Sensor Data Exploration: Making Stream Data Management Systems More Usable in Monitoring Applications

Shengliang Xu
Computer Science & Engineering
University of Washington, Seattle
slxu@cs.washington.edu

ABSTRACT

This paper is concerned of making the sensor stream data management systems more usable in monitoring applications for average users. The advances of sensor technologies have promoted a large number of monitoring applications. Stream data management systems (SDBMS) provide sophisticated tooles for the users to organize the monitoring data and set up complex event monitors. Although very powerful, the SDBMSes generally require the users to be both 1) experts of their query interfaces, 2) experts in their domain so that they know what to query, and 3) be able to precisely describe each monitoring task in every detail since the query languages are all rigorous. All the requirements are too restrictive for average users and prevent SDBMS systems from being usable. This paper discusses a sensor stream data exploration mechnism which is aimed at lowering the high usage barriers of SDBMSes in monitoring applications for average users. Specifically, a data cube based exploration model is proposed, and two new operations, i.e. clustering and model-based slicing are designed in addition to the traditional data cube operations. Preliminary experiments using a hydro dataset show that the proposed exploration mechnism can effectively help the users identify interesting patterns.

1. INTRODUCTION

The vast advances of sensor technologies have promoted a large number of sensing applications in various areas, e.g. supply-chain management [22], environment monitoring, elder care [7] and activity recognition [9], etc.

One dominant task in sensing applications is to monitor whether any abnormal events happen. To identify abnormal events directly in the raw sensing streams is impossible for average users. In elder health care monitoring systems (hereafter we use this as a running example), there typically would not be direct sensors due to privacy considerations, such as video cameras or audio recorders. Mostly, the elders are monitored by indirect sensors like motion sensors [7], electricity usage sensors [11] or water usage sensors [10]. Therefore, the nurses cannot directly observe the conditions of the elders or patients. To find out abnormal events manually from a pool of indirect sensing data is impossible.

A great number of stream database management systems (SDBMS) have emerged to help the users efficiently and effectively manage stream data. In general, these systems will hide the underlying messy of raw sensor data and provide a much easy-to-use querying interface. Some of them view the stream data as relational and support SQL-like query languages [4] for data processing. Some of them treat each sensor data item as an elementary event and provide pattern based languages[23] [17] [18] or tree style languages [20] to find complex combinations of events.

With no doubt, all the SDBMS systems have dramatically reduced the pain of managing the stream data for the users. However, is a data management system and a querying interface enough for the average users? For us, the answer is definitely NO. In general all the SDBMSes require the users to:

- master the query interfaces. It's simply because the query interfaces are the only way to direct the actions of SDBMSes. However even if the users are experts in the querying interfaces, it is still not enough to easily set up a monitoring task. Moreover, they need to
- be domain experts. The SDBMS systems are only data management tools. For each specific application, the users are responsible for telling the SDBMS systems what to query. And now the question is are the above two experties all the SDBMSes usage requirements are about? Not yet. Furthermore, the users need to
- be able to describe every monitoring task exactly precise. Because of the rigorousness of the query languages, if the users have any vagueness in the monitoring task understanding and describing, they cannot construct a correct query.

The preceding three requirements are all too restrictive for average users. The query interface barrier is a common usage problem in all database systems [13]. There have been several research efforts try to lower this barrier, including [16] [8] etc. In this paper, we try to focus on the latter two barriers. In the following we identify three more concrete usage barriers that can be ascribed to one of the above latter two general barriers but specific to monitoring applications.

Hereafter, an elder care monitoring application is used as a running example for clear statement.

1. *Unawared Monitoring Patterns.* No matter how profound in the expertise of the monitoring application area, a user may miss some important monitoring patterns. As an example, a nurse may have set up a bunch of monitors on an elder for monitoring the motion, the diet, and the midnight actions. But he/she may be unaware that the frequency of shower is also a good indicator for sickness. While in a SDBMS, there is no way for the users to realize this useful monitoring pattern.

