International Journal of Computational Intelligence Systems
Vol. **12(1)**; 2019, *pp.* 311–317
DOI: https://doi.org/10.2991/ijcis.2019.125905648; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

A Validation Approach for Ontology-Based Real-time DBMS

Wided Ben Abid*, Mohamed Ben Ahmed Mhiri, Malek Ben Salem, Emna Bouazizi, Faiez Gargouri

Laboratory MIRACL, University of Sfax, The Higher Institute of Computer Science and Multimedia, Sfax-3021, Tunisia

ARTICLE INFO

Article History
Received 15 Apr 2018
Accepted 11 Jan 2019

Keywords
Real-time DBMS
Ontology-based databases
Real-time ontology
Quality of service management
Feedback control scheduling

ABSTRACT

Real-time DBMS (DataBase Management Systems) are an appropriate storage system under real-time constraints. However real-time DBMS do not implement inference or reasoning mechanisms. Ontologies on the other hand allow these mechanisms by creating formal representations of concepts, properties and relationships between concepts. Traditionally ontologies have been applied to static domains, in the sense that entities represented in these ontologies do not change over time. However objects in real time databases are dynamic; it is possible that an object changes their value along time. To overcome this limitation, in this paper, we propose a Feedback Control Scheduling Architecture for Real-Time Ontology that develops the temporal aspect within an ontology in the real time databases and that guarantee of Quality of Service (QoS) for real-time DBMS under unpredictable workloads.

1. INTRODUCTION

Many real-time applications are becoming very sophisticated in their data needs such as information retrieval systems, airline reservation systems, aircraft and spacecraft control, robotics, and so on. Real-time DBMS (DataBase Management Systems) are an appropriate formalism to handle such applications. These systems are designed to maintain the logical consistency of databases and to ensure a compliance with timing constraints. However, these systems are limited to describe concepts, without expressing their semantics. For these, several studies used the ontologies in the field of database to provide an adequate representation of needs, negotiate semantics of the concepts in a domain, share heterogeneous data, represent a domain in an explicit way without ambiguity, and deduce a new knowledge.

Furthermore, the massive use of ontologies generates a big amount of semantic data. To facilitate their managements, persistent solutions to store and query these loads of semantic data were proposed. These gave raise to a new type of databases called ontology-based databases (OBDBs). Several OBDBs have been proposed, such as RDFSuite [1], OntoMS [2], OntoDB [3], and OntoDB2 [4]. All the OBDBs architectures do not handle the temporal aspect forthe advanced applications. In this case, the designer of a database application must reconcile the ontological concepts of the real-time application domain with the ontology constructed in these databases. For that purpose, we will need to propose a new approach allows to fill in the limits of these studies and to suggest a method to conceptualize the ontology-based real-time databases (OBRTDB). This approach needs to integrate an ontology representing the real-time domain into the design process of a real-time database in which we will introduce a specific management of the real-time ontological data based on techniques of scheduling with feedback control. It consists in proposing an architecture for a Quality of Service (QoS) enhancement in real-time ontological database. The proposed architecture is called Feedback Control Scheduling (FCS) Architecture for Real-Time Ontology denoted by FCS-RTO architecture.

In this paper, we begin first by presenting the related work containing the ontology-based databases, the real-time ontology and the QoS approaches for real-time DBMS. In Section 2, we describe our ontology-based real-time database through QoS management architecture in the real-time DBMS. In Section 3, we have described the functional architecture of the developed simulator and whose evaluation performance. We conclude this paper by discussing this work and by focusing on its major perspectives.

2. RELATED WORK

In this section, we will present the OBDB and the real-time ontology adapted to real-time applications. Then, we briefly describe the quality of service approaches for real-time DBMS.

2.1. Ontology-Based Databases

These last years have witnessed the emergence of several works of rapprochement between databases and ontologies to facilitate their design. These solutions gave raise to a new databases architecture called OBDBs. The different approaches to building an OBDB integrate a local ontology in their construction process. The constructed ontology presents knowledge about a specific field. The

*Corresponding author. *Email:* widedbenabid@gmail.com

data presented in the advanced applications must have specific mechanisms for their manipulation in order to manage the temporal constraints related to them. Moreover, the transactions applied on this type of data have temporal characteristics in which they must be validated before the expiration of the deadlines fixed to them. The following types of information about data and transactions are not considered in the OBDBs modeling. For that, several characteristics relating to data and transactions will be necessary to take them into account in the construction of the ontology presented in these types of systems (OBDBs) such as timing constraints (e.g., deadlines), data requirement, periodicity, time of occurrence of events, and transactions conflict.

2.2. Construction of the Real-Time Ontology

Creating a new ontology [5] becomes a necessity to represent the real-time domain in an explicit way without ambiguity, describing the most general categories and relations is central, and defining the temporal information and resolving the semantic conflicts is vital. In the following section, we present the construction process for real-time ontology.

To better explain the different tasks performed in the process of real-time ontology construction and to validate the approach proposed in this paper, we propose as application case, the application of traffic management assistance. In the transport domain, the base station for collecting real-time information detect measurements concerning the vehicles (throughput, speed, vehicle position, etc.). These measures are intended to operating requirements for road traffic management such as travel time display, automatic congestion detection, traffic status information. The measures thus collected by the sensors are transmitted periodically to the base station which records for each: the type of measure, the value, the instant, the duration during which this data remains valid, and the predefined thresholds which must not be exceeded. The highway traffic control system (HTCS) produces a message in the form of an alert containing the results produced. These alerts are characterized by instants of production, validity interval, type of alert, and so on. The messages generated by the controller are displayed on variable message panel (VMP) to be operable.

The ontological modeling approach for real-time applications [5] adopt a strategy based on the reuse of OWL-Time ontology and their construction process consists of three steps. The ontology construction process (see Figure 1) starts by the first step which is the domain analysis. In this step, authors concentrate on the understanding of the real-time applications' domain in which they analyzed the characteristics as well as the features of the real-time applications. In the next step, authors passed to concepts extraction and relationships determination based on the descriptions proposed and validated by the experts of domain or from the conceptual representations of the real-time applications proposed by researchers, let us quote for example [6] and [7]. They have identified two types of concepts, concepts which are specific to the real-time domain (domain concepts), and concepts that are strongly related to time which describe the temporal aspect of a real-time application. The relationships between concepts are classified into two categories: one is the conceptual relationship and the other is the semantic one. In addition, they presented the temporal semantic relationships that define the semantics of time, such that Allen's

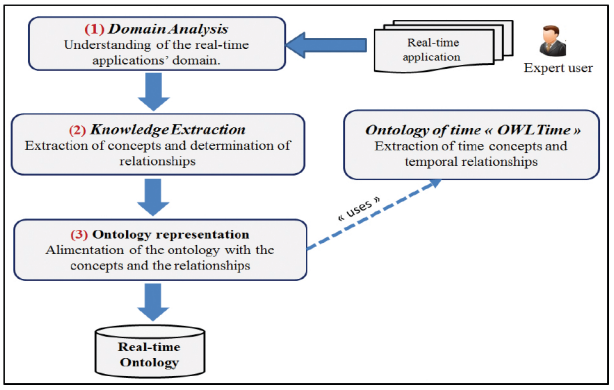

Figure 1 The construction process of a real-time ontology.

temporal relationships [8]. The third step is to represent the ontology based on the concepts and relationships identified in the previous step and which is validated by the domain's experts.

As a summary, the real-time ontology allows the description of the functional aspects and also the temporal aspects of real-time applications for designing the real-time systems. It represents a reference to all concepts and relationships and especially the relationships that are not represented by the other methods, such as the temporal relationships.

2.3. QoS Approaches for Real-time DBMS

Several works based on the QoS as [9, 10], is proposed to manage the instability phases of real-time DBMS and to control their behavior. These approaches make this system more robust and stable against the unpredictable arrivals of transactions. QoS in real-time DBMS is defined by a collective measures of a service level provided to the customer. It is characterized by different performance criteria. The main performance criteria is the success ratio (SR) which measures the percentage of transactions that meet their deadlines. The QoS approaches are based on feedback scheduling control techniques in which they introduced two main concepts: quality of data (QoD) and quality of transaction (QoT). The FCS architecture is considered as the basic model for QoS management in real-time DBMS [10]. This architecture is based on the feedback control loop. The feedback control loop serves to stabilize the system during the overload and under-use phases. It is based on the principle of observation and self-adaptation. The observation is to consider the operating state of the system and to determine if it matches the initially specified QoS parameters, such as the system utilization rate and the transactions' SR. From the observation, the system adjusts its parameters, via the quality of service controller, to increase or decrease the accepted transactions in the system.

