International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 05 | May -2017 www.irjet.net p-ISSN: 2395-0072

A Comparative Study of Database Connection Pooling Strategy

Sohel S. Shaikh[1], Dr. Vinod. K. Pachghare[2]

[1]Student, Dept. of Computer Engineering, College of Engineering Pune, Maharashtra, India
[2]Associate Professor, Dept. of Computer Engineering & IT, College of Engineering Pune, Maharashtra, India

---

Abstract - *In recent times, with fast advancement in technology there is a faster need for reducing the dormant user time while interacting with software applications. Every application makes a use of databases to store their data for several purposes. The conventional database is a good fit for application with fewer database interactions. As the number of interactions increases the load on the application increases thus leading to rise in response time. This paper proposes a novel technique of database interaction with the advent of database Connection Pooling. It details about the proposed model and elaborates its working principle. The proposed methodology advances the efficiency of accessing database significantly.*

Key Words: Database Connection Pooling, persistent systems, performance optimization.

1. INTRODUCTION

In today's world, there is enormous data available. These data have a great potential to expand ones' information base. These data are needed to be stored and well retrieved as and when obligatory. To stock this data databases are used. Database is a set of information that is schematized so that it can be fluently retrieved, sustained and restructured. To smoothen these actions a database management system is established. Connections are developed to communicate with these database management systems. For every request a connection is formed, resources such as locks, transactional logs, etc. are assigned. These connections are luxurious to create, owing to the overhead of starting network acquaintances and adjusting database connection sessions in back-end chronicles. Database resources such as locks, memory cursors, transaction logs, statement handles and temporary tables increase with increase in the number of concurrent connection sessions. [1]

In such scenarios, there is a compulsive need for an alternative mechanism called the Database Connection Pooling. Database Connection Pooling is a reserve of live database connections. The admission of Database pools facilitates faster connection of applications with the database management systems. Also, the chief feature is that the connections within a pool can be reused once its purpose is accomplished. This paper proposes a model implementing connection pools and provides the comparison among the applications developed using traditional database and database connection pools.

The paper progresses as follows:

Section 2 provides insight about the existing conventional database connection. Section 3 gives an overview of the Database Connection Pooling. Section 4 delivers the proposed methodology. Section 5 details the features that were set for the proposed system implementations. Section 6 describes the experimental results by comparing the different techniques. Section 7 concludes the accomplishments of the paper.

2. CONVENTIONAL DATABASE CONNECTION TECHNIQUE

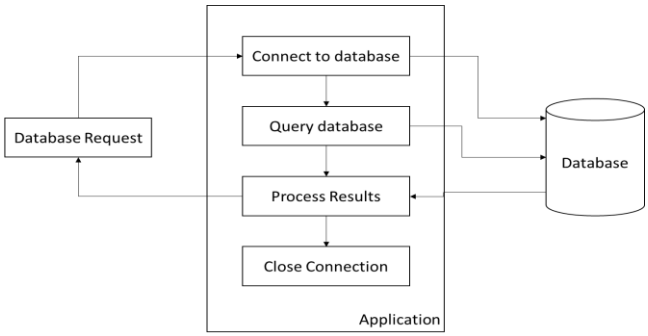

Fig 1: Conventional Database Connection

At the present time, web applications have become an inevitable necessity for every individual. Such applications necessitate run time processing of requests from the users. Database connections are to be established on the fly to facilitate these huge number of requests. This implies substantial load on the servers to create a connection for every request from scratch, allocate resources and close the connections. Hence runtime creation of such connections becomes too expensive. Also, garbage collection becomes an additional overhead. Connection initialization involves time consuming processing to perform user authentication, establishing transactional contexts and establishing other aspects of session that are required for succeeding database usage. It is not only expensive to create but also expensive to maintain overtime. In addition, if the connections are left opened even after completion of tasks, it may lead to more serious issues like memory leakage leading the application to break.

The correct remedy for avoiding such critical scenarios is use of Database Connection Pooling. It is well elaborated in the next section.

3. DATABASE CONNECTION POOLING

Database connection pools are a group of live connections to the database systems. They are maintained by the applications which require frequent connections to be established with the database. On every connection request, a connection is pulled from the pool and the demand is sufficed. This enhances performance of applications and facilitates efficient concurrency and scalability. It reduces or nearly eliminates the waiting time of applications requesting a connection. Furthermore, use of connection pooling is transparent, it does not hinder the business applications in any way. They do not require any modifications for communicating with the database pools. The pools can be tweaked and tuned by application administrators without the knowledge of the applications. If the applications use generic JDBC connections, one could simply point it at different vendors' database without even altering any code.

