Journal of Engineering and Applied Sciences 13 (Special Issue 7): 5561-5564, 2018
ISSN: 1816-949X

Querying Scalable and Redundant Data via. Recursive Queries in Orm Systems

[1]Priyadharshini Muthukrishnan, [1]V. Sakthivel, [2]Baskaran Ramachandran, [3]K. Srihari and [4]P. Balaji
[1]KPR Institute of Engineering and Technology (KPRIET), Coimbatore, India
[2]Anna University, Chennai, India
[3]SNS College of Technolog (SNSCT), Coimbatore, India
[4]Wipro Technologies, Bengaluru, India

Abstract: Persistent data are stored in database server and is often used than archived data. Most business applications handling large databases contain recursive data and processing such recursive data is difficult. Relational algebra doesn't provide adequate facility to manipulate such data, even initial SQL too doesn't, however, it is introduced in SQL: 1999. MySQL an open source database management system doesn't support recursive query yet. Hibernate being an open source Object Relational Mapping (ORM) tool provides Hibernate Query Language (HQL) for querying databases. Using which recursive queries are executed as HQL with the options provided by the corresponding Data Base Management System (DBMS). In DBMS such as MySQL where recursive query is not possible our proposed extension of hibernate provides the option to execute recursive queries by executing a combination of normal queries in a particular order.

Key words: Hibernate, persistent data, recursive query ORM, combination, relational, particular

INTRODUCTION

The host languages commonly used for application development such as Java, C++, C#, etc. are object oriented and the databases used are relational databases such as SQL Server oracle, Mysql etc., Object Relational Mapping (ORM) tools facilitate the storage of data in object oriented host language into the relational database. Hibernate is one such open source tool that performs object relation mapping. Business applications today produce and handle data in huge volume which consists of recursive data. Relational languages lack adequate facility to manipulate recursive data for which an alternative is to be devised.

Data models devised for numerous business enterprises include recursive data structures that are in the form of hierarchies and networks. There are various ways to query such data. Obviously, a dedicated client code can be written. Then, the data processing is done on the client side. In this case, a significant amount of complex source code must be created, debugged and maintained. Therefore, a server side solution is called for. It was proposed as extensions to SQL, e.g., Oracle's connect by clause or recursive Common Table Expressions (CTE) eventually adopted in SQL: 1999. These extensions have been implemented in numerous database systems.

However, there are still database managements systems that do not support recursion in queries, e.g., MySQL. Since, they are widely adopted and used, applications programmers often face the question how to query their recursive data. Hence, it is chosen to hardcode suitable logic in the application. In spite of deceptive simplicity of these solutions, it causes merely troubles such as lower efficiency, increased cost and complexity as well as reduced maintainability.

On the other hand, Object Relational Mapping (ORM) system is a possible way to solve the above problem. They bridge the gap between data models of relational storage and object-oriented code. Along with the basic functionality ORM also establishes a thick abstraction layer that can be augmented with abundant features.

Literature review: The optimization methods for executing recursive queries from hibernate are discussed by Ordonez (2010) and Szumowska *at al.* (2011). Object Relationship Mapping (ORM) system is discussed by various researchers (Bauer and King, 2006; Boniewicz *et al.*, 2013; O'Neil, 2008; Melnik *et al.*, 2008) in the hibernate framework for object persistence. The various methods executing recursive queries are discussed by Boniewicz *et al.* (2013), Burzanska *et al.* (2010), Ordonez (2010) and Przymus *et al.* (2010) discussesabout the optimization methods for executing

Corresponding Author: Priyadharshini Muthukrishnan, KPR Institute of Engineering and Technology (KPRIET), Coimbatore, India

recursive queries. Boniewicz *et al*. (2013) discusses about the implementation ORM which provides materialization of methods for executing recursive queries. The disadvantage is that users have to choose appropriate materialization method manually to record their decisions in configuration files. The discussion regarding the ORM and object persistence in Java is made where in it is found that they are not suitable for complete graph (Bauer and King, 2006).

