Towards Hybrid Graphs: Unifying Property Graphs and Time Series

Mouna Ammar
Leipzig Univ. & ScaDS.AI
Leipzig, Germany
ammar@informatik.uni-leipzig.de

Christopher Rost
Leipzig Univ. & ScaDS.AI
Leipzig, Germany
rost@informatik.uni-leipzig.de

Riccardo Tommasini
INSA Lyon, LIRIS
Lyon, France
riccardo.tommasini@insa-lyon.fr

Shubhangi Agarwal
Lyon 1 Univ., LIRIS
Lyon, France
shubhangi.agarwal@liris.cnrs.fr

Angela Bonifati
Lyon 1 Univ., LIRIS, IUF
Lyon, France
angela.bonifati@univ-lyon1.fr

Petra Selmer
Bloomberg
New York, United States
pselmer@bloomberg.net

Evgeny Kharlamov
Bosch Center for AI
Renningen, Germany
evgeny.kharlamov@de.bosch.com

Erhard Rahm
Leipzig Univ. & ScaDS.AI
Leipzig, Germany
rahm@informatik.uni-leipzig.de

ABSTRACT

Graphs effectively represent structural relationships, while time series capture temporal dynamics, both of which are critical to understanding complex systems, such as asset management, IoT optimization, and micromobility demand predictions. In these contexts, the interplay between evolving entities and their relationships, captured by graphs and large volumes of time-series data, remains challenging to fully exploit, due to the absence of a unified approach. Practitioners are thus forced to treat both data structures as isolated and must create connections with manual effort. Our vision, HyGraph, includes a hybrid data model and operator concept designed to integrate the expressive power of temporal graphs with time-series analysis, providing a holistic approach for complex queries, analytics, and predictive tasks, which are currently unfeasible by working solely on isolated data structures. This vision has the potential to drive significant advancements in both research and practice, addressing limitations associated with isolated data models and fostering new opportunities for interdisciplinary insights.

1 INTRODUCTION

Data interconnection and temporal evolution are emerging as essential modeling aspects in modern data management. Graphs were proved to be a working abstraction to capture the former [67], while data streams [37] and time series [35, 47] effectively describe the latter. However, analysts are currently forced to put manual efforts when both modeling needs appear together [14, 15, 63] because a hybrid solution for this integration *is currently missing*.

Figure 1 (left) illustrates the issue: (streaming) time series are ideal for representing evolving value but lack the semantic richness to represent meaningful relationships [6, 11]. Conversely, graph data models, like labeled property graphs, are already sufficiently expressive to capture structural changes [42, 65], but fail to represent the sequential nature of time-series data, reducing their integration to a simple attribute and/or losing the opportunity to interact with them as objects in the graph. This combination presents research challenges in data management, impacting data mining and AI communities, where attention to

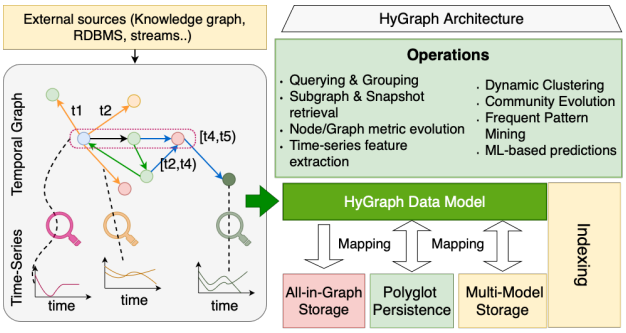

Figure 1: HyGraph's vision: data model and architecture.

evolving connected data is growing [58]. Current trends increasingly explore approaches to enrich time series with structural information [14, 62, 75] or, inversely, to augment entities and relationships within a graph using evolving contextual data [63, 86]. Such investigations are a first step towards a unified data abstraction. Indeed, this is promising since both data structures excel at different purposes; their combination may reveal patterns that would remain undetected when analyzed in isolation.

This paper presents our vision, namely HyGraph, for a unified view of graph and time-series data. Our goal with HyGraph is to propose a research direction for unifying property graphs and time-series data into a cohesive data model that treats both as first-class citizens. This novel approach enables hybrid operations, analyses, and machine-learning opportunities that address questions involving structural and temporal dimensions (as illustrated in Figure 1). By allowing users to interact with a single HyGraph instance, we eliminate the need to switch between different data models or systems through a seamless integration. While we do not claim to address all challenges in a single project, this work provides a foundational framework to inspire further exploration in this domain, leading to three key contributions:

- We introduce use cases that demonstrate the necessity of HyGraph and identify the requirements to address;
- We present preliminary results on HyGraph, including data model and operations, for solving a running example;
- We outline a research roadmap and present a selected dataset to discuss and substantiate our vision.

Outline. Section 2 presents industrial use cases and analyzes the requirements for a solution. In Section 3, a running example from the financial domain is introduced, illustrating the benefits of our

vision. Section 4 elaborates the vision in more detail, while Section 5 presents our preliminary work. Finally, Section 6 outlines the research roadmap, and Section 7 concludes the paper.

2 FROM USE CASES TO REQUIREMENTS

This section presents industrial use cases that combine graphs and time series. By highlighting the challenges such use cases unveil, we elicit some foundational requirements for our vision.

Uncovering **financial fraud** [81, 82] requires navigating connections between assets (e.g., cards, bank accounts) and to classify or aggregate value fluctuations within a time frame. These operations demand graph traversal queries and advanced time-series analysis, often achieved through complex, ad-hoc solutions. Managing financial entities involves capturing the temporal dynamics of both physical and virtual entities, such as companies, equities, funds, and portfolios. The dynamic structure (e.g., entity relationships) and data (e.g., time series) drive many query patterns. For example, a query spanning a long period (e.g., in backtesting) needs to cover a number of stages and milestones for some company C, such as its inception, being privately held, having an IPO event, and going public, being listed on stock exchange(s) with varying membership levels over the years, being acquired by a company D, being sold to another company E, and E (and all its subsidiaries) going bankrupt. All these changes in C's state impact the topology of the graph. Moreover, these stages reflect distinct properties, such as daily stock prices for publicly listed companies, highlighting the interplay between graph topology and time-series data. In this paper, **credit card fraud detection** will serve as our use case, which we will further develop and analyze in Section 3 to show the potential of the HyGraph model.

The Internet of Things (IoT) and smart manufacturing is another application domain containing scenarios with thousands of time series structurally connected [74]. **IoT data analytics** is crucial for cost reduction and optimization [18] but it must consider the devices' physical and logical disposition and structure, how the devices are connected (topology) as well as the environment they are operating in.

In **urban micromobility**, dynamic sensor topologies on vehicles add complexity. Smart bike and scooter providers must predict demand at stations and districts to optimize distribution [29]. Rental station networks, user relationships, and evolving metrics like bike availability and battery levels can significantly enhance prediction accuracy. Our published dataset of temporal property graphs (TPGs) and time-series data enables research on optimizing vehicle distribution [52].

