EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article ICST.ORG

Applying Log Data to Context-Awareness in Home Network System

Yuichi Watanabe[1], Shinsuke Matsumoto[1], Sachio Saiki[1] Masahide Nakamura[1]

[1]Kobe University, 1-1 Rokkodai, Nada, Kobe, Hyogo 657-8501, Japan

Abstract

In the conventional context-aware services of the home network system (HNS), every context has been defined by current (or recent) situations only. Considering *past situations* in a house would significantly extend the expressive power of the context-aware services. In this paper, we propose a new type of context, called *log context*, by using *house log* of the HNS, extensively. The log context is defined with both the current and past situations, where the current situation is obtained by sensors or device status of the HNS while the past situations are derived by queries to the house log. We also develop a system that can derive the log contexts within an actual HNS. To manage individual log contexts efficiently, the system is designed by four layers: application layer, log context layer, log query layer, and DB connector layer. To evaluate the execution performance of our system, we have conducted an experiment which measures execution time of some variety of log queries.

Keywords: home network system, context-aware service, house log

Received on 11 December 2014; accepted on 17 January 2015; published on 12 March 2015

doi:10.4108/casa.2.3.e3

1. Introduction

Research and development of *Home Network Systems (HNS)* (also called *smart home*) have gathered great attention [1][2][3][4]. HNS provides value-added services for home users by connecting sensors (e.g., temperature, humidity, brightness, etc.) and household appliances (e.g., TVs, air-conditioners, lights, etc.) to the home network. We are developing an actual HNS, called *CS27-HNS* [5]. Although there are many types of services in HNS, *context-aware services* are considered to be most valuable but challenging services. In HNS, the context-aware services automatically control the appliances (as desired by home users), based on situational contexts characterized by the sensors.

In the conventional context-aware services, every context was usually defined by the *current* (or *recent*) values of the sensors [6][7][8][9][10]. For example, context *"it is cold"* can be defined by *"the value of the temperature sensor is less than 8 degree"*. To efficiently manage such context-aware services within CS27-HNS, we developed a framework called *sensor service framework (SSF)* [11].

As we developed various context-aware services with SSF, we have found that contexts with the current values only are insufficient to meet individual requirements of home users. For example, compared to *"it is cold"*, contexts like *"it is colder than yesterday"*, or *"it is the coldest day for three years"* are much more informative for a user to make a decision. To define such contexts, we need to consider *past situations*, i.e., the past values of sensors. However, they are beyond the scope of the conventional framework.

Our goal here is to extend the expressive power of the conventional methods, so that every context can be defined with both the current and past situations. Here the term "past" is not limited to "recent", which can vary from minutes to years. As for similar but different approaches, there exist studies using *context history* [12][13][14][15]. The context history is a database which records time-series satisfactions of contexts. It is used to predict user preference or to recommend services. However, the context history is basically the record of the past contexts, but not to define new contexts using

*Corresponding author. Email: nabe@ws.cs, masuda@ws.cs, shinsuke@cs, sachio@carp, masa-n@cs.kobe-u.ac.jp

the past situations. Thus, the history can say *"it was cold, yesterday"*, but does not say *"it is colder than yesterday"*.

To achieve our goal, we extensively use *house log* in this paper. The house log is a history of data acquired from the sensors and the appliances of HNS [16]. Typical data include sensor values, appliance status, power consumption, etc. Here we use the house log to retrieve past situations in HNS. For example, *yesterday's temperature* can be obtained by a *query* to the house log. Comparing this with the current temperature evaluates the context *"it is colder than yesterday"*. We call such a context defined with queries to the house log, *log context*.

In this paper, we first present a method to define the log contexts. The original context was defined by an expression over the current sensor values. For the log context, we extend the expression so that it can contain *log queries*, which are queries to the house log. We then develop a system that can define and derive actual log contexts within the CS27-HNS environment. To manage individual log contexts efficiently, the system is designed by four layers: application layer, log context layer, log query layer and DB connector layer. Additionally, we develop a graphical user interface (GUI) that facilitates creation of log query and log context. By using the GUI, a user can create an own log context with only knowledge of SQL.

To evaluate the execution performance of our system, we have conducted an experiment which measures execution time of some variety of log queries. The object of house log database comprises 50 million records of sensor data for approximately four years in our laboratory. The experimental results show that some simple queries can be executed within a few dozen of milliseconds. These queries would be practical to real-time context-aware service.

2. Preliminaries

2.1. Home Network System

Home network system (HNS) provides value-added services by connecting household appliances and sensors to home network. Each appliance (or sensor) has API, by which external systems can control the appliance. We have been developing a real HNS, called CS27-HNS [5]. Adopting the *service oriented architecture (SOA)*, every device in CS27-HNS can be used as a Web service. A client can access any device with a platform-independent protocol (i.e., REST or SOAP), which achieves high interoperability and programmability. For example, to get a temperature from a sensor, a client just accesses `http://cs27-hns/TemperatureSensor/getValue`. Accessing `http://cs27-hns/AirConditioner/on?mode=heating` turns on an air-conditioner in a heating mode.

2.2. House Log

House log is history of data that are acquired from the sensors and the appliances in HNS. In [16], we classify the house log into the following three types.

- **Energy Log:** It refers to log data of energy consumption in a house, including electricity, water, gas and so on.
- **Device Log:** It refers to log data taken from appliances, including status, operations and errors. It characterizes human activities within a house.
- **Environment Log:** It refers to log data taken from sensors, including temperature, humidity and illuminance. It characterizing the environment a house.

Traditionally, the context-aware services consumed only interesting data to evaluate the context. All irrelevant data were discarded without being stored, since the storage was limited and expensive. Now in the era of cloud computing and big data, we can manage large-scale data with inexpensive storage. Thus, it is realistic to store any kinds of data from HNS as house log.

In our CS27-HNS, we have been accumulating large-scale house log for several years, using cloud technologies [16]. For example, our environment log comprises of 50 million records of sensor data for approximately four years. In this paper, we assume that the house log is managed by a certain DBMS, and data can be searched by a *query* (e.g., SQL, Hive, Pig, etc.).

2.3. Current Context-Aware Services

A *context-aware service* is a service that autonomously executes appropriate actions when a context is established. A context is defined by situational information characterized by sensors. We have developed the *sensor service framework* which efficiently creates and manages context-aware services in CS27-HNS [11].

In this framework, a context is defined according to a format **[context_name: context_expression]**. In the format, **context_name** is an identifier of the context, while **context_expression** is an expression over a sensor value that defines the context. Figure 1 shows a syntax diagram for the context expression. An *atomic context expression* is defined by a current sensor value (CurrentValue) and a constant threshold (FixedValue) connected by a relational operator. CurrentValue (or FixedValue) takes a value over a primitive type. Combining multiple expressions by logical operators builds a *composite context expression*.

For instance, let us define a context `Cold` by the following atomic expression:

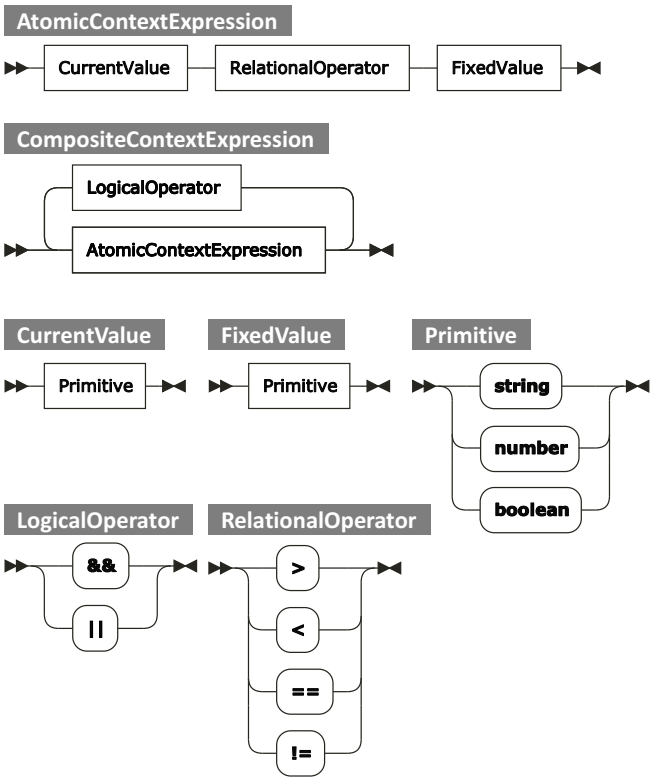

Figure 1. Syntax diagram of context expression (original version)

```
[ Cold: tempSensor01 < 8 ]
```

We assume that the variable `tempSensor01` holds the current value of a temperature sensor `tempSensor01` in HNS. Thus, `Cold` is defined by a situation that the temperature captured by `tempSensor01` is under 8 degree. A context takes a truth value (true or false) by *evaluating* the context expression. For example, `Cold` is evaluated to be true when the temperature is under 8 degree. Note in this framework that every context expression is constructed by *current* sensor values. Thus, every context is defined by a current situation only.

3. Log Context: Considering Past Situation in Context

3.1. Key Idea

This paper aims to extend the previous framework so that a context can be defined by both current and *past* situations. For this, we extensively use the house log. As seen in Section 2.2, the house log is comprised of time-stamped records gathered from various devices in HNS. Hence, the past situation in HNS can be retrieved by a "query to the house log", which we call *log query*.

Our key idea is to allow the context expression to contain log queries. Now, the current situation is

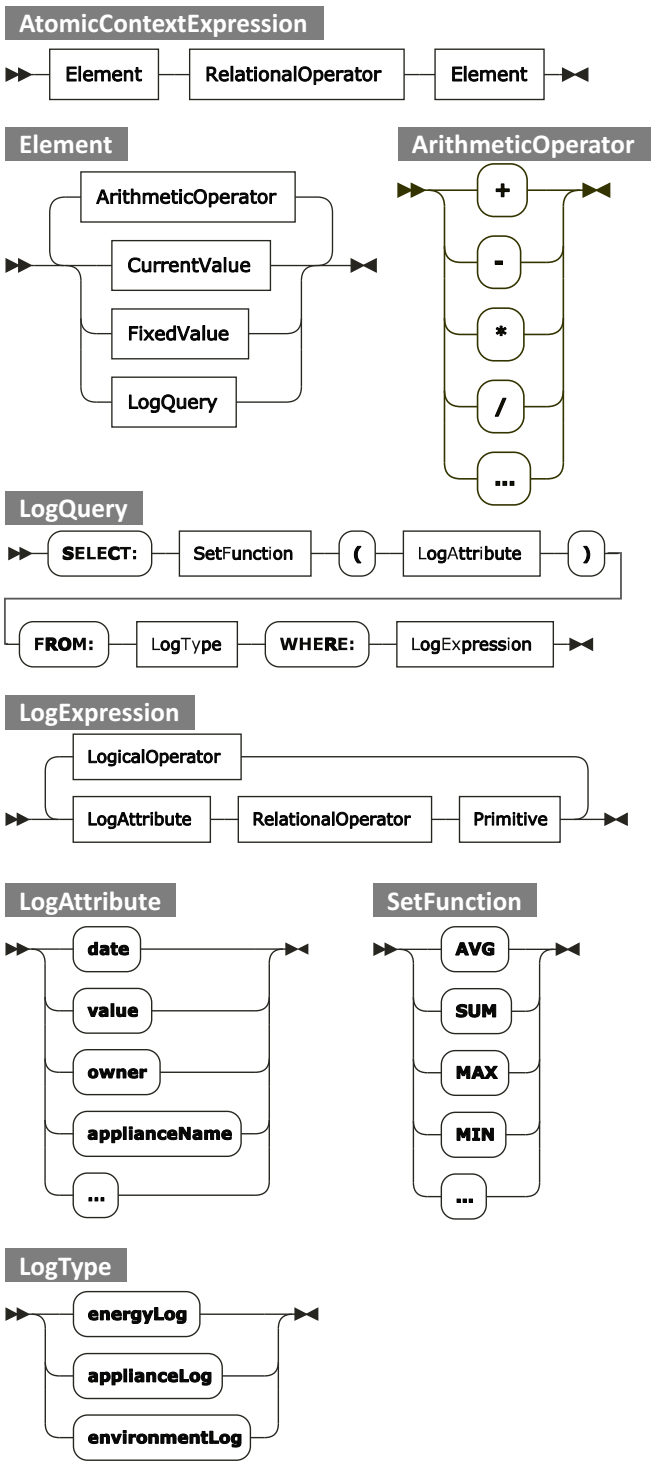

Figure 2. Syntax diagram of context expression (extended version)

captured by the sensor, while the past situation may be characterized by the log query. We call the extended context, *log context*.

Let us consider a log context *"it is colder than yesterday"*. To define the context, we need to compare the

current temperature and yesterday's temperature. The current temperature may be obtained by tempSensor01 as seen in Section 2.3. Yesterday's temperature may be obtained by a log query retrieving the value of tempSensor01 of yesterday in the same time as now. Details of the extension of context expression is explained as follows.

3.2. Extending Context Expression with Log Query

Figure 2 shows a syntax diagram of the extended context expression. In the extended version, an atomic context expression is defined by Elements, which are arithmetic expressions over CurrentValue, FixedValue and new LogQuery.

LogQuery defines a query to the house log that extracts a *single value* used in the context expression. According to an SQL-like format, a log query is constructed by SELECT, FROM and WHERE clauses. The SELECT clause specifies what attribute in the house log should be computed into a single value by which set function. The log attributes include date of the log, value of data, device appliance, etc. The set functions involve average, sum, max, min, etc.

The FROM clause defines which type of house log should be used. As seen in Section 2.2, the house log is is classified into three types: energy log, device log or environment log. The WHERE clause determines a condition what log data should be considered (log expression). The log expression is constructed by the logical combination of comparative expressions, each of which is defined by a log attribute and a primitive value connected by a relational operator.

3.3. Illustrative Examples of Log Context and Log Query

Here we describe some examples to support understanding. Let us start with a log query for yesterday's temperature. More specifically, we define a log query yesterdayTemp01, which calculates "the room temperature measured by tempSensor01 of yesterday around the same time as now" from the house log.

```
yesterdayTemp01: {
//the room temperature measured by
//tempSensor01 of yesterday
//around the same time as now
 SELECT: AVG(temperature)
 FROM:   EnvironmentLog
 WHERE:  date >= NOW() - 24 HOUR - 5 MIN &&
         date <= NOW() - 24 HOUR + 5 MIN &&
         sensorName == tempSensor01    }
```

This log query first obtains data records of tempSensor01 from the environment house log, where the recorded date is the same time (with 5 minutes margin) of yesterday. It then calculates the average of the temperature attribute.

The next example shows a log query that returns "the lowest temperature around the same time of the same day for past three years".

```
lowestTemp01For3Years: {
//the lowest temperature measured by
//tempSensor01 around the same time
//of the same date for the past 3 years
 SELECT: MIN(temperature)
 FROM: EnvironmentLog
 WHERE: date.DAY  == NOW().DAY &&
        date.TIME >= NOW().TIME - 1 HOUR &&
        date.TIME <= NOW().TIME + 1 HOUR &&
        date.YEAR <  NOW().YEAR &&
        date.YEAR >= NOW().YEAR - 3 YEAR &&
        sensorName == tempSensor01  }
```

Using the log queries, we define two log contexts: (C1) "it is 5 degree colder than yesterday" and (C2) "It is the coldest day for the past three years":

```
(C1) [Colder5DegThanYesterday:
      tempSensor01 + 5 < yesterdayTemp01]

(C2) [ColdestDayFor3Years:
      tempSensor01 < lowestTemp01For3Years]
```

We suppose that the log query is dynamically executed when the log context is evaluated. For instance, when we evaluate (C1), the value of tempSensor01 is obtained from the temperature sensor, while yesterdayTemp01 is calculated by executing the log query. Thus, the truth value of (C1) is determined.

The next section describes how to evaluate the log contexts systematically.

4. Implementing System to Derive Log Context

4.1. System Architecture

In this section, we implement a system that can derive the log contexts within an actual HNS. The system should be able to manage individual log contexts and queries as efficiently as possible. Also, the created log contexts and queries should be used by various applications. To achieve the requirement, we design the system based on a layered architecture, consisting of the following four layers:

1. *Application layer*: Manages context-aware services with log contexts.
2. *Log context layer*: Manages log contexts and evaluates each context based on its context expression.

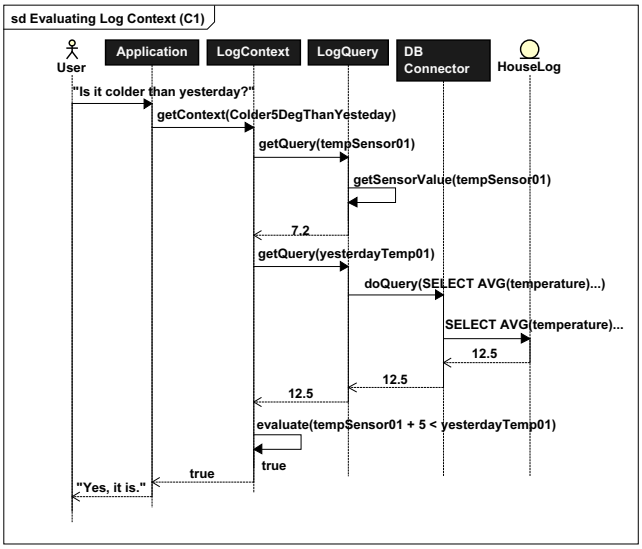

Figure 3. Sequence diagram for evaluating log context (C1) "it is 5 degree colder than yesterday."

3. *Log query layer*: Manages log queries to designated house log.

4. *DB connector layer*: Connects the database and executes the query.

4.2. Evaluating Log Context within Layers

Figure 3 shows a sequence diagram representing a scenario that evaluates the log context (C1) "it is 5 degree colder than yesterday" (see Section 3.3). In the proposed system, the log context is evaluated stepwise through the four layers.

In this scenario, we assume that a user asks an application "is it 5 degree colder than yesterday?" First, the application layer asks the log context layer if `Colder5DegThanYesterday` holds by executing `getContext()` method. Next, the log context layer tries to update values of all queries (i.e., `tempSensor01`, `yesterdayTemp01`) by asking the log query layer through `getQuery()` method. In the log query layer, the value of `tempSensor01` is obtained from the temperature sensor, as it is the current sensor value. Here we suppose that the value of 7.2 is obtained. Since `yesterdayTemp01` is a log query, it is delegated to the DB connector layer to query the designated house log database. In the DB connector layer, the query is executed for the database. Here, we obtain the value 12.5 as the result of the query. In the log context layer, the values of `tempSensor01` and `yesterdayTemp01` are updated to 7.2 and 12.5, respectively. Hence, the context expression of `tempSensor01 + 5 < yesterdayTemp01` is evaluated to be true. That is, the log context

