Ontology Driven Conceptualization of Context-Dependent Data Streams and Streaming Databases

Shreya Banerjee, Anirban Sarkar

Ontology Driven Conceptualization of Context-dependent Data Streams and Streaming Databases

[1]Shreya Banerjee, [2]Anirban Sarkar

[1,2]Department of Computer Applications, National Institute of Technology, Durgapur, India
[1]shreya.banerjee85@gmail.com, [2]sarkar.anirban@gmail.com

Abstract. Heterogeneous stream formats, related contexts, vocabularies and schema structures are key difficulties to facilitate sharing and extracting knowledge from stream databases. To resolve these heterogeneities, the key challenge is how to provide common semantic representation for context-dependent data stream formats along with streaming databases. To address such issues, this paper proposes an ontology driven formal semantics of context-dependent data streams together with a universal conceptualization of streaming databases. The novelty of this work is to handle heterogeneity, large volume and availability of streaming data, such as web content, commercial broadcasting data etc. It also facilitates to recognize evolving information from semantic representation of data streams at conceptual modelling level. Besides, the proposed conceptual model is flexible to represent finite partition of stream and thus help in data stream storing and further querying. The conceptualization is implemented using an ontology editorial tool Protégé for the initial validation of proposed set of formal semantics. Several crucial properties of the proposed conceptualization are specified in order to exhibit the benefits of the proposed work. The expressiveness of proposed model is illustrated using a suitable case study.

Keywords: Data Stream, Conceptual model, Context Modelling, Streaming Databases, Ontology

1 Introduction

In recent years, with the advancement of information and web technology, several applications need to work with continuous data generating processes. Those data are dynamic, time sensitive and continuous in nature. Data generated from Web-clicks, network monitoring, commercial broadcasting, sensor nets and stock quotes are few examples of such data [7]. These types of data are considered as a stream (data stream) rather than static snapshots [6]. Distinct *Data Stream Management Systems (DSMS)* are developed for processing and analysis of these data streams. Those DSMS are built due to limitations of traditional data management systems towards managing distinct data streams [2]. Hence, a well-organized model of data streams is the key requirement for proficient management of those data streams by DSMS. However, data streams have several exceptional characteristics, which make them difficult to model. *Firstly*, a data stream is usually defined as "an unbounded sequence of values continuously appended,

each of which carries a time stamp that typically indicates when it has been produced" [6]. *Secondly*, in different applications these continuous data are represented in different ways such as a discrete signal, an event log or a combination of trained series [15]. *Thirdly*, rapid changing of underlying contextual information of data streams generated in diverse domains has serious consequences in deriving useful decisions from complex real time applications [4]. *Fourthly*, distinct back-end databases ranging from strict schema-based (for example Relational Databases) to flexible schema based (for example NoSQL Databases) are used to store theses data streams in structured, semi-structured or unstructured way. *Finally*, a fixed or flexible finite partition, called *window*, are made from this continuous unbounded sequence while streams have to be stored or retrieved from databases [11]. Hence, several challenges exists in efficient modelling of data streams in order to facilitate sharing of information related to data streams across different applications and DSMS. *Starting with*, how to represent common description of heterogeneous data streams semantically and syntactically. *Secondly*, how different surrounding contexts (contextual information) of data streams are represented in a uniform way. *Thirdly*, how evolving contexts of data streams can be recognized so that realization of dynamically added contextual information towards data streams is achieved efficiently. To handle these issues, ontology will be beneficial. The key reason for applying ontology is that it can establish consensus on unifying conceptualization of heterogeneous data stream formats and related contexts. Ontology is defined as a formal, explicit specification of shared conceptualization in terms of concepts, relationships present between those concepts and related axioms [8].

Existing research works, primarily, focus on semantic representations of resources and devices producing data streams. However, less attention is paid towards uniform semantic representation of distinct context dependent data streams and further heterogeneous streaming databases. In [1, 3, 7, 15], authors have described abstract semantics of streams. Authors in [7] have described an extensible framework that facilitates experimenting with different algorithms related with data stream mining tasks. In [2, 11], authors have described powerful operator algebra for data streams. Both of these approaches have facilitated in supporting multiple query languages and data models. Semantic Sensor Network (SSN) ontology [5] represents a high-level general schema of sensor systems. IoT-A and IoT.est [12] provide architectural base for utilization and representation of domain knowledge in sensor networks with some services and test concepts. The Observation & Measurement (O&M) description of sensory Data are described as a part of Sensor Web Enablement (SWE) standards from the Open Geospatial Consortium (OGC) [9]. However, this description is based on XML (Extensible Markup Language) which has weak semantic structure for expressing and describing stream data ontology in more detail. Through approaches regarding Semantic Sensor Web (SSW), context information such as time, space is added with sensors. However, these approaches are mostly specific to certain domain and thus are not in high-level abstraction [13]. Besides, none of these approaches has explored the representation of contextual information related to data streams and streaming databases.

To address aforementioned challenges regarding modelling of data streams, an effort has been made in this paper to provide precise semantics towards data streams, related

contexts and streaming databases. For this purpose, an ontology driven conceptualization of data streams along with its related context is devised. The novelties of the proposed ontology driven conceptual model are many-folds. The proposed conceptualization efficiently deals with generic semantics towards modelling of variety of data streams, resources producing those data streams and streaming databases. It further facilitates in sharing and preserving strong interoperability in heterogeneous applications and DSMS. Next, the proposed conceptualization aids in recognizing static and evolving contextual information related to data streams along with a set of distinct relationships. This essence of context sensitivity approach helps in reducing search spaces during the time of querying on data streams. Besides, the proposed conceptual model may assist in future in the extraction of new knowledge from data streams since it is ontology driven and hence based on Open World Assumption (OWA) [8]. Moreover, it has also provided discreetness in continuous streaming by representing finite, indefinite, fixed and flexible partition of data streams.

2 Proposed Ontology Driven Conceptual Model for Context-Dependent Data Streams

The proposed conceptual model formalizes a common set of constructs and relationships for conceptualization of context-dependent data streams and streaming databases. The proposed model comprises of three interrelated layers (*Collection*, *Family* and *Stream*) and their identifiable construct types. Besides, the constructs are related with each other using different relationships. The proposed model is specified axiomatically using both first order and higher order logic to represent semantics of data-stream constructs and their interrelationships. The key constructs and distinct relationships of the proposed model are specified in Fig. 1. In this figure, *Collection*, *Family* and *Stream* layers are represented using shapes of rectangle, rounded rectangle and oval respectively. Details of the proposed model are specified in following sections.

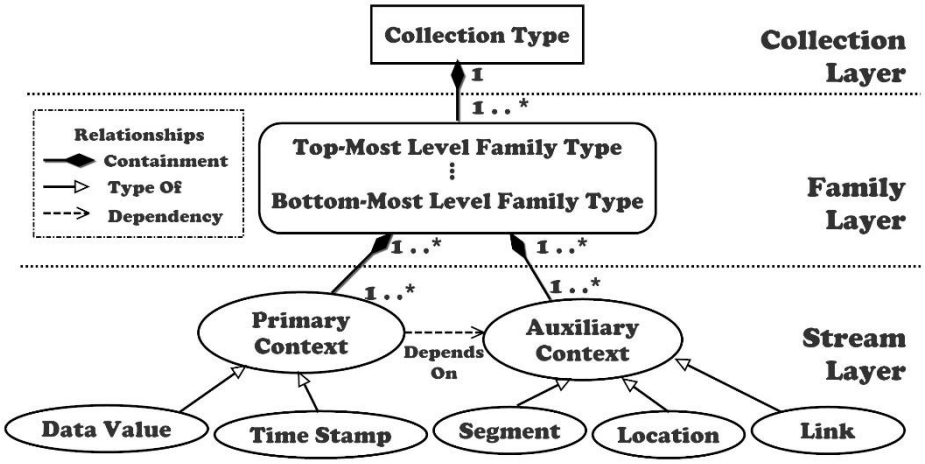

Fig. 1. Proposed Conceptual Model for Data-Streams

2.1 Constructs in Proposed Conceptual Model

Proposed model consists of three main layers, namely, *Collection*, *Family* and *Stream*. These three layers have their respective construct types *Collection* (*Col*), *Family* (*FA*) and *Stream_Context* (*Str*). Formal axioms of these constructs are specified below and different interrelationship among those constructs are described in section 2.3.

(*a*) *Collection Layer*: It is the upper-most layer of the data model. *Collections(Col)* are main identifiable constructs of this layer. Semantically related *Families(FA)* (Intermidiate layer of the proposed model) are grouped together to form a *Collection(Col)*.

F1: $\forall x \exists l \exists c \exists v (Col(x) \leftrightarrow (HT(c) \wedge FA(l) \wedge Cnt_{col}(v))$

Explanation: Here, *HT* is *Has Time* relationship and Cnt_{col} is *Containment relationship*. F14 and F19 formalizes *Containment* and *Has Time* relationship. Further, *x, l, c, v* are instances of *Collections*, *Family*, *Has Time* and *Containment* relationship.

(*b*) *Family Layer*: It is the intermediate layer of the conceptual model. *Families* are main identifiable constructs of this layer. This layer may be composed of number of levels to reflect the fact of continuous encapsulation of data. Further, the lowest level of *Family* layer may be combined of semantically related data streams and its contexts.