We have elicited the four requirements for the HyGraph vision based on the use cases. (R1) **Expressiveness.** In HyGraph data model, the query primitives, and the analytical operations must preserve the expressiveness of their counterparts for time series and TPGs, i.e., integrating existing graphs or time series in the hybrid model without losing structural or temporal information. (R2) **Consistency.** The HyGraph model must accurately represent the dynamic interplay between structure and time. This includes ensuring chronological integrity in time series [69] and temporal integrity in the graph [65], and enabling users to define and manage alternative logical views over a model instance, e.g., via grouping or sampling. (R3) **Timeliness.** The HyGraph model must be designed for replacing stale data without compromising the structure's integrity, even for high ingestion. Moreover, structural updates must satisfy the velocity requirements of time-sensitive scenarios. (R4) **Scalability.** Additionally, the HyGraph model must handle high-velocity data ingestion, the increasing volume, and the complexity of the hybrid graph time series data. It must enable scalable data processing with minimal overhead as data volume increases.

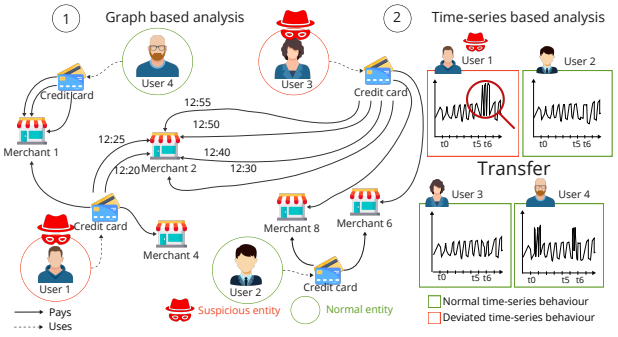

Figure 2: Existing methods to enhance Fraud Detection.

3 RUNNING EXAMPLE

This section elaborates on the financial fraud use case to present a running example. It focuses on *Credit Card Fraud Detection* [22] as it emerged as a crucial area within the financial domain where graph and time-series analysis have already been independently applied. Assume having recorded transactions between users and merchants to identify fraudulent users. We discuss possible solutions using graph models and time-series data models individually (Figure 2), then a combined approach using existing systems, and finally through HyGraph in Section 4. The time series in Figure 2 illustrate the expenditure behavior of the users.

The graph-based way. Existing graph-based approaches adopt pattern matching [17] and machine learning algorithms [59] to detect fraudulent transactions. The former can be a graph pattern query like the one in Listing 1, which finds users with high transaction amounts to at least three merchants nearby within an hour. For our example graph in Figure 2, a user who briefly conducts transactions with multiple merchants might be considered suspicious if such behavior deviates from the usual pattern. The query in Listing 1 would return *User 1* and *User 3* as potentially suspicious, as they meet the specified criteria.

The time-series way. Time-series data is used to analyze transaction patterns over time (e.g., transactions frequency or balance) to detect anomalies [27]. This method involves outlier detection by employing distance-based methods to identify transactions that deviate significantly from a user's typical spending pattern (Listing 2). For instance, User 1 in Figure 2 is marked as suspicious due to several significant peaks of transactions within a short interval $[t_5, t_6)$ which may indicate fraudulent activity. However, this separation often fails to capture the complete picture, as graph analysis might identify suspicious entities without time-contextual transactional behaviors, and time-series analysis might spot irregular spending patterns without knowledge of the correlation to other related entities. Thus, (R2) and (R3) are only partially fulfilled as each data model is treated separately, while (R1) remains unfulfilled.

The combined way. Some single-core and multi-model databases can handle both graph and time-series data but typically treat them as separate silos. For example, MongoDB [54] supports time-series collections and basic graph capabilities. While this allows basic querying, it leads to complexity and performance

```
MATCH (u:User)-[:USES]->(cc:CreditCard)-[t1:TX WHERE t1.
      amount>1000]->(m1:Merchant),
MATCH (u:User)-[t1:TX WHERE t1.amount>1000]->(m1:Merchant)
      , path=(u)-[:TX]->(m2:Merchant)
WITH u, collect(m2) as mrs, relationships(path) as txs
WHERE ALL(tx IN txs WHERE tx.amount > 1000 AND tx <> t1)
  AND ALL(m IN mrs WHERE distance(m.loc, m1.loc) < 1000)
  AND ALL(t IN txs WHERE t.time-t1.time <= duration("1H"))
  AND length(mrs) > 2
  RETURN u.name AS suspiciousUser
```

Listing 1: Fraud detection using graph model only.

```
def detect_fraudulent_transactions(timeseries):
    centroid = np.mean(timeseries[:10])
    distances = np.linalg.norm(timeseries[:10]-centroid)
    threshold = np.max(distances)
    is_fraudulent = any((np.linalg.norm(timeseries[10:])-
        np.mean(timeseries[10:])) > threshold)
    return is_fraudulent
```

Listing 2: Fraud detection using time-series model only.

bottlenecks due to a lack of optimization techniques for advanced analyses like graph traversals or time-series-based computations (R4). Existing solutions either lack a unified language or provide limited ones (e.g., AQL [7]), failing to provide seamless hybrid capabilities and requiring ad-hoc, task-specific solutions that compromise consistency (R2) and maintainability (R3). The simple alternation for a dedicated solution enforces a physical separation of the specified pipeline, limiting the model expressiveness (R1). In contrast, HyGraph unifies data models and analyzes all levels, paving the way for new combined representations that fulfill all requirements.

4 THE HYGRAPH VISION

HyGraph is a transformative approach that unites temporal property graphs (TPGs) and time series into a coherent model without losing any expressiveness of each data structure. HyGraph treats time series as a living part of the graph, contributing to the evolving narrative of connections. Similarly, the graph represents the flow of structural and contextual changes, with each graph element enriched with temporal information and seamlessly integrating time series. The distinction between graph and time-series data dissolves, allowing users to interact with a unified model. The HyGraph model's internal core masks the complexity, eliminating the need to toggle between different data models and enabling more efficient querying and analysis. **HyGraph operations** can analyze and manipulate TPG and time series simultaneously: the existing time-series analyses can be redefined to use the semantic graph data for enhanced results, while operations of the graph domain can benefit from the information captured by the underlying patterns in the time series. Moreover, new operators can be defined that can be executed on a HyGraph instance. For instance, the creation of logical graph patterns from nodes that exhibit similar time-series patterns and conduct frequent pattern mining for community detection.

Figure 3 depicts such intuition at the bottom (marked by ⑩): HyGraph introduces a higher level abstraction over two simpler data models, enabling interactions between them.

Related Work on Graphs and Times Series Management. The need for a unifying view over graphs and time series is emerging in the data management community [84]. Figure 3 shows the state of the art on existing data models and transformations between them.

