Electrical Big Data's Stream Management for Efficient Energy Control

Jean Gane Sarr(✉), Ndiouma Bame, and Aliou Boly

Cheikh Anta Diop University, Dakar Fann, BP 5005, Senegal
{jeangane.sarr,ndiouma.bame,aliou.boly}@ucad.edu.sn

Abstract. Energy is crucial for any activity of a country. Therefore, electrical organizations need to analyze the data generated in the form of data streams by network equipment's for decisions making. These data are voluminous and velocious, which make impossible to process or store them with conventional methods. Hence the need to have a tool that allows to exploit these electrical data from the user's consumption and the network. In this paper, we propose a tool that summaries data streams by using data cubes structures and gives the ability to face the need to make the best decisions required by the customer as well as the supplier. This tool is composed by a data stream summary model and two algorithms used to create, load, and update the data cubes. This proposal was performed with Big Data tools that give to this summary the capacity to scale up. To demonstrate the effectiveness of the proposition, a detailed experimental evaluation over a real electrical data stream is presented.

Keywords: energy · electrical data · data stream · summary · big data · data cube

1 Introduction

The need to manage power energy in developing country constitute a real challenge. Indeed, all other sectors like health, education, telecommunications, and so on, crucially depend on electricity. Therefore, with the advent of smart meters, electricity consumption can be measured remotely from consumers by electricity suppliers such as EDF for France, ENEL for Italy or SENELEC for Senegal with a certain periodicity between measured (seconds, minutes, hours,...). This has the effect of generating a continuous stream of electrical data, in a fast and volumetric way. Thus, sensors can be connected to the supplier's information system (IS) to allow billing, control, aggregation of load curves, for one thing. The analysis made on the consumption readings of the meters of the customers can prove to be of great use both for the supplier and for the customer. Indeed, the customer could use this analysis, for example, to detect the reasons of high-power consumption (a damaged freezer, a high number of household appliances,...). The supplier, could use these data, for example, to know the profile of the consumption to propose adapted prices to each customer, to anticipate electrical needs, to

R. A. Saeed et al. (Eds.): AFRICOMM 2022, LNICST 499, pp. 414–429, 2023.
https://doi.org/10.1007/978-3-031-34896-9_25

react to demand in the advent of a critical situation, or to supervise its network, to process the overall efficiency of the network or to issue some alerts, among other things. In addition, the recorded measures allow to analyze the load curves corresponding to the evolution of power consumption at a time interval ranging from one second to many days for a well-defined period. Thus, a communicating meter can, during its operation, transmit a records (tuple) according to a customizable time interval. This tuple can contain the meter number, the reading time, the date, voltages, intensities, active power, and other energy attributes. Therefore, a stream of consumption data is composed by events coming from several equipments. Data streams are often described by multiple qualitative and quantitative attributes from different objects. These data can be stored in data structures called cubes [1] where the qualitative objects can be viewed are dimensions and the quantitative objects are measures. They are also unloaded from the system if it does not exist a retention mechanism that can be used to query them in the future. In this sense, this paper aims to present a summary technic that allows to analyze electrical data streams with Online Analytical processing (OLAP) [2] technics by using NoSQL [3] tools that give the possibility to deal with data streams constraints like storage and processing and to scale up. In the remainder of this article, the Sect. 2 presents a study of some propositions for the processing of electrical data streams. Then, in Sect. 3, the proposed summary and the architecture are described. The data stream used and its modeling as well as some use cases are presented in Sect. 4. The results obtained are given in Sect. 5. Finally, in Sect. 6, an assessment and future directions are discussed.

2 State of the Art

Many relevant works on the processing of electrical data streams have been carried out [4]-[10]. The authors of [4] presented an experiment performed with two public Data Stream Management Systems (DSMS), STREAM and Telegraph [6] based on queries with the aim of proposing a new distributed data stream management system. This work allows to discover the usage of these frameworks but do not propose a summary. In [7], authors proposed an approach for real-time analysis of the energy consumption of manufacturing equipment based only on electrical energy data streams. This approach followed the paradigm of event-driven systems and used complex event processing methods, however, it did not retain data. In [8], event stream processing techniques are applied to automate the monitoring and analysis of energy consumption in manufacturing systems. Methods to reduce the use according to the specific patterns discerned are discussed. However, in these works, the processing is performed only on the fly. So, there is no mechanism that retains a certain part of the observed data stream, that's why they are unloaded after analysis. Another proposal for the processing of electrical data streams is PQStream [9]. It is a data stream processing system which determines the quality of electrical energy. This system has been designed for the processing and management of electrical data streams from the Turkish electricity transmission system. However, the output is stored in a relational

database that does not allow multidimensional analysis and scaling. In [10], the authors presented an approach related to Big Data and the Internet of Things that involved models that learn gradually. They used a strategy that consists of storing the first events of the stream to apply a set of learning techniques on them that make the streaming algorithms ready to be able to handle incoming events. In contrast, this proposition doesn't allow to query the unloaded data because of a lack of data storage structure. These works constitute real advances in the processing of electrical data streams. Nevertheless, they do not address the multidimensional aspect of them. Also they do not propose summary techniques that enable to preserve and exploit certain relevant data from these data streams in the future. Indeed, with data streams, the data are unloaded from the system after each expiration of time windows. However, the end user may want to query the past data to be able to make decisions. Therefore, with these systems, it will not be possible to give results for this kind of queries. In addition, to satisfy the storage and processing constraints related to data streams, the system must evolve, which is not the case with these proposals. Thus, this paper gives a summary of multidimensional data stream performed on electrical data with Big data tools to deal with scalability and cope with storage and processing limits for future analysis.

3 Summary Model and Architecture

This section describes the proposition that consists of a multi-level cascading cubes with two algorithms to create, load and update these cubes. A cube is a data structure that involves axes of analysis like equipment, time, geograpic area known as dimension to analyse insights called measures as power, voltage, intensity, gathered into a fact table [1]. A cube can be modelized in a star schema (where dimensions dont have links) or in a snowflakes schema (where dimensions may be linked) [1]. For the implementation of this model, a complete architecture based on Big data approach will be proposed.

3.1 A Cascading Cube Model

The data streams are often characterized by several dimensions and facts. Thus, it becomes necessary to model them in a multidimensional way for OLAP applications. In this sense, a multidimensional data streams summary technic is proposed. These summaries consist of cascading data cubes with algorithms to create and update the different cubes.

The Summary Definition. The proposed summary is composed by different sub summaries corresponding each to a cube. Thus, let S be the summaries set presented by Fig. 1, Tw a tilted-time window [11] (where the most recent data are registered at the finest granularity and the more distant data are registered at a coarser granularity). We also define the set of cubes C representing the differents cubes arranged in a manner to form a cascade. In other words,

the summary is composed by cubes having different levels of data aggregation following the time granularity. Therefore, we have :

- $T_w = \{T_{w1}, T_{w2},..., T_{wn}\}$ where each T_{wi} in this time dimension corresponds to a time window and $T_{wi} < T_{wi+1}$.
- $C = \{C1, C2,..., Cn\}$ where each Ci $(1 \leq i \leq n)$ corresponds to a summary (cube of the cascade) computed after the time window T_{wi} $(1 \leq i \leq n)$. In the set of cubes C, Ci and $Ci+1$ are disjoint. In addition, each $Ci+1$ is the result of the update function (presented in the following section) applied on Ci.
- Thus, the definition of the global summary is gived by the union of cubes or summaries Ci calculated during the time window T_{wi}. Thus, $S = U\{(Ci, T_{wi})\}$.

The expiration of a window initiates the call of the aggregation and propagation algorithm of the model. Thus, the different sizes of these time windows constitute indicators used as timers to aggregate and propagate data from one cube level to another. The cubes of this model differ in the duration of conservation of their data before aggregation and propagation thereof to the cube having the highest temporal value as well as the level of aggregation of the data of each of these cubes. In this sense, the temporal granularity is not the same at all levels of the cascade and it is possible to customize it with minimum ($gTMin$ minimal time window parameter) and maximum ($gTMax$ maximal time window parameter) limits presented in Fig. 1. The number of cubes in the cascade is defined by the $gTMax$ parameter. For example, when gTMin = minute and $gTMax = day$ then we have time levels *minute, hour, day* so the cascade will contain 3 cubes.

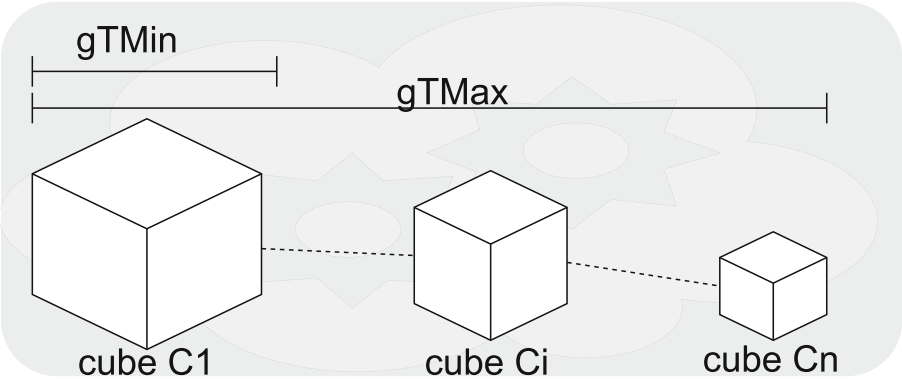

Fig. 1. Multi-level cascade of cubes.

The Algorithms of the Model. The initialization and the updating of the different cubes of the cascade are performed by two algorithms. The first algorithm, presented by the Fig. 2, creates and loads the data cubes. The second is the update algorithm described by Fig. 3 and used to propagate the aggregated data between two successive cubes of the cascade. These two algorithms are customizable in the aim to be able to consider the user's preferences and are presented as follows.