The system performances are enhanced by using database connection pools as it reduces the frequency of creating and closing database connections. Also, it provides flexibility in creating, configuring and closing connection pools. It further enhances the stability in interaction with the databases.

The basic requirement of connection pooling is to predetermine the number of connections provided by the database systems during the initialization stage of application system, that are to be accommodated within the memory called the database connection pools. These pools are to be organized by using container objects such as Vector, Stacks, etc. Thus, the applications only have the overhead for creating these connections in the beginning and closing them after conclusion of the tasks. It is relieved of the operations of acquiring connections and terminating them, thus reducing huge amounts of system resources consumption thus improving the execution speed.

Connection pooling increases efficiency, because scarcer connections are created and unbolted. Furthermore, connection pooling allows the application server to actually handle more concurrent instantaneous client sessions than the maximum number of open connections allowed by the database at any instant. It leads to conservation of resources.

4. METHODOLOGY

The connection pool is a concept of hoarding the connections. It is a technique of creating and handling a pool of connections that are ready for use by any thread that needs them. The connection pools initialize several connections on startup. It obeys the datum that most applications only require a thread to have access to a jdbc connection when they are enthusiastically processing a transaction, which typically takes milliseconds to complete. When not processing a transaction, the connection would otherwise sit idle. Instead, connection pool allows the idle connection to be used by some other thread to do valuable work. When the connection is lent out from the pool, it is used exclusively by the thread that requested it. After its completion, the connection is again reimbursed into the connection pool.

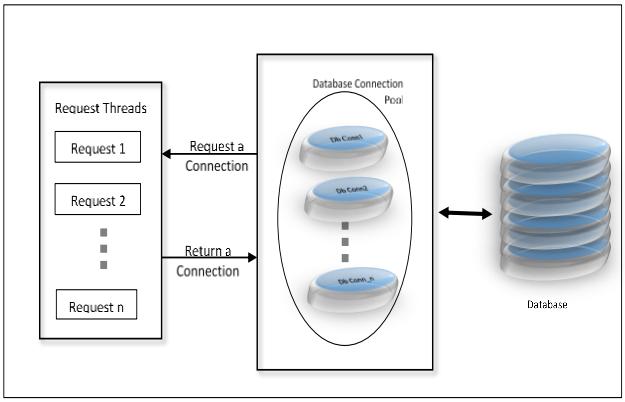

Fig 2: Working of Proposed System

5. IMPLEMENTATION

The technology stack used for implementing the proposed Database Connection Pooling are Java 7 and MySQL database.

The attributes of proposed Database Connection Pool were set as follows:

Table 1: Proposed System Attributes

ATTRIBUTE	DESCRIPTION
Resource Name	Name of database (E.g. jdbc/database_name)
Auth	Container
Driver classname	com.mysql.jdbc.Driver
url	Jdbc Connection url to the specified database
Username & password	Database username & password
Initial size	The initial size of pool
maxActive	Maximum number of allowed active connections
maxIdle	Maximum number of allowed idle connection

minIdle	Minimum number of allowed idle connections
maxWait	Longest allowed waiting time
Validation query	To check if connections are valid (E.g. Select 1)
Validation interval	The interval for validation (30000)

Methods of Proposed Database Connection Pool are described as follows:

Table 2: Proposed System Methods

METHOD	DESCRIPTION
createConnection()	Creating a database connection
init()	Initializing the database connection pool and invoking it at server start up
getConnection()	Retrieving the database connection
freeConnection()	Return the connection to the database connection pool

6. EXPERIMENTAL RESULTS

A traditional database system was developed and tested against the proposed database connection pooling system. The experimentation was simulated for both the techniques to compare their performances. The database access requests varied from 1, 10, 100, …, 1000 and execution time was recorded in milliseconds. The results thus obtained were very substantial. It is depicted as below:

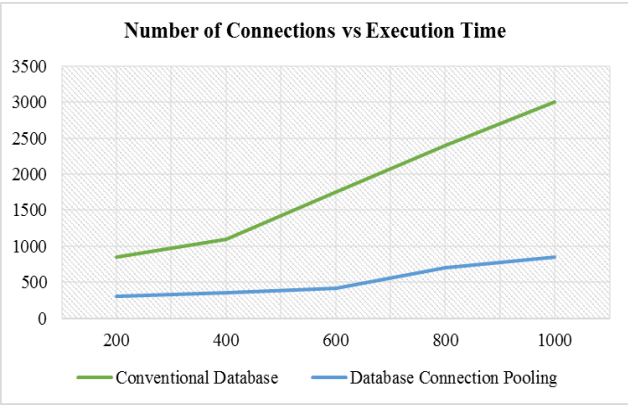

Chart 1: Number of Connections vs Execution Time

It can be observed that for an intricate application, recurrent establishment and termination of connections will significantly decrease the system's performance. Hence, the usage of the traditional database system was becoming a bottleneck of systems' throughput. Considering all these the proposed system avoids excessive feasting of resources by recycling the connections to the connection pools. Also, it was observed that the conventional database produced loads of garbage as the connections are destroyed each time, leading to reduce in speed of machine execution speed. The proposed connection pool emerges as the efficient alternative for scenarios where the database requests are large in number.

