An ORM Based Context Model for Context-Aware Computing

Annet Nishantha Anton Yogarajah, Shiluka Raveen Dharmasena, Gobinath Loganathan(✉), Srinath Perera, Vishnuvathsasarma Balachandrasarma, and Malaka Walpola

Department of Computer Science and Engineering, University of Moratuwa, Moratuwa, Sri Lanka
slgobinath@gmail.com

Abstract. Context-aware applications are the future of modern smartphones. Now we have mobile devices with enough sensing and processing capabilities but combining them and developing a context-aware application for mobile devices is still a challenging task for developers. Context-aware middleware support is a solution to reduce the complexity in developing context-aware applications. Context modeling is one of the key requirement for a successful context-aware middleware for context representation and reasoning. This paper presents a new Object-Role Modeling (ORM) based context model which uses the advantage of modern graph databases and overcomes the problems associated with previous context models including their lack of context reasoning ability and poor spatial and temporal context modeling support.

Keywords: Context modeling · Context reasoning · Context awareness · Context-aware computing · Pervasive computing · Object-role modeling

1 Introduction

The most profound technologies are those that disappear [1]. In today's world, smartphones become an inevitable requirement for people to capture their surrounding, listen to music, watch videos, read email, access social media, chat with friends and even more. Here, a single device has access to various contextual information of its owner and others who share their information through the network which is connecting all the smartphones. The numerous sensors packed in modern smartphones let them track user's activities without interrupting the user. For example, my phone knows where I am and what I am doing right now to a certain extend without my concern. This seamless interaction of smartphones makes them the suitable candidate for context-aware computing. Context-awareness is the ability of a software to adapt according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as

P. Cong Vinh et al. (Eds.): ICCASA 2016, LNICST 193, pp. 132–141, 2017.
DOI: 10.1007/978-3-319-56357-2_14

to changes to such things over time [2]. Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [3].

However, still developing a context-aware application is a challenge for mobile application developers because implementing a context-aware system requires addressing many issues: (1) How does the system represent context internally? (2) How frequently does the system need to consult contextual information? What is the overhead of considering context? (3) What are the minimal services that an environment must provide to make context awareness feasible? (4) What are the relative merits of different location-sensing technologies? [4].

In this paper, we address the first issue and present an ORM context model using Neo4j graph database to support context modeling and reasoning as the solution. Our model represents the contextual information of people using nodes and links in a graph. The underlying database engine is used to store and process the contexts. This paper is organized as follows. Related work is discussed in Sect. 2. An introduction to Neo4j database and why we have selected it are described in Sect. 3. We discuss our context model in Sect. 4 and its context reasoning ability in Sect. 5. Section 6 describes the evaluation and results followed by the conclusion and future work in Sect. 7.

2 Related Work

There are remarkable researches conducted on context modeling and reasoning in middlewares. In this section, we analyze the major modeling techniques which were successfully adapted to general context modeling and the most successful implementation of each type.

Ontology-oriented modeling: This modeling is used by most of the existing middlewares because of its expressiveness and reasoning support. Dejene Ejigu et al. [5] developed a multi-domain ontology based reusable model. Comprehensive Structured Context Profiles (CSCP) was developed based on Resource Description Framework (RDF) as an improved representation technique [6]. SOCAM middleware uses the ontology to represent generic contexts along with the temporal changes and quality of contexts [7]. Dynamic spatial ontology was developed to represent snapshots of the geographical data and to support spatial queries [8]. Frederico T. Fonseca et al. [9] proposed ontology-based geographical knowledge system which can be developed using partial data.

Graphical modeling: Graphical modeling is used to represent the contexts as a conceptual model. Unified Modeling Language (UML) and ORM are the two major techniques used to represent the contexts. ContextUML, a UML-based language for model-driven context-aware services was proposed by Quan Z. Sheng et al. [10]. Henricksen et al. proposed a graphical context model which supports context classification, quality of contexts and temporal characteristics of contexts [11]. Their model uses timestamps to represent temporal contexts and additional relationships to represent the quality of contexts.

However, none of these models have proposed a native spatial model or a temporal model to represent history of events. In this paper, we present our object-role model which can be seen as an extended model of Henricksen's proposal. Our model overcomes the spatial and temporal modeling issues of existing models with alternative approaches.

3 Neo4j

Neo4j [12] is an open source NoSQL graph database with an expressive query language called Cypher query. Cypher query is an SQL-inspired language for describing patterns in graphs visually using an ASCII-art syntax [13]. Our model is built on top of Neo4j 2.3.0 and all the queries given in this paper follow the Neo4j 2.3.0 Cypher query syntax. Even though the model is built on Neo4j, it can be deployed on top of any graph databases with suitable changes in low level CRUD operations.