Initialization Algorithm. This algorithm is responsible for the creation of each data cube of the model. It is presented by the Fig. 2. For the first level of the cascade, the function verifies if the table corresponding to the data cube already exists; if it is not the case, it creates the cube's table where column families correspond to each dimension and to the fact(s) [1] defined from the received data structure before loading data. After creating this data cube, this function will then inject data in the corresponding table.

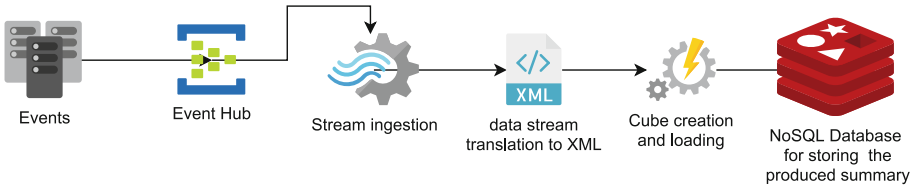

Fig. 2. Initialization of summaries.

The Update Algorithm. The data cube update task is made by this function also known here as *f_update* and presented by the Fig. 3. It is executed after time windows expiration to update the differents cubes. This function works in different steps. In the first time, it discards data contained in the cube *Ci+1*. In the second step, it generates a new data stream with data contained in the cube *Ci*. In the next step, it calls over this new data stream a continuous query to aggregate these data according to the new time window granularity. Afterwards, in the fourth step, it injects the results in the cube *Ci+1* table. The update function is run to these different processes between all the cubes of the cascade in a descending way. If we assume to have three cubes *Ci*, *Ci+1* and *Ci+2*, the process can be defined like following: discard data from *Ci+2*, *Ci+2* = *f_update(Ci+1)*, discard data from *Ci+1*, *Ci+1* = *f_update(Ci)*, discard data from *Ci*, load new data in Ci. Discarding the data of the last cube allows one to maintain a bounded size for the system.

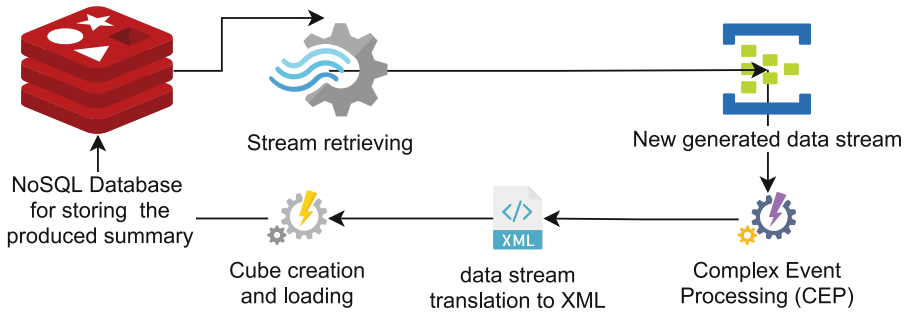

Fig. 3. Summaries (cubes) updating process.

3.2 Architecture

The architecture presented by the Fig. 4, consists of different layers or phases ranging from the ingestion of data streams to the visualization of these data streams through the processing of data and their storage. These different phases define a multilayer architecture whose levels are interdependent. At each level of this architecture, there are different tools to manipulate data streams for well-defined processing.

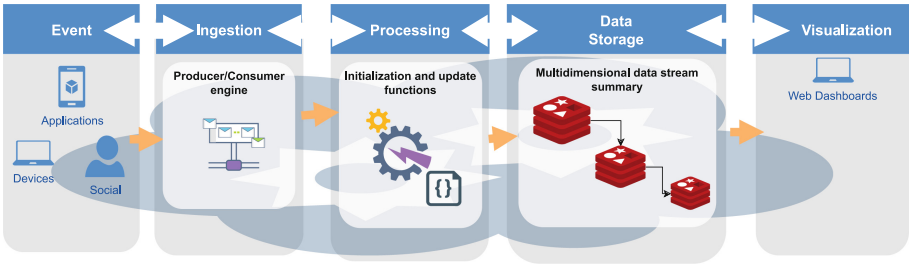

Fig. 4. The global architecture.

3.3 Data Stream Ingestion Layer

The data stream ingestion layer connects the various data stream sources such as, for example, the electricity consumption data collected in real time on the meters of all the network equipments (substations, transformers, feeders, ...) to the processing layer. These data streams are collected and then injected into the system by various tools that work in producer/consumer mode by using a queue. This step may involve tools such as Kafka [12], Flume [13]. In this work, Kafka is used to implement this layer because it can handle voluminous data from various sources in a fast way. It is also simple to implement and can be linked with many other processing tools.

3.4 Data Stream Processing Layer

This processing can be performed into two ways, namely a stream or batch processing [14]. In batch processing, data are collected and grouped into blocks of a certain temporal granularity (second, minute, hour, ...) after that, they are injected into a processing system. Batch processing is best suited when data streams are received later. For example when the data source only provides its information every 30 min. As usage, it can be employed to process all electrical measures taken by the meters after 10 min. This type of processing is also suitable when it is more important to process large volumes of data to obtain more detailed information than to obtain fast analysis results. For batch processing, there are various distributed platforms that provide scalable processing on clusters. To process the data streams as they arrive in the system, it is necessary

to perform on-the-fly processing on them to be able to draw knowledge from them. In fact, their storage in their globality is not allowed by the available resources. The continuous processing of data streams is usually done on clusters of distributed machines to allow to scale. This ensures a certain availability of processing resources (memory and CPU) [20]. This layer can be achieved by tools like Apache Storm, Spark Streaming [21].

