Expanding Continuous Queries Scope for Better Event Detection

Shenoda Guirguis Panayiotis Neophytou Panos Chrysanthis
Alexandros Labrinids Kirk Pruhs
{shenoda, panickos, panos, labrinid, kirk}@cs.pitt.edu

1 Motivation

Data Stream Management Systems (DSMS) were developed to be at the heart of every monitoring application (from environmental monitoring, to patient care and monitoring for outbreaks of diseases, to financial market monitoring, to monitoring for interesting cosmic phenomena). DSMSs are designed to handle with ease large volumes of data and large volumes of user queries (i.e., exhibit scalability), while at the same time providing fast response times. Essentially, DSMSs deviate completely from the store-then-query paradigm of traditional database management systems, since, in a DSMS, it is the queries that are stored (or registered) ahead of time, while the data is coming in at high rates (i.e., without the need for storing). That is, in DSMSs, monitoring applications register Continuous Queries (CQs) which continuously process unbounded data streams looking for data that represent events of interest to the end-user.

A key assumption in current DSMSs (such as STREAM [3], Borealis [1], Aurora [2], Streambase [13] and Coral8[4]) is that data is precise. However, in many applications, this assumption is not valid. For example, temperature readings from wireless sensors in an environmental monitoring application cannot be always assumed to be accurate. Due to communication noise and loss, an average temperature readings of $99\,^{\circ}\mathrm{F}$ is not 100% accurate. Given that such a value is fuzzy, if a user wants to be alerted of the event when the average temperature rises above, say, $100\,^{\circ}\mathrm{F}$ (as indication of potential fire), the user should be alerted of the event even when the average temperature rises above $99\,^{\circ}\mathrm{F}$. This is to be distinguished from approximate query answering [7]. Approximate answering is concerned with providing approximate answers to queries over precise data for performance and efficiency reasons by utilizing sampling techniques, while in our case, we are concerned with giving the most accurate answer, given fuzzy data.

Although probabilistic models have been proposed to handle uncertain data [5, 6], they cannot handle data that is fuzzy by nature. Consider a climate system inside a building which receives people's requests for setting the temperature and attempts to adjust to a specific temperature in order to make them comfortable. Inherently the people's opinion is a fuzzy timeseries/datastream since every user has a different interpretation of words like "warm", "hot", "cold", and "freezing". The system should include an operator that is able to weigh these opinions and map them into temperatures or temperature ranges, and finally decide whether to increase or decrease the temperature. Therefore, a fuzzy data model that utilizes fuzzy logic and fuzzy sets theory [8, 9] is needed.

Another implicit assumption in current DSMSs is that users have perfect knowledge of data; that is users know exactly how to express their queries in terms of the data readings, which is not always the case. For example, consider a weather application where environmental data streams tuples are of the schema: (temperature, wind speed, pressure, icy). Assume a user is interested to be notified with dangerous road conditions. Typically, the user would prefer to register a CQ on the form: "Alert on dangerous driving conditions", rather than precisely providing the values of different attributes (temperature, wind speed, etc) that define safe/dangerous conditions. Thus, there is a need for a Fuzzy Continuous Query language (Fuzzy-CQL) to enable users to express their vague queries, or Fuzzy Continuous Queries (Fuzzy-CQs).

Given the Fuzzy-data, and the Fuzzy-CQs, a new fuzzy processing model for data streams is needed. We prognosticate the fuzzy processing scheme should follow or build upon the imprecise queries' answering

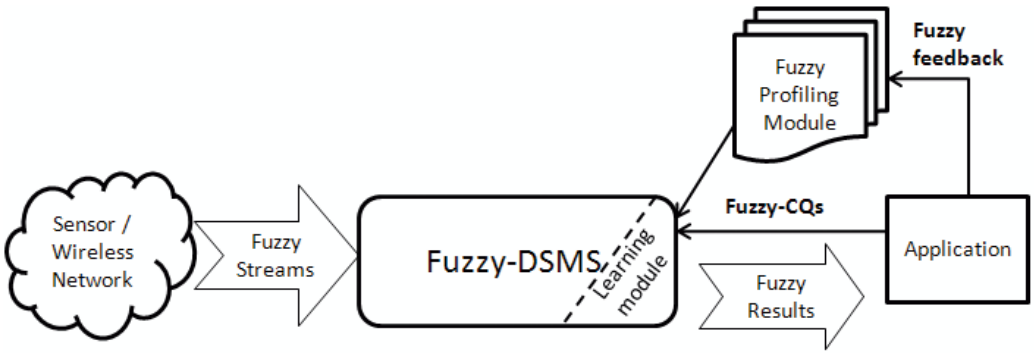

Figure 1: Overview of the Fuzzy Environment

techniques [10]. However, two modifications are mandatory: 1) handling fuzzy logic, and 2) learning from the user feedback in order to improve the quality of the returned fuzzy results.