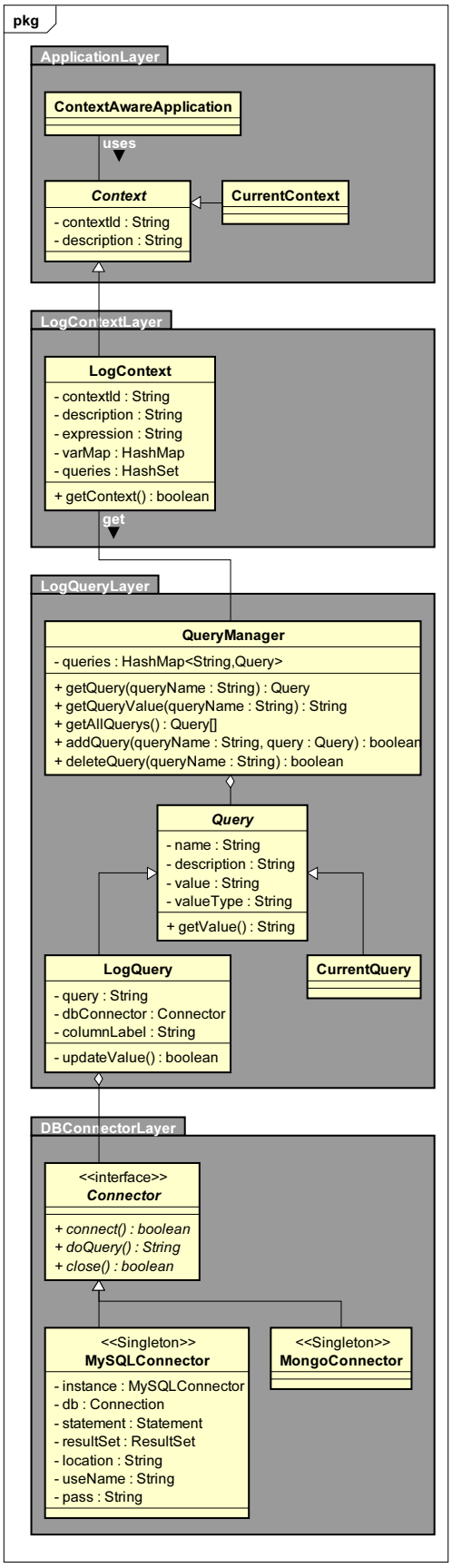

Figure 4. Class diagram of the developed system

`Colder5DegThanYesterday` becomes true, and the answer *"Yes, it is"* is returned to the user.

Thus, the task of evaluating a log context is coordinated by the four layers so that the responsibility is well distributed to the layers.

4.3. Detailed Design

Figure 4 shows a class diagram, representing the detailed design of the developed system. The four layers are enumerated as four packages from the top to the bottom. In the application layer, `ContextAwareApplication` is supposed to be an application providing a context-aware service. The application uses some `Context` which is either `CurrentContext` or `LogContext`.

The class `LogContext` implements `Context` in application layer. Based on the definition of the log context, it contains `contextId`, `expression` and `description` as attributes. Moreover, it contains `varMap` storing pairs of a log query and its value used to evaluate the expression, and `queries` specifying a set of all log queries involved in the log context.

`QueryManager` in the log query layer manages all the existing queries. It has a hashmap containing all the queries, and methods to add, get and delete a query within the hashmap. `Query` is an abstract class implemented by either `LogQuery` or `CurrentQuery`. `CurrentQuery` corresponds to a query to the current sensor value. `LogQuery` represents the log query, which contains a query statement and a database connector for the house log. Method `updateValue()` connects the database and executes the query statement to update its own value.

`Connector` in the DB connector layer specifies interface of DB connector. Various kinds of DBMS are adapted by this interface, so that the log query layer does not care the difference of DBMS when executing the query. In the figure, there are `MySQLConnector` for MySQL, and `MongoConnector` for MongoDB.

4.4. Graphical User Interface

In order to facilitate the definition of log query and log context, we implement a graphical user interface (GUI) for operating the system. By using the GUI, a user can create an own log context with only knowledge of SQL. Creation forms for log query and log context are shown in Figure 5 and Figure 6.

- **Log Query Creation Form:** Figure 5 shows GUI screen of log query creation form. Red asterisk represents a required field. First, a user inputs a query name, selects query type (i.e., current or log) and selects database name. The most essential field is SQL query field. According to the syntax diagram, which described in Section 3.3, a user can freely define or customize his/her required log query.