Several studies have been carried out on the data presented in real-time applications using architectures based on a QoS approach like [11] and [12]. In addition, these architectures manipulated other types of data such as multimedia data [13]. In all these works, they did not consider the management of semantic data which they are data that refer to an ontology, and that will be presented in the form of structured data linked to a concept and a set of properties. In the current work, we propose to adapt the FCS architecture to manage knowledge (data presented in an ontology).

3. QoS MANAGEMENT ARCHITECTURE OF ONTOLOGY-BASED REAL-TIME DBMS: FCS-RTO

The integration of the ontology into real-time DBMS has brought us to propose an ontology-based real-time DBMS model. This model is based on the FCS architecture. Our goal is to extend the functionality of FCS to take into account the manipulation of semantic data used in the proposed ontology. The proposed architecture is called the Feedback Control Scheduling Architecture for Real-Time Ontology that we note by the FCS-RTO architecture [14]. In this extension, we exploit the manipulation of the data presented in the proposed ontology for the real-time domain (real-time knowledge) instead of managing data generated randomly by the system in the various extensions proposed previously in the literature [9, 11] and [15], and we propose mechanisms to manage the transactions applied to this type of data (ontological data).

Data in our architecture are classified into either real-time ontological data or nonreal-time ontological data. Transactions can be classified into two classes: update transactions and user transactions. Update transactions are used to update the values of real-time ontological data in order to reflect the state of real world. Update transactions execute periodically and have only to write real-time ontological data, but in this case, we have another treatment that is used to check the real-time characteristics (timestamp, lifetime, and validity) of the data accessed by accessing to their properties defined in the ontology. User transactions, representing user requests, arrive aperiodically and may read real-time ontological data, and read or write nonreal-time ontological data.

The transactions in our model when they will access to a real-time ontological data, they must access to the different time properties, it is the difference between RTDB and OBRTDB. The example below (see Figure 2) represents the "Event" class and its different time dependent properties. The "during" property (p3) defines the validity period as the validity interval for the classical data of an instance of the "Event" class. The properties "start" (p1) and "finish" (p2) respectively define the capture time of the value by the sensor and the moment when the value of the data becomes useless. In our case, a transaction to update a real-time ontological data needs to access to these three properties. For QoS management approaches, it is important to grant a high level of Data Quality (QoD) which is defined according to the accuracy and freshness of the real-time ontological data.

Our proposed architecture (see Figure 3) keeps the same components of the conventional FCS architecture, but the functionality of some are different. In what follows, we describe the functioning of the various modules of the FCS-RTO architecture.

The admission controller has also to check whether a user transaction may be terminated before reaching its deadline or not, according to its arrival date and its execution time. In our case, the user transaction can reach ontological data in a read mode. In this case, the operations of this transaction necessarily have to reach the properties of the datum in question, to decide to accept or to reject the transaction to be executed.

Transaction scheduling is based on maturity, which will be recalculated by integrating a specific mechanisms for our data type. If a transaction has access to a real-time ontological data, it must access to its different properties in order to determine the observation time and the validity interval of the requested data. This mechanism is used to calculate the deadline and the execution time of the transaction. The transaction handler consists of a freshness manager (FM) that checks the freshness of real-time ontological data before being

