

Continuous Query Engine to Detect Anomalous ATM Transactions

Fernando Martín-Canfrán, Daniel Benedí, Amalia Duch and Edelmira Pasarella

Universitat Politècnica de Catalunya, Barcelona Tech, Barcelona, Spain

Abstract

Nowadays data are in motion, change continuously and are –possibly– unbounded implying data sources that are also constantly evolving. From the data persistence point of view this reality breaks the usual paradigm of having dynamic but stable data sources. This, together with the increasing number applications based on data streams for taking critical decisions in real time, raises the need for re-thinking both the data and the query models to fit these new requirements. Therefore, under these circumstances, it seems reasonable that a suitable data model is a *continuously evolving data graph*. In this work we tackled the problem of querying continuously evolving data graphs in the context of ATM transactions, in particular anomalous ones. Under this context, evaluating continuous queries corresponds to recognize patterns usually associated with anomalous behaviors in the volatile subgraph of ATM transactions. We propose an evaluation process based on the dynamic pipeline computational model, a stream processing technique allowing the emission of alerts as soon as anomalous patterns are identified. Stream based Bank applications that monitor ATM transactions are direct beneficiaries of our proposal since they can continuously query data graphs to get “fresh” data as are produced, avoiding the computational overhead of having to discard non-valid data.

Keywords

continuous query evaluation, property graphs, dynamic pipeline approach, stream processing, ATM transactions

1. Introduction

Although from a classical point of view databases are thought of for persistent data, nowadays this perspective is changing since data are in motion, continuously changing and (possibly) unbounded. So, the following questions arise: (i) What is the proper data model? and (ii) What is the proper query model?

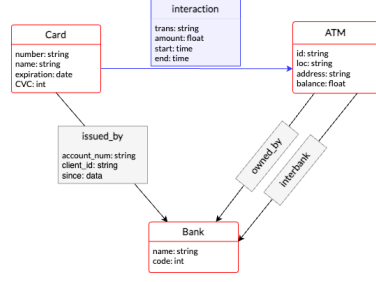
Regarding the data model, the new nature of data requires a *de facto* new database paradigm -*continuously evolving databases*- where data can be both *stable* and *volatile*. Even though evolving databases can be implemented according to any approach, graph databases seem especially well suited here [1, 2]. Indeed, the natural way to process evolving graphs as streams of edges gives insights on how to proceed in order to maintain dynamic graph databases. Hence, we consider that a suitable data model is a *continuously evolving data graph*, a graph having persistent (*stable*) as well as non persistent (*volatile*) relations. Stable relations correspond to edges occurring in standard graph databases while volatile relations are edges arriving in

AMW2024: 16th Alberto Mendelzon International Workshop on Foundations of Data Management, September 30th - October 4th, 2024, Mexico City, Mexico

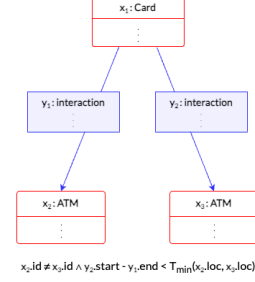
✉ fernando.martin.canfran@estudiantat.upc.edu (F. Martín-Canfrán); daniel.benedi@estudiantat.upc.edu (D. Benedí); duch@cs.upc.edu (A. Duch); edelmira@cs.upc.edu (E. Pasarella)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



(a) Part of a schema of a PG



(b) Pattern of anomalous transactions

Figure 1: Part of a PG schema specifying volatile (**interaction** edges) and stable (**issued_by**, **owned_by**, **interbank** edges) relations in an evolving ATM Network and a continuous query pattern.