Burzanska *et al*. (2010) discusses about the execution of recursive queries using ORM. Przymus *et al*. (2010) discusses about the list of possible ways for executing recursive queries in hibernate. In particular, Szumowska *et al*. (2011) discusses about the execution of recursive queries using hibernate. It parses recursive query information from Extensible Markup Language (XML) mapping file, executes the recursive queries and returns the list of objects. Melnik *et al*. (2008) discusses about the compiling of mapping files and building persistent object. Keller discusses about the ORM, object persistence and methods for persisting objects. The related works corresponding to ORM is categorized as materialization methods for ORM, compiling of mapping files in ORM, persisting objects in Java and ORM with recursive queries are clearly shown in the mind map as Fig. 1.

In current scenario user chooses the appropriate materialization method for executing recursive query or they have to write XML mapping file manually for executing recursive queries, so, they are forced to strictly follow XML DTD. There is no generic method for executing recursive queries.

User has to write native Structured Query Language (SQL) queries for executing recursive query which is possible only if the back end supports recursive query. Otherwise, users have to create suitable hardcode for executing recursive queries. This usually causes a noteworthy increase of the budget and a shift in the delivery schedule. Execution time of recursive CTE and Oracle's connect by clause are also more when using native SQL queries.

The proposed system eliminates the issues so that the users need not choose any materialization method and they need not write XML mapping files for executing recursive queries. Instead, they have to call the method by passing recursive query and the method will execute recursive queries and returns the list of objects. The proposed system is implemented as an extension to Hibernate package, called hibernatex package that realizes and executes recursive queries. In particular, recursion is added on top of database systems that do not implement it directly. It will be executed with the help of hibernatex

Fig. 1: Mind map of the object relationship mapping

package which will parse and execute the recursive query. The proposed system executes recursive queries irrespective of the backend used via. hibernate ORM systems. Though MySQL like databases does not support recursive querying, the proposed system executes the recursive queries and produce the result. The proposed system also provides solutions which speed up the recursive queries.

MATERIALS AND METHODS

System architecture: In the proposed system, objects of the application and their references will be stored in database in form of persistent objects. Hibernate consists of recursive querying API and configuration, through which recursive querying API is configured and it is connected to the database. The core application of the proposed system is executing recursive queries in hibernate irrespective backend via. object relationship mapping systems and reduces execution time of the recursive queries.

The entire proposed system is categorized into two phases, i.e., implementation of hibernate framework and creation of hibernatex package which consist methods for parsing and executing recursive queries. The overall system architecture including hibernatex package is shown as in Fig. 2 where two methods in which one validates and parses the recursive query and pass the class name, condition and recursive variables to another method which executes the recursive query and returns the list of objects.

Parsing recursive query: The proposed system must follow SQL connect by syntax for executing recursive queries. First method in hibernatex package will validate the recursive query, if any syntax error occurs, it will through the error. If the given query is syntactically correct, it will parse the recursive query and get all required information for executing recursive queries such as class name, condition and recursive fields. Finally, it will call another method by passing class name, condition and recursive fields for executing recursive queries.

Executing recursive query: The proposed system is a replacement to Oracle's recursive CTE which follows SQL connect by syntax for the simplicity, it uses algorithm of recursive CTE queries. Let us discuss about recursive CTE queries.

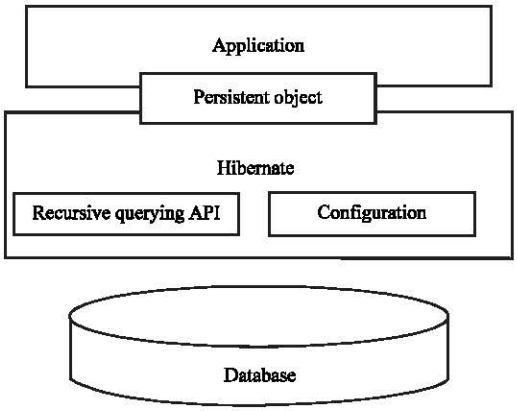

Fig. 2: System architecture

Algorithm 1; Syntax of recursive CTE:

```
WITH cte_name ( column_name [, ..., n] )
AS (

CTE_query_definition -- Anchor member is defined
    UNION ALL
CTE_query_definition -- Recursive member is defined referencing cte_name.
    ) -- Statement using the CTE
    SELECT *
    FROM cte_name
```