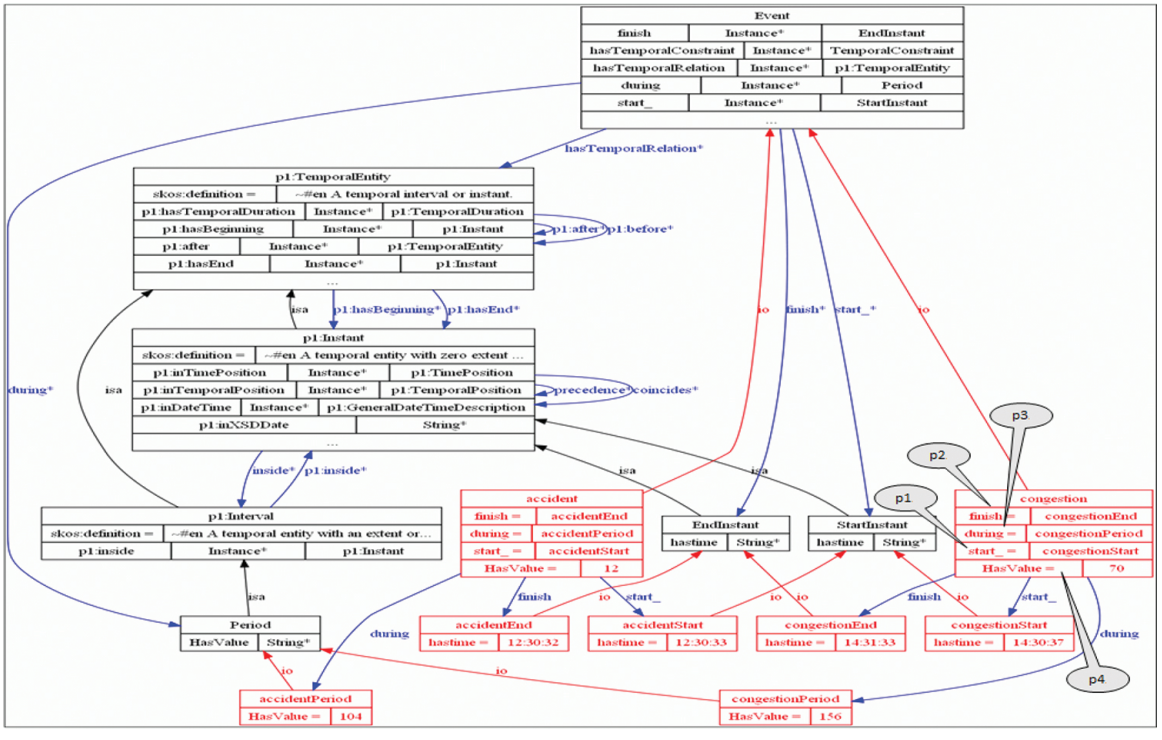

Figure 2 Examples of real-time ontological data from the SAGT application.

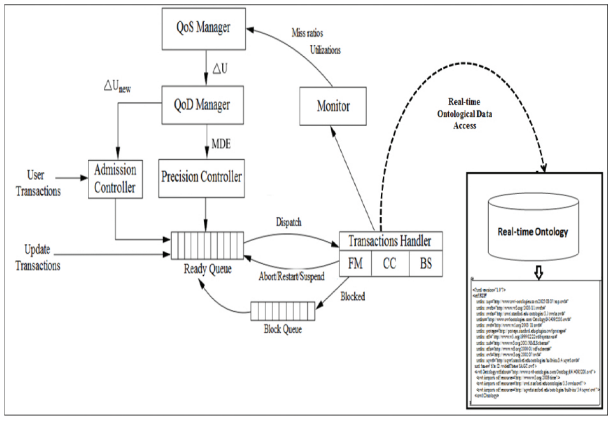

Figure 3 The feedback control scheduling architecture for real-time ontology (FCS-RTO).

accessed by transactions, a concurrency controller (CC) resolving accessing data conflicts appearing between transactions, and a base scheduler (BS) determining the execution time of each transaction.

The consistency of the database is maintained by the CC. The 2PL-HP protocol [16] was proposed to manage conflicts between transactions in real-time context. However, in order to use this algorithm in real-time ontological database, we propose to adapt this algorithm for real-time ontological data structure. The various cases of conflicts which can arise between the transactions that can exist in our architecture can be summarized in both cases following ones:

- Read–Write conflict or Write–Read conflict: In the first case, a "user" transaction requests a read lock on real-time ontological data held by an update transaction for the same data. In the second case, a read lock is held by one or more "user" transactions on a real-time ontological data, while the transaction requesting the lock is an update transaction for the same real-time data.
- Write–Write conflict: In this case, the accessible data is conventional ontological data (non-real time) and the conflict is between two "user" transactions.

In the FCS-RTO architecture, we propose a specific CC that takes into account the real-time ontological data. In this proposed algorithm, we will deal with case-by-case conflicts to show how we manage transactions regardless of their type.

The following definitions show the different notations used in the above algorithm:

- Tr_{up} and Tr_{user}: update transaction and user transaction.
- ST_H: hold sub-transaction.
- ST_R: request sub-transaction.
- *Conflict_List*: A list containing ST_H in conflict with ST_R. It contains a single sub-transaction in the case of a Write–Write or Read–Write conflict, and one or more sub-transactions in the case of a Write–Read conflict.