The top half of Figure 3 shows the static and temporal data models. In the top layer, semantically rich graph data models, labeled graphs (LG) [61] and labeled property graphs (LPG) [6] add properties as key-value pairs to vertices and edges while the data series model, a sequence of values ordered in some manner [5], enable operations on serialized data, e.g., byte streams. The arrows ① and ② represent operations on these data models, e.g., frequent pattern mining, subgraph matching, path queries, etc., while ④ represents data series sampling, filtering, grouping, etc. The middle layer shows the temporal context with TPGs [42, 65], which have equal expressiveness as LPG but extend it with the time dimension to model all changes in the graph over time. The arrow ③ represents operations on TPGs, like snapshot retrieval [45], temporal pattern matching [87], etc., that solely operate on TPGs. The ⑤ represents operations on the time series [21], like classification [20] or subsequence mining [60].

Some approaches already use both data models in their tasks: arrow ⑥ in Figure 3 represents a graph representation of a novel low-dimensionality embedding of time-series subsequences in recent research works [14, 15]. Similarly, a novel approach is proposed in [33], where time series are connected by edges based on their similarity. The FeatTS approach [75] shaping time-series data into a labeled graph form is also represented by arrow ⑥. The transformation of an LPG into a data series is done through simple pattern-matching queries returning property values or aggregates as a series of values (marked by ⑦).

An instance of interaction between LPG and time series (⑧ in Figure 3) consists of LPG augmented with time-series data as properties. Time-series analysis is then used to encode relationships between two time series that can be navigated with the help of pattern matching. Arrow ⑨ in Figure 3 represents operations that use both LPG and time series. While recent research has made significant strides towards integrating time-series data within graph databases satisfying arrows partially (⑧ and ⑨), most existing frameworks [10, 72] treat time series as secondary properties attached to nodes and edges, rather than as first-class citizens. Such a hierarchical approach causes discrepancies where graph components such as nodes and edges are prioritized over time series data. In practice, such models do not allow time series and graph components to interact on an equal footing, potentially limiting the depth of analysis that can be derived.

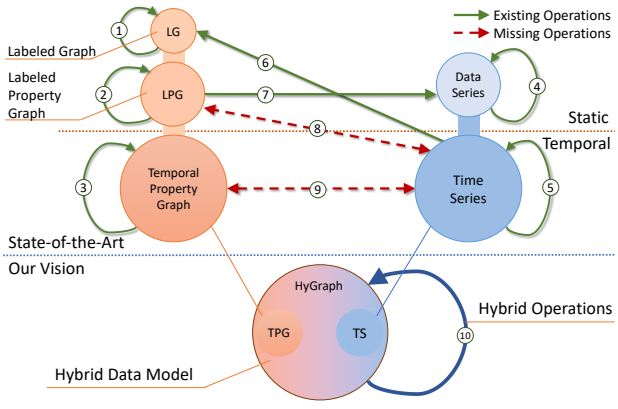

Figure 3: State-of-the-art data models and operations (top), Proposed HyGraph data model and operations (bottom).

Moreover, works on RDF stream processing (RSP) [16, 79] are worth mentioning. RSP aims at extending the Linked Data infrastructure for continuous querying [13]. To this extent, RSP solutions ingest a streaming extension of RDF based on timestamped graphs or triples [16, 78], their relation to HyGraph is limited. In particular, RSP focuses on reactiveness and recency, limiting the temporal aspect to window operators. Instead, HyGraph aims at making time series first-class citizens in the model.

Research on HyGraph is also related to multi-databases [73], which include **multistores** and **polystores**. The former exposes a unified declarative query interface over heterogeneous data models. The latter combines the benefits of the multistores with *polyglot querying* [39], i.e., they expose multiple query interfaces over heterogeneous data models [28]. Regardless of their architecture, multi-databases aim to reduce the amount of jobs required to have a uniform view of heterogeneous data. Although data integration is one of the objectives of HyGraph, our vision goes beyond system federation and integration. HyGraph aims at enabling a whole new family of complex workloads that combine the navigational nature of graph queries and the sequential analysis of time series. It represents a paradigm shift towards a new data model that inherently supports the complexity of modern data, combining the strengths of property graphs and time series.

In the past, researchers have extensively explored hybrid models to enable data processing and analysis techniques that surpass the limitations of individual models. For example, the authors of [51] highlight the usefulness and reusability of relational platforms for scalable linear algebra due to their robustness and inbuilt cost-based optimizations. An efficient processing technique for such hybrid platforms is proposed in [70]. The compilation framework presented in [71] paves the way for enhanced in-database machine learning. Similarly, our vision with HyGraph aims to unify time series and graph databases, enabling advanced analytics that transcend the capabilities of either data model.

5 PRELIMINARY WORK

Basic data model. Various approaches exist to integrate the TPG model with time-series data. One approach enriches time series with graph elements, adding structural connectivity [15], while another embeds time series as graph properties [10], enabling values to vary over time. Our preferred method treats both as first-class citizens, ensuring equal graph and time-series data representation. We propose the HyGraph Model (HGM) to unify LPGs [6], TPGs [65], and time-series data [25], supporting univariate and multivariate time series [1, 46] as 1) vertices, 2) edges, or 3) properties.

Given a set of property keys K, a set of property values $\mathcal{N}$, a set of labels L, a set of tuples Y, and a set of ordered timestamps T. The HyGraph model is a tuple $HG = (V, E, S, TS, \eta, \gamma, \lambda, \phi, \rho, \delta)$, where V is a set of vertices, E is a set of edges, S is a set of logical subgraphs and TS is a set of (multi-variate) time series. To emphasize the equality of graph data and time-series data as first-class citizens of the model, we define *property graph vertices/edges* and *time-series vertices/edges*. Formally, $V = V_{pg} \cup V_{ts}$ with $v_{pg} \in V_{pg}$ is a property graph vertex and $v_{ts} \in V_{ts}$ is a time-series vertex. Similarly, $E = E_{pg} \cup E_{ts}$ with $e_{\text{pg}} \in E_{pg}$ is a property graph edge, and $e_{\text{ts}} \in E_{ts}$ is a time-series edge. The function $\delta : (V_{ts} \cup E_{ts}) \rightarrow TS$ maps each time-series vertex and edge to a multivariate time series in TS. A multi-variate time series $ts \in TS$ is an ordered set of tuples $ts = \{(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n) | n \in \mathbb{N}\}$ with $t \in T$ represents a timestamp and $y \in Y$ represents a tuple of values of the time series with $y = (val_1, val_2, \ldots, val_k)$. Whenever a time series semantically represents an entity or relationship, specialized vertex v_{ts} and edge e_{ts} are used. Such an approach is optimal when the focus is on the evolution of a given entity attribute as it neglects the need to monitor the entity's other properties. The function $\rho : (V_{pg} \cup E_{pg} \cup S) \rightarrow T \times T$ retrieves the timestamps, $\langle t_{\text{start}}, t_{\text{end}} \rangle$, between which an object is valid (t_{end} is initialized to $max(T)$).