F2: $\forall x \exists u \exists a \exists r \exists m \exists d \exists l \exists c \exists v \exists n (FA(x) \leftrightarrow (FA_{llev}(u) \vee FA_{ulev}(a) \vee FA_{lev}(r) \vee Icnt_{FA}(m) \vee Col(d)) \wedge HT(l) \wedge Cnt_{FA}(c) \wedge primary\_context(v) \wedge auxiliary\_context(n)))$

Explanation: Here, FA_{llev}, FA_{ulev} and FA_{lev} are denoted as *Families* in the bottom-most level, in the top-most level and in any level respectively. *IcntFA* is *Inverse Containmnet* and *CntFA* is *Containment relationship*. Later, in section 2.3 axiom *F17* formalizes *Inverse Containment* relationship. Further, *primary_context()* and *auxiliary_context()* are predicates representing *Primary Contexts* and *Auxiliary Contexts* of Stream. Related axioms are specified in axioms *F3*, *F4*, *F5* and *F6*.

(c) *Stream_Context Layer*: This is the lower-most layer of the proposed conceptual model. Data-Streams may be represented in this layer formally. Data stream is an indefinite ordered sequence of data points, each of which carry a time stamp. These data points can be ranged from structured to unstructured type. Besides this, these data points may be related with precise contextual information that are useful to characterize the features of streams which are necessary in order to interact between users and applications. Detailed formalizations of *Stream_Context* are specified in section 2.2.

2.2 Conceptualization of Stream Context

Stream Context is precise information useful to describe data stream, and its surrounding concepts. Two types of contexts are used to represent data streams in *Stream_Context* layer. One is *Primary Context* (*PC*) and another is *Auxiliary Context* (*AC*) that may provide useful information towards *Primary Contexts*. Hence, a data-stream may be defined as an ordered indefinite sequence of *Primary Contexts* and its related *Auxiliary Contexts*. Related axiom is

F3: $\forall x \exists y \exists z \exists t1 \exists t2 (stream(x) \leftrightarrow ((\cup union\_of((\square\, PC(data\_value(y) \wedge time\_stamp(t1)) \wedge AC(z)), (\square\, PC(data\_value(y) \wedge time\_stamp(t2)) \wedge AC(z)))) \wedge (t1 < t2) \wedge (number\_of\_data\_value(\infty))))$

Explanation: Here, *number_of_data_value()* is a predicate implying that a data-stream may be infinite; *data_value()* and *time_stamp()* are predicates implying data values and their corresponding time stamps respectively; □ operator implies mandatory participation of the argument and *Union_of()* is a predicate implying the union of arguments. Axioms F4 and F6 formalize the *Primary* and *Auxiliary Context* respectively.

(*a*) *Primary Context*: This represents basic information about data stream. Basic information of data stream mandatorily includes the data value at a specific time. The data value and its related specific time collectively can be called as a *Frame*.

F4: $\forall x \exists v \exists t1 \exists t2 \exists t(PC(x) \leftrightarrow (\square(data\_value(x,v) \wedge starting\_time(t1) \wedge ending\_time(t2) \wedge (existence(x) \leftrightarrow (t1 \wedge t2 \wedge duration(t1,t2))) \wedge ((\neg existence(x) \wedge time(t)) \leftrightarrow (t \wedge (t < t1) \wedge (t > t2))))))$

F5: $\forall x \exists v \exists t(Frame(x) \leftrightarrow (\square\, PC(data\_value(y) \wedge time\_stamp(t1))))$

Explanation: Here, *starting_time()* and *ending_time()* are predicates implying start time and end time of respective arguments. Further, predicate *existence()* implies the existence time duration of the argument.

(*b*) *Auxiliary Context*: This context provides additional information relevant for *Primary Context*. For example, let assume humidity sensor generates data stream of humidity values. Then location may be an auxiliary context related to the primary context humidity. *Auxiliary Context* can be of several types as specified below.

F6: $\forall x \exists a1 \exists a2 \exists a3((PC(x) \wedge AC(a1) \wedge AC(a2) \wedge AC(a3)) \leftrightarrow ((pair(a1,a2,a3)) \rightarrow x))$

Explanation: Here, *pair()* is a predicate implying the pairing of the respective *Auxiliary Contexts*.

(*i*) *Segment Context*: *Segment* represents a finite partition of the data-stream containing ordered sequences of *Primary Contexts* when the stream is to be going to store in a database. This size of partition may be fixed or flexible depending on the number of instances of a *Frame*. Axiom *F5* formally represent *Frame*. Axioms related to *Segment Context* are specified below.