In this adaptation of 2PL-HP, we distinguished two cases:

Algorithm 1: Concurrency control process for real-time ontological data

```
begin
    if Read-Write conflict
        /* ST_R comes from Tr_user. */
        /* Two possible cases : (1) conflict between
        two Tr_user (access to a non real-time
        ontological data) ou (2) conflict between a
        Tr_user and a Tr_up (access to a real-time
        ontological data). */
        if ST_H comes from Tr_user
            /* Case (1) : Sub-transactions can reach
            the data itself or one of its properties that
            do not depend on time.*/
        else
            /* Case (2) : La ST_H will necessarily
            access the different properties of the
            requested data. */
        fin if
    fin if
    if Write-Read conflict
        /* The ST_H come from Tr_user. */
        /* Two possible cases : (3) only conflict
        between user transactions (Tr_user) (access to a
        non real-time ontological data) or (4) conflict
        between a Tr_up and one or more Tr_user (access
        to a real-time ontological data in reading
        mode). */
        if ST_R comes from Tr_user
            /* Case (3) : Sub-transactions can reach
            the data itself or one of its properties that
            do not depend on time.*/
        else
            /* Case (4) : Access to different properties
            that depend on time for update
            transactions..*/
        fin if
    fin if
    if Write-Write conflict
        /* Case (5) : conflict between two Tr_user. */
        /* Sub-transactions can reach the data itself or
        one of its properties that do not depend on
        time.*/
    fin if
end
```

i. The real-time ontological data is managed by an update transaction: in this case, the transaction that holds the data is forced to access their different properties that depend on time.

ii. The real-time ontological data is managed by a user transaction: in this case, the transaction that holds the data is in read mode. For this, we have the choice between a simple access to the data itself (its value), or one of its properties.

At each sampling period, the local monitor samples the system performance data, by referring to statistics about transactions' execution, as the number of successful, blocked, aborted transactions, and so on, which it retrieves from the transaction manager. Measured values (the system utilization and the transactions' miss ratio)

belong to the feedback control loop and are, then, reported to the local controller. The local controller includes the local utilization controller and the local miss ratio controller, which generate, respectively, the local miss ratio and the local utilization control signals, based on the received values and on the system reference parameters. According to these control signals, the local controller sets the system target utilization parameter to be considered at the next sampling period. The scheduler is used to schedule transactions according to the Earliest Deadline First (EDF) algorithm.

The real-time ontology container provide the selected real-time ontology to be validated with QoS management approach. This real-time ontology store a structured data associated with an other ontologies.

4. SIMULATIONS AND RESULTS

4.1. Simulation Model

To evaluate the performance of the FCS-RTO architecture, we have developed a simulator with a java programming called FCS-RTO-Simulator that allows the execution process of a real-time ontology-based DBMS. It involves to implementing the feedback theory taking into account the functionality of all FCS-RTO components.

The performance metrics used to evaluate the proposed approach with the FCS-RTO simulator is the SR which represents the ratio between the number of transactions that meet their deadlines and the total number of handled transactions. This simulator is the implementation of a started FCS Architecture and its improvements. In the following, we briefly describe its components.

In our simulator, we chose to use the same real-time application described previously. The data generator is represented by the ontological data in the form of an owl file. The access to the different ontological data of the real-time ontology will take place at the level of the execution module of the transactions. For this, and as shown in Figure 4, this module is seen interacting with the various other modules of the simulator, in order to ensure the execution of transactions under the required conditions. The transactions generator generates the two kinds of transaction (user and update). The scheduler implements the scheduling algorithm, especially the EDF algorithm which sorts transactions according to their priorities. The FM checks the ontological data freshness that transactions will be accessed. We apply the proposed protocol which represents an adaptation of the 2PL-HP protocol for taking into account the ontological data.