We define the set of property values as $\mathcal{N} = \mathcal{N}_\Sigma \cup \mathcal{N}_{\text{TS}}$, with $\mathcal{N}_\Sigma \cap \mathcal{N}_{\text{TS}} = \emptyset$, where $\mathcal{N}_\Sigma = \{\sigma_1, \sigma_2, \ldots\}$ and $\mathcal{N}_{\text{TS}} = \{ts_1, ts_2, \ldots\}$ with $ts_i \in TS$ ($i \in \mathbb{N}$), represent static and time-series property values, respectively. A time series is represented as a property when it is supplementary to the primary entity, providing additional context but not the main focus. The assignment function $\eta : E \rightarrow V \times V$ maps an edge to vertices, while the function $\gamma : S \times T \rightarrow \mathcal{P}(V) \times \mathcal{P}(E)$ is a function that assigns to every subgraph at each point $t \in T$ a subset of vertices V and edges E where $\mathcal{P}(\cdot)$ denotes the power set. Furthermore, the functions $\lambda : V \cup E \cup S \rightarrow \mathcal{P}(L)$ and $\phi : (V_{pg} \cup E_{pg} \cup S) \times K \rightarrow \mathcal{N}$ assign labels to the entities and values to the property keys, respectively.

First HyGraph analysis prototype. The HyGraph analysis package [3, 4] is a Python-based package that implements our data model and provides a range of queries and operations for managing and analyzing hybrid data. It leverages two main libraries, Xarray [88] and NetworkX [57], for in-memory storage and computation, enabling efficient handling of both (multivariate) time-series data and graph structures within the same framework. The predefined operators rank from simple read, write, and update of the HyGraph instance to a set of hybrid operators like hybrid pattern matching, enabling users to query patterns span both temporal and structural dimensions. We allow transformations between graph and time series data to guarantee that both data types are treated as first-class citizens. Starting from graph data, we can generate time series by applying operations that aggregate edges into super-edges, storing edge information in a time series format, and then applying time-series operators. Conversely, we can build a graph on top of time series data, enabling graph operators. This bidirectional capability guarantees that our system treats both data models equally.

First HyGraph storage prototype. We conducted benchmarking experiments using our bike-sharing dataset [52] by comparing Neo4j as a native graph database with TimeTravelDB [68], our polyglot persistence research prototype combining Neo4j with TimescaleDB that uses an extended Cypher-based language for querying. We store the time series in Neo4j as properties of nodes and edges, where each timestamp and its corresponding value

Table 1: Performance benchmarking of Neo4j and TimeTravelDB (TTDB): Mean Response Time (MRS) and Coefficient of Variation (CV).

	Neo4j		TTDB	
Query	MRS (ms)	CV (%)	MRS (ms)	CV (%)
Q1	3.40	20.50	4.33	5.39
Q2	41.47	20.92	7.02	4.08
Q3	56.09	21.28	20.48	5.43
Q4	31109.26	21.41	71.86	50.19
Q5	73814.52	21.37	62.85	41.14
Q6	73446.80	21.61	64.95	26.53
Q7	48299.03	21.29	48.39	38.57
Q8	54494.19	21.19	48.61	7.66

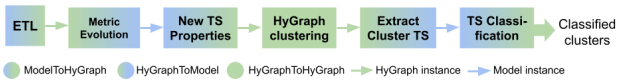

Figure 4: HyGraph pipeline to solve the running example.

are stored as separate properties. This approach significantly increases the number of properties, resulting in high write overhead. We design eight distinct queries to evaluate performance, ranging from straightforward time-range queries to more complex queries involving aggregations of time series values. Table 1 presents the performance metrics for Neo4j and TimeTravelDB (TTDB). The results demonstrate TTDB's significant advantage in terms of query performance, particularly for complex queries. For instance, in Q4, Neo4j exhibits a mean response time of 31,109.26 ms, while TTDB achieves a mean response time of 71.86 ms, representing a speedup of several orders of magnitude. Similarly, for Q5 and Q6, where Neo4j records response times exceeding 73,000 ms, TTDB maintains response times below 100 ms. Even for simpler queries apart from Q1, which is a simple time range query, TTDB outperforms Neo4j, with mean response times as low as 7 ms for Q2 compared to 41.47 ms for Neo4j. Although Neo4j exhibits relatively consistent performance with CV values around 21% across all queries, its significantly higher response times make it unsuitable for efficiently handling time-series data. We also refer to refer the reader to [80] for more experiment details. Moreover, the queries in Neo4j were significantly longer and more complex due to the need to manually handle time series data stored as properties. This shows that storing time series data directly within a graph database can lead to performance bottlenecks. In Figure 1, we highlighted in red this limitation, i.e., the 'All-in-graph Storage' approach; conversely, we show in green the more promising 'Polyglot persistence' approach.

Other works. Our previous work on Gradoop [64, 65] introduces a scalable open-source framework for managing and analyzing temporal graphs. The TPG model provides operators and algorithms; it serves as the basis for our vision and uses its architecture to build a scalable system. Moreover, we explore the evolution of graph metrics over time in TPGs [63]. This work is pertinent to the HyGraph model as it underscores the importance of time-series analysis in understanding graph structures.

Basic operations: Solving the motivation use case. The HyGraph vision suggests solving the credit card fraud example by classifying clusters based on the behaviors of their members as either "ordinary" or "suspicious", as follows.

Users and merchants are represented as PG (property graph) vertices while credit cards are described as TS (time series) vertices, as the focus is only on the evolution of the balance. Edges connecting the user to the credit card are represented as PG edges. In contrast, edges connecting the credit card to the merchant are TS edges, highlighting the dynamic nature of transaction flows. Two credit cards can also be connected to highlight their similarity. The similarity edge between two credit cards is a TS edge because the entity represents a time series encapsulating the time-varying nature of their transactional similarities. Such a method utilizes distinct interfaces to facilitate interaction between those data models. Thus, we identify below three different interfaces into which the operations will fall and create a complete pipeline, as shown in Figure 4. The placeholder <*X*> represents either a TPG or a time-series model.

Table 2: Time Series vs Graphs: Querying, Analysis, and ML. Legend: [Q]uerying, [D]etection, [P]attern [M]ining, [E]mbeddings, [C1] Classification and [C2] Clustering.

	Time Series	Graph
Q1	Subsequence matching [89]	Subgraph matching [11]
Q2	Downsampling [48]	Graph aggregation [90]
Q3	Correlation [55]	Reachability [11]
Q4	Segmentation [26]	Snapshot [45]
D	Anomalies [8]	Communities [34]
PM	Sequence, Motif [32]	Subgraph, Motif [11]
E	Subsequence matching [89]	Vertex/Edge/Path/Graph [19]
C1	Temporal FAT, trends [36]	Labels, edge/vertex feat. [43]
C2	Temporal proximity [31]	Connectivity, Density [40]

The *<X>ToHyGraph* interface allows the integration of graphs and time series into a given or new HyGraph instance. It involves operations such as the addition of vertices, edges, and time series to properties. An example of an operation in this class is transforming temporal graph data with time series into the HyGraph (the first step in Figure 4).