2. *Task/Subject Specific Information.* A user may be quite familar with SDBMS query interface, but many monitoring queries need subject specific information, so that the user cannot construct a query without enough apriori information of the task/subject. For example, a nurse may want to monitor an elder of whether he/she gets up too frequently at midnight. A SQL-like query may be like:

   ALERT WHEN (SELECT COUNT(*) FROM Whole_House_Activities AS H WHERE H.timestamp within time_window_midnight) >=THRESHOLD

   where time_window_midnight is a time window describing midnight. The question here is what is the meaning of "frequently", i.e. what should be the value of THRESHOLD in the query? Different people may have totally different living habits at night. It's very hard for the nurses to set a threshold to classify the midnights into frequent and infrequent without enough familarity with the monitored elder.

3. *Vague Information Need.* In this case, a user may be aware that some conditions may be valuable for monitoring, but he/she may not be able to clearly give an exact description. For example, a nurse may be aware that the midnight actions of an elder is valueable for helth condition monitoring, but he/she may fail to clearly identify what actions to monitor, should it be the overall frequency of any actions or only the toilet actions.

In view of the above usage barriers, we claim that if the users can explore historical stream data convieneantly, the pain that an average user may suffer can be dramatically mitigated. For example, if a nurse can explore the historical records of an elder's midnight events, he/she can define the normal midnight pattern of that elder quite accurately.

Specifically, in this paper we propose a stream data exploration system. The system is able to organize historical monitoring data into a bunch of data cubes. Traditional data cube operations such as drill down, roll up and slice as well as two new operations, i.e. data cube clustering and model-based slicing, can be carried out on the data cubes. The data model and the operations together form an effective exploration mechnism for the users in assisting them identifing interesting patterns, task specific information, etc.

As for experimentation, we use a HydroSense dataset [10], which is a stream of water event records on different fixtures, such as shower, bath faucet, kitchen faucet, etc. collected in several houses. Preliminary experimental results show that our data exploration mechnism is able to help the users identify interesting patterns.

The rest of this paper is organized as follows. Section 2 lists some prior studies related to our work. In Section 3, the data exploration mechnism is discussed in detail. In Section 4 we report the experiment results. Finally, we conclude our work and list some future work in Section 5.

2. RELATED WORK

Research on stream databases has been investigated quite thoroughly. A lot of systems have been proposed to process stream data. Some representatives include [2] [5] [1] [23], etc. These systems focus on improving stream processing ability from various perspectives. Although these researches do not relate to our work very much, they provide the base of our work. In other words, without them, our work will be meaningless.

In recent years, a trend of making the DBMS usable has attracted a large number of audiences. Although there exists a standard and user-friendly interface (at least comparing to computer programming languages) across all DBMSes, i.e. SQL, it still requires a lot of efforts for an ordinary user to translate her/his information needs into a correct SQL expression. Because of this, almost in every company or organization, there will be an army of database administrators, consultants, and other technical experts busily helping users get data into and out of a database [13]. As a pioneering work, Jagadish et al. in [13] present six challenges in using databases today. QueRIE [8] analyzes a user's query log, finds other users who have executed queries over similar parts of the database, and recommends new queries to retrieve relevant data. Nodira et. al. in [16] proposes a similar process to recommend query snippets to the users. Generally, these work rely on a large amount of query logs to make personalized query recommendation. In our stream DB systems, the situation is quite different. The main problem lies in that typically stream DBMS queries are predefined continuous queries [5]. In general, a user will define a set of queries at the very beginning and then leave them running in the system for a long time. Therefore, these systems usually cannot collect a large number of query histories.

Although typical stream DB applications may be monitoring events online and discarding or archiving stream data [2] [23], historical stream data warehousing systems have shown to be useful in many aspects. Jiawei et.al. in [12] proposed an architecture to calculate stream cubes online. Eric et. al. in [19] proposed a Sequence OLAP model. They extended traditional data cube by adding pattern-based sequence summarization. And then they designed a SQL-like language for the users to build sequence data cubes. The Moirae system [6] found out that the historical information can be very useful in improving the fine-grained clustering of events. Many other systems have been applied to monitoring applications such as web complexes [3], highway traffic [21], and wide-scale networks [15]. These researches do not relate to our work because they do not try to make use of the historical stream data for improving stream DB system usability.