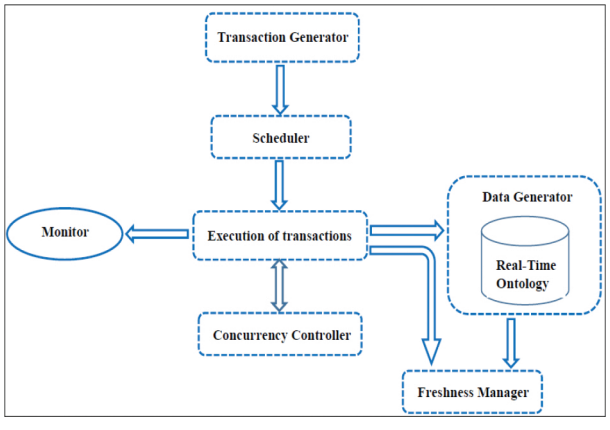

Figure 4 Global architecture of FCS-RTO Simulator.

The proposed architecture serves to resolve the existing conflicts between the transactions applied to the ontological data presented in our HTCS case study and to guarantee the execution of a large number of transactions before the termination of their maturity.

4.2. Simulation Settings

We have evaluated our approach according to a set of simulation experiments, where a set of parameters have been varied. The system parameters for simulations are shown in Table 1. The parameter settings of *user* transactions are summarized in Table 2. We note that arrival times of these transactions are generated according to the "Poisson" process with lambda (λ) is varied between 0.4 and 0.9. We generate the update transactions according to the universal method that assigns random values to their periodicities which can be higher, lower, or equal to the validity of the real time data that it manipulates.

Table 1 System parameter settings.

Parameter	Value
Simulation execution time	5000
Number of real-time ontological data	24
Number of nonreal-time ontological data	29
Number of sampling period	5
Real-time data updating method	Universal
Scheduling algorithm	EDF
Concurrency control algorithm	Adapted-2PL-HP

Table 2 *User* transactions parameter settings.

Parameter	Value
Number of write operations	[1, 2]
Number of read operations	[1, 2]
Write operation time (ms)	2
Read operation time (ms)	1
Slack factor	10
Remote data ratio	20%

4.3. Results and Discussions

We have performed a set of experiments which consists of simulating the behavior of a ontology-based real-time DBMS. In addition, in our experiments, we varied the execution time of a simulation and the value of lambda (λ) to vary the number of "user" transactions that will be accepted by the admission controller in our system.

In Figures 5 and 6, we propose to study the lambda (λ) variation according to accepted user transactions number, this variation may influences on the SR. This study is performed for each sampling period for a simulation execution time equal to 5000 ms. The results are summarized in Table 3, where *nbTUser* is the number of user transactions committed during each period, *SR* (Success Ratio) is the success rate of transactions, and *NbTotalTrUser* is the number of "user" transactions accepted by the admission controller for each value of λ. We give in the following the graphical representation of these results.

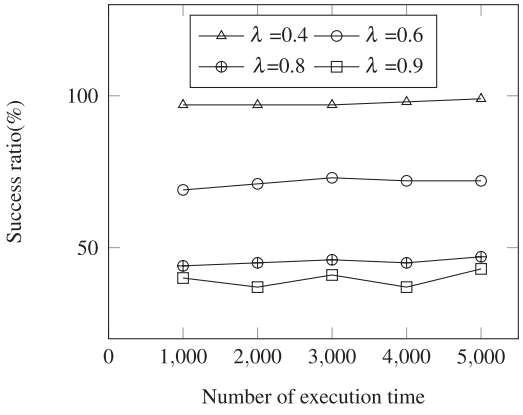

Figure 5 | Simulation results for user transactions.

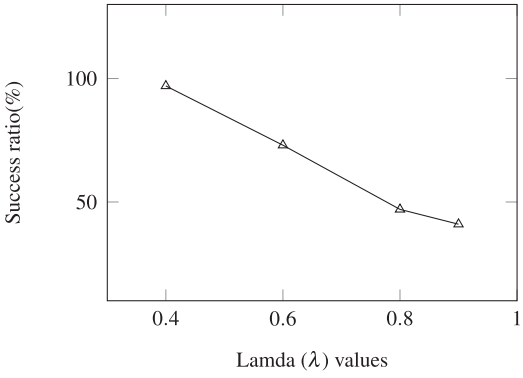

Figure 6 | Impact of λ variation on the success ratio for user transactions.

In Figure 5, each curve has a value of lambda. When lambda value is low, the number of user transactions that will be accepted to be executed in our system is also low, so we will have more user transactions that will meet their deadlines. That's why the best SR is presented for lambda equal to 0.4. In addition, the SR deviations for each curve from one sampling period to another are due to the feedback loop adjustment. In fact, the system adapts its parameters, via the QoS controller, in order to increase or decrease the accepted user transactions to stabilize the behavior of the real-time ontology-based DBMS, thus avoiding overloading or under-utilization of the system.