3.5 Storage Layer

This layer, is used to store summaries build by the processing layer. The implementation of this layer is ensured by a model consisting of cascading cubes representing the summary of data streams obtained after processing. The first level is a cube that stores detailed data, and the other levels aggregate the data according to the time window coarsity. The data migration between the cubes of the cascade is performed using an update algorithm later presented. This data stream summary layer allows to visualize the history or a part of them in order to give to the decision-makers the ability to perform analysis as well as to query past data. This layer is achievable by solutions such as HBase, Hive or Cassandra [14]. In this work, HBase [15], a NoSQL Document-oriented database, is used for the implementation of this layer. The reason of this choice is that HBase helps to deal with scalability and gives high performance in read/ write tasks. Indeed, NoSQL databases support a variety of data models that are ideal for building applications that require large data volumes and low latency or response times. They provide flexible schemas that allow to handle multiple data format (non structure, semi-structured or structured) [15]. NoSQL databases can scale-out by using commodity hardware which has the ability to support increased traffic [18,19]. In addtion, NoSQL databases can become larger and with hight-performance which make them to be more suitable for evolving datasets like data streams [16]. They are automatically replicated wich give a high availibilty and are designed for distributed data stores with large data storage needs which make them best choice for big data. These solutions involve tools such as HBase, Hive or Cassandra for the management of NoSQL databases and are most often used to store Big Data [15]. They are grouped into four categories depending on the type of representation implemented. Thus, key-value-oriented, column-oriented, document-oriented, and graph-oriented models can be distinguished. Data streams summaries using Column-oriented databases: The columns-oriented databases were used in various propositions [16,17] to store data streams. Indeed, they are built for highly analytical, complex-query tasks and are most often suitable for large data model. They can also be utilized to perform data warehouses. HBase [16] works very well for real-time analysis of large data streams.

3.6 Data Visualization Layer

The visualization layer allows the extraction of crucial information for analysis with real-time data stream mining or learning tools to allow strategic and/or tac-

tical decision-making (detecting outliers [17], detecting fraud, ...). Thus, to help analysis of the data streams, obtained information must be described according to a certain number of types of representation such as tables, graphs, curves and stuff. And these different displays, combined, represent a dashboard. In this sense, this phase of data analysis is the one in which can be detected intrinsic patterns, extract relationships and knowledge. It thus helps the decision-makers to easily exploit the results obtained from the data stream analysis. SAP Hana, Power BI [22] are different tools that can be utilized to carry out this analysis. Power BI is employed for the need of this work.

4 Implementation

This section describes the implementation of the summary based on the presented architecture. The used data stream and his modelling are presented. Also the ingestion is achieved with Kafka, the processing layer with Apache Storm, and storage layers with HBase [3,20]. The visualization layer is achieved with Power BI Desktop [22].

4.1 Electrical Data Stream

The used data stream in the experiments describes data measured by smart equipments. They are continuously obtained from the different components of the electrical network of Senelec (national electricity company of Senegal). The database used as source has an initial size more than 100GB and is continuously growing with millions of records (measures) each record having 42 columns (40 numerical values) 1 for the timestamp and 1 for the meter number. This data stream is described by the following parameters:

- The **name** gives the name of the equipment.
- The **equipment type** item describes the type of equipment.
- The **longitude** and **latitude** headings describe the geographical the equipment location.
- The **MeterNo** the counter number,
- The **dataTstmp** the timestamp,
- The attributes **P8311** (U1), **P8312** (U2) and **P813** (U3) the values measured of the average voltages of phases 1, 2 and 3 in volts (V),
- The attributes **P8391** (I1), **P8392** (I2) and **P8393** (I3) the measured values of the average intensities of phases 1, 2 and 3 in ampere (A),
- The **P8341** (kW) attribute defines the value measured of the imported active power from which the reactive power (in kW) will be processed with the values of the voltages as well as those of the intensities.

4.2 Use Case

The electrical data streams analysis allows to gain information that helps to know the network behaviour. With these results the end user can make investments,

to know zones that need to be linked to the electrical network, to reduce power wastage, to detect frauds, among other analysis tasks. Therefore, in this paper we provide to a decision maker a way to query multidimensional data streams. This proposition allows then to the end user to analyse electrical data stream measures (intensities, voltages, powers) by using different axes of analysis (equipment, time, geographic) in the aim to take decisions like what zones require to be liked to the electrical network, how to detect frauds, how to predict production, how to reduce power wastages, among others. In this sense, we build a summary that gives a great possibility of analysing data streams and is applicable to all use cases. In this sense, some cases are defined related to the data stream described above. The queries are built in windows defined on the streams emitted in the system by the higher layers of the architecture. These queries are formulated as follows:

- **Query 1**: Determines the average for the Phase 2's voltage of all equipments. The result of this query allows to detect for example the causes of abnormal or excessive consumption by detecting values lower or greater than these measures.
- **Query 2**: Gives the maximum, minimum and the average values for the phase 2 voltage of the HTA transformers. The supplier could use this query, for example, to anticipate electrical needs, to react to power energy needs of the customers demand in the advent of a critical situation, or to supervise its network.
- **Query 3**: Provides U2 values for all equipment the 09-27-2020 for example. This query can help to detect the equipments that excessively consume power for a precise day.
- **Query 4**: Gets maximum, minimum after getting the average values of U2 for equipment which name contains "Classic" for period between the 09-27-2020 at 23:50 and the 09-28-2020 at 04:20 for example. This query can be used to detect outliers [17].

4.3 Modeling

In the following, the star model [1] is described by the Fig. 5. It has been developed from the presented data stream. This model contains dimensions which are the axes of analysis and a fact table. These dimensions are the following: the "DimEquipment" dimension describing each equipment present in the network having attributes like the meter number, the equipment name. Then, the "DimTime" dimension that stores different time granularities such as data tstmp, minute, hour, day. The "DimService" dimension gives the service to which the equipment is attached. The "DimGeo"' dimension represents the location of the counter and allows a geographical analysis. The "DimEquipmentType" dimension that categorizes the equipment as source station, substation, feeder, transformers. Finally, the model has a central fact table, "FactEnergy" which groups together a certain number of measures to be analyzed. More formally, we have the following diagram.

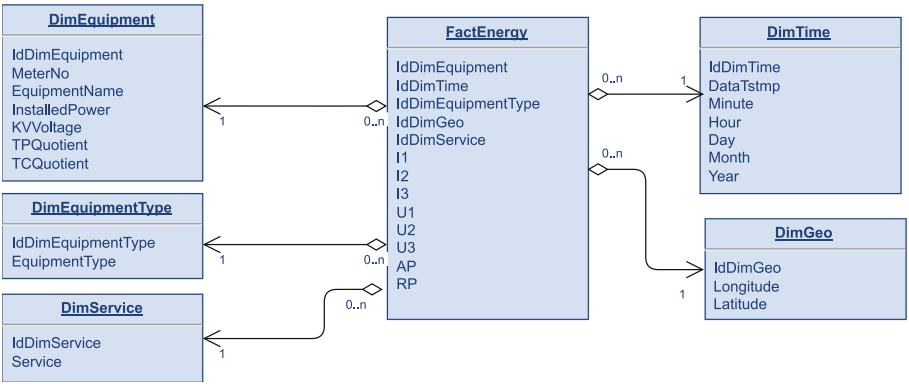

Fig. 5. Star schema based on the used data stream

4.4 Data Stream Retrieval

The source data stream is retrieved by this part of the system that is built with kafka event producer. This producer is implemented with Java. It defines the kafka configuration by using java.util.Properties object and it uses the *send* method of the classe *org.apache.kafka.clients.producer.Producer* to emit events read from th source into a Kafka topic. A topic is similar to a folder in a filesystem, and the events are the files in that folder and works as a queue. And before writing events in a topic, it must be created by giving it a name with the following command :

– *bin/kafka-topics.sh –create –topic fluxDBTestUpdate –bootstrap-server localhost:9092*

Thus, the producer firstly retrieves events from the data source. Then, it sends them to the Kafka topic where they are taken by a Storm's spout (consumer) to send them to the various bolts for processing.

4.5 The Summary Construction and Update

The initialization and update algorithms previously presented for the summary's cubes creation and updating are implemented in this part. This layer ensures the processing of the tuples into the Kafka topic. To achieve these tasks, different functions performed by Storm bolts are used for retrieving these data. They also execute over data streams continuous queries. In the final step, the presented initialization function is ran to create the cube data tables (multidimensional summaries) in HBase and to load obtained results into them. After retrieving the data injected by a spout, a bolt is defined with the goal to translate data into a JavaBean having the following prototype: *MeasuresCounter (String id, String meterNo, String dataTstmp, String P8311, String P8312, String P8313, String P8391, String P8392, String P8393, String 8341)*. The *JavaBean* instantiated

is transmitted to a second bolt which role is to check the existence of the table of the first cube (summary) in HBase. If the table does not exist, one method which, after defining the HBase table structure, creates it. This method performs the description of its column families as well as the columns of the HBase table. Then data are inserted into the previous created table. The updating cubes (summaries) process is performed according to different steps. In the first step, the data of the cube Ci are retrieved via a new Kafka producer. After aggregation, they are loaded in a new Kafka topic (a queue) named *fluxDBTestUpdate*. Then by a new spout, we load these data into an update bolt. This update bolt performs its processing via queries executed in the Esper engine following time windows corresponding to the different time values of the different levels of the cascade. Indeed, each cube updated corresponds to a welldefined time window. To do this, a Java class called *EsperOperation* is defined. The constructor of the *EsperOperation* class itializes the Esper listener and defines the Esper query to be executed, presented by the Fig. 6. For example, if we want to update the data of the cube corresponding to the last 30 min electricals measures (cube *Ci*), the Esper query buffers the events over 30 min and then returns the result of the aggregate functions applied to them for each equipment during the 30 min. For data propagation, the bolt that achieves the update task calls the data insertion bolt to load the results into an HBase table corresponding to the cube to be updated (cube *Ci+1*). Fixed time windows are used here.

