

[SIGMOD 2016](#)**ACM SIGMOD Conference 2016**

June 26 - July 1, San Francisco, CA, USA

Reviews For Paper**Track** Research, November 2015**Paper ID** 533**Title** AQuery, a query language for order in data analytics: Language, Optimization, and Experiments**Masked Reviewer ID:** Assigned_Reviewer_6**Review:**

Question	
Overall rating	Reject; I have serious concerns about this paper that cannot be addressed with a revision
Briefly summarize your review, and rationale for the chosen rating	In this paper authors describe a new query language with ordered table (arrayable) that is claimed to be a better language for data analytics with its builtin semantics for ordered data not only because of its expressive power but also allows better optimization compared to sql. Authors describe the syntax and semantics of AQuery (actually introduced in 2003), offer a number of optimizations that leverage order semantics, and show viability of their model through experiments comparing AQuery to a number of language/systems for data analytics such as Pandas, MonetDB, and Sybase IQ. Their experiments suggest improved performance against current systems.
List at least 3 strong points, numbered S1, S2, S3, ...	<ul style="list-style-type: none"> - Clear presentation - Clear performance improvements of their techniques - Nice optimization rules for ordered tables
List at least 3 weak points, numbered W1, W2, W3, ...	<ul style="list-style-type: none"> - Need to discuss more on expressive power of the language - Need to link how the suggested optimization techniques leverage the query language - Discuss how the relation algebra is extended.
	<p>In this paper authors describe a new query language with ordered table (arrayable) that is claimed to be a better language for data analytics with its builtin semantics for ordered data not only because of its expressive power but also allows better optimization compared to sql. Authors describe the syntax and semantics of AQuery (actually introduced in 2003), offer a number of optimizations that leverage order semantics, and show viability of their model through experiments comparing AQuery to a number of language/systems for data analytics such as Pandas, MonetDB, and Sybase IQ. Their experiments suggest improved performance against current systems.</p> <p>Novelty: The key novelty is not in the AQuery language (introduced earlier) but in the several optimization techniques authors described that can be applied to AQuery. While these optimization techniques currently are implemented as rules it is not difficult to incorporate them into cost-based optimization approaches that most modern systems use. Two key questions are (1) whether these optimization techniques are novel in an of themselves, and (2) whether these optimization techniques can be equally well applied to sql. Regarding (1) many of the proposed techniques are straightforward such</p>

<p>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.</p>	<p>as eliminating column sorts if they are sorted, sorting columns only not the whole table, sequence selection, etc., some of which I assume (I am not an expert in query optimization) are implemented in current systems such as DB2. Regarding (2) I examined several of the examples in the paper and there seems to be very little difference in the way the queries are expressed in sql and aquery. For example, avgs function seems to be a shorthand for order by with a range of index, assuming clause can equally be expressed in the order by clause. So from expressive power specifically I am not sure AQuery (though a bit simpler) is not significantly better in my own opinion. I wish the authors did a bit of user experience study to evaluate the expressive power with db experts (such as asking them to write queries in both aquery and sql and see what dbadmins think).</p> <p>Depth: I would have liked to see how the Aquery extends the relational algebra. For example, there is also quite a bit of row dependency on semantics. For example avgs(x, 24) authors suggest to represent a day of records where each row represents an hour. How is this represented in the schema? And also how about missing records. In practice there are a lot of missing records, if for example a couple of hours of data is missing the query semantics would not be sound. Query semantics could not depend on missing data.</p> <p>Presentation: The paper is written very clearly and is quite easily readable for a broad audience. Unfortunately performance charts are unnecessarily complex, I would suggest authors to remove all color bar charts and use line charts instead. (Authors can also reduce redundant charts such as 5c).</p> <p>Soundness/Evaluation: I think overall I am seeing the missing connection between the language and the particular optimization techniques. How do the language enable these optimizations in an easier way? Does it impact the query plans yielding a better match to particular optimization techniques? While the evaluations look at performance issues (and kudos to authors their optimization techniques do yield better results) but I think the real question is the connection between the language and optimization techniques. Otherwise these optimization techniques can equally easily be built in to current database optimization approaches, into pandas, etc.</p>
<p>If you recommended a Conditional Accept, describe specific issues you would like to see addressed in a revision.</p>	<p>more discussion on the connection between the language and specific optimization techniques more theory regarding what this does to relational algebra.</p>

Masked Reviewer ID: Assigned_Reviewer_7

Review:

Question	
Overall rating	Reject; I have serious concerns about this paper that cannot be addressed with a revision
Briefly summarize your review, and rationale for the chosen rating	This paper presents a case of applying AQuery, an extension of SQL that supports ordered relations as first-class objects, to data analytical queries. Given prior work on AQuery and other similar languages, I have serious concerns about the novelty of the paper.