In Figure 6, the transaction SR according to each lambda value was presented for an execution time of 5000 ms, regardless of their values during each sampling period. We note that the user transactions SR is influenced by the increase of the lambda values. In this case, the number of user transactions handled by the system will increase and therefore the number of conflicts in accessing data, while the user SR will decreased.

The update transactions number are defined according to the simulation execution time. In Figure 7, we present the SR of update transactions in each sampling period. The decrease in the SR from one period to another is due tothe increase of the update transactions number that will be executed. For example, in period number one, fewer transactions are ready to be executed, so more transactions that will meet their deadlines. In addition, the SR for update transactions is always high compared to user transactions, because the system promotes update transactions than user transactions.

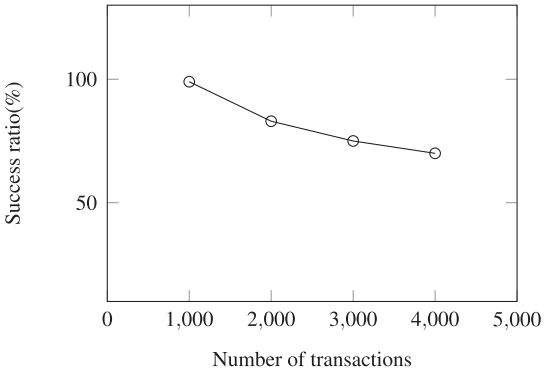

Figure 7 | Simulation results for update transactions.

The system guarantees a good conditions for the execution of all transactions, and a overall system performance in terms of QoS by increasing the number of transactions that meet their deadlines

Table 3 | Simulation results for an execution time of 5000 ms.

Lambda / Period	λ = 0.4		λ = 0.6		λ = 0.8		λ = 0.9	
	nbTUser	**SR**	**nbTUser**	**SR**	**nbTUser**	**SR**	**nbTUser**	**SR**
Period 1	384	97	400	69	336	44	315	40
Period 2	386	97	408	71	342	45	322	37
Period 3	386	97	835	73	686	46	660	41
Period 4	753	98	1239	72	1026	45	945	37
Period 5	754	99	1659	72	1443	48	1400	43
NbTotalTrUser (λ)	*763*		*2287*		*3203*		*3577*	

using ontological data. Then, the result provided by FCS-RTO confirms that the contribution of the feedback loop remains relevant for the ontology-based real-time DBMS.

The ontology-based real-time DBMS is proposed to manage a new type of data that depends on semantics. For this, our system offers a reliable access to ontological data and a good conditions for the transactions that accessed to them to respect their deadlines, and therefore to have a good performance.

5. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a ontology-based real-time DBMS model. The performance study of this model require the use of QoS-based architectures which the feedback control scheduling technique is the best case. In this order, FCS-RTO is proposed. This architecture is suitable to make the system in stable state and enhance its performance. The FCS-RTO ensures a best functioning of the system components face to ontology-based real-time database. To evaluate the performances of the FCS-RTO, we have built a simulator allowing to measure the transactions SR. The various simulations that we carried out confirmed the contribution of our system, that is we obtain more successful transactions before their deadlines for both types of the transactions (user and update), in particular with the increase of the size of the system. This returns to the fact that with our approach, the requirements for transactions to enter the system and complete their execution before deadlines are less. Our system thus offers a reliable access to ontological data and a good conditions for the transactions that reach it to meet their deadlines, and thus have a good performance.

A short-term extension of our work is to extend the FCS-RTO architecture, by the various extensions proposed for the basic FCS architecture. Our objective will always remain to improve the QoS in real-time DBMS, by giving more chances to the transactions to succeed their executions, without affecting the freshness of the real-time ontological data, and by guaranteeing a certain stability of the system in the face of the unpredictable workloads. It would be interesting to integrate the notion of (m, k)-firm constraints, applied to update transactions in order to relax the atomicity property. We will also propose to integrate other techniques for the concurrency control and the scheduling algorithms like the Mixed Real-Time Scheduling (MRTS) protocol [17].