ESPER QUERY EXEMPLE

```
SELECT MeterNo,DataTstmp,
       SUM(P8311),SUM(P8312),SUM(P8313),
       SUM(P8391),SUM(P8392),SUM(P8393),
       SUM(P8341)
FROM MeasureBean.win.time_batch(30min)
GROUP BY MeterNO,DataTstmp
```

Fig. 6. An Esper query sample

5 Results and Discussion

This section presents some results obtained after execution of some queries using the time and equipment dimensions. The Fig. 7 presents the phase 2 voltage (U2) over all the network, the Fig. 8 the phase 2 voltage for electrical HTA transformers, the Fig. 9 phase 2 voltage for the 09/27/2020 and Fig. 10 phase 2 voltage for 09/20/2020 at 04:20. In each of these figures, we also represent MAX, MIN and AVG phase 2 voltage values.

5.1 Results Presentation

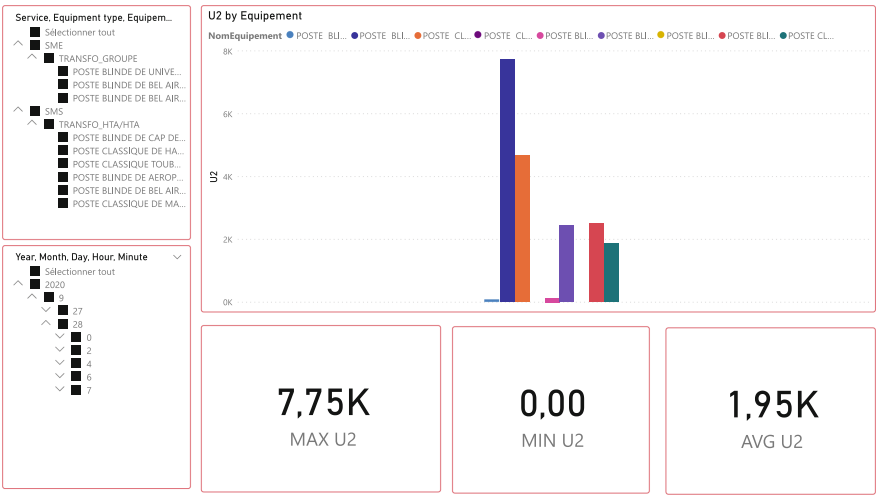

Fig. 7. Phase 2 voltage (U2) measure over the electrical network

The Fig. 7 shows the voltage measured over all the network. It also shows the max, min and average of the phase 2 voltage. These results allow to the supplier to supervise the equipment load. The Fig. 8 gives the maximum, the minimum and the average values for the phase 2 voltage of the HTA transformers. It shows the ability of the summary to answer queries with filters in other dimensions. This query can help the supplier to supervise the electricity transportation. The Fig. 9 shows phase 2 voltage for the given day 27-09-2020. It allows to see that the summary can aggregate data by also applying a filter. This query demonstrates the summary ability to analyze the network data in a particular date. The Fig. 10 shows that the summary can be used to analyze a certain part of the electrical network.