4 An ORM-Based Model

4.1 Design Considerations

A context-aware middleware must have the capability of processing, modeling, storing and distributing the context among various context-aware applications which are interested in a specific context. The context can be any information about relationships between users, physical objects, and applications. The interrelation of contexts helps to derive some high-level contexts which are not directly available from the raw sensors. For example, if Alice is in her office and Bob's mobile is connected to Alice's mobile via Bluetooth, we can conclude that Bob is closed to Alice. There are exceptions of course, for example, if Alice knows Bob, we cannot conclude that Bob knows Alice. Therefore, domain-specific rules on interrelationships of contexts are required to derive more contexts using existing knowledge.

Some contexts are inconsistent while some are not. For example, "Alice knows Bob" this relationship never changes once created, but "Alice is in city center" is not consistent with time. History of contexts which are changing with time must be available for some interested applications. For example, history of visited locations may be required for a tour guide application to decide on best places to visit. Again, a domain-specific rule is necessary on deciding which contexts to be stored with their past history.

4.2 Modeling Contexts

Our model is user oriented so the contexts are represented as the relationships of a user with other users, physical objects, and the environment. The wide range of contexts is classified into three major categories: social context, spatial context, and temporal context. Social context is the relationship of a user with

other people. Social context also includes the personal contexts like home location, work location, and user's device related information. Spatial context is the relationship of a user with a location and the nearby physical objects. Temporal context is the context changing with time and its history is required to derive some other useful information. For example, the location data of the user which are periodically collected and maintained as a temporal context can be used to find the traveling pattern of the user and suggest him/her to adjust it based on real-time traffic data and weather forecast.

All these three categories can be further classified into direct context and indirect context based on the means by which the context is obtained. Direct context is collected using the sensors connected to the middleware. This type of contexts can be stored in the model without complex processing. Indirect context is derived from direct context through aggregation and reasoning using a processing engine. Indirect contexts can be stored in the context model or immediately sent to the interested applications and discarded in the middleware depending on the context type and its validity period.

All the contexts are modeled using ORM model because it is more expressive to represent the contexts in various dimensions and modern graph databases provide powerful processing support.

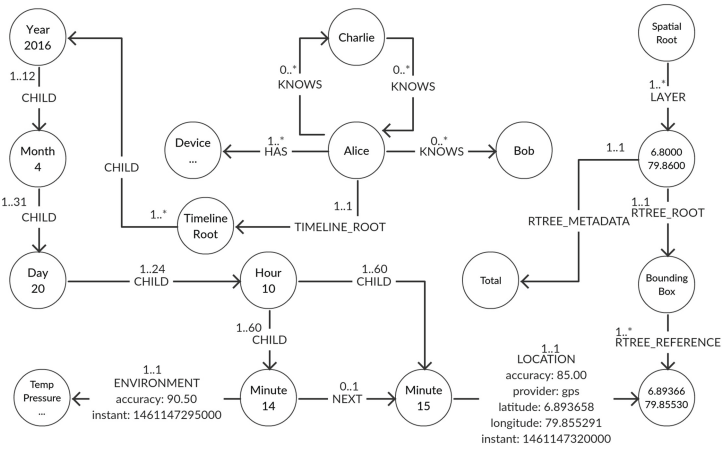


Fig. 1. A partial ORM model representing the contexts of Alice (Alice knows Bob but Bob doesn't know Alice. Alice and Charlie know each other. Alice has a Device. On 20-04-2016 at 10.15, Alice was at 6.89366, 79.85530 location)

The Nodes: Person, Device, Environment, Location, Year, Month, Day, Hour, Minute and WiFi represent the real world entities as their name suggest. Nodes can have certain properties depending on their class. For example, Person node has a name, email address and user_id and Device node has manufacturer name, device_id, operating system version, Wi-Fi MAC address, Bluetooth MAC

address, last seen time, current battery level and a list of available sensors. The links: KNOWS, HAS, ACTIVE_DEVICE, ENVIRONMENT, LOCATION, LOCATED_AT, CHILD, NEXT, etc. represent the relationship between entities. All the links are directed links but the direction can be ignored during context reasoning depending on the domain and requirement. If a two-way direction is explicitly required, two links have to be used to achieve it. For example in Fig. 1, Alice knows Bob but Bob does not know Alice is represented by a directional link from Alice to Bob. Alice and Charlie know each other is represented by two directional links; one from Alice to Charlie and the other from Charlie to Alice. Links also can have properties based on their class. For example, the links ENVIRONMENT and LOCATION have a property 'accuracy' which represents the quality of the context. All devices which are with the user at the moment are linked using ACTIVE_DEVICE relationship. This link is useful to resolve contradicting contextual information reported by two devices. For example, if Alice has two phones and both are sending two different locations, our model stores the location reported by the ACTIVE_DEVICE only. If two active devices are reporting different locations, one with the highest accuracy will be stored in the model.