In addition, in FCS-RTO, we did not consider the distribution of transactions. We focused on QoS in the case of a centralized environment. Another extension of our work will be to apply it to the distributed context. Indeed, the presence of multiple sitesin a distributed system poses problems that were not present in the centralized systems and the performance of the distributed systems depends on the distribution of the workload between the sites.

REFERENCES

[1] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle, The ICS-forth RDF suite: managing voluminous RDF description bases, in Proceedings of the Second International Conference on Semantic Web—Volume 40, SemWeb'01, Aachen, Germany, 2001, pp. 1–13, CEUR-WS.org.

[2] M.J. Park, J. Lee, C.H. Lee, J. Lin, O. Serres, C.W. Chung, An Efficient and Scalable Management of Ontology, Springer, Berlin, Heidelberg, 2007, pp. 975–980.

[3] H. Dehainsala, G. Pierra, L. Bellatreche, OntoDB: An Ontology-Based Database for Data Intensive Applications, Springer, Berlin, Heidelberg, 2007, pp. 497–508.

[4] C. Fankam, Ontodb2: support of multiple ontology models within ontology based database, in Proceedings of the 2008 EDBT Ph.D. Workshop, Ph.D. '08, ACM, New York, 2008, pp. 21–27.

[5] W. BenAbid, M. Mhiri, E. Bouazizi, A. Rhayem, An ontological approach to design real-time applications, J. Data Semantics. 5(3) (2016), 195–209.

[6] OMG, A UML Profile for MARTE: modeling and analysis of real-time embedded systems, Beta 2, ptc/2008-06-09. Technical report, Object Management Group, 2008.

[7] S. Rekhis, N. Bouassida, R. Bouaziz, C. Duvallet, B. Sadeg, Modeling real-time design patterns with the UML-RTDP profile, in: Domain Engineering. Product Lines, Languages, and Conceptual Models, Springer, Berlin, Heidelberg, 2013, pp. 59–82.

[8] J.F. Allen, An interval-based representation of temporal knowledge, in Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'81, Morgan Kaufmann Publishers Inc., San Francisco, 1981, pp. 221–226.

[9] K.D. Kang, S.H. Son, J.A. Stankovic, T.F. Abdelzaher, A qos-sensitive approach for timeliness and freshness guarantees in real-time databases, in Proceedings 14th Euromicro Conference on Real-Time Systems. Euromicro RTS 2002, IEEE, Vienna, Austria, 2002, pp. 203–212.

[10] M. Amirijoo, J. Hansson, S.H. Son, Specification and management of qos in real-time databases supporting imprecise computations, IEEE Trans. Comput. 55(3) (Mar. 2006), 304–319.

[11] E. Bouazizi, C. Duvallet, Utilisation des contraintes (m, k)-firm pour la gestion de la qds dans les SGBD temps réel, in Actes du XXIXème Congrès INFORSID, Lille, May 24–25, 2011, pp. 95–110.

[12] M. BenSalem, F. Achour, E. Bouazizi, R. Bouaziz, Applicability of the (m,k)-firm Approach for the QoS Enhancement in Distributed RTDBMS, Springer International Publishing, Cham, 2013, pp. 166–175.

[13] B. Alaya, C. Duvallet, B. Sadeg, Feedback architecture for multimedia systems, in The 8th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2010, Hammamet, May 16–19, 2010, pp. 1–8.

[14] W. BenAbid, M. BenAhmedMhiri, M. BenSalem, E. Bouazizi, F. Gargouri, A feedback control scheduling architecture for real-time ontology, in International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2017, Nanjing, Nov. 24–26, 2017, pp. 1–7.

[15] M. BenSalem, E. Bouazizi, R. Bouaziz, Multi-versions data and epsilon-serializability for qos enhancement in distributed RTDBMS, in IEEE/ACS International Conference of Computer Systems and Applications, AICCSA 2015, Marrakech, Nov. 17–20, 2015, pp. 1–6.

[16] R. Abbott, H. Garcia-Molina, Scheduling real-time transactions: a performance evaluation, ACM Trans. Database Syst. 17(3) (Sep. 1992), 513–560.

[17] F. Achour, E. Bouazizi, W. Jaziri, Scheduling approach for enhancing quality of service in real-time DBMS, in Databases and Information Systems - 12th International Baltic Conference, DB&IS 2016, Riga, July 4–6, 2016, pp. 126–135.