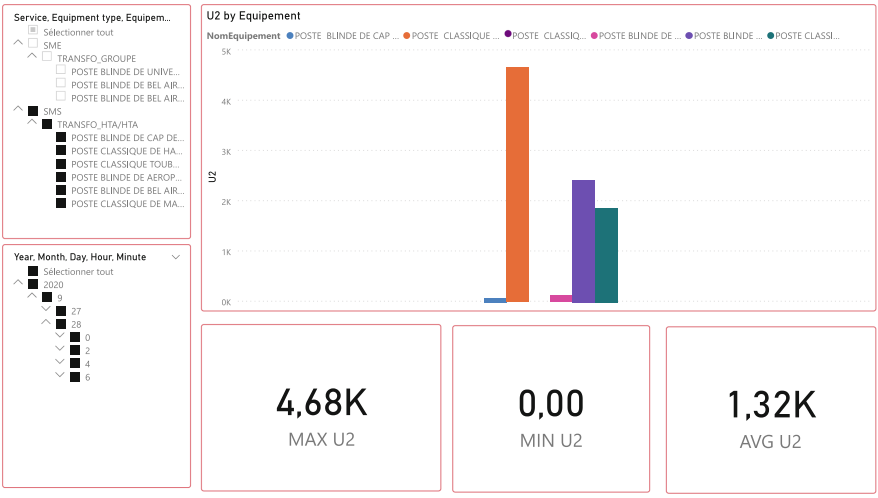

Fig. 8. Phase 2 voltage (U2) voltage measure for HTA transformers

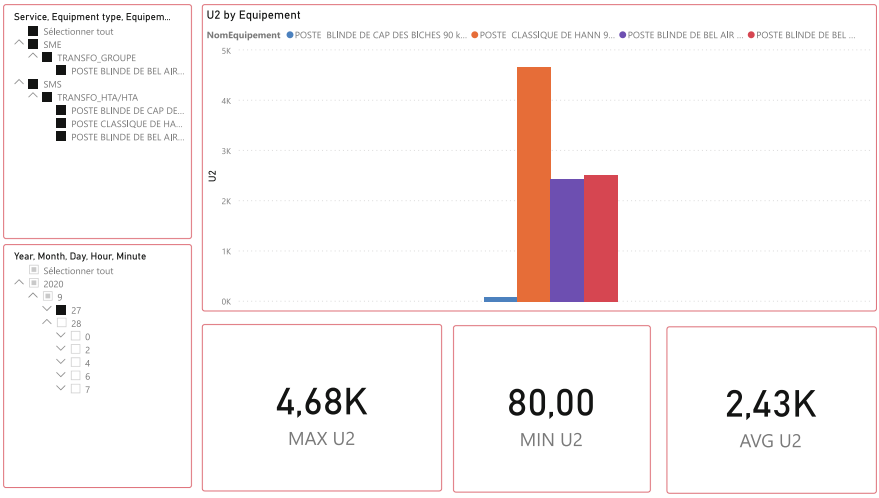

Fig. 9. Phase 2 voltage (U2) voltage measure for the 27-09-2020

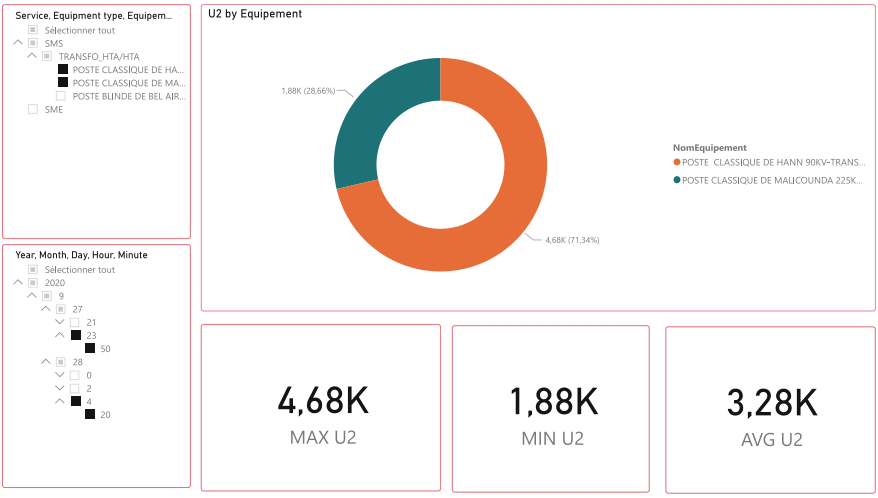

Fig. 10. Phase 2 voltage (U2) measured values with maximum, minimum and the average for equipment which name contains "Classic" for between timestamp "09-27-2020 23:50" and "09-28-2020 04:20" ?

5.2 Discussion

In this section, a certain number of queries have been executed against the summaries. We observed from the results that the waterfall cube model was able to meet the demands of a user who would like to view stream data on different dimensions. The results obtained by the summaries show that the contribution of this model is different from works presented in the section II. Indeed, it allows to give answers for queries by using different dimensions over data streams with filtering and aggregation. Moreover, this model can scale up. This model's ability is allowed by the fact that it was achieved by Big Data tools. This functionality is not offered by the propositions studied from the literature. In this sense, the model can be used to deal with CPU and memory resources.

6 Conclusion

Analyzing electrical data streams can give the opportunity to take decisions that will help to evolve different sectors of a developing country. However, it is impossible to efficiently process and store these multidimensional data streams with existing methods. Therefore, in this paper, different propositions on electrical data stream processing have been studied. This study shows that these different interesting systems do not propose a summary technic for analyzing in the future electrical data streams but only do it on the fly. Also, these works do not consider the multidimensional aspect of electrical data streams. In this sense, a generic cascading cubes data model and an architecture have been proposed and

implemented with Big Data tools. The architecture includes ingestion, processing, storage and visualization layers. Similarly, the generic cascading cubes data model and its OLAP modeling have been described. The implementation of the different architecture's layers was also presented to test the waterfall model over a real electrical data stream. Finally, the execution of some queries has been performed on the summaries. The execution of these queries shows that the proposed model met the expectations of decision-making users. The perspectives are to study and to develop data mining and machine learning algorithms to analyze data present into the obtained summaries.