F7: $\forall x \exists y \exists z \exists m(Segment(x) \leftrightarrow (stream\_store((Union\_of((\square\, Frame(y) \wedge Other\_AC(z)) \wedge (\square\, Frame(m) \wedge Other\_AC(z)))) \wedge (t1 < t2) \wedge number\_of\_data\_value(finite))))$

F8: $\forall x((segment(x) \wedge size(x)) \leftrightarrow (number\_of\_instance(Frame))$

F9: $\forall x \exists y \exists l((PC(x) \wedge segment(y) \wedge has\_auxiliary\_context(l)) \leftrightarrow l(x,y))$

Explanation: Here, *has_auxiliary_context()* has created a relationship instance *l* that has attached *Primary Context* with *Auxiliary_Context*. Details of this relationship are stated in section 2.3. Besides, *size()* is a predicate implying the length of the *Segment*.

(*ii*) *Location Context*: This represents the current location/resources those holding the *Primary Context* with a specific time stamp. Assume, *L* is the set of locations. Related axioms are specified below.

F10: $\forall l((l \in Location) \leftrightarrow Q(L(Location)))$

Explanation: Here, *Q* is a predicate and *L* (*Location*) is a function returning locations of a *Primary Context*.

F11: $\forall x \exists y \exists t((PC(x) \wedge location(y) \wedge has\_auxiliary\_context(l)) \leftrightarrow l(x,y))$

(*iii*) *Link Context*: This represents how the *Primary Contexts* are communicated over communication channel in terms of simplex, duplex etc. Besides, multiple segments of

similar or multiple data streams may be available in communication channel following some sequences. Hence, segment wise communications may be present within both of single stream and multiple data streams.

F12: $\forall x \exists y \exists z \exists l \exists k \exists m \exists n((Link(x) \wedge stream(l) \wedge stream(k) \wedge l(segment(PC(y) \wedge PC(z))) \wedge k(segment(PC(m) \wedge PC(n)))) \leftrightarrow ((Communication\_type\_between(((PC(y) \wedge PC(z)) \vee ((PC(m) \wedge PC(n)))) \vee (communication\_type\_between((PC(y) \vee PC(z)) \wedge (PC(m) \vee PC(n))))))$

F13: $\forall x \exists y((PC(x) \wedge Link(y) \wedge has\_auxiliary\_context(l)) \leftrightarrow l(x, y))$

Explanation: Here, *Communication_type_between()* is a predicate implying the type of communication between *Segments* of single or multiple *Primary Contexts*.

These proposed *Auxiliary Contexts* are of minimal set. More distinct *Auxiliary Contexts* may be appended towards *Primary Contexts* based on design demand. Hence, proposed conceptualization realizes both static and evolving contextual information.

(*c*) *Finite Partition of Data Stream*: *Segment* has represented the finite partition of infinite data-streams for storing data-streams in database. Similarly, for the retrieval purpose another finite partition of data-streams can be defined as a *Window*. The size of *Window* may be fixed or flexible depending on numbers of instances of time stamps. Later, different data-stream query operators can be defined on this *Window*. The axiom of *Window* is as follows.

F14: $\forall x \exists y \exists z \exists m(Window(x) \leftrightarrow (stream\_retrieve((Union\_of((\square\, Frame(y) \wedge Other\_AC(z)) \wedge (\square\, Frame(m) \wedge Other\_AC(z)))) \wedge (t1 < t2) \wedge number\_of\_data\_value(finite))))$

F15: $\forall x((window(x) \wedge size(x)) \leftrightarrow (number\_of\_instance(Frame))$

2.3 Relationships in Proposed Conceptual Model

Distinct constructs of proposed conceptual model are interrelated. These relationships can be of two types – *Inter layer* and *Intra layer* [14]. *Inter-layer relationships* can be between dissimilar construct types of three different layers. *Intra-layer relationships* can be between similar construct types of identical layer. Different relationships may be present within a data stream, data stream and its related contextual information, and in the layer hierarchy of streaming databases. These relationships may be *Containment*, *Inverse Containment*, *Has_auxiliary_Context*, *Reverse_has_auxiliary_context*, *Sequence*, and *HasTime*. Former two are of *Inter-layer* and *Intra-layer* kind of relationship and the rest all are of *Intra-layer* kind of relationship.

(*a*) *Containment* (*Cnt*): *Containment* relationships can be present between two construct types when one encapsulates similar or different types of constructs.

F16: $\forall x(Cnt(x) \leftrightarrow \exists y \exists^{n} z(Cmp(y) \wedge Cmp(z) \wedge ((sl(y, z) \wedge lev(y) \wedge lev_{Next}(z)) \vee (dl(y, z) \wedge layer(y) \wedge layer_{Next}(z))) \wedge x(y, z) \wedge \neg(y = z) \wedge (k(range(x) = z)) \wedge (m(domain(x) = y)) \wedge ((p(value(k) = 1) \vee greaterthan(value(k), 1)))))$

Explanation: Let, *sl()* is a predicate. Arguments of *sl()* are construct types (*Cmp*) and express the fact that they are of similar type. In contrast, *dl()* is a predicate that takes *Cmp* (construct types) as arguments and articulates that its arguments are in different layers and are of dissimilar types. Besides, *range()* and *domain()* are functions returning

target and source domain of relationships respectively; *k* and *m* are predicates specifying those functions; *value()* is a function returning number of instances encapsulated; *greaterthan()* is a predicate implying whether the first argument is greater than second argument; *lev()* and $lev_{Next}()$ are predicates implying whether arguments of these belongs to a level and its next lower level respectively; *layer()* and $layer_{Next}()$ are predicates implying whether arguments of these belongs to a layer or its next lower layer respectively; and *p* is a predicate.