4.3 Modeling Spatial Contexts

R-Tree is a balanced tree data structure to store and process spatial objects in databases. It allows mapping geographical coordinates and polygons with high precision and acceptable performance. R-Tree does not guarantee a good worst-case performance but it works well on average cases for most kinds of data [14]. We use an existing R-Tree implementation for Neo4j database [15] which supports spatial queries like Contain, Cover, Covered By, Cross, Disjoint, Intersect, Intersect Window, Overlap, Touch, Within, and Within Distance. The R-Tree layer is a combination of geometries used to represent a collection of geometric objects with the same attribute. Our current model contains only one layer to represent the GPS coordinates of locations. It can be extended to include multiple layers with various geographical information. The layer has a bounding box and a metadata node to store the number of nodes connected to that layer and the range of the stored nodes. An ID is generated for Locations using approximated latitude and longitude values. The level of approximation is determined by the level of precision required by the application. Current model approximates the coordinates to 5 digits which provides a precision of 1.1 m [16]. Before storing a location, our model searches for an existing location with the same ID and if it is available, rather than creating a new Location node, existing node will be shared among the users in order to reduce the number of nodes in the model. The LOCATION link contains the accuracy reported by the GPS sensor and the exact latitude and longitude sent by the device. LOCATION link also contains an optional property 'provider' which stores the location provider used to derive the location. Currently, there are three location providers available in Android operating system which are GPS, network and

passive. Location provider and the accuracy of the location can be used to define the quality of the context later in context reasoning process.

Nearby Wi-Fi terminals reported by devices are also associated with the location. The relationship 'LOCATED_AT' between Wi-Fi node and the location node contains a property named 'strength'. If a Wi-Fi terminal which is already available in the model, is available in another location with more than 500 m distance, that Wi-Fi terminal is considered as a portable terminal and the 'LOCATED_AT' relationship will be deleted. This information can be used to identify the location of a user with the help of available Wi-Fi networks instead of using GPS which consumes more power than Wi-Fi (Fig. 2).

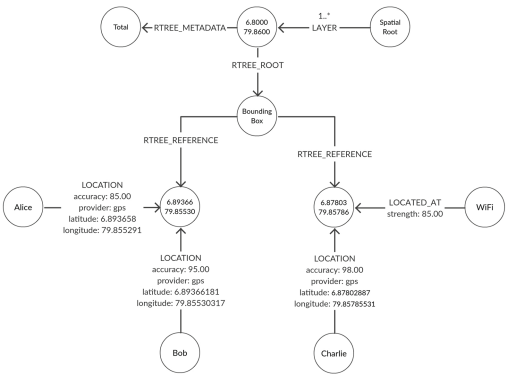


Fig. 2. Spatial model

4.4 Modeling Temporal Contexts

Even though modern smartphones synchronize the time with global time, there can be devices with wrong clock time. When multiple individual devices are sending contextual information to a central server, storing them using the device clock time will cause to event ordering problem due to the variation in devices time. As a solution, our model uses the server time to store time sensitive contextual information, ignoring network delays. The zero network delay assumption leads to another assumption that the HTTP requests sent by devices reach the server in the same order. In other words, we assume that if device A sent an event before device B, the request of device A reached the server before the request of device B. The time zone of a user is identified using his/her current location and stored in the TimelineRoot node. When an application requests a time sensitive contextual information, device local time is calculated using the time zone of the user and the server. Additional options are provided to override the user's time zone if required.

This model represents the time using a tree structure. Each user has a link to a unique TimelineRoot node. TimelineRoot node has CHILD links to Year nodes. Each year nodes can have 1–12 CHILD links to Month nodes. Similarly,