Algorithm 2; Recursive CTE:

Step 1: Executes the Anchor part and sends the output and also write it to temporary table in tempdb
Step 2: Picks the Last element (Recently inserted) from the temporary and table and execute the predicate in recursive part and get the newly qualified rows
Step 3: Sends the qualified rows to output and also write them to temporary table in tempdb
Step 4: Now remove the previously selected element in tempdb
Step 5: Go to step2 and repeat the same process until no more elements in temporary table

The proposed system uses same logic as recursive CTE for executing recursive queries but it deals everything in terms of objects. Second method in hibernatex package executes anchor part and put the results into list and result list. It picks the last elements from list and executes the recursive part. Now, the results are stored in temporary list and result list, once the list is empty and then temporary list is assigned to list. This process continues till the temporary list is empty. Finally, it returns the list of objects from result list.

RESULTS AND DISCUSSION

The proposed system is tested and the performance analysis shows various execution times for a sample recursive queries using Hibernatex package, i.e., proposed system and the traditional native SQL query are 479.2 and 1833.4 msec. The results shows a remarkable difference in executing recursive query using Hibernate, i.e, via. ORM system Table 1.

Table 1: Execution times for executing recursive query

Run. No.	Execution time of proposed system (msec)	Execution time of Existing system (msec)
1	524.0	1778.0
2	483.0	1835.0
3	481.0	1810.0
4	452.0	1888.0
5	456.0	1856.0
Average execution time	479.2	1833.4

CONCLUSION

The system proposed for querying scalable and redundant data via. recursive queries in ORM systems provides an extension to hibernate which is used to execute recursive queries irrespective of the backend. Though backend does not support recursive queries, the proposed system executes recursive queries and produces the result. The proposed system executes recursive queries much faster than the existing system does. The proposed system creates an extension to Hibernate in order to execute recursive queries. It does not alter HQL syntax to recognize and execute recursive query but it calls a method which executes recursive queries. In future, HQL syntax can be altered, so that, it can recognize and execute recursive queries as it does the normal query execution.

REFERENCES

Bauer, C. and G. King, 2006. Java Persistence with Hibernate. Manning Publications, Greenwich, Connecticut, USA.,.

Boniewicz, A., P. Wisniewski and K. Stencel, 2013. On redundant data for faster recursive querying via ORM systems. Proceedings of the 2013 Federated Conference on Computer Science and Information Systems (FedCSIS'13), September 8-11, 2013, IEEE, Krakow, Poland isBN:978-83-60810-52-1, pp: 1451-1458.

Burzanska, M., K. Stencel, P. Suchomska, A. Szumowska and P. Wisniewski, 2010. Recursive Queries Using Object Relational Mapping. In: Future Generation Information Technology, Kim, T., Y. Lee, B.H. Kang and D. Slezak (Eds.). Springer, Berlin, Germany isBN:978-3-642-17568-8, pp: 42-50.

Melnik, S., A. Adya and P.A. Bernstein, 2008. Compiling mappings to bridge applications and databases. ACM. Trans. Database Syst., 33: 67-78.

O'Neil, E.J., 2008. Object-relational mapping 2008: Hibernate and the Entity Data Model (EDM). Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, June 09-12, 2008, ACM, Vancouver, Canada isBN:978-1-60558-102-6, pp: 1351-1356.

Ordonez, C., 2010. Optimization of linear recursive queries in SQL. IEEE. Trans. Knowl. Data Eng., 22: 264-277.

Przymus, P., A. Boniewicz, M. Burzanska and K. Stencel, 2010. Recursive Query Facilities in Relational Databases: A Survey. In: Database Theory and Application, Bio-Science and Bio-Technology, Zhang, Y., A. Cuzzocrea, J. Ma, K.I. Chung and T. Arslan et al. (Eds.). Springer, Berlin, Germany isBN-13:978-3-642-17621-0, pp: 89-99.

Szumowska, A., A. Boniewicz, M. Burzanska and P. Wisniewski, 2011. Hibernate the recursive queries-defining the recursive queries using hibernate ORM. Adv. Databases Inf. Syst., 789: 190-199.