The *HyGraphTo<X>* interface enables data extraction from a HyGraph instance, ensuring compatibility with existing pipelines. Users can extract graph or time series instances in their original formats and apply operations without losing functionality. For example, the *metricEvolution* operator, inspired by [63], computes meta-properties like *node degree* and *community ID* over time stored as time series properties of the corresponding vertices. It shows the duality of the *<X>ToHyGraph* and *HyGraphTo<X>* operations, enriching graph metrics into time-series data and storing them as properties.

Finally, the *HyGraphToHyGraph* interface enables advanced processing of HyGraph instances. For example, the *clustering algorithm* in HyGraph will analyze transactional interactions (graph data) and account balance (time series) to produce enriched clusters with meta-properties, transaction edges, and credit card similarity measures. A temporal evolution of these metrics across clusters can be further used for classification to detect fraudulent clusters and annotate the HyGraph instance, reducing false positives and negatives. This would aid in detecting "*User 3*" as a false positive and "*User 1*" as a "*suspicious entity*" due to its subgraph connections and time series behavior. By enriching transitions between graph and time-series models, HyGraph enables more accurate analyses.

6 RESEARCH ROADMAP

The HyGraph's research roadmap weaves the definition of simple querying and analytics and paves the way to multi-model intelligent data management. Table 2 overviews related technologies for graph and time-series research areas. The table includes querying, analytics, and machine learning. We show how each respective combination leads to new hybrid operators.

Querying HyGraph. Querying involves defining primitives based on graph pattern matching, reachability, and spatiotemporal constructs of property graphs [11, 12, 44]. Unlike previous work that separated queries into static graphs or time-series data [11, 41], HyGraph unifies these approaches to enhance querying efficiency across both domains. (*Q1*) introduces an operator that matches specific temporal patterns with corresponding structural patterns (subgraphs), thus detecting complex hybrid

patterns. ($Q2$) is useful for summarising high-frequency data, or even in streaming [66, 78], without losing context. This operator summarizes and aggregates graph elements and adjusts the frequency of associated time series to a user-defined granularity, improving scalability and performance. ($Q3$) measures the correlation between time-series data of vertices to enhance reachability analysis, aiding in identifying entities with similar temporal patterns. ($Q4$) creates graph snapshots at significant time intervals identified through time series segmentation, allowing a detailed analysis of graph evolution [58].

Analyzing HyGraph. Analytics within HyGraph can be intricate, often necessitating a multi-stage process that combines basic operators with established algorithms for graphs and time series. We will emphasize analytical tasks crucial to the data communities, such as detection and pattern mining. *Detection* (D) tasks differ slightly between time series and graphs: the former typically focuses on identifying anomalies, while the latter concentrates on detecting communities. HyGraph exploits such a duality to enrich anomaly detection with contextual data from graph communities and to use trends in data for informed anomaly identification. *Pattern Mining* (PM) in HyGraph involves identifying recurring subgraphs. It refers to generating potential subgraphs through the HyGraphToGraph interface, testing their occurrence frequency, and integrating time-series data to analyze trends in sub-structures featuring common vertex types.

HyGraph and AI. HyGraph explores various methodologies for merging graph-based and time-series machine learning models, such as combining them into a cohesive unit like the GC-LSTM model [24] or enhancing existing models with time-series capabilities, such as TISER-GCN [9]. We also explore hybrid techniques that involve training separate models and integrating their findings through joint learning techniques [38, 83]. We plan to design specialized *embeddings* (E) to capture the topological *and* temporal data characteristics within HyGraph, such as incorporating node2vec [53] or FastRP [23] for structure and context and PCA [91] for time-series aspects. Besides, enhancing *classification* (C1) and *clustering* (C2) by developing methods that utilize features from time series for clustering based on the graph structure, and employing trend analysis for graph-based classification.

Graph Retrieval-Augmented Generation (GraphRAG) employs a graph as a source of contextual information for the LLM [30]. Since HyGraph maps not only structural information but also its evolution over time and masses of correlated time-series data, it serves as an ideal extended knowledge base for GraphRAG. Our plan for integrating HyGraph into a GraphRAG pipeline comprises three steps. Initially, the HyGraph implementation must provide a query API and support vector similarity searches. Subsequently, as previously described, nodes are augmented with new embeddings that represent both evolutionary graph- and time series features. Finally, the HyGraph is integrated into an orchestration framework, so that relevant nodes are found by similar embeddings. Those nodes can either be directly used as knowledge or used as the starting point for subsequent queries.

HyGraph Implementation feasibility and challenges. Implementing HyGraph in real-world scenarios requires addressing challenges in managing and querying hybrid data efficiently (OLTP) and enabling advanced analysis (OLAP). This section explores the implementation feasibility and proposes an architecture (Figure 1, right)) to address indexing, scalability, and integration concerns. One of the key challenges in implementing HyGraph is to unify both data models into a single, coherent data model without treating them separately. In this unified approach, users primarily interact with a HyGraph instance. To deal with only one data model, functions are implemented to extract the necessary single data model when needed.

We propose an architecture that leverages existing data management systems through polyglot persistence (already tested) or multi-model databases (in our roadmap, explaining the yellow color in Figure 1), integrating our HyGraph data model that is responsible for unifying approach, deeply on top of the storage layer. This HyGraph architecture (see Figure 1) allows users to interact with hybrid data as if stored in a single system.

In designing HyGraph's storage layer, we plan to leverage the scalability mechanisms provided by existing data management systems [2, 49]. Selecting inherently scalable systems will be crucial for efficiently managing hybrid graph and time-series data. Initially, we aim to adopt single storage systems to simplify design and maintenance, avoiding the complexities of distributed systems. However, these systems may lack the horizontal scalability required for large-scale datasets. As workloads grow, we plan to transition to distributed storage systems that enable horizontal scaling by distributing data across nodes while addressing challenges such as network overhead and data consistency [85].

To optimize performance, we plan to apply in-memory caching techniques. For efficient querying and analysis, we plan to use existing indexing mechanisms at the storage layer, such as Neo4j's adjacency index and TimescaleDB's B-Tree and hypertable partitioning [76, 77] in our polyglot prototype. We aim to implement combined indexing mechanisms at the HyGraph data model layer to integrate graph and time-series indices. For example, Neo4j's property index could be extended to include aggregated time-series features, enabling the grouping of nodes by shared characteristics. Moreover, we will develop semantic indices [56] that combine embeddings from graph structures and time-series data. Such advanced indexing methods support future applications such as HyGraph-RAG, going beyond the separate indexing approaches used in current multi-model databases [50].