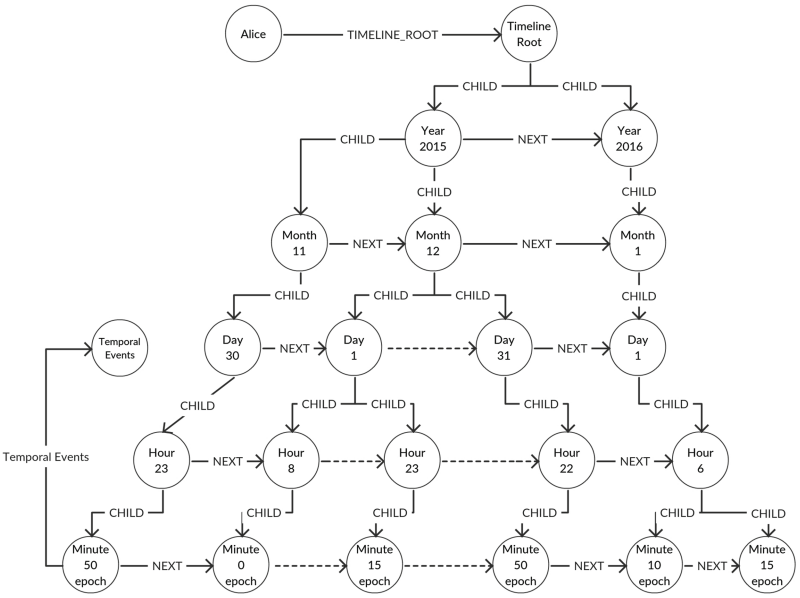


Fig. 3. Temporal model

the tree grows until Minute nodes. Nodes in the same depth have intermediate links named NEXT which points the next node if available. We stopped with the precision of minutes because of two reasons. First, the minimum time interval of mobile sensors to collect and report new data is 5 min if the battery level is 100% because frequent usage of sensors in a mobile device drains the power of the mobile device. We have ended with 5 min after trial and error experiments and this interval is dynamically adjusted based on the battery level of the device. Second, modeling with a precision of seconds leads to too many nodes and links in the database which needs more memory. The Minute nodes contain an extra property 'milliseconds' which is the number of milliseconds from Java epoch truncated to that specific minute. The actual instant in milliseconds is stored in the relationship which connects the Minute node and the context node (Fig. 3).

Since our model supports multiple devices and the precision is up to minutes, there is a possibility of two devices reporting the same context within a minute. If there are two different types of contexts reported by two active devices of the user within a minute, both will be linked to that specific time in this model. If the context types are same, the one with the highest accuracy is linked to the time.

5 Context Reasoning

Context reasoning is the process of deriving deduced contexts from raw contexts. It also provides a solution to resolve context inconsistency and conflict that

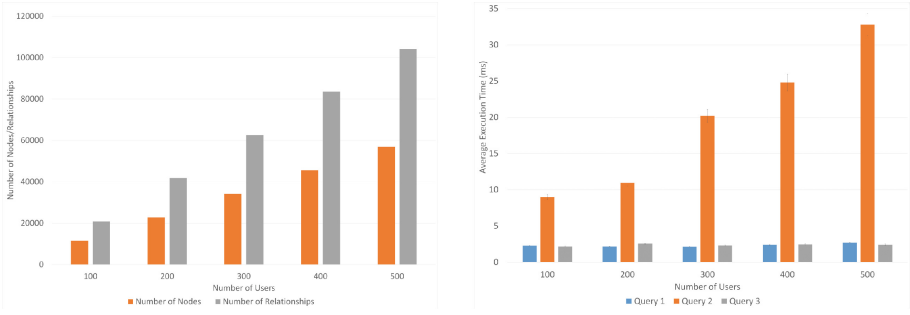
caused by imperfect sensing [14]. Most of the modern graph databases provide query support to aggregate data and retrieve the interested information. Since our model is built on Neo4j, Cypher query of Neo4j is used to process the contexts and to derive the complex contexts based on the defined rules. Our middleware uses WSO2 Complex Event Processor [17] along with the database for realtime context processing. Combination of both CEP and database is used to derive high level contexts and to improve the quality of contexts as well.

For example, finding the visited locations of Alice from January 1st to April 15th in last five years can be done using the following Neo4j query.

```
MATCH (n:Person {name: 'Alice'})-[:TIMELINE_ROOT]->(:
TimelineRoot)-[:CHILD]->(y:Year)-[:CHILD]->(m:Month)-[:
CHILD]->(d:Day)-[:CHILD*]->(:Minute)-[:LOCATION]->(l:
Location) WHERE (y.value >= 2011 AND y.value < 2016)
AND (m.value < 4 OR (m.value = 4 AND d.value <= 30))
RETURN l
```

6 Evaluation

Five sample data sets were created using 100 to 500 users per sample along with 10 visited locations and 10 environmental conditions per user. Further, each model contain 100 known relationships among the users. The data sets were deployed on Neo4j server and tested through REST API of Neo4j. Therefore the time measured in these tests includes the delay in sending HTTP requests. The test has been done only for data retrieval because the behaviors of both writing to the model and reading from the model are identical in our context model. Testing the read-write performance of underlying database is out of the scope of this research. As shown in Fig. 4a, our model requires more nodes and relationships than the models suggested in existing works because of the temporal model we use. However, it does not heavily impact the overall performance of the model.