Figure 5. Log query creation form

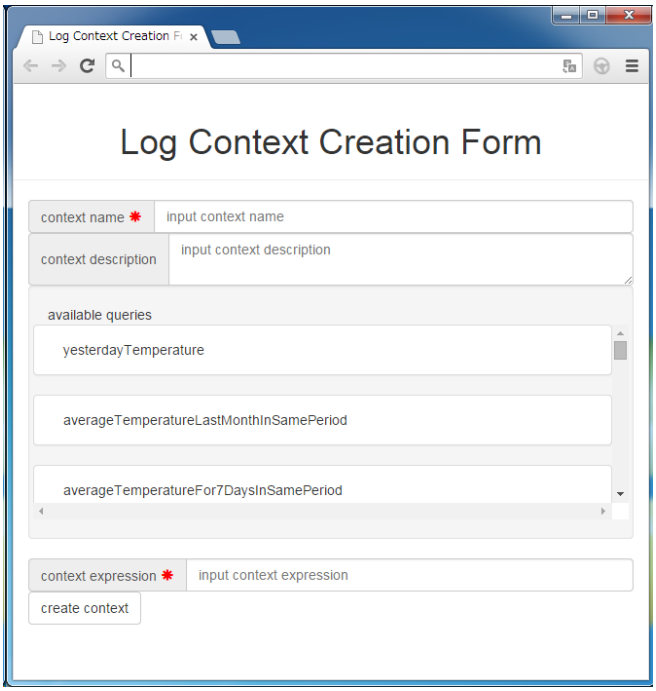

Figure 6. Log context creation form

- **Log Context Creation Form:** Figure 6 represents GUI screen of log context creation form. By using the defined log query, created in the above creation form, a user can define a new log context. User-defined log queries are

listed in the "available queries" field (i.e., `yesterdayTemperature`, `averageTemperatureLastMonthInSamePeriod` and `averageTemperatureFor7DaysInSamePeriod` in Figure 6). User defines a new context expression which depicted in Section 3.3 while viewing the available queries.

5. Experiment

5.1. Overview

To evaluate the execution performance of our system, we conduct an experiment which measures execution time of 11 variety of log queries. The object of house log database comprises 50 million records of sensor data for approximately four years in our laboratory.

5.2. Experiment Design

We defined 11 log queries and measured its execution time to evaluate the performance of our system. These queries have variety of viewpoints in terms of the number of referenced records. Figure 7 illustrates the difference of the referenced records in each log query. The dotted arrow shows a timeline and the gray vertical line shows current time. Referenced records are represented by circle or oval.

The first three queries (QY1, QY2 and QY3) retrieve a single temperature value from the different period of time. The subsequent four queries (QY4, QY5, QY6 and QY7) retrieve some temperature values and then calculate the average value. The subsequent two queries (QY8 and QY9) retrieve a lot of temperature values and then calculate the average value. The last two queries (QY10 and QY11) calculate the maximum temperature in a specified period of time.

Every query is in practical for some context-aware services that are triggered by temperature sensors. By measuring execution time of these log queries, we can confirm that how much can our proposed system be applied for real log-based context-aware services.

In the experiment, we measure the execution time of these queries 10 times. Database cache is disabled by "SQL_NO_CACHE" option.

5.3. Result

The experiment results are shown in Table 1. Coefficient of variation is the ratio of the standard deviation to the mean. It represents the amount of variation as a percentage of the total. The "actual number of referenced records" are calculated by executing each query with "COUNT" function instead of "SELECT" function.

At first, the coefficients of variations indicate that every execution time of log query has small variation. In other words, our system provides stable response time for same query execution in a single-user situation.

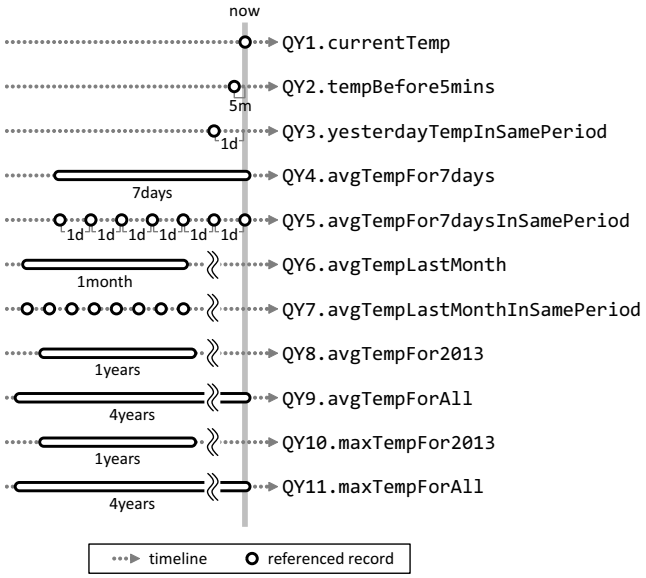

Figure 7. Referenced records of 11 log queries evaluated in experiment

Table 1 shows the most of log queries (QY1 to QY6) can be executed within one second. Especially, queries for accessing a referenced record (i.e., QY1 to QY3) is executed within a few dozen of milliseconds. These queries would be practical to real-time context-aware service. However, the execution time for QY7 to QY11, which requires certain amount of referenced records or calculation, is about one to six seconds. This execution time may be enough practical for non real-time services. However, it is difficult to apply these queries for real-time context-aware service without any improvement or modification to our system.