(*b*) *Inverse Containment* (*Icnt*): This relationship enables one construct type to de-encapsulate itself in order to encapsulate *Families* towards *Collections* dynamically.

F17: $\forall x(Icnt(x) \leftrightarrow \exists y \exists^{n} z(Cmp(y) \wedge Cmp(z) \wedge ((FA(y,z) \wedge lev(y) \wedge lev_{Next}(z)) \vee (Col(y) \wedge FA(z) \wedge layer(y) \wedge layer_{Next}(z))) \wedge x(z,y) \wedge \neg(y = z) \wedge (k(range(x) = y)) \wedge (m(domain(x) = z)) \wedge ((p(value(k) = 1) \vee greaterthan(value(k), 1)))))$

(*c*) *Has_auxiliary_Context* (*HAC*): This relationship connects *Primary Context* with *Auxiliary Context*.

F18: $\forall x \exists y \exists l((PC(x) \wedge AC(y) \wedge HAC(l)) \leftrightarrow l(x,y))$

(*d*) *Reverse_has_auxiliary_context* (*RHAC*): This relationship may connect *Auxiliary Contexts* with *Primary Contexts* dynamically. This relationship is in reverse order of *Has_auxiliary_Context*.

F19: $\forall x \exists y \exists l((PC(x) \wedge AC(y) \wedge RHAC(l)) \leftrightarrow l(x,y))$

(*e*) *Sequence* (*Seq*): This is the relationship between two or more *Primary Contexts* when the data value of a particular time stamp is connected with the data value of the successive time stamp.

F20: $\forall x \exists f1 \exists f2 \exists f3 \exists y1 \exists y2 \exists y3 \exists l \exists t1 \exists t2((Frame(f1, f2, f3) \wedge data\_value(y1, y2, y3) \wedge time(t1, t2, t3) \wedge f1(y1 \wedge t1) \wedge f2(y2 \wedge t2) \wedge f3(y3 \wedge t3) \wedge Seq(l) \wedge (t1 < t2 < t3)) \leftrightarrow (l(y1, y2, y3) \wedge \neg(y1 = y2 = y3) \wedge (((succecive\_order(y1, y2) \wedge succecive\_order(y2, y3)) \rightarrow succecive\_order(y1, y3))) \wedge ((succecive\_order(y1, y2)) \rightarrow \neg(succecive\_order(y2, y1)))))$

Explanation: Here, *succecive_order()* is a predicate implying that the second argument is coming in next sequence of the first argument. Contrary, by reversing the order of arguments of the predicate *succecive_order()*, flow of the data stream can be realized in the reverse direction. In this way, dynamically appended data values towards data streams can be recognized.

(*f*) *Has Time* (*HT*): This relationship represents the connection between data value and its existence time stamps. The axiom of this relationship is

F21: $\exists i \exists j[HT(i) \leftrightarrow \exists t \exists x[(t \in Tm) \wedge data\_value(x) \wedge i(x,t)]$

Explanation: Here, *Tm* is a set of timestamps.

The proposed conceptual model thus represents formal and universal vocabularies of context-dependent data streams and streaming databases. Using the axioms of stream layer, semantics of data stream and its associated heterogeneous context is specified. Likewise, through the entire layer hierarchy the proposed model is capable to represent common conceptualization of different streaming databases ranging from strict to flexible schema based. Thus, the proposed conceptualization deals with heterogeneity issue of data streams. Further, the proposed conceptual model is in high level abstraction.

Hence, representation of large volume of data streams can be managed using this proposed conceptualization efficiently in conceptual level. Besides, using *Has_auxiliary_Context* and *Inverse Containment* relationships dynamically added contextual information towards the domain have been recognized. In this way, the proposed conceptualization may facilitate in future in deriving new knowledge from data streams. Further, using *Sequence* relationship the rapid availability of data points towards data stream is realized. Furthermore, *Segment* and *Window* partition has facilitated in realizing discreteness among continuous stream. Moreover, the proposed conceptualization model is flexible as it provides flexible finite size towards data-streams using *Segment* and *Window*. It has also recognized the communication and available sequences between *Segments* or *Windows* of similar or multiple data-streams. Several other related crucial features are described in section 5.