7 CONCLUSION

In this paper, we present our vision for HyGraph, a new powerful model that unifies temporal graphs and time series to form a single coherent hybrid model. It bridges the gap between disparate data models and opens up new dimensions of analytical capabilities with its flexible model and operator pipelines. Our proposed research roadmap lays the foundations for realizing the full potential of HyGraph. It aims to take HyGraph from a visionary concept to a working tool, concentrating on developing the hybrid data model, intuitive querying, and simple and complex hybrid analytics. HyGraph will pave the way for joint research in data management, and machine learning communities.

ACKNOWLEDGMENTS

The DFG, under grant number RA 497/25-1, and ANR, under grant number ANR-22-CE92-0025-01, fund this project. We further acknowledge the Federal Ministry of Education and Research of Germany's financial support and the Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus for ScaDS.AI.

REFERENCES

[1] Ratnadip Adhikari and Ramesh K Agrawal. 2013. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613* (2013).
[2] Divyakant Agrawal, Amr El Abbadi, Sudipto Das, and Aaron J Elmore. 2011. Database scalability, elasticity, and autonomy in the cloud. In *International*

conference on database systems for advanced applications. Springer, 2–15.
[3] Mouna Ammar. 2024. HyGraph-core py package installation. https://pypi.org/project/hygraph-core/ Accessed: 2024-12-10.
[4] Mouna Ammar. 2024. HyGraph-core python package: Combined graph and time series analysis. https://github.com/dbs-leipzig/HyGraph-package/tree/develop Accessed: 2024-12-10.
[5] Eric Anderson, Martin Arlitt, Charles B Morrey III, and Alistair Veitch. 2009. DataSeries: An efficient, flexible data format for structured serial data. *ACM SIGOPS Operating Systems Review* 43, 1 (2009), 70–75.
[6] Renzo Angles. 2018. The Property Graph Database Model. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018 (CEUR Workshop Proceedings)*, Dan Olteanu and Barbara Poblete (Eds.), Vol. 2100. CEUR-WS.org.
[7] ArangoDB. 2023. *ArangoDB: Multi-model NoSQL Database.* https://arangodb.com/graph-document/
[8] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. 2021. A review on outlier/anomaly detection in time series data. *ACM computing surveys (CSUR)* 54, 3 (2021), 1–33.
[9] Stefan Bloemheuvel, Jurgen van den Hoogen, Dario Jozinović, Alberto Michelini, and Martin Atzmueller. 2023. Graph neural networks for multivariate time series regression with application to seismic data. *International Journal of Data Science and Analytics* 16, 3 (2023), 317–332.
[10] E. Bollen, R. Hendrix, and B. Kuijpers. 2024. Managing Data of Sensor-Equipped Transportation Networks using Graph Databases. *Geoscientific Instrumentation, Methods and Data Systems Discussions* 2024 (2024), 1–30. https://doi.org/10.5194/gi-2024-3
[11] Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. *Querying Graphs.* Morgan & Claypool Publishers.
[12] Angela Bonifati, Wim Martens, and Thomas Timm. 2017. An Analytical Study of Large SPARQL Query Logs. *Proc. VLDB Endow.* 11, 2 (2017), 149–161.
[13] Angela Bonifati and Riccardo Tommasini. 2024. An Overview of Continuous Querying in (Modern) Data Systems. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 605–612. https://doi.org/10.1145/3626246.3654679
[14] Paul Boniol and Themis Palpanas. 2020. Series2Graph: Graph-based Subsequence Anomaly Detection for Time Series. *Proc. VLDB Endow.* 13, 11 (2020), 1821–1834. http://www.vldb.org/pvldb/vol13/p1821-boniol.pdf
[15] Paul Boniol, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2020. GraphAn: Graph-based subsequence anomaly detection. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2941–2944.
[16] Pieter Bonte, Jean-Paul Calbimonte, Daniel de Leng, Daniele Dell'Aglio, Emanuele Della Valle, Thomas Eiter, Federico Giannini, Fredrik Heintz, Konstantin Schekotihin, Danh Le Phuoc, Alessandra Mileo, Patrik Schneider, Riccardo Tommasini, Jacopo Urbani, and Giacomo Ziffer. 2024. Grounding Stream Reasoning Research. *TGDK* 2, 1 (2024), 2:1–2:47. https://doi.org/10.4230/TGDK.2.1.2
[17] Fabian Braun, Olivier Caelen, Evgueni N Smirnov, Steven Kelk, and Bertrand Lebichot. 2017. Improving card fraud detection through suspicious pattern discovery. In *Advances in Artificial Intelligence: From Theory to Practice: 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, 2017, Proceedings, Part II 30*. Springer, 181–190.
[18] Hongming Cai, Boyi Xu, Lihong Jiang, and Athanasios V Vasilakos. 2016. IoT-based big data storage systems in cloud computing: perspectives and challenges. *IEEE Internet of Things Journal* 4, 1 (2016), 75–87.
[19] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering* 30, 9 (2018), 1616–1637.
[20] Remy Cazabet, Pablo Jensen, and Pierre Borgnat. 2018. Tracking the evolution of temporal patterns of usage in bicycle-Sharing systems using nonnegative matrix factorization on multiple sliding windows. *International Journal of Urban Sciences* 22, 2 (2018), 147–161.
[21] Chris Chatfield. 2003. *The Analysis of Time Series.* 352 pages. https://doi.org/10.4324/9780203491683
[22] Khyati Chaudhary, Jyoti Yadav, and Bhawna Mallick. 2012. A review of fraud detection techniques: Credit card. *International Journal of Computer Applications* 45, 1 (2012), 39–44.
[23] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. 2019. Fast and accurate network embeddings via very sparse random projection. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 399–408.