3. SENSOR STREAM DATA EXPLORATION

3.1 Problem Analysis

The first step to solve a problem is to understand the problem thoroughly. In this section, we present the analysis of the monitoring applications. In general, three properties are identified as listed bellow:

- *Repeated Events.* Since our goal is to help users identify potential interesting abnormal events from normal events, we implicitly aim at those events that happen repeatedly, otherwise, it is meaningless to classify an event as abnormal or normal.
- *Data Mining and Analysis.* Different from the monitoring task of SDBMSes, our task is more like a data mining and analysis one. For example, the identification of the abnormal events can be scribed as a rare pattern mining task. This property is to play an important role in data model selection for our system.
- *Clustering Property.* Generally speaking, the normal events should share some similarity between each other. In opposite, the abnormal events should be more different from the normal events. Therefore, we consider the events have a clustering property, i.e. under an appropriate similarity measure, the normal events most likely tend to get clustered together and the abnormal events do not. But note that, the normal events may not get clustered into a single cluster. They may tend to group into several seperate clusters. For example, the normal kitchen usage patterns in weekdays may be very different from those in weekends.

3.2 Data cube as data model

Before any data exploration operation can be defined and carried out, a data model should be set up. In our system, we make two assumptions of the underlying data.

1. *Raw Monitoring Stream Data Model.* The raw monitoring stream data is the input data that our system is to be built on. We assume that the raw stream data is a stream of tuples. Each tuple denotes an elementary event with two general fields, timestamp and state. Timestamps record when the elementary event happened. States are a set of property fields which describe the details of the elementary events.
2. *Exploration Data Model.* The exploration data model is the core data model that the users are to operate on. We require this model to be simple so that the users can easily understand and also suitable for data analysis. The data models in SDBMS systems such as relational and event streams are flexible for monitoring tasks but are too complex or restrictive for data analysis tasks. In view of this, we find that the OLAP task in traditional DBMS is very similar to our goal, as Jim et. al. pointed out in [14], "Data analysis applications look for unusual patterns in data. They categorize data values and trends, extract statistical information, and then contrast one category with another.". Therefore, we borrow the data cube model as the data model in our exploration system.

   Specifically, we model a data stream as a huge data cube with the time dimension going to infinite as illustrated in Figure 1.

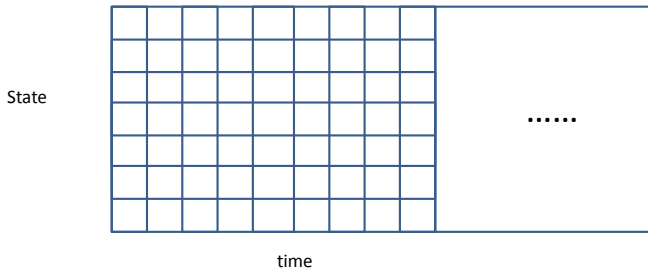


Figure 1: Stream Cube

3.2.1 Pattern based sub-cube construction

The infinite time dimension is not operable, we need a mechnism to allow the users break the huge data cube into smaller bounded operational units. Recall that the goal of our task is to help users in identifying interesting abnormal events among repeated events. Therefore, the smallest operational units of our system should be these events.

> **Definition** *Event Sub Cube.* An event sub cube is a sub cube of the stream infinite cube representing an event.

Here "event" is a more general meaning than a typical concrete event in any other monitoring systems. For example, an event in the elder care applications can be the group of actions fall into the midnight time window. In this scenario, a normal event may be the number of actions within midnight less than or equal to 10 and an abnormal event may be the number of actions more than 10.