3 Protégé Implementation of the Proposed Model

The proposed meta-model has been implemented in this section using OWL (Web Ontology Language) based ontology editorial tool Protégé [10]. Protégé facilitates representation of formally expressed axiom set of this proposed conceptualization towards OWL logic. It is composed of a number of reasoners for automated inference on ontological theory expressed in OWL logic. OWL is based on Description Logic.

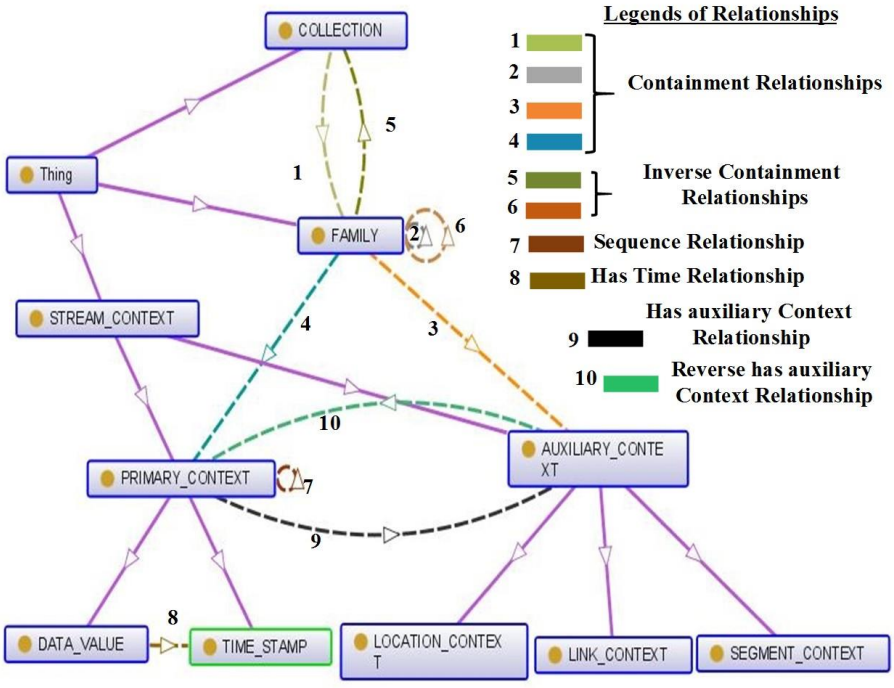

Fig. 2. Ontological graph of the proposed Conceptual Model using OntoGraf plug-in in Protégé

Table 1. Mapping from Proposed Conceptual Model towards Protégé

Constructs in Proposed Conceptualization	Corresponding Constructs in Protégé
Collection	COLLECTION
Family	FAMILY
Stream Context	STREAM_CONTEXT
Primary Context	PRIMARY_CONTEXT
Auxiliary Context	AUXILIARY_CONTEXT
Segment Context	SEGMENT_CONTEXT
Link	LINK_CONTEXT
Location Context	LOCATION_CONTEXT
data_value	DATA_VALUE
time_stamp	TIME_STAMP
Containment	INTER_CONTAINMENT
	INTRA_CONTAINMENT
Inverse Containment	INTRA_INVERSE_CONTAINMET
	INTER_INVERSE_CONTAINMENT
Has_auxiliary_context	HAS_AUXILIARY_CONTEXT
Reverse_has_auxiliary_context	REVERSE_HAS_AUXILIARY_CONTEXT
Sequence	SEQUENCE
Has Time	HAS TIME

Three layers and their construct types have been mapped towards Protégé Classes. Besides, six key relationships of the proposed conceptualization are specified as Object Properties in Protégé. The mapping from the proposed conceptualization towards Protégé is specified in Table 1. Further, several Object Properties in the proposed conceptualization may have multiple sub object properties. Such as INTER_CONTAINMENT has sub property called INTER_CONTAINMENT_COLLECTION_FAMILY. Fig 2 describes the graph obtained through OntoGraf plug-in in Protégé.

4 Illustration of the Proposed Conceptual Model

Let, an application is aimed to determine whether a car-driver is relaxed or stressed when the driver has to drive in a predefined route from one starting point to a specific destination and return to the starting point within predefined time duration. Besides, drivers are warned about the remaining time to reach the destination. Five sensor signals - Heart Rate (HR), Finger Temperature (FT), Respiration Rate (RR), Carbon-di-oxide (CO2) and Oxygen Saturation (SpO2) have been recorded. This case study has been adopted from [4].

In this case study, all five sensor signals are of stream data Heart Rate (HR) sensor's primary context is recorded heartbeat. Heartbeats have specific values in a specific time. Besides, Heart Rate is dependent on particular location of the driver. Similarly, other sensors' recorded data have specific values in specific time. Further, all of them have auxiliary contexts. Such as Heart Rate has auxiliary contexts location, age, weight etc. According to this case study, the driver's recorded sensor data will be a *Collection*. Driver will be a *Family*. Further, driver has five *Primary Contexts – Heartbeat*, *Finger Temperature, Respiration Rate, Carbon di oxide* and *Oxygen Saturation*. Each *Primary*

Context has values and related time stamps. Besides, all are related to *Auxiliary Context* such as location, body size. Key elements of this case study have been listed below.