List at least 3 strong points, numbered S1, S2, S3, ...	S1: The paper gives a thorough review of AQuery's syntax and semantics. S2: Some of the experiments show substantial performance improvement with AQuery compared to Pandas and MonetDB.
List at least 3 weak points, numbered W1, W2, W3, ...	W1: Unclear what is the paper's contribution given prior work on AQuery W2: Experiments do not fully discuss why the current paper's implementation of AQuery is better than the other storage engines used in comparison. W3: Lack of novelty in the translations presented in the paper.
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.	<p>This paper describes AQuery, a language that supports ordered relations. The paper includes a number of transformations that can be used to optimize AQuery queries, along with some experiments comparing data analytical queries expressed using AQuery and Pandas.</p> <p>My main concern about the paper is its lack of novelty. Query languages that support ordered relations have been proposed for a long time (starting with temporal databases, time-series databases, relations on ordered domains, etc), and both theoretical and systems work have been done in the past. While the current paper is not a "language" paper per se, it does spend a substantial number of pages (2.5 pages) describing the syntax and semantics of AQuery, which I find strange. In comparison, the AQuery transformations are only briefly described in about 1.5 pages, with some of them being obvious (e.g., 4.1 and 4.3, which seem similar to those proposed in the original AQuery paper in VLDB 2003), and details missing from the rest. For instance, is the "full strategy sketch" described in 4.5 a heuristic that is occasionally applied, or is that part of the AQuery execution engine?</p> <p>Also, while the AQuery implementation performs much better in comparison to Pandas, the authors did not explain why that is the case, and what are the sources of the performance improvement. Are they coming from prior optimization already proposed in AQuery, or are they due to the ones mentioned in the paper? The settings for the experiments were also somewhat strange in that they were run on a machine with very limited memory (8GB), which I don't think is the way that modern data analytics engines are hosted (especially for financial companies, where the examples are based from). How big was the dataset (in terms of disk size, not the number of rows), and what happens when the amount of memory available increases?</p>

Masked Reviewer ID: Assigned_Reviewer_8

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	<p>The paper reviews AQuery, an ordered SQL variant, proposes some optimization rules (e.g. reorder sorts vs. joins), and evaluates optimized AQuery versus a python-based system for ordered data called Panda on financial analysis workload.</p> <p>Applying relational-esque optimization to ordered data/query approaches that currently use non-relational technology e.g. python/Panda seems like a great way to extend the reach of our community's ideas. On the other hand, I'm wondering whether AQuery can be more closely unified with relational approaches, and what the tradeoffs would be -- it would be nice if the paper</p>

	addressed that at least with some discussion.
List at least 3 strong points, numbered S1, S2, S3, ...	<p>S1. Applying relational-esque optimization to ordered data/query approaches that currently use non-relational technology e.g. python/Panda seems like a great way to extend the reach of our community's ideas.</p> <p>S2. Amazing performance results vs. Panda, showing the huge gains that query optimization can offer.</p> <p>S3. Very well-written.</p>
List at least 3 weak points, numbered W1, W2, W3, ...	<p>W1. My main question is how well it would work to position AQuery as syntactic sugar on top of SQL, and performing the proposed optimization heuristics inside a modern relational optimizer. The approach to handling the "assuming" clause seems equivalent to the old "interesting orders" idea, applied to both virtual and materialized views, which makes me wonder how far apart the two approaches really are. I would like the paper to at least entertain this approach, and perhaps discuss whether the proposed optimizations would be realistic to achieve in a relational optimizer, or whether the patterns to be optimized would be too obfuscated for the optimizer to be able to exploit them.</p>
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.	<p>If this paper ultimately doesn't make the bar for SIGMOD, I would like to encourage the authors to keep pursuing this line of work on bringing relational-style optimizations to Panda etc. It would be great to bring Panda etc. "into the sigmod fold". Thanks for undertaking this project!</p> <p>Minor comments:</p> <ul style="list-style-type: none"> - please consider citing/discussing Ibis (an Impala extension) in related work
If you recommended a Conditional Accept, describe specific issues you would like to see addressed in a revision.	W1