References

1. Zafar, K.M., Ghulam, M., Nadeem, S., Syed, W., Junaid, Q., Shaista, S.: A Review of Star Schema and Snowflakes Schema, pp. 129-140, May 2020. ISBN:978-981-15-5231-1, https://doi.org/10.1007/978-981-15-5232-8_12
2. Zhan, C., et al.: AnalyticDB: real-time OLAP database system at Alibaba cloud. Proc. VLDB Endow. **12**(12), 2059–2070 (2019). https://doi.org/10.14778/3352063.3352124
3. Wingerath, W., Gessert, F., Ritter, N.: NoSQL & real-time data management in research & practice. In: Meyer, H., Ritter, N., Thor, A., Nicklas, D., Heuer, A., Klettke, M. (Hrsg.), BTW 2019 - Workshopband. Gesellschaft für Informatik, Bonn, S. 267–270 (2019). https://doi.org/10.18420/btw2019-ws-28
4. Abdessalem, T., Chiky, R., Hébrail, G., Vitti, J.: Traitement de données de consommation électrique par un Système de Gestion de Flux de Données, pp. 521–532 (2007)
5. Arasu, A., et al.: STREAM: the Stanford data stream management system. In: Garofalakis, M., Gehrke, J., Rastogi, R. (eds) Data Stream Management. DSA, pp. 317–336. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-540-28608-0_16
6. Chandrasekaran, S., et al.: Telegraphcq: continuous data stream processing for an uncertain world. In: Proceeding of CIDR, ACM (2003)
7. Chiotellis, S., Grismajer, M.: Analysis of electrical power data streams in manufacturing. In: Dornfeld, D., Linke, B. (eds) Leveraging Technology for a Sustainable World. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-29069-5_90
8. Dornfeld, D., Vijayaraghavan, A.: Automated energy monitoring of machine tools. CIRP Ann. Manuf. Technol. **59**, 21–24 (2010)
9. Küçük, D., et al.: PQStream : a data stream architecture for electrical power quality (2015). ArXiv: abs/1504.04750
10. Lobo, J.L., Ballesteros, I., Oregi, I., Del Ser, J., Salcedo-Sanz, S.: Stream learning in energy IoT systems: a case study in combined cycle power plants. Energies **13**, 740 (2020). https://doi.org/10.3390/en13030740
11. Pitarch, Y., et al.: Multidimensional data streams summarization using extended tilted time windows. In: INA: Frontiers of Information Systems and Network Applications, Bradford, United Kingdom. pp. 102–106, 1–20, May 2009
12. Kreps, J., Narkhede, N., Rao, J.: Kafka: a distributed messaging system for log processing. In: ACM NetDB 2011, Athens, Greece, June 12 2011 Vohra, Deepak. (2016). Apache Flume. https://doi.org/10.1007/978-1-4842-2199-0_6. 978-1-4503-0652-2/11/06.10.00

13. Vohra, D.: Apache flume, pp. 287–300 (2016). ISBN: 978-1-4842-2198-3. https://doi.org/10.1007/978-1-4842-2199-0_6
14. Sarr, J.G., Boly, A., Bame, N., et al.: Data stream summary in big data context: challenges and opportunities. Adv. Sci. Technol. Eng. Syst. J. **6**(4), 414–430 (2021)
15. Ahmed, R., Khatun, A., Ali, A., Sundaraj, K.: A literature review on NoSQL database for big data processing **7**(2) (2018). https://doi.org/10.14419/ijet.v7i2.12113
16. Zheng, T., Chen, G., Wang, X., Chen, C., Wang, X., Luo, S.: Real-time intelligent big data processing: technology, platform, and applications. Sci. China Inf. Sci. **62**(8), 1–12 (2019). https://doi.org/10.1007/s11432-018-9834-8
17. Duraj, A., Szczepaniak, P.S.: Outlier detection in data streams - a comparative study of selected methods, procedia computer science; knowledge-based and intelligent information & engineering systems. In: Proceedings of the 25th International Conference KES2021, vol. 192, pp. 2769–2778, ISSN, pp. 1877–0509 (2021). https://doi.org/10.1016/j.procs.2021.09.047, https://www.sciencedirect.com/science/article/pii/S1877050921017841
18. Khalid, M., Kjell, O., Tore, R.: Wrapping a NoSQL datastore for stream analytics, pp. 301–305, August 2020. https://doi.org/10.1109/IRI49571.2020.00050
19. Mahmood, K., Orsborn, K., Risch, T.: Comparison of NoSQL datastores for large scale data stream log analytics, pp. 478–480, June 2019. https://doi.org/10.1109/SMARTCOMP.2019.00093
20. Shaikh, S.A., Kitagawa, H., Matono, A., Mariam, K., Kim, K. -S.: GeoFlink: an efficient and scalable spatial data stream management system. In: IEEE Access, J. Name Stand. Abbrev, vol. 10, pp. 24909–24935 (2022). (In Press) https://doi.org/10.1109/ACCESS.2022.3154063.e
21. Hoseiny Farahabady, M., Taheri, J., Zomaya, A.Y., Tari, Z., Energy efficient resource controller for apache storm. Concurr. Comput. Pract. Exp. (2021). https://doi.org/10.1002/cpe.6799
22. Amrapali, B., Upadhyay, A.K.: Microsoft power BI. Int. J. Soft Comput. Eng. (IJSCE) **7**(3), 14–20 (2017). ISSN: 2231–2307