Drivers' recorded Data (*Driver*)

Driver (*Heart Rate*, *Finger Temperature*, *Respiration Rate*, *Carbon di Oxide*, *Oxygen Saturation*);

Heart Rate ({heartbeat, time stamp}, {BODY_WEIGHT, AGE, BODY_POSITION, GENDER, MEDICAL_HISTORY, DRIVER'S LOCATION});

Finger Temperature ({temperature, time stamp}, {AGE, GENDER, MEDICAL_HISTORY, DRIVER'S LOCATION});

Respiration Rate ({respiration_rate, time stamp}, {GENDER, MUSCLE TYPE, DRIVER'S LOCATION});

CO2 ({amount of CO2, time stamp}, {DRIVER'S LOCATION});

SpO2 ({amount of SpO2, time stamp}, {DRIVER'S LOCATION})

Nomenclatures of key elements in the case study are represented as, (i) Collections are in "**bold**" letters; (ii) Families are in "*italics*" letters; (iii) Primary Stream Contexts are in "small" letter cases; and (iv) Auxiliary Stream Contexts are in "CAPITAL" letter. In this section, the case study has been implemented using the ontology editorial tool Protégé. Key constructs of the case study have been mapped towards Protégé as specified in section 2. Fig 3 is displaying the partial ontology graph of this case study showing only heart rate stream along with its auxiliary contexts. The graph is obtained through OntoGraf plug-in of Protégé.

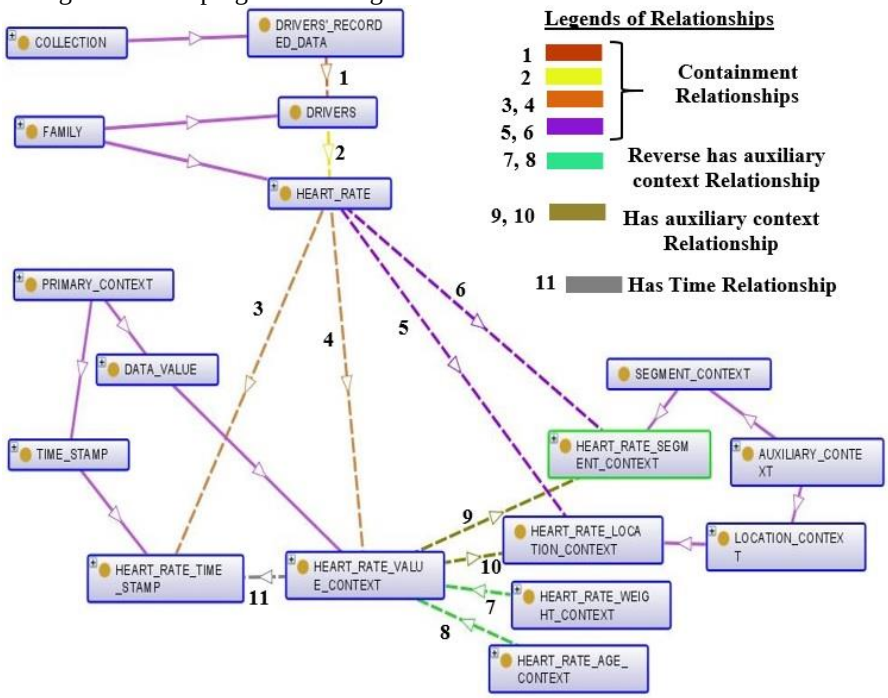

Fig. 3. The partial ontological graph displaying Primary Context of Heart Rate using Auxiliary Context and obtained through OntoGraf plug-in in Protégé

5 Features of Proposed Conceptualization

Proposed conceptualization possess several crucial features. Those features are *Abstraction* and *Reusability*, *Adaptability*, *Flexibility*, *Interoperability*, *Productivity* and *Context Sensitivity*.

(*i*) *Abstraction* and *Reusability*: Proposed conceptualization is in high-level abstraction due to representation of data streams independent of any domain. Hence, it is reused in large numbers of domain.

(*ii*) *Adaptability*: Proposed conceptualization is able to recognize evolving contextual information using *Reverse_has_auxiliary_context*. Thus, it is adaptable towards changing surrounding environment.

(iii) *Flexibility*: Using this proposed conceptualization bounded, unbounded, fixed and flexible partition of bounded sequence of data streams are represented through Frame, Segment and Window. In this way, the proposed conceptualization is flexible.

(*iv*) *Interoperability*: With the aid of generic formal semantics, proposed conceptualization provides interoperable uniform representation towards heterogeneous data streams and streaming databases.