[24] Jinyin Chen, Xueke Wang, and Xuanheng Xu. 2022. GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence* (2022), 1–16.
[25] Fatoumata Dama and Christine Sinoquet. 2021. Time series analysis and modeling to forecast: A survey. *arXiv preprint arXiv:2104.00164* (2021).
[26] Vitor de Castro Silva, Bruno Bogaz Zarpelão, Eric Medvet, and Sylvio Barbon. 2023. Explainable time series tree: An explainable top-down time series segmentation framework. *IEEE Access* 11 (2023), 120845–120856.
[27] R Devaki, V Kathiresan, and S Gunasekaran. 2014. Credit card fraud detection using time series analysis. *International Journal of Computer Applications* 3 (2014), 8–10.
[28] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magdalena Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stanley B. Zdonik. 2015. The BigDAWG Polystore System. *SIGMOD Rec.* 44, 2 (2015), 11–16. https://doi.org/10.1145/2814710.2814713
[29] Sathishkumar V. E., Jangwoo Park, and Yongyun Cho. 2020. Using data mining techniques for bike sharing demand prediction in metropolitan city. *Comput. Commun.* 153 (2020), 353–366. https://doi.org/10.1016/J.COMCOM.2020.02.007
[30] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).
[31] Benjamin Ertl, Jörg Meyer, Matthias Schneider, and Achim Streit. 2021. Semi-Supervised Time Point Clustering for Multivariate Time Series.. In *Canadian AI*.
[32] Philippe Esling and Carlos Agon. 2012. Time-series data mining. *ACM Computing Surveys (CSUR)* 45, 1 (2012), 1–34.
[33] Leonardo N Ferreira and Liang Zhao. 2016. Time series clustering via community detection in networks. *Information Sciences* 326 (2016), 227–242.
[34] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.
[35] Tak-chung Fu, Fu-lai Chung, Robert Luk, and Chak-man Ng. 2008. Representing financial time series based on data point importance. *Engineering Applications of Artificial Intelligence* 21, 2 (2008), 277–300.
[36] Ben D Fulcher and Nick S Jones. 2014. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* 26, 12 (2014), 3026–3037.
[37] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. 2005. Mining data streams: a review. *ACM Sigmod Record* 34, 2 (2005), 18–26.
[38] Snezhana Gocheva-Ilieva, Hristina Kulina, and Antoaneta Yordanova. 2022. Stacking Machine Learning Models using Factor Analysis to Predict the Output Laser Power. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. IEEE, 1–6.
[39] Philipp Marian Grulich, Steffen Zeuch, and Volker Markl. 2021. Babelfish: Efficient Execution of Polyglot Queries. *Proc. VLDB Endow.* 15, 2 (2021), 196–210. http://www.vldb.org/pvldb/vol15/p196-grulich.pdf
[40] Erez Hartuv and Ron Shamir. 2000. A clustering algorithm based on graph connectivity. *Information processing letters* 76, 4-6 (2000), 175–181.
[41] Martin Hirzel, Guillaume Baudart, Angela Bonifati, Emanuele Della Valle, Sherif Sakr, and Akrivi Vlachou. 2018. Stream Processing Languages in the Big Data Era. *SIGMOD Rec.* 47, 2 (2018), 29–40.
[42] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125.
[43] Saiful Islam, Md Nahid Hasan, and Pitambar Khanra. 2024. A Structural Feature-Based Approach for Comprehensive Graph Classification. *arXiv preprint arXiv:2408.05474* (2024).
[44] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. 2008. Discovery of convoys in trajectory databases. *Proc. VLDB Endow.* 1, 1 (2008), 1068–1080.
[45] Udayan Khurana and Amol Deshpande. 2013. Efficient snapshot retrieval over historical graph data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 997–1008.
[46] Thi Kieu Khanh Ho, Ali Karami, and Narges Armanfard. 2023. Graph-based Time-Series Anomaly Detection: A Survey and Outlook. *arXiv e-prints* (2023), arXiv–2302.
[47] Gebhard Kirchgässner, Jürgen Wolters, and Uwe Hassler. 2012. *Introduction to modern time series analysis.* Springer Science & Business Media.
[48] Xingni Li, Yi Gu, Po-Chun Huang, Duo Liu, and Liang Liang. 2017. Down-sampling of time-series data for approximated dynamic time warping on nonvolatile memories. In *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 1–6.
[49] André Lopes, Diogo Rodrigues, João Saraiva, Maryam Abbasi, Pedro Martins, and Cristina Wanzeller. 2023. Scalability and Performance Evaluation of Graph Database Systems: A Comparative Study of Neo4j, JanusGraph, Memgraph, NebulaGraph, and TigerGraph. In *2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. IEEE, 537–542.
[50] Jiaheng Lu and Irena Holubová. 2019. Multi-model databases: a new journey to handle the variety of data. *ACM Computing Surveys (CSUR)* 52, 3 (2019), 1–38.
[51] Shangyu Luo, Zekai J Gao, Michael Gubanov, Luis L Perez, and Christopher Jermaine. 2018. Scalable linear algebra on a relational database system. *ACM SIGMOD Record* 47, 1 (2018), 24–31.
[52] Lyft Bikes & Scooters and Constantin Urbainsky. 2024. New York City Bike Sharing Network: Time-Series Enhanced Nodes and Edges Dataset. https://doi.org/10.5281/zenodo.13846868 Accessed: 2024-09-27.
[53] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. 2018. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 3762–3765.
[54] mangodb. 2023. *MongoDB: Time-series collection: time-series data and query optimization.* https://www.mongodb.com/docs/manual/core/timeseries-collections/
[55] Johan Medrano, Abderrahmane Kheddar, and Sofiane Ramdani. 2024. Assessing time series correlation significance: A parametric approach with application to physiological signals. *Biomedical Signal Processing and Control* 94

