The Logic of Query Languages for Data Streams

[Extended Abstract]

Carlo Zaniolo
Computer Science Department
University of California at Los Angeles
Los Angeles, California USA
zaniolo@cs.ucla.edu

Data Stream Management Systems (DSMS) represent a vibrant research area that is rich in technical challenges, which many projects have approached by extending database query languages and models for continuous queries on data streams [1, 3, 4, 9, 5]. These database-inspired approaches have delivered remarkable systems and applications, but have yet to produce solid conceptual foundations for DSMS data models and query languages—particularly if we compare with the extraordinary ones of relational databases. A cornerstone of the success of relational databases was the tight coupling between their data model and their logic-based query language. In this paper, we show that a similar approach can succeed for data streams and propose a tight-coupled design for DSMS data models and query languages. To express more naturally the behavior of a data stream and attain more powerful on-line queries, we abandon the set-of-tuples model of relational databases, and instead use sequences of tuples ordered by their time-stamps as our data stream model. This approach allows us to overcome the blocking problem that severely impairs the expressive power of data stream query languages. As elucidated in [1]: A blocking query operator is one that cannot produce the first tuple of the output until it has seen the entire input. Previous work had characterized blocking query operators by their non-monotonic behavior [7, 6, 8]. In this paper, we instead use the closed-world assumption [11, 10] to characterize blocking/nonblocking behaviors with respect to the incompleteness/completeness of the streaming database. From this, we infer simple syntactic conditions that make Datalog rules immune from blocking. A significant and surprising new result is that the use of negated goals in the bodies of rules does not imply a blocking behavior: in fact, many very useful nonblocking queries can be expressed using negation. The flip side of this exciting result is that additional conditions must then be imposed on the rules to ensure that (i) the results produced by Datalog programs are ordered according to their time-stamps, and (ii) possible time-skews between streams are also managed explicitly by the rules [2]. These problems, and their possible remedies, are captured and expressed quite naturally using Datalog, which thus emerges as a powerful framework for analyzing and expressing continuous queries. Related problems, including the treatment of data streams without time-stamps, the characterization of monotonic query operators [7, 6, 8], and the use of more general closed-world assumptions were also studied and answered in the course of this research [12].

1. REFERENCES


Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

LID 2011 March 25, 2011, Uppsala, Sweden
Copyright 2011 ACM 978-1-4503-0609-6 ...$10.00.