(*v*) *Productivity*: The proposed conceptualization is productive as through this specification compatibility among different heterogeneous data streams, streaming databases and applications can be maximized.

(*vi*) *Context Sensitivity*: The proposed conceptualization is able to recognize related contextual information of both data streams and resources. Thus, the proposed model is context sensitive. This further facilitates validation and analysis of data streams.

6 Conclusion and Future Work

The paper has proposed an ontology driven common semantics towards context-dependent data streams and heterogeneous streaming databases. The objective of the proposed work is to model data streams and related contexts in a uniform way so that strong interoperability can be sustained among heterogeneous applications utilizing data streams. The novelty of the proposed ontology driven conceptualization is to support in realization of continuous temporal nature, static and evolving contexts related to data streams, homogeneity in heterogeneity formats, and rapid availability of data streams. The proposed conceptualization is capable to provide generic semantics towards contents of data streams and resources producing those data streams. Further, the proposed conceptualization is flexible enough to represent discreteness within infinite data streams and provide choices of fixed or variable partitions of data streams for storing and retrieval purpose. In this way, the proposed conceptualization may facilitate in future in deriving knowledge and decisions from data streams.

Future work will include semantical validation of the proposed ontology driven conceptualization of stream data. Further, ontology driven formal specification of a query language for retrieval of data streams is another important future work.

References

1. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. The VLDB Journal—The Int. J. on Very Large Data Bases. 12(2), 120-139 (2003).
2. Appelrath, H., Geesen, D., Grawunder, M., Michelsen, T., Nicklas D.: Odysseus: a highly customizable framework for creating efficient event stream management systems. In: 6th ACM Int. Conf. on Distributed Event-Based Systems, pp. 367-368. ACM, New York (2012).
3. Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J.: Stream: The stanford data stream management system. Technical Report 641, Stanford InfoLab (2004).
4. Begum, S., Barua, S., Filla, R., Ahmed, MU. : Physiological sensor signals classification for healthcare using sensor data fusion and case-based reasoning. Expert systems with applications. 41(2), 295-305 (2014).
5. Compton, M., Barnaghi, P., Bermudez, L., Castro, R.G., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W. D., Phuoc, D. L., Lefort, L., Leggieri, M., Neuhaus, H., Nikilov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web. 17, 25-32 (2012).
6. Ediger, D., Jiang, K., Riedy, J., Bader, D. A.: Massive Streaming Data Analytics: A Case Study with Clustering Coefficients. In: 2010 IEEE Int. Sympo. on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW, 2010), pp. 1-8 (2010).
7. Forrest, J.: stream: A Framework for Data Stream Modeling in R. B.S. Thesis, Southern Methodist University, Dallas, TX, United States, (2011).
8. Guarino, N., Oberle, D., Staab, S.: What is an Ontology?. Handbook on Ontologies, second edition, ed: Springer-Verlag, pp. 1-17 (2009).
9. Henson, C. A., Neuhaus, H., Sheth, A. P., Thirunarayan, K., Buyya, R.: An Ontological Representation of Time Series Observations on the Semantic Sensor Web. In: 1st Int. Workshop on the Semantic Sensor Web (SemSensWeb 2009), pp. 79-94 (2009).
10. Horridge, M.: "A Practical Guide to Building OWL Ontologies Using Protégé 4 and COODETools", Edition 1.3, The university of Manchester (2011). https://mariaiulianadascalu.files.wordpress.com/2014/02/owl-cs-manchester-ac-uk_-eowltutorialp4_v1_3.pdf.
11. Kramer, J., Seeger, B.: Semantics and implementation of continuous sliding window queries over data streams. ACM Trans. on Database Systems (TODS). 34(1), 4:1- 4:49 (2009).
12. Wang, W., De, S., Toenjes, R., Reetz, E., Moessner, K.: A Comprehensive Ontology for Knowledge Representation in the Internet of Things. In: 11th Int. Conf. on Trust, Security and Privacy in Computing and Communications (IEEE TRUSTCOM), pp. 1793-1798. IEEE Computer Society (2012).
13. Wang, X., Zhang, X., Li, M.: A Survey on Semantic Sensor Web: Sensor Ontology, Mapping and Query. Int. J. of u- and e- Service, Science and Technology, 8(10), 325-342 (2015).
14. Banerjee, S. Sarkar, A.: Ontology Driven Meta-Modeling for NoSQL Databases: A Conceptual Perspective. Int. J. of Software Engg. and Its Applications, Science & Engineering Research Support Society (SERSC). 10(12), 41-64 (2016).
15. Zaniolo, C.: Logical foundations of continuous query languages for data streams. In: Second Int. Workshop, Datalog 2.0, pp. 177-189. Springer-Verlag Berlin Heidelberg (2012).