(2024), 106235.
[56] Neo4j. 2024. *Neo4j: Semantic indexes.* https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/overview/
[57] networkX. 2024. *NetworkX: Network analysis in python.* https://networkx.org/
[58] Axel Polleres, Romana Pernisch, Angela Bonifati, Daniele Dell'Aglio, Daniil Dobriy, Stefania Dumbrava, Lorena Etcheverry, Nicolas Ferranti, Katja Hose, Ernesto Jiménez-Ruiz, Matteo Lissandrini, Ansgar Scherp, Riccardo Tommasini, and Johannes Wachs. 2023. How Does Knowledge Evolve in Open Knowledge Graphs? *TGDK* 1, 1 (2023), 11:1–11:59. https://doi.org/10.4230/TGDK.1.1.11
[59] Debachudamani Prusti, Daisy Das, and Santanu Kumar Rath. 2021. Credit card fraud detection technique by applying graph database model. *Arabian Journal for Science and Engineering* 46, 9 (2021), 1–20.
[60] Thanawin Rakthanmanon, Bilson J. L. Campana, Abdullah Mueen, Gustavo E. A. P. A. Batista, M. Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn J. Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*. ACM, 262–270.
[61] Marko A. Rodriguez and Peter Neubauer. 2010. Constructions from dots and lines. *Bulletin of the American Society for Information Science and Technology* 36, 6 (2010), 35–41. https://doi.org/10.1002/bult.2010.1720360610 arXiv:https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/bult.2010.1720360610
[62] Christopher Rost, Philip Fritzsche, Lucas Schons, Maximilian Zimmer, Dieter Gawlick, and Erhard Rahm. 2021. Bitemporal Property Graphs to Organize Evolving Systems. *CoRR* abs/2111.13499 (2021). arXiv:2111.13499 https://arxiv.org/abs/2111.13499
[63] Christopher Rost, Kevin Gomez, Peter Christen, and Erhard Rahm. 2023. Evolution of Degree Metrics in Large Temporal Graphs. In *Datenbanksysteme für Business, Technologie und Web (BTW 2023) (LNI)*, Vol. P-331. Gesellschaft für Informatik e.V., 485–507. https://doi.org/10.18420/BTW2023-23
[64] Christopher Rost, Kevin Gómez, Philip Fritzsche, Andreas Thor, and Erhard Rahm. 2021. Exploration and Analysis of Temporal Property Graphs. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*. 682–685. https://doi.org/10.5441/002/EDBT.2021.83
[65] Christopher Rost, Kevin Gómez, Matthias Täschner, Philip Fritzsche, Lucas Schons, Lukas Christ, Timo Adameit, Martin Junghanns, and Erhard Rahm. 2022. Distributed temporal graph analytics with Gradoop. *VLDB J.* 31, 2 (2022), 375–401. https://doi.org/10.1007/S00778-021-00667-4
[66] Christopher Rost, Riccardo Tommasini, Angela Bonifati, Emanuele Della Valle, Erhard Rahm, Keith W. Hare, Stefan Plantikow, Petra Selmer, and Hannes Voigt. 2024. Seraph: Continuous Queries on Property Graph Streams. In *Proceedings of the 27td International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28, 2024*, Letizia Tanca, Qiong Luo, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani (Eds.). OpenProceedings.org. https://doi.org/10.48786/edbt.2024.21
[67] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71.
[68] Axel Schuster. 2023. TimeTravelDB: Polyglot persistence combining Graph and Time series. https://git.informatik.uni-leipzig.de/ammar/time-travel-db-v-2-benchmark Accessed: 2024-12-10.
[69] Arie Segev and Rakesh Chandra. 1993. A data model for time-series analysis. *Advanced Database Systems* (1993), 191–212.
[70] Amir Shaikhha, Mathieu Huot, Jaclyn Smith, and Dan Olteanu. 2022. Functional collection programming with semi-ring dictionaries. *Proceedings of the ACM on Programming Languages* 6, OOPSLA1 (2022), 1–33.
[71] Amir Shaikhha, Maximilian Schleich, and Dan Olteanu. 2021. An intermediate representation for hybrid database and machine learning workloads. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2831–2834.
[72] Benjamin Steer, Felix Cuadrado, and Richard Clegg. 2020. Raphtory: Streaming analysis of distributed temporal graphs. *Future Generation Computer Systems* 102 (2020), 453–464.
[73] Ran Tan, Rada Chirkova, Vijay Gadepally, and Timothy G. Mattson. 2017. Enabling query processing across heterogeneous data models: A survey. In *2017 IEEE International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017*, Jian-Yun Nie, Zoran Obradovic, Toyotaro Suzumura, Rumi Ghosh, Raghunath Nambiar, Chonggang Wang, Hui Zang, Ricardo Baeza-Yates, Xiaohua Hu, Jeremy Kepner, Alfredo Cuzzocrea, Jian Tang, and Masashi Toyoda (Eds.). IEEE Computer Society, 3211–3220. https://doi.org/10.1109/BigData.2017.8258302
[74] Zhipeng Tan, Zhuoxun Zheng, Antonis Klironomos, Mohamed H. Gad-Elrab, Guohui Xiao, Ahmet Soylu, Evgeny Kharlamov, and Baifan Zhou. 2023. Literal-Aware Knowledge Graph Embedding for Welding Quality Monitoring. In *ISWC 2023 (CEUR Workshop Proceedings)*, Vol. 3632.
[75] Donato Tiano, Angela Bonifati, and Raymond Ng. 2021. FeatTS: Feature-based Time Series Clustering. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2784–2788.
[76] Timescale. 2024. *TimescaleDB: About hypertables.* https://docs.timescale.com/use-timescale/latest/hypertables/about-hypertables/
[77] Timescale. 2024. *TimescaleDB vs. InfluxDB: Purpose Built Differently for Time-Series Data.* https://www.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/
[78] Riccardo Tommasini, Pieter Bonte, Femke Ongenae, and Emanuele Della Valle. 2021. RSP4J: An API for RDF Stream Processing. In *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings (Lecture Notes in Computer Science)*, Ruben Verborgh, Katja Hose, Heiko Paulheim, Pierre-Antoine Champin, Maria Maleshkova, Óscar Corcho, Petar Ristoski, and Mehwish Alam (Eds.), Vol. 12731. Springer, 565–581. https://doi.org/10.1007/978-3-030-77385-4_34
[79] Riccardo Tommasini, Pieter Bonte, Fabiano Spiga, and Emanuele Della Valle. 2023. *Streaming Linked Data.* Springer.
[80] Constantin Urbainsky. 2023. TimeTravelDB: Polyglot persistence combining Graph and Time series. https://git.informatik.uni-leipzig.de/ammar/time-travel-db-v-2-benchmark/-/blob/main/benchmark/Benchmark_timetraveldb_neo4j.pdf?ref_type=heads Accessed: 2024-12-10.
[81] Atif Usman, Nasir Naveed, and Saima Munawar. 2023. Intelligent anti-money laundering fraud control using graph-based machine learning model for the financial domain. *Journal of Cases on Information Technology* 25, 1 (2023), 1–20.
[82] Salvatore Vilella, Arthur Thomas Edward Capozzi Lupi, Giancarlo Ruffo, Marco Fornasiero, Dario Moncalvo, Valeria Ricci, and Silvia Ronchiadin. 2023. Exploiting graph metrics to detect anomalies in cross-country money transfer temporal networks. In *Companion Proceedings of the ACM Web Conference 2023*. 1245–1248.
[83] Zhihong Wang, Hongru Ren, Renquan Lu, and Lirong Huang. 2022. Stacking Based LightGBM-CatBoost-RandomForest Algorithm and Its Application in Big Data Modeling. In *2022 4th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. IEEE, 1–6.
[84] Klaus Wehmuth, Artur Ziviani, and Eric Fleury. 2015. A unifying model for representing time-varying graphs. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 1–10.
[85] Lena Wiese. 2015. *Advanced data management: for SQL, NoSQL, cloud and distributed databases.* Walter de Gruyter GmbH & Co KG.
[86] Giles Winchester, George Parisis, and Luc Berthouze. 2023. On the Temporal Behaviour of a Large-Scale Microservice Architecture. In *NOMS*. IEEE, 1–6.
[87] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014. Path Problems in Temporal Graphs. *Proc. VLDB Endow.* 7, 9 (2014), 721–732. https://doi.org/10.14778/2732939.2732945
[88] xarray. 2024. *Xarray: Dealing with multidimensional arrays in python.* https://docs.xarray.dev/en/stable
[89] Haoran Xiong, Hang Zhang, Zeyu Wang, Zhenying He, Peng Wang, and X Sean Wang. 2024. CIVET: Exploring Compact Index for Variable-Length Subsequence Matching on Time Series. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2123–2135.
[90] Jiajia Xu, Yichang Qiu, Haiying Zhang, Meng Li, and Manli Li. 2017. Large-scale time series data down-sampling based on Map-Reduce programming mode. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, 409–413.
[91] Kiyoung Yang and Cyrus Shahabi. 2004. A PCA-based similarity measure for multivariate time series. In *Proceedings of the 2nd ACM international workshop on Multimedia databases*. 65–74.