SIGMOD 2016
ACM SIGMOD Conference 2016
June 26 - July 1, San Francisco, CA, USA

Reviews For Paper

Track	Research, November 2015
Paper ID	448
Title	Portal: A Query Language for Evolving Graphs

Masked Reviewer ID: Assigned_Reviewer_1

Review:

Question	
Overall rating	Conditional Accept; I will fight for this paper if the authors address the concerns listed below in a revision
Briefly summarize your review, and rationale for the chosen rating	I enjoyed reading this paper. It presents a problem that I had not considered before; the ability to query evolutionary aspects of graph data. I think the authors did a great job in organizing the discussion, which gives a clear description of the query language and semantics. However, it is vague in its description of the system architecture, most notably the compiler/optimizer and how certain plans are chosen. Moreover, the experimental section does not give a clear understanding of the scalability nor a comparison to a naive GraphX implementation. If addressed in a revision, then I would be glad to revise my review to Accept.
List at least 3 strong points, numbered S1, S2, S3, ...	S1. An elegant solution to a very interesting class of queries. S2. Very well organized paper that deals with tackles a class of queries at the language, optimizer and operator levels. S3. Protal offers a good variety of distributed graph data structures and tradeoffs in terms of query performance.
List at least 3 weak points, numbered W1, W2, W3, ...	W1. Paper writing needs polishing and clarifications described below. W2. Experimental section is weak; does not really show web scale experiments, even though such problems are referenced in the introduction. Nor does it give a comparison to a pure GraphX solution. W3. Unclear how optimizer determines an optimal plan based on statistics. Paper needs a clear architecture picture that includes some form of catalog and statistics e.g., Figure 10 does not provide sufficient insight into the system level and compiler components.
Paper review; use this section to provide authors with your detailed feedback, and suggestions on how to improve the paper. Comment on	I did not fully understand the difference between TAnd and TOr. It seems that hey are both simply combining the respective graphs T1 and T2 for the structural aggregation operations. Please clarify on how the combining step differs, or is it somehow correlated with the type of structural aggregation operator Any vs. All. Computing pagerank() on the fly for each snapshot could be expensive. Perhaps you could mention that this value, or any other snapshot analytic, can be materialized into an attribute for improved performance. Figure 7 has a bug. Your result for 2010-2011 is the result for 2011-2012; I believe. This bug cascades and is the source of my confusion between TAnd and TOr (earlier comment), which I now understand. It is not clear how you decide that TAnd can be pushed before temporal selection for improving loading time. Perhaps you should mention a catalog

novelty, depth, presentation quality, and soundness and thoroughness of experimental evaluation.	and statistics that tell you the necessary information. After reading Section 4, I now understand the difference between TAnd and TOr. I'm leaving my earlier comment that arose in Section 3 because I think it needs better clarification there. I'm curious whether equivalent queries can be written in GraphX and a comparison can be made to Portal, both in terms of query complexity (i.e., how much more programming effort it takes to express queries Q1-Q8 in GraphX) and performance w.r.t. the graph data structures offered in Protal. The GraphX queries could be implemented in a naively i.e., by some data scientist that does not understand system level bottlenecks. IMO this would greatly improve the contributions presented in this paper.

Masked Reviewer ID: Assigned_Reviewer_7

Review:

Question	
Overall rating	Conditional Accept; I will fight for this paper if the authors address the concerns listed below in a revision
Briefly summarize your review, and rationale for the chosen rating	The paper presents by example a declarative SQL-like language for querying evolving graphs named Portal. It also report on an implementation of this language on top of GraphX and in turn the Spark environment.
List at least 3 strong points, numbered S1, S2, S3, ...	- Evolving graphs are an important emerging use case in big data analytics and being able to query these declaratively adds value for end users. - The design of the Portal language, by staying close to SQL in its mapping of graphs as two tables V,E preserves certain of its advantages and its temporal semantics seem well chosen. - Sound system architecture: the implementation of Portal in a system based on based on GraphX allows integration of graph analytics developed for it and leverages the power of optimized parallel data processing with Spark.
List at least 3 weak points, numbered W1, W2, W3, ...	- defining a new query language by example is not sufficient to fully nail its semantics, leaving the reader with many questions - The paper scope is too ambitious as besides defining this temporal extension of an undefined declarative query language also tries to evaluate representation and partitioning choices in the system that implements it - The integration if GraphX specific algorithms such as pagerank() and trend analysis is not well explained and seems like black magic. In such a language definition one would expect to find the semantics of a mapping between the query language and the graph algorithm inputs/outputs such would allow to envision how the language can be extended to incorporate further algorithms.
	The Portal language for querying evolving graphs is novel and addresses interesting use cases. The paper depth suffers because of the choice to combine the presentation an motivation of the new language with a comparison of implementation techniques in the system that implements this new language. The presentation quality is good. The experimental evaluation focuses on issues of data representation and partitioning, however it is difficult to assess the general validity of the conclusions since quite little other information (beyond the partitioning and data representation strategy) is