2 F-DSMS Model

As discussed above, there is a need for a new Fuzzy Data Streams approach to model imprecise data and vague querying. Such a new approach will introduce a fuzzy data model, as well as a fuzzy continuous query language for data streams, which necessitates a fuzzy continuous query-processing scheme. An example of a fuzzy data streams environment is illustrated in Figure 1, in the core of which is the Fuzzy-DSMS (F-DSMS). In addition to the Quality of Service (QoS) and Quality of Data (QoD) performance metrics in existing DSMSs [11, 12], this model must also introduce new metrics, which we call: Quality of Results (QoR). QoR refers to the degree of user satisfaction with the fuzzy results. In the current non-fuzzy continuous query language, the user gets results that she knows or expects, while in our proposed fuzzy continuous query language, users who are not certain what they need exactly, can get useful results of their interest. This is expected to dramatically increase the number of potential users and the number of fuzzy continuous queries they register. This in turn adds to the criticality of performance tuning and scalability of the F-DSMS.

The Fuzzy Data Streams Model illustrated in Figure 1 shows three levels of fuzziness. First level is the data collection level, where the input streams are fuzzy. Second level of fuzziness is the processing level in the F-DSMS itself. Given fuzzy input streams, and fuzzy Continuous Queries (fuzzy-CQs), a mechanism for processing them is needed. Fuzzy-CQs processing is should be designed to optimize for the QoR new performance metric, as well as the traditional QoD and QoS metrics. The design of the scheduler, query optimizer and load shedder need to be revisited accordingly.

Finally, the third level of fuzziness is in the application level, where users use a fuzzy CQ language to query a fuzzy data collection. Thus, there is a need for a standard fuzzy-CQL. Moreover, many applications now utilize a profiling module to best serve the users. Given the fuzzy paradigm, it makes better sense to allow users to express their general interest (profiles) and their feedback in a vague (i.e., fuzzy) way. Such fuzzy profiling modules along with users' fuzzy feedback should be fed to the learning module of the F-DSMS to further fine tune the performance.

3 Research Challenges

To develop an effective fuzzy environment, three key challenges must be addressed, one for each level of fuzziness: 1) handling uncertain and fuzzy data streams, 2) handling fuzzy (vague) queries, and 3) maintaining the fuzzy semantics (i.e., the learning module). In the first challenge, a new fuzzy data streams model is needed to allow the system to handle the fuzziness in the three aforementioned levels. Another

challenge in this level is to design a model that can efficiently model fuzziness, while maintaining the main characteristics of data streams, such as in-memory processing.

For the second challenge, a new CQ processing scheme is needed to process Fuzzy-CQs. The new fuzzy processing scheme would optimize for the QoR, in addition to the QoS and QoD. This involves three challenges: 1) how to implement and process such Fuzzy-CQs, 2) how this affects the modules in the F-DSMS, such as the query optimizer, scheduler, load shedder and dissemination modules, and 3) a fuzzy-CQ language is needed. In traditional databases, two fuzzy database query languages were proposed based on fuzzy calculus and fuzzy algebra [14], as an extension to the relational calculus and algebra respectively. The fuzzy calculus and fuzzy algebra were proved to be relationally complete. One potential direction is to extend the fuzzy algebra to incorporate streams querying primitives such as sliding windows.

Finally, in the third challenge, we are to consider practical techniques to maintain both data streams processing semantics as well as application specific fuzzy-semantics, through a learning mechanism. The challenge here is to achieve that in an efficient way and automatically, i.e., human free to avoid the population biasness problem. More specifically, the fuzzy CQ language would allow linguistic terms, to express vagueness, such as ”more or less”, ”almost equal”, etc. How does a user interpret these words differ from one user to another. The F-DSMS should learn from the application's feedback and adjust the processing accordingly. In addition to the application's feedback, the F-DSMS should utilize profiling modules, in which a profile is maintained for each user. This helps the system to learn the user-specific semantics of the vague linguistic terms, which will subsequently help to further fine tune the QoR.

References

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. C etintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. B. Zdonik. The design of the borealis stream processing engine. In *CIDR*, 2005.

[2] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. *VLDB Journal*, 2003.

[3] Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Itaru Nishizawa, Justin Rosenstein, and Jennifer Widom. Stream: The stanford stream data manager (demonstration description). In *SIGMOD*, 2003.

[4] http://www.coral8.com/, 2004.

[5] Nilesh Dalvi and Dan Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, 2007.

[6] Amol Deshpande, Carlos Guestrin, and Samuel R. Madden. Using probabilistic models for data management in acquisitional environments. In *CIDR*, 2005.

[7] Chris Jermaine, Subramanian Arumugam, Abhijit Pol, and Alin Dobra. Scalable approximate query processing with the dbo engine. *ACM Trans. Database Syst.*, 33(4), 2008.

[8] Zadeh L.A. Fuzzy sets. *Inform. Cont.*, 8(3):338–353, 1965.

[9] Zadeh L.A. Fuzzy set theoretic interpretation of linguistic hedges. *J. Cyb.*, 1972.

[10] Ullas Nambiar and Subbarao Kambhampati. Answering imprecise queries over web databases. In *VLDB*, 2005.

[11] Mohamed A. Sharaf, Panos K. Chrysanthis, Alexandros Labrinidis, and Kirk Pruhs. Efficient scheduling of heterogeneous continuous queries. In *VLDB*, 2006.

[12] Mohamed A. Sharaf, Panos K. Chrysanthis, Alexandros Labrinidis, and Kirk Pruhs. Algorithms and metrics for processing multiple heterogeneous continuous queries. *TODS*, 2008.

[13] http://www.streambase.com, 2006.

[14] Y. Takahashi. Fuzzy database query languages and their relational completeness theorem. *TKDE*, pages 122–125, 1993.