To implement the event sub cube construction, we propose a simple patten based window language. The main part of the syntax is as the follows:

$$
\begin{array}{rcl}
PatternWindow & := & Begin\ Predicate;\ End\ Predicate \\
Predicate & := & Predicate\ and\ Predicate \\
 & & \mid Predicate\ or\ Predicate \\
 & & \mid not\ Predicate \\
 & & \mid (Predicate) \mid Atom \\
Atom & := & Property\ op\ C \\
 & & \mid Property\ op\ Property \\
 & & \mid C\ op\ Property \\
op & := & < \mid > \mid >= \\
 & & \mid <= \mid <> \mid == \\
Property & := & state\ property \mid timestamp \\
 & & \mid function(Property)
\end{array}
$$

For example, a pattern window for midnight (from 10:00 PM to 4:00 AM) can be defined as:

$$Begin\ hour(timestamp) >= 22;\ End\ hour(timestamp) <= 4$$

3.2.2 Dimension Hierarchy

As in OLAP systems, we also allow the users define hierarchies on the properties of the input stream, including both the timestamp and the state properties. For example, in elder care monitoring applications, the monitoring of hydro events requires differnt sensors in different rooms, for example a sensor for water events of the toilet, a sensor for

Table 1: A sample event tuple stream

timestamp	fixture	interval	fixture type
2010-02-01 08:09:59.000	Bathroom Sink	26.45097	SingleHandleFaucet
2010-02-01 08:20:38.000	Bathroom Sink	25.30426	SingleHandleFaucet
2010-02-01 08:27:25.000	Kitchen Sink	9.074509	SingleHandleFaucet
2010-02-01 20:11:19.000	Kitchen Sink	7.922266	SingleHandleFaucet
2010-02-01 21:50:05.000	Toilet	27.67681	Fixture
2010-02-01 21:50:13.000	Bathroom Sink	22.66671	SingleHandleFaucet
2010-02-01 21:58:01.000	Shower	927.2068	SingleHandleFaucet

water events of the shower. All the sensors can be organized into rooms and the rooms can be further organized into the house. Therefore, the house → rooms → sensors is a three-layer hierarchy. And the same to the timestamp. Figure 2 illustrates two examples.

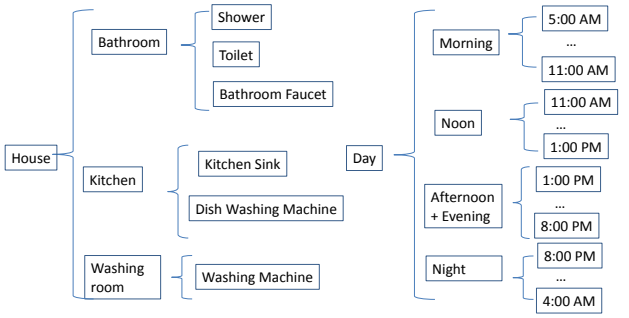


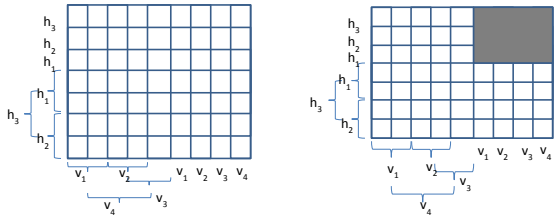
Figure 2: Dimension Hierarchy

3.3 Operations on sub-data cubes

Since the sub-data cubes are all data cubes, all the operations on traditional data cube OLAP can be directly applied here, including roll up, drill down, slice, etc. We do not discuss them in detail here. The traditional cube operations all perform on a single data cube. In our goal of data exploration, these operations are not enough. Specifically, we define two new operations, clustering of event sub cubes and model-based slicing of event sub cubes.

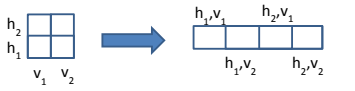
3.3.1 Distance calculation between sub-data cubes

The two new operations both require a distance measure between sub cubes. A cube is essentially a tensor. The dimensions are orthogonal between each other. And within each dimension, the different cells are also independent to each other. Therefore, it won't lose infomation by vectorizing a cube into a vector, as illustrated in Figure 4(a). By this simple transition, all the distance measures in L^p space can be directly applied. However in our work, for each dimension, there may be an associated hierarchy structure. This information is semantically rich and should be kept in cube distance calculation. One way to incoporate the hierarchy information into calculation is to add the non-leaf nodes of a hierarchy structure of a dimension into that dimension to form a larger expanded cube, as illustrated in Figure 3(a). But in this way, in the worst case, there may be $(2^m - 1) \times (\textit{original cube size})$ new cells, where m is the number of dimensions. Therefore the new cells will dominate the cube distance calculation. To mitigate this problem, we further restrict that only the cells that contain at most one hierarchy dimension should be considered. The cells with more than one dimensions of hierarchy are dropped. In this way, the will be at most $(m-1) \times (\textit{original cube size})$ new cells. Figure 3(b) shows the reduced cube dimension expansion. And Figure 4(b) shows the vectorization of reduced dimension expanded cubes.