data streams during a set time interval. Once this time interval is over, the relations are not longer valid so that there is no need to store them in the (stable) graph database. However, when required -as for further legal or auditing purposes- timestamped occurrences of volatile relations can be kept in a log file. Volatile relations induce subgraphs that exist only while the relations are still valid. Without loss of generality, in this work we consider property graphs (PG) [3, 4] as the basic reference data model. As an example, Figure 1a depicts part of a schema of a PG database where stable relations correspond to the data that a bank typically gathers on its issued cards, ATMs (Automated Teller Machines) network, etc. Volatile relations model the interaction between cards and ATM entities. Concerning the query model, fixed queries evaluated over data streams are known as *continuous queries* [5, 6]. Thus, instead of classical query evaluation processes we envision *incremental/progressive* query evaluation processes. A query on a PG database can be seen as a PG graph pattern with constraints over some of its properties. Evaluating such a query consists on identify if there is a subgraph of the database that matches the given pattern and satisfies its constraints. The problem of progressively identify and enumerate bitriangles (i.e. a specific graph pattern) in bipartite evolving graphs using the *Dynamic Pipeline Approach* [7] have been successfully solved by Royo-Sales [8]. We claim that the problem of evaluating continuous queries over *continuously evolving PGs* belongs to the same family of problems and hence, we propose to address it using the same stream processing approach. However, in this case, in addition to identify the query pattern, the constraint satisfaction over properties must be checked also. Figure 1b shows a constrained graph pattern corresponding to a continuous query. In this work, as a proof of concept, we tackled the problem of evaluating continuous queries corresponding to anomalous patterns of ATM transactions against a continuously evolving PG representing a bank database. To be concrete, the anomalous patterns of ATM transactions are identified in the volatile (PG) subgraph of the considered database. The evaluation process is based on the dynamic pipeline computational model and emits answers (alarms) as soon as anomalous patterns are identified. Additionally, a log of all the volatile relations of the PG is maintained. Figure 2 illustrates a possible anomalous situation associated to this query.

Related work. Recently, in Rost et al. [9] authors formalize an extension to the query language Cypher that allows for evaluating continuous queries over property graph streams

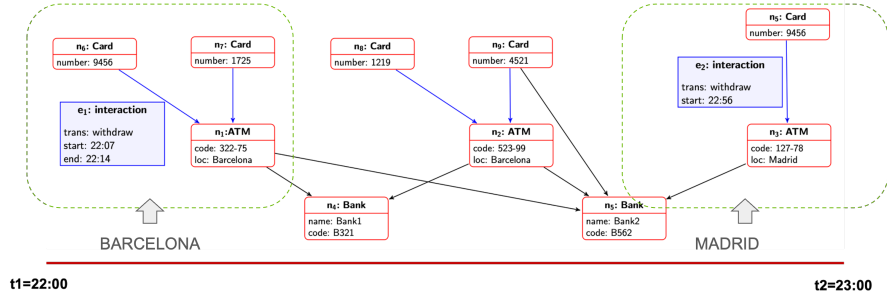


Figure 2: Example of the occurrence of anomalous ATM transactions in (a part of) a continuously evolving PG over a time interval: the card 9456 is used twice at ATMs in different cities, within one hour. However, to get from one of the cities to the other and vice versa requires more than one hour using any means of transport. This example could represent a possible case of *skimming* and *cloning*.

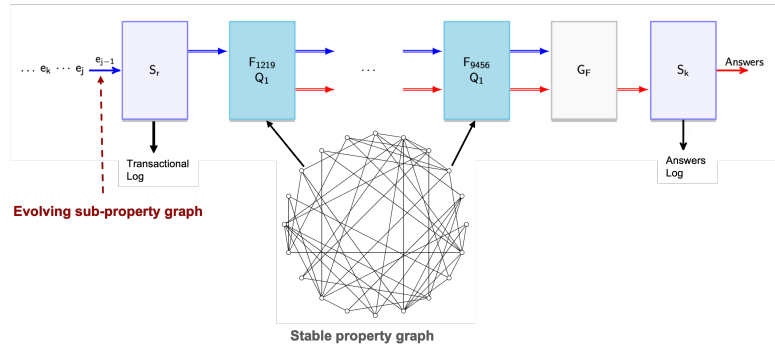


Figure 3: Preliminary continuous query engine architecture for detecting anomalous ATM transactions based on the dynamic pipeline computational model. Considering the schema given in Figure 1a, in this directed (multi) graph presentation of the DP_{ATM} , the arriving input data is a stream $\langle \dots e_k \dots e_j e_{j-1} \rangle$ corresponding to interactions (volatile relations). Boxes (vertices) represent stateful processes called *stages* and internal arrows in the pipeline represent channels. Blue channels carry interaction edges and red channels carry detected anomalies (answers). S_r and S_k correspond to the Source and Sink stages which receive input data and results, respectively. Filter stages, F_N , are parameterized with the value of the property number (N) of Card vertices. The Generator stage, G_F , is in charge of spawning new filters, when required. The stable PG is a standard bank database (i.e. without volatile relations). Transactional log and Answers log keep input interactions and answers, respectively.

according to a specified frequency, during a defined time interval. The main differences of their approach with our proposal are: first, the data model because they merge incoming PGs to the persistent database and, second, we evaluate continuous queries as new edges are added to the volatile part of the property graph (i.e. as the PG evolves) by processing (identifying) patterns (subgraphs). To our knowledge there is no other work approaching the problem of modeling and implementing a continuous query engine for detecting anomalous ATM transactions. The related work that we have found is mainly based on using machine learning, big data and data visualization approaches[10, 11, 12, 13, 14].

Contribution. The main contribution of this work is to provide, using a stream processing approach, a general technique for addressing the problem of continuous query evaluation against an evolving graph database by decomposing the datagraph into volatile and stable well defined subgraphs. Among the advantages of using the dynamic pipeline computational model are its parallel/concurrent nature and its suitability for developing real-time systems that emit results as they are computed, in a progressive way. In regards to detecting abnormal or suspicious ATM transaction, to our knowledge, most of the work addressing this topic provide a delayed detection based on predictions given by ML systems. Also, it is frequent the classical treatment of the problem by consulting log files because of the complaint of customers when detecting by themselves some weird movement in their accounts. This involves annoying processes for customers in order to have their money back. The idea is that, in presence of some weird finding in an ATM transaction, banks have a tool able to either ask card holders for authorizations or to take any other fraud preventing action at real-time.

2. Challenges for achieving a real implementation

Defining and implementing a continuous query engine requires to address many different problems, each of different nature. In addition, the proof of concept we intend to provide has itself its own complications.

2.1. Defining anomalous patterns of transactions

It is not trivial to establish what is and in which circumstances a transaction can be considered anomalous. Based on a work that have addressed this characterization [15] we intend to find a proper characterization and then define the graph patterns associated to these anomalies. The exact topology of an anomaly will depend on its own nature. Figure 1b depicts an example characterizing a possible card cloning, among many other possibilities. For instance, using a (stolen) card many times over a period of time at different ATMs to withdraw small amounts. In this latter case, there will arrive to the evolving PG many volatile (interaction) edges having the same card vertex and different ATM vertices. There could also be patterns related with frequent/very high expenses; transactions located in an ATM out of the threshold distance of the usual/registered address of the card holder and so on. Moreover, definition of patterns can be beyond ATM transactions by considering online card transactions.

2.2. Modeling and implementing the continuous query engine

To define a proper architecture for a continuous query engine is one of the most challenging activities of our work. Among other tasks, this comprises: to define a graph-based query language expressive enough to allow for capturing the different patterns representing anomalous queries; to establish the algorithms for identifying the patterns associated to anomalous queries; to choose and manage the right windowing approach and other features related to distributed query-evaluation; to deal with the evaluation of many continuous queries simultaneously; to evaluate the suitability of the implementation language, tools and the proper system configuration. Figure 3 depicts a preliminary architecture for a continuous query engine

for detecting anomalous ATM transactions, DP_{ATM} . We propose a solution that follows the Dynamic Computational Model [7]. Briefly speaking, in this approach, *stages* are processes that execute tasks concurrently/in-parallel. The multiset underlying the input data stream is partitioned [16] and distributed along filters according to a grouping relationship, usually based on filters' parameters. Each filter applies the same function to its block of data (stored as its state). Accordingly, the DP_{ATM} algorithm is specified as follows: During a pre-defined time interval window, when an interaction e (together its properties' values) arrives to the DP_{ATM} , the stage S_r register it into a standard transactional log file. Then, S_r passes e to the next stage. If there exists a filter parameterized with the value of the property number of the Card vertex that is incident to e , this filter keeps e in its state. In this way, filters' states store subgraphs induced by the edges in the volatile subgraph. Notice that these sets of edges in each filter correspond to blocks of the (multiset) input data stream. Otherwise, the filter passes e to the next stage. The task/function of each filter is to decide if there is a match with (some of) the continuous query pattern(s) evaluated by the engine DP_{ATM} by means of the graph that it stores and the information retrieved from the stable PG to identify patterns and solve constraints. This is, indeed, the way to evaluate continuous queries. In case of matching a pattern, filters emit an alert reporting the finding. Hence, answers are the detected anomalies and they are emitted as they are obtained in filters. When answers arrive to S_k , this stage post-process and output them. In addition, S_k maintains an answer log file. The fact that an interaction arrives to G_F means that there were not previous interactions having the same value of Card property number and thus, a new filter parameterized with this new value is spawned. When the time interval window is over, the DP_{ATM} is, in some sense, reset according to the given window policy. Note that the window policy must take into account stored data that might be valid in between two windows and handle the transition properly.

3. Discussion on Ongoing Work and Outlook

The main luck stone, in order to be able to provide the proof of concept that we propose, is to have a proper dataset. Given the confidential and private nature of bank data, it has been impossible to find a real dataset for our experiments. In this regard, we are currently constructing a synthetic property graph bank dataset based on the *Wisabi Bank Dataset*¹. In fact, we consider that our synthetic dataset will be an important contribution of this work.

A prototype implementation of the DP_{ATM} for detecting the anomalous ATM transaction pattern presented in Figure 1b has been developed using Go language. For testing this implementation² we have constructed a small stable bank PG using Neo4j graph database management system. We plan to extend the prototype including window management and multiple continuous query evaluation. Once we had generated a big enough stable bank property graph, to be able to conduct experiments to assess our proposal, we need to tackle two important issues: (i) to find appropriate baselines to compare our solution and (ii) to establish what are the proper statistic frequency distributions for including anomalous ATM transactions in the input streams of volatile relations.

¹<https://www.kaggle.com/datasets/obinnaiheanachor/wisabi-bank-dataset>

²<https://github.com/FCanfran/ATM-DP>

Acknowledgments

This work has been supported by MCIN/AEI/10.13039/501100011033 under grant PID2020-112581GB-C21.

References

- [1] R. Angles, C. Gutierrez, Survey of graph database models, *ACM Computing Surveys (CSUR)* 40 (2008) 1–39.
- [2] R. Kumar Kaliyar, Graph databases: A survey, in: *International Conference on Computing, Communication & Automation*, IEEE, 2015, pp. 785–790.
- [3] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, D. Vrgoč, Foundations of modern query languages for graph databases, *ACM Computing Surveys (CSUR)* 50 (2017) 1–40.
- [4] R. Angles, The property graph database model, in: *CEUR Workshop Proceedings*, CEUR-WS.org (AMW2018), volume 2100, 2018. URL: <https://ceur-ws.org/Vol-2100/paper26.pdf>.
- [5] S. Babu, J. Widom, Continuous queries over data streams, *ACM Sigmod Record* 30 (2001) 109–120.
- [6] C. Zaniolo, Logical foundations of continuous query languages for data streams, in: *International Datalog 2.0 Workshop*, Springer, 2012, pp. 177–189.
- [7] E. Pasarella, M.-E. Vidal, C. Zoltan, J. P. Sales, A computational framework based on the dynamic pipeline approach, *Journal of Logical and Algebraic Methods in Programming* 139 (2024) 100966.
- [8] J. P. Royo-Sales, An algorithm for incrementally enumerating bitriangles in large bipartite networks (master thesis), 2021. URL: <http://hdl.handle.net/2117/361615>.
- [9] C. Rost, R. Tommasini, A. Bonifati, E. Della Valle, E. Rahm, K. W. Hare, S. Plantikow, P. Selmer, H. Voigt, Seraph: Continuous queries on property graph streams, in: *EDBT*, 2024, pp. 234–247.
- [10] M. Ahmed, A. N. Mahmood, M. R. Islam, A survey of anomaly detection techniques in financial domain, *Future Generation Computer Systems* 55 (2016) 278–288.
- [11] Y. Heryadi, L. A. Wulandhari, B. S. Abbas, et al., Recognizing debit card fraud transaction using chaid and k-nearest neighbor: Indonesian bank case, in: *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, IEEE, 2016, pp. 1–5.
- [12] K. Singh, P. Best, Anti-money laundering: Using data visualization to identify suspicious activity, *International Journal of Accounting Information Systems* 34 (2019) 100418.
- [13] M. M. Rahman, A. R. Saha, et al., A comparative study and performance analysis of atm card fraud detection techniques, *Journal of Information Security* 10 (2019) 188.
- [14] R. Kian, H. S. Obaid, Detection of fraud in banking transactions using big data clustering technique customer behavior indicators, *Journal of applied research on industrial engineering* 9 (2022) 264–273.
- [15] F. Magdalena-Laorden, Artificial intelligence and ontology applied to credit card fraud detection (bachelor thesis), 2021. URL: <https://oa.upm.es/69050/>.
- [16] E. A. Bender, Partitions of multisets, *Discrete Mathematics* 9 (1974) 301–311.