There is no difference between the SQL preset aggregation functions, such as average and maximize (QY8 to QY10 and QY9 to QY11). They were executed within 10 seconds even if the query accesses all records. This result shows that simple log query, which only uses preset aggregate function, can be enough to apply non real-time services.

6. Discussion

6.1. Extending Context-Aware Service by Past Context

We discuss feasibility of the proposed system through the experience of implemeting an actual context-aware service with introducing past context. The subject service is called "Umbrella Reminder Service (*URserv*)" which reminds a user to bring back his/her umbrella when it was rainy in the morning, but now it's sunny. *URserv* uses one composite context *"it was rainy morning but now it's sunny"* which is composed of the following three atomic contexts.

Table 1. Experiment result

query name	average execution time (ms)	coefficient of variation (%)	actual # of referenced records
QY1.currentTemp	14	5.1	1
QY2.tempBefore5mins	18	8.8	5
QY3.yesterdayTempInSamePeriod	14	5.4	10
QY4.avgTempFor7days	41	4.1	9,894
QY5.avgTempFor7daysInSamePeriod	42	4.6	66
QY6.avgTempLastMonth	229	1.0	43,173
QY7.avgTempLastMonthInSamePeriod	1,440	1.8	294
QY8.avgTempFor2013	5,400	0.4	482,984
QY9.avgTempForAll	4,865	0.6	2,374,638
QY10.maxTempFor2013	5,407	0.3	482,984
QY11.maxTempForAll	4,862	0.3	2,374,638

```
[LeaveOffice: userState == "leaving"]
[ItsSunnyNow: currentWeather == "sunny"]
[RainyAtMorning: morningWeather == "rainy"]
```

The first and the second contexts are conventional current context, and the third context is our proposed log context. The third context uses a log query `morningWeather`. This log query is defined as follows.

```
morningWeather: {
 SELECT: weather
 LIMIT: 1
 FROM: EnvironmentLog
 WHERE: date.DAY == NOW().DAY &&
        date.TIME >= "08:00:00" &&
        date.TIME <= "09:00:00"}
```

When every atomic contexts are satisfied, *URServ* invokes a speech synthesis service to remind a user to bring back an umbrella. In our CS27-HNS, the speech synthesis service can be invoked through `http://cs27-hns/SpeechSynthesizedService/speak?str=did_you_have_an_umbrella?`.

Please note that, establishing the defined composite context and invoking the speech service are controlled by our proposed management system of context-aware services. For more detailed information please refer to our previous work [17].

One of the most important aspect of this case study is how much time is required to accomplish the execution of *URServ*. It is necessary to accomplish the execution within a several seconds of leaving the office. The average time of the execution was about 2.1 seconds. So, we regard the proposed system have high feasibility and can be applied to actual and useful context-aware service.

6.2. Advantages and Limitations

We have extended our framework to provide log context by exploiting house log. We then have developed a system that can define and derive actual log contexts within the CS27-HNS environment. We assume that the extended log context can expand the availability of context-aware services. Furthermore, our developed GUI helps to create and manage log context. A user can create an own log context with only knowledge of SQL.

Comparing the extended framework with the previous one indicates a possibility of degradation of system performance because execution of log query requires access to database. Performance of the log query execution strongly depends on complexity of query itself. Real-time context aware service may be impractical depending on the log query. One of the solution is to introduce a materialized view which improves the speed of query on a large database by storing the processing result of the query.

6.3. Related Works

Many works are done on the context-aware services [6][7][8][9][10][18][19]. Some researchers have reported about context-awareness in a smart home environment [20][21], as same as this study. Gellersen et al. investigated multi-sensor context-awareness in mobile devices [9]. They reported "Mediacup" which embed multiple sensors and controlling software in a coffee mug. Mediacups detects some variety of situational contexts (e.g., *cup is stationary*, *drinking out of the cup* and *cup is carried around*) from a rule-based heuristic algorithm with environmental sensors. Henricksen and Indulska presented a software engineering approach to support context-aware application development [10]. These studies are only focused on "current" situational context. The concept of context-awareness can be applied not only to the physical environment but also to the computing environment. [22] and [23] have applied

concept of context-awareness as security controlling system. The novelty of this study is to investigate historical data to context modelling compared with these studies.

As a similar concept of our work, some researchers reported "context-history" [12][13]. Alaa et al. developed a general structure for storing and managing past contexts [12]. Hong et al. proposed a framework for providing personalized services with using context history [13]. Context-history is a concept of a series of someone's (or somewhere's) context occurred in the past. In contrast, the focus of our study is how to define and detect the current context which compared with the past environmental situation.

7. Conclusion

In this paper, we have extended the previous framework of context-aware services in home network system (HNS), so that every context can consider past situations. Using the house log gathered within HNS, we have proposed log contexts and log queries to define richer contexts with both current and past situations. We also designed and implemented a system that can derive the log contexts. The system was designed with four layers. The experimental result showed that the developed system correctly derived the log contexts with reasonable time.