7. CONCLUSION

The proposed database connection pool is a feasible key for intricate complications in database applications. It is best suitable for applications that require concurrent database accesses. It surpasses the traditional database systems with traits of low efficiency and bulky occupancy to provide faster execution and lesser garbage production. This paper offers a novel management technique to enhance database connections. Reusing of connections secures the execution time by reducing the interaction with the database to create and terminate connections. It achieves the target of providing a better service while utilizing limited system resources.

REFERENCES

[1] Zhang, T. F., Zhang, Y. J., & Yao, J. (2014). A study of database connection pool. In *Applied Mechanics and Materials* (Vol. 556, pp. 5267-5270). Trans Tech Publications.

[2] GUIa, D., LIa, G., Zhanga, C., & Yina, P. A Method On Connection Pool Service For Distributed Heterogeneous Databases In Urban Geographic Informaiton Public Platform.

[3] Reese, George. *Database Programming with JDBC and JAVA*. " O'Reilly Media, Inc.", 2000.

[4] SUN, Y. F., & SONG, Z. S. (2004). Database Access Technology Based on Connection-pool in JSP [J]. *Computer Applications*, *6*, 80-81.

[5] Othman, L. A., Hosny, H. M., & Aly, S. G. (2006, September). Aspectizing Database Connection Pooling for Improved Run-Time Performance Measures in Web Applications. In *Information Reuse and Integration, 2006 IEEE International Conference on* (pp. 528-532). IEEE.

[6] Othman, L. A., Hosny, H. M., & Aly, S. G. (2011, December). A comparative analysis of database connection pooling implementations with emphasis on the added value of aspect orientation. In *Computer Systems and Applications (AICCSA), 2011 9th IEEE/ACS International Conference on* (pp. 102-111). IEEE.

[7] Li, B. Z., Zheng-jun, J., Yi-jun, L., Ye, L., & Zhi-min, Y. (2009, August). XML configuration-based self-adaptive database connection pooling in NMVS. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on* (pp. 130-133). IEEE.

[8] Hohenstein, U., & Jaeger, M. C. (2009). Database Connection Monitoring for Component-based Persistence Systems. *International Journal on Advances in Intelligent Systems Volume 2, Numbers 2&3, 2009*.

[9] Liu, F. (2012, August). A Method of Design and Optimization of Database Connection Pool. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on* (Vol. 2, pp. 272-274). IEEE.

[10] Othman, L. A., Hosny, H. M., & Aly, S. G. (2011, December). A comparative analysis of database connection pooling implementations with emphasis on the added value of aspect orientation. In *Computer Systems and Applications (AICCSA), 2011 9th IEEE/ACS International Conference on* (pp. 102-111). IEEE.

[11] Souza, F. N., Arteiro, R. D., Rosa, N. S., & Maciel, P. R. (2008, December). Performance models for the instance pooling mechanism of the JBoss application server. In *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International* (pp. 135-143). IEEE.

[12] Yang, J., Zhang, Z., & Zhao, Y. (2010, August). Analysis on database connection mechanism of web application system in dreamweaver. In *Internet Technology and Applications, 2010 International Conference on* (pp. 1-4). IEEE.

[13] Luo, R., & Tang, X. (2004). Design and Realization for JDBC-based Database Connection-pool [J]. Computer Engineering, 9, 036.

[14] Singh, V., Sawant, U. V., Prateek, G. O. E. L., & Deshaveni, N. G. (2014). U.S. Patent No. 8,874,609. Washington, DC: U.S. Patent and Trademark Office.

[15] Xiufen, Z. G. F. (2011). A New Model of Database Connection Pool [J]. Computer & Digital Engineering, 2, 043.

[16] HOU, C., YANG, Z., & LIU, W. (2006). Application and improvement of database connection pool based on J2EE architecture [J]. Computer Technology and Development, 16(10), 8-10.

[17] Siyun, Q. (2005). Design for Web-based Supply Chain Management System Database Connection-pool. *Industrial Control Computer*, *12*, 027.

[18] Liu, F. (2012, August). A Method of Design and Optimization of Database Connection Pool. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on* (Vol. 2, pp. 272-274). IEEE.