(a) Space required by the model (b) Performance of the model

Fig. 4. Evaluation results

Three queries were used to test the performance of the model and the performance is compared in Fig. 4b. Query 1 searches for a given person using the user id to evaluate overall read performance of the model. The performance remains constant regardless of the number of nodes and the same result is expected for any entities searched using their indexed properties. Query 2 retrieves the nearby known people of a given user within a given interval to check the performance of spatial and temporal models. The query was purposefully designed to first retrieve the nearby locations using the R-Tree implementation and then search for known people in those locations within the given interval. Therefore the major impact on the performance is made by the spatial model. The R-Tree implementation provides more inbuilt features to the model but the performance is the limiting factor in using the spatial model. Performance decreases with the number of locations and the average time to retrieve the data is much higher than the time to retrieve other contextual information. However, the same information can be retrieved without using the advantage of the spatial model which is implemented in Query 3.

The exact information retrieved using the query 2 was retrieved using query 3 in another way. Query 3 starts from the known people and then searches for their location within the given interval. Once the visited locations were retrieved, it calculates the euclidean distance between those locations and the given location. Those who were within the given distance are returned as nearby known people. While both query 2 and 3 return the same result, the time taken to execute the queries significantly varies. As Fig. 4b shows, the performance of query 1 and 3 are quite similar which concludes that the performance of the model is not impacted by the temporal model we used. The spatial model does not provides better performance and should not be used if there are any other ways to retrieve the spatial information.

7 Conclusion

In this paper, we have presented an extensible ORM model to represent and process contexts with the support of graph database. Our context model represents social, spatial and temporal contexts along with the quality of contexts. The underlying database provides storage and also acts as a context processing engine. We have already used this model in our context-aware middleware ConTra and proved the validity of this model. We are looking at creating multiple R-Tree layers to represent various geographical information like buildings in one layer and streets in another layer. We are also focusing on developing this model as a generic framework which can be deployed on top of any graph databases with less or no effort.

References

1. Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. **3**(3), 3–11 (1999)
2. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications. WMCSA 1994, pp. 85–90. IEEE Computer Society, Washington, DC (1994)
3. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999). doi:10.1007/3-540-48157-5_29
4. Satyanarayanan, M.: Challenges in implementing a context-aware system. IEEE Pervasive Comput. **1**(3), 2 (2002)
5. Ejigu, D., Scuturici, M., Brunie, L.: An ontology-based approach to context modeling and reasoning in pervasive computing. In: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops. PERCOMW 2007, pp. 14–19. IEEE Computer Society, Washington, DC (2007)
6. Held, A., Buchholz, S., Schill, A., Schill, E.: Modeling of context information for pervasive computing applications (2002)
7. Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An ontology-based context model in intelligent environments. In: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 270–275 (2004)
8. Grenon, P., Smith, B.: Snap and span: towards dynamic spatial ontology. Spat. Cogn. Comput. **4**(1), 69–103 (2004)
9. Fonseca, F.T., Egenhofer, M.J.: Ontology-driven geographic information systems. In: Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems. GIS 1999, pp. 14–19. ACM, New York (1999)
10. Sheng, Q.Z., Benatallah, B.: ContextUML: a UML-based modeling language for model-driven development of context-aware web services development. In: Proceedings of the International Conference on Mobile Business. ICMB 2005, pp. 206–212. IEEE Computer Society, Washington, DC (2005)
11. Henricksen, K., Indulska, J., Rakotonirainy, A.: Generating context management infrastructure from high-level context models. In: 4th International Conference on Mobile Data Management (MDM) - Industrial Track, pp. 1–6 (2003)
12. Neo4j: The world's leading graph database. http://neo4j.com. Accessed 07 Feb 2016
13. What is a graph database? A property graph model intro. http://neo4j.com/developer/graph-database/. Accessed 05 Jan 2016
14. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data. SIGMOD 1984, pp. 47–57. ACM, New York (1984)
15. Taverner, C.: Neo4j-contrib/spatial. https://github.com/neo4j-contrib/spatial. Accessed 05 Jan 2016
16. Hijmans, R.J., Guarino, L., Cruz, M., Rojas, E.: Computer tools for spatial analysis of plant genetic resources data: 1. DIVA-GIS. Plant Genetic Resources Newsletter, pp. 15–19 (2001)
17. Complex event processor - WSO2 inc. http://wso2.com/products/complex-event-processor. Accessed 03 May 2016