Our future work is to consider concrete services with the log contexts. We also plan to conduct an experiment where various users define their own contexts. Evaluation of context precision and user satisfaction is interesting.

Acknowledgements. This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (C) (No.24500079, No.24500258), Scientific Research (B) (No.26280115), Young Scientists (B) (No.26730155)] and Kawanishi Memorial ShinMaywa Education Foundation.

References

[1] C. L. Wu, C. F. Liao, and L. C. Fu, "Service-oriented smart home architecture based on osgi and mobile agent technology," in *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, vol. 37, no. 2, 2007, pp. 193–205.

[2] J. Bourcier, A. Chazalet, M. Desertot, C. Escoffier, and C. Marin, "A dynamic-soa home control gateway," in *International Conference on Services Computing*, 2006, pp. 463–470.

[3] V. Ricquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge, "The smart home concept: our immediate future," in *E-Learning in Industrial Electronics*, 2006, pp. 23–28.

[4] M. Cabrer, R. Redondo, A. Vilas, J. Pazos Arias, and J. Duque, "Controlling the smart home from tv," *IEEE Trans. on Consumer Electronics*, vol. 52, no. 2, pp. 421–429, May 2006.

[5] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto, "Constructing home network systems and integrated services using legacy home appliances and web services," *International Journal of Web Services Research*, vol. 5, no. 1, pp. 82–98, 2008.

[6] H. Andy, H. Andy, S. Pete, W. Andy, and W. Paul, "The anatomy of a context-aware application," *Wireless Networks*, vol. 8, no. 2/3, pp. 187–197, 2002.

[7] X. Bai, D. White, and D. Sundaram, "Towards an adaptive visualization system in context-aware environments," in *Context-Aware Systems and Applications*, vol. 128. Springer, 2014, pp. 271–282.

[8] B. B. Kristensen, "Awareness of entities, activities and contexts in ambient systems," in *Context-Aware Systems and Applications*, vol. 128. Springer, 2014, pp. 144–156.

[9] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mobile Networks and Applications*, vol. 7, no. 5, pp. 341–351, 2002.

[10] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach," *Pervasive and mobile computing*, vol. 2, no. 1, pp. 37–64, 2006.

[11] M. Nakamura, S. Matsuo, S. Matsumoto, H. Sakamoto, and H. Igaki, "Application framework for efficient development of sensor as a service for home network system," in *International Conference on Services Computing*, 2011, pp. 576–583.

[12] A. Alaa, A. Ammar, M. Mubarak, and A. Vangalur, "Storing and managing context and context history," in *Context-Aware Systems and Applications*, vol. 128. Springer, 2014, pp. 35–46.

[13] H. Jongyi, S. Eui-Ho, K. Junyoung, and K. SuYeon, "Context-aware system for proactive personalized service based on context history," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7448–7457, 2009.

[14] L. Mengmeng, H. Ogata, H. Bin, N. Uosaki, and K. Mouri, "Context-aware and personalization method in ubiquitous learning log system," *Journal of Educational Technology & Society*, vol. 16, no. 3, pp. 362–373, 2013.

[15] A. Sofiane, B. Mokrane, and L. Stéphane, "Context-aware recommender systems: A service-oriented approach," in *International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases*, 2009, pp. 31–36.

[16] S. Yamamoto, S. Matsumoto, and M. Nakamura, "Using cloud technologies for large-scale house data in smart city," in *International Conference on Cloud Computing Technology and Science*, 2012, pp. 141–148.

[17] H. Takatsuka, S. Saiki, S. Matsumoto, and M. Nakamura, "Design and implementation of rule-based framework for context-aware services with web services," in *International Conference on Information Integration and Web-based Applications & Services*, 2014, pp. 233–242.

[18] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[19] M. Martin, O. Brdiczka, D. Snowdon, M. Jean-Luc *et al.*, "Learning to detect user activity and availability from a variety of sensor data," in *International Conference*

on Pervasive Computing and Communications. IEEE Computer Society, 2004, pp. 13–13.

[20] Y.-S. Chen, I.-C. Chen, and W.-H. Chang, "Context-aware services based on osgi for smart homes," in *Ubi-media Computing*, 2010, pp. 38–43.

[21] D. Shin, G. Lee, D. Shin, and D. Shin, "System architecture using human interaction markup language for context awareness in home network," in *Future Information Technology*. Springer, 2014, pp. 439–444.

[22] S. H. Park, J. S. Cho, Y. J. Han, and T. M. Chung, "Design and implementation of context-aware security management system for ubiquitous computing environment," in *Frontiers of High Performance Computing and Networking*. Springer, 2007, pp. 235–244.

[23] S. H. Park, Y. J. Han, and T. M. Chung, "Context-aware security management system for pervasive computing environment," in *Modeling and Using Context*. Springer, 2007, pp. 384–396.