ATLANTIS PRESS
Advances in Intelligent Systems Research (AISR), volume 155
2018 International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018)

Research on Database Fragmentation Technology Based on Hibernate Shards Framework for SaaS Development

Fei Zhou* and Yurong Guan
Huanggang Normal University, Huangzhou, Hubei, China
*Corresponding author

***Abstract*—As a child frame of Hibernate framework, Hibernate shards provides an effective implementation scheme for the database slicