interval. The query reads in T-GQL (cPath computes the CPs of length two for every node):

```
SELECT path
MATCH (n:Person), path = cPath((n)-[:Friend*2]->(:Person))
```

Analogously, PCPs can be also computed using the `pairCPath` function, and four functions are supported for CSPs: `fastestPath`, `earliestPath`, `shortestPath`, and `latestDeparturePath`.

The intermediate results of a query can be filtered by an interval I , provided by the user, that filters out the paths whose interval does not intersect I . The temporal granularity of the starting and the ending instants of the interval must be the same. Also, the `BETWEEN` operator receives a temporal interval and performs a temporal filter. The `WHEN` clause has the form `MATCH-WHERE-WHEN` and can reference variables in an outer query. This clause is used to compute events occurring during concurrent intervals.

4 Current Work and Open Problems

We are currently working on different topics that extend the work presented in the previous section. We comment on this work next.

Indexing Continuous Paths. Computing temporal paths over the TGraph model, particularly CPs, turns out to be costly. We studied how to improve the computation of CPs by indexing them. For this, two index structures are proposed and implemented, one that indexes all the paths and another one that indexes all paths of length two. In the latter case, computing the paths of length higher than two requires additional processing. We implemented additional commands in T-GQL to create and make use of indices. We also consider reducing the search space by limiting the time window to consider the one in which queries will most likely fit. Although we have already implemented this proposal, this is a fertile field to work in and improve current results.

Temporal Modeling of Sensor Networks. The model in the previous section can represent graphs whose nodes and edges change over time, like in the case of social networks. There are other kinds of interesting networks in real-world scenarios where the model can be applied. This is the case of transportation networks, like roads or river networks, which differ from social networks in that there is an element that flows through the network (e.g., cars, water). These networks are rather *stable*, in the sense that changes occur occasionally. For example, the

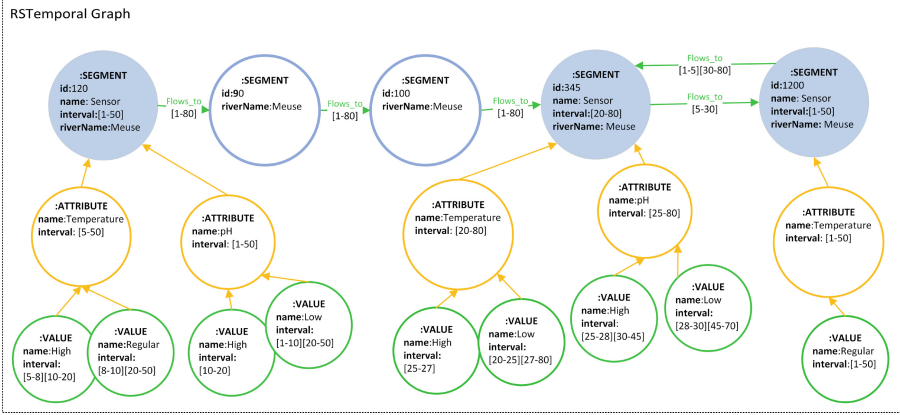


Fig. 2. A temporal graph for a river network equipped with sensors.

direction of the water flow in a river may change due to a flood or a branch may disappear during long dry weather periods. More interesting is the case where sensors are attached to segments in transportation networks, producing time-series data. These are called *sensor-equipped transportation networks* [1], sensor networks for short. The data captured by sensors can be used in various application areas, like traffic control and river monitoring. For example, the Internet of Water project of the Flemish government in Belgium will deploy a network with 2,500 wireless water quality sensors. Bollen et al. [4] proposed a formal model and a calculus to query sensor networks, where the network topology and the time-series data are stored in a graph database for querying and analysis. However, this model does not account for changes over the network. Thus, we propose to model sensor networks using the TGraph model, where Object nodes represent network segments, Attribute nodes represent the temporal properties of the segments, and Value nodes represent the time-series data captured by the sensors.

Modeling sensor networks requires extending TGraph in many ways and opens up a wide variety of paths that are worth studying using Allen's Algebra [2]. We call this model *SN-TGraph*. In this model we distinguish Object nodes that hold a sensor from the ones which do not, and call them *Sensor* and *Segment* nodes, respectively. Also, a list of time intervals indicates the periods of time where a segment had a working sensor on it. Properties that do not change across time are represented as usual in property graphs. Figure 2 shows a scheme of the SN-TGraph model for a river network. Shaded nodes are Sensor nodes. All nodes are of type `:Segment`, and sensor nodes have the value `Sensor` for the property `name`. There are two kinds of Attribute nodes, for representing series of temperature and pH values. Also, in the Value nodes there is an ordered sequence of time intervals for each value of the variables.

The temporal paths presented in Sect. 3 are more involved in SN-TGraphs, since paths must now be defined based on a function over the sensors measure-

ments. This way, CPs are now defined as paths where the *value* of some function is the same throughout the path during a certain interval. At this time, we only consider functions over categorical rather than continuous variables. This is the case, for example, where there is a sequence of consecutive sensors such that each sensor has measured the value of a variable *Temperature* categorized as *High* throughout a certain interval. We denote this as an SN-CP. PCPs over a sensor network are defined as paths such that for every pair of consecutive sensors there is an overlapping interval where the value of the variable is the same. We denote these paths SN-PCP. In the case of Consecutive paths, there is a sequence of consecutive sensors where the value of the variable is the same and the time intervals do not overlap. We denote these paths as SN-CSP.

We note that the paths above cannot completely capture the flow in a transportation (sensor) network, since an event captured by a sensor (e.g., a traffic jam in a road network or the presence of a pollutant in a river) will probably be detected by a sensor located after the previous one in the sense of the movement. Further, if the sensors were placed close to each other, it is possible that a value measured by one sensor would still be valid while it is valid in the next one. If this condition is fulfilled for each pair of sensors, the path can be considered a PCP, while in case the sensors were placed far apart, it can be a CSP. To combine both situations in one path in which the valid time of a sensor measurement just starts before the next one we introduce the notion of *Flow path (SN-FP)*.

Temporal Relations Between Paths. The difference between the four kinds of paths introduced so far lies in the way in which the temporal interval of each node (or edge) is related to the next one. Temporal relations between every pair of consecutive intervals in those paths can be described in terms of Allen’s Algebra. There are thirteen possible Allen’s relations covered by CPs, PCPs, CSPs, and SN-FPs. Even though not all possible combinations of Allen’s intervals are interesting in real-world situations, we want to study if there are interesting paths that can be identified in addition to the ones already defined. In order to determine if our paths are not covering some important combination, we need to define a path taxonomy, and map paths to real-world cases. We are currently carrying out this study.

5 Conclusion

In this Ph.D. project, we study the evolution of different kinds of property graphs over time. As our first result, we proposed a temporal model (called TGraph) and a high-level query language (called T-GQL) to account for changes on nodes and edges in property graphs. Temporal paths are first-class citizens in this model. Although TGraph is applicable to, for example, social networks, it cannot capture transportation networks (i.e., networks where some element flows from one node to another) equipped with sensors which produce time-series data measuring the evolution of different variables over time. Thus, we proposed SN-TGraph, a model that extends TGraph. In SN-TGraph, new kinds of temporal

paths can be defined in terms of the variables measured by the sensors. We are currently studying the impact and applicability of these paths to real-world cases, like river systems and road networks, two typical cases of transportation networks. In addition to the above, since computing temporal paths is costly, we are studying different ways of indexing such paths.

Acknowledgements. Valeria Soliani was partially supported by Project PICT 2017-1054, from the Argentinian Scientific Agency.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. *IEEE Commun. Mag.* **40**(8), 102–114 (2002)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. *Commun. ACM* **26**(11), 832–843 (1983)
3. Angles, R., Thakkar, H., Tomaszuk, D.: RDF and property graphs interoperability: status and issues. In: Hogan, A., Milo, T. (eds.) *Proceedings of AMW, Asunción, Paraguay, 3–7 June 2019. CEUR Workshop Proceedings*, vol. 2369. CEUR-WS.org (2019)
4. Bollen, E., Hendrix, R., Kuijpers, B., Vaisman, A.A.: Time-series-based queries on stable transportation networks equipped with sensors. *ISPRS Int. J. Geo Inf.* **10**(8), 531 (2021)
5. Byun, J., Woo, S., Kim, D.: Chronograph: enabling temporal graph traversals for efficient information diffusion analysis over time. *IEEE Trans. Knowl. Data Eng.* **32**(3), 424–437 (2020)
6. Campos, A., Mozzino, J., Vaisman, A.A.: Towards temporal graph databases. In: Pichler, R., da Silva, A.S. (eds.) *Proceedings of AMW, Panama City, Panama, 8–10 May 2016. CEUR Workshop Proceedings*, vol. 1644. CEUR-WS.org (2016). <http://ceur-ws.org/Vol-1644/paper40.pdf>
7. Debrouvier, A., Parodi, E., Perazzo, M., Soliani, V., Vaisman, A.: A model and query language for temporal graph databases. *VLDB J.* **30**(5), 825–858 (2021). <https://doi.org/10.1007/s00778-021-00675-4>
8. Elmasri, R., Wu, G.T.J., Kim, Y.: The time index: an access structure for temporal data. In: *Proceedings of VLDB 1990, August 13–16, 1990, Brisbane, Queensland, Australia*, pp. 1–12. Morgan Kaufmann (1990)
9. Francis, N., et al.: Formal semantics of the language cypher. *CoRR* abs/1802.09984 (2018)
10. Kvet, M., Matiasco, K.: Impact of index structures on temporal database performance. In: *EMS 2016, Pisa, Italy, 28–30 November 2016*, pp. 3–9. IEEE (2016)
11. Pokorný, J., Valenta, M., Troup, M.: Indexing patterns in graph databases. In: *DATA 2018, Porto, Portugal, July 26–28, 2018*, pp. 313–321. SciTePress (2018)
12. Rizzolo, F., Vaisman, A.A.: Temporal XML: modeling, indexing, and query processing. *VLDB J.* **17**(5), 1179–1212 (2008)
13. Wu, H., Cheng, J., Huang, S., Ke, Y., Lu, Y., Xu, Y.: Path problems in temporal graphs. *Proc. VLDB Endow.* **7**(9), 721–732 (2014)
14. Wu, H., et al.: Core decomposition in large temporal graphs. In: *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29–November 1 2015*, pp. 649–658 (2015)