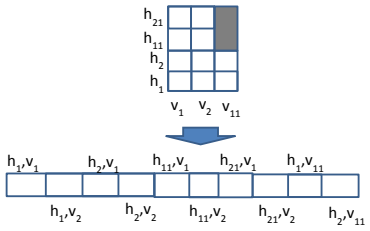


(a) Cube dimension expansion (b) Reduced cube dimension expansion

Figure 3: Dimension expansion of the data cubes for distance calculation



(a) Cube vectorization



(b) Reduced dimension expanded cube vectorization

Figure 4: Cube vectorization for distance calculation

3.3.2 Clustering

The preceding definitions of cube vectorization and distance calculation open the door to the fantastic world of linear algebra based data mining and analysis. Recall the cluster property of our task, a clustering operation of the event sub cubes can be of helpful in identifying potential abnormal events. Based on the preceding cube distance definition, the clustering operaiton can be easily implemented. And all the standard clustering methods, such as Kmeans

or Hierarchical clustering can be applied here without any problem.

3.3.3 Model-based slicing

The other new operation is the model-based slicing. As in the clustering operation, the model-based slicing operation allows the users to firstly select one or more event data cubes as the seeds; and then the system will automatically rank all the event data cubes according to the distance to the seed cubes; and finally let the users to select a bunch of similar sub data cubes and drop all the rest.

4. EXPERIMENTATION

We use the HydroSense data [10] for experimentation, which are water event records on different fixtures, such as shower, bath faucet, kitchen faucet, etc. collected in several houses. In the evaluation, we focus on testing the newly added two operations.

4.1 Clustering

In evaluating the clustering operation, we choose Kmeans as the clustering algorithm and set the number of clusters as 8. Figure 5 shows the clustering result. The x-axis is a list of fixtures with in the house which the data are collected, the y-axis represents 24 hours, and the z-axis shows the number of actions performed on a fixture at a moment. From the graph, it is clear that the different clusters present quite different distribuion properties. This is exactly what we expect. The different clusters denote different patterns of events. What we provide is a general tool for the users to help them organize the data.

4.2 Model based Slicing

For model based slicing, we present two experimental results. Figure 6 presents the 1-seed ranking of the cubes. And Figure 7 presents the 2-seed ranking of the cubes. The two experiments both demonstrate the soundness of this operation.

5. CONCLUSION AND FUTURE WORK

Making the data management systems more usable is a hot research topic in recent years. A lot of research efforts have been put into the sub area of making traditional DBMS systems usable. As for stream data management systems, there is not yet a research paper working on this direction. The main contributions of this paper include 1) identifying the difficulities in SDBMSes usage, specifically in monitoring applications; 2) the proposal of the sensor stream data browsing mechnism to assist the users managing stream data more easier.

As we have pointed out in the introduction section that there are generally three usage barriers. This paper focuses only on the latter two. However, the first barrier, i.e. the query language barrier, is also a famous bottleneck in usability. As a future work, we will work on this problem too. Generally, we will extend our system to incoporate an automatic query generation phase. In addition, this paper only discusses one data mining technique, i.e. clustering, actually our distance model can be applied to many other data mining techniques such as classification. As a second future work, we will investigate some more data mining methods to aid the monitoring application users.

6. REFERENCES

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, M. Cherniack, J. hyon Hwang, W. Lindner, A. S. Maskey, E. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik. The design of the borealis stream processing engine. In *In CIDR*, pages 277–289, 2005.

[2] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12:120–139, August 2003.

[3] M. Ahuja, C. C. Chen, R. Gottapu, J. Hallmann, W. Hasan, R. Johnson, M. Kozyrczak, R. Pabbati, N. Pandit, S. Pokuri, and K. Uppala. Peta-scale data warehousing at yahoo! In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 855–862, New York, NY, USA, 2009. ACM.

[4] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15:121–142, June 2006.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, pages 1–16, 2002.

[6] M. Balazinska, Y. Kwon, N. Kuchta, and D. Lee. Moirae: History-enhanced monitoring. In *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR)*, 2007.

[7] T. Barger, D. Brown, and M. Alwan. Health-status monitoring through analysis of behavioral patterns. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35:22 – 27, 2005.

[8] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM 2009, pages 3–18, 2009.

[9] T. Choudhury, M. Philipose, D. Wyatt, and J. Lester. Towards activity databases: Using sensors and statistical models to summarize peopleâĂŹs lives. *IEEE Data Eng. Bull*, 29:2006, 2006.

[10] J. E. Froehlich, E. Larson, T. Campbell, C. Haggerty, J. Fogarty, and S. N. Patel. Hydrosense: infrastructure-mediated single-point sensing of whole-home water activity. In *Proceedings of the 11th international conference on Ubiquitous computing*, Ubicomp '09, pages 235–244, 2009.

[11] S. Gupta, M. S. Reynolds, and S. N. Patel. Electrisense: single-point sensing using emi for electrical event detection and classification in the home. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 139–148, 2010.

[12] J. Han, Y. Chen, G. Dong, J. Pei, B. W. Wah, J. Wang, and Y. D. Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distrib. Parallel Databases*, 18:173–197, September 2005.

[13] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proceedings of the 2007*

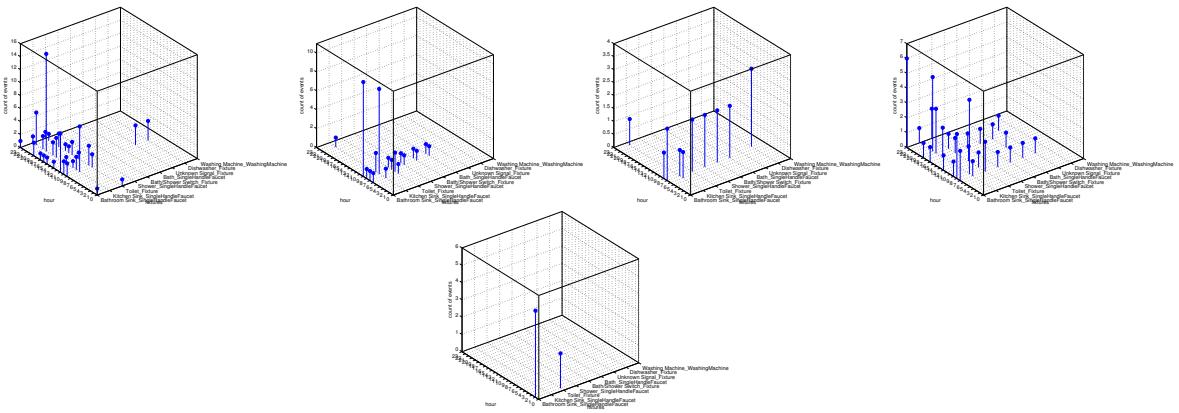
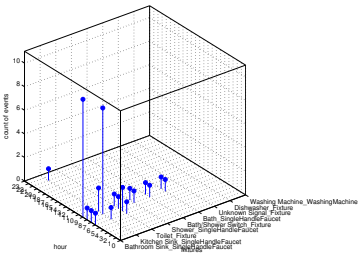
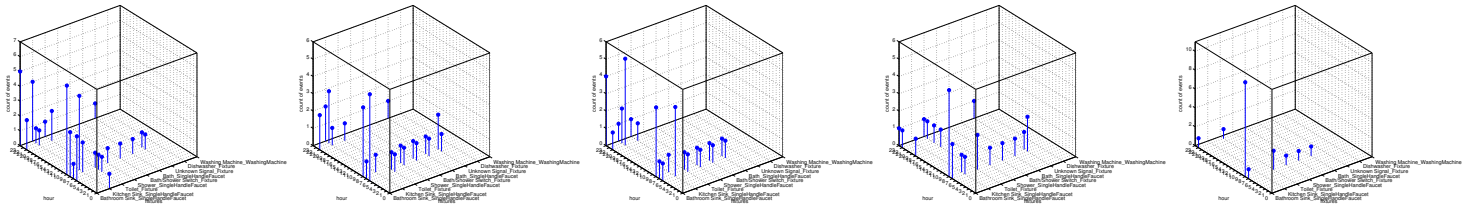


Figure 5: A sample clustering results for the HydroSense dataset

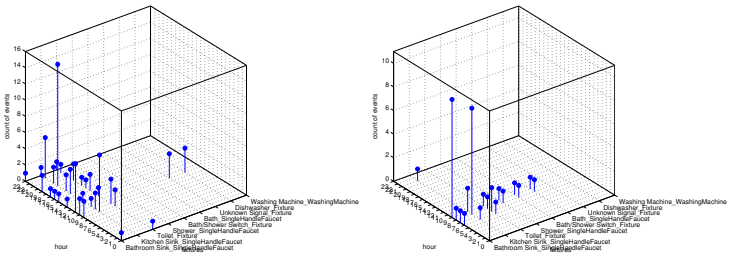


(a) Seed Cube

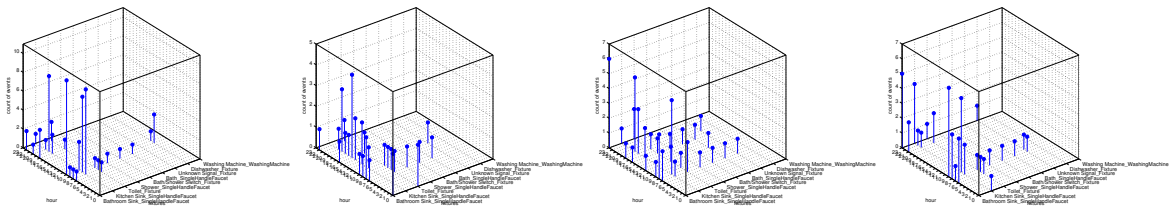


(b) Top 5 rankded cubes

Figure 6: Model based slicing result 1. The first cube is served as a seed and the top 5 most similar cubes are presented



(a) Two seed cubes



(b) top four ranked cubes

Figure 7: Model based slicing result 2. The first and the second cube are served as two seeds

ACM SIGMOD international conference on Management of data, SIGMOD '07, pages 13–24, 2007.

[14] A. B. A. L. D. R. M. V. F. P. Jim Gray, Surajit Chaudhuri and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1:29–53.

[15] C. P. D. S. J. v. d. M. J. A. J. Kalmanek, C.R.; Ihui Ge; Seungjoon Lee; Lund. Darkstar: Using exploratory data mining to raise the bar on network reliability and performance. In *Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop on*, pages 1 – 10, 2009.

[16] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. Snipsuggest: Context-aware autocompletion for sql. *Proceedings of the VLDB Endowment*, 4, 2011.

[17] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Challenges for event queries over markovian streams. *IEEE Internet Computing*, 12:30–36, November 2008.

[18] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Lahar demonstration: warehousing markovian streams. *Proc. VLDB Endow.*, 2:1610–1613, August 2009.

[19] E. Lo, B. Kao, W.-S. Ho, S. D. Lee, C. K. Chui, and D. W. Cheung. Olap on sequence data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 649–660, New York, NY, USA, 2008. ACM.

[20] Y. Mei and S. Madden. Zstream: a cost-based query processor for adaptively detecting composite events. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 193–206, New York, NY, USA, 2009. ACM.

[21] K. Tufte, J. Li, D. Maier, V. Papadimos, R. L. Bertini, and J. Rucker. Travel time estimation using niagarast and latte. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 1091–1093, New York, NY, USA, 2007. ACM.

[22] F. Wang and P. Liu. Temporal management of rfid data. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 1128–1139. VLDB Endowment, 2005.

[23] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 407–418, New York, NY, USA, 2006. ACM.