	provided on the system -- performance is a combination of processing algorithms and data representation/distribution.
Paper review; use this section to provide authors with your detailed feedback, and suggestions on how to improve the paper. Comment on novelty, depth, presentation quality, and soundness and thoroughness of experimental evaluation.	Some more detailed remarks: def 2.3: variable L in \gamma^V_{vid,L} is not defined. Q2: it is unclear what further restrictions the use of the function pagerank() imposes on the query, nor how it obtains the initial connection strength (is this required to be "score"??). Please explain this in detail in the paper. sec 4.1, third example: "T1.Start >= 2010 And T*1*.End <= 2014" sec 5.1: "The default number of partitions based on the Hadoop Block size proved inefficient in practice" =>?? be more specific. I presume it was too large? The DBLP dataset is just too small for such an evaluation. You would do better with larger graph sizes. If real graphs cannot be found, maybe try the LDBC social network benchmark graph generator. sec 5.5: the rules reason about the denseness of the graph affecting the runtime of GraphX programs. However, a properly generated GraphX program (we suppose something like this happens, though the paper tells us nothing about the system in terms of query processing) would not be affected by that. It is just a matter of pushing down filters that ensure that only the relevant edges fire messages (relevant = those belonging to the snapshot being queried). As such, even though the graph is dense, properly programmed communication needs not grow; just because of the data representation.
If you recommended a Conditional Accept, describe specific issues you would like to see addressed in a revision.	address: - the integration of pagerank() and other GraphX functions in the query processing logic of the system as well as in the language semantics of Portal - include more information about the query processing in GraphX in your implementation and enhance the evaluation section with that information (deeper analysis, maybe some more detailed graphs).

Masked Reviewer ID: Assigned_Reviewer_8

Review:

Question	
Overall rating	Reject; I have serious concerns about this paper that cannot be addressed with a revision
Briefly summarize	The paper introduces Portal, a temporal graph query language to query evolving graphs. The majority of the paper introduces the motivation and notation behind Portal, while the end of the main section of the work discusses three system-related issues of the work, focusing on query operator reordering, evolving graph representation, and graph partition strategies. The presented experiments seem to focus more on evaluating the graph representation and partition strategies and less on the aspects of the introduced language.

your review, and rationale for the chosen rating	By far, the biggest drawback of the work is the over-simplifying assumption stated by the authors as "However, we require that all vertices (resp. edges) of G have the same schema, i.e., V and E are homogeneous sets.". Into this context, the evolving "graph" that the authors are studying are actually simple relational table (with no nulls). They share little with real graphs encountered in practice (that are fairly unstructured). Even in the very limited context of graphs considered, the authors are obliged to compare their work essentially with past works on temporal relational databases since their is an easy way to map one problem/dataset to the other.
List at least 3 strong points, numbered S1, S2, S3, ...	S1: Temporal graphs are indeed an interesting area of research so the paper is well-placed in this regard.
List at least 3 weak points, numbered W1, W2, W3, ...	W1: The assumption that ".. all vertices (resp. edges) of G have the same schema, i.e., V and E are homogeneous sets." is crippling in this context and the work has little value given this.
Paper review; use this section to provide authors with your detailed feedback, and suggestions on how to improve the paper. Comment on novelty, depth, presentation quality, and soundness and thoroughness of experimental evaluation.	C1: The assumption in 2.1 that ".. all vertices (resp. edges) of G have the same schema, i.e., V and E are homogeneous sets." boils down to this: the graphs can be thought of as two relational tables, one with key vid, and another with key vid1, vid2. Both tables should have no nulls (since all vertices must have the same schema). This is a rather strong assumption and considers a very specific set of graphs that are unlike a lot of graph found “in the wild” (where nodes are heterogeneous and fairly unstructured). With this assumption in place, the authors just cannot ignore past works in temporal relational data in provide an clear comparison that indicates that they don't re-invent the wheel. C2: Definitions 2.7 and 2.8 are a bit strange: it seems your union and intersection of sequences results in constructs that are not sequences. what makes this weird is that defining the union and intersection in such a fashion makes them non-composable. For example, you cannot consider the union of three sequences P1 \union P2 \union P3 since P1 \union P2 has not guarantee that it results in a sequence. C3: It is not clear why SQL is used as a language to query graphs, when other more popular languages (e.g. SPARQL) exist for this purpose. Why not extent SPARQL of have a SPARQL-like syntax? C4: When it comes to temporal reconciliation, the authors should make clear that they focus on numerical attributes. C5: In the experimental section, the sizes of datasets considered hardly justified distribution. I think it would make more sense to consider also a single-cpu system. As things stand, the experiments focus on graph-representation and graph-partitioning, none of which seems to be the focus of this work. Focus on a single-cup system would allow for a comparison with competing systems (e.g., Miao) which is currently avoided using a rather superficial justification that Portal is geared towards distribution.
If you recommended a Conditional Accept, describe specific issues you	n/a

would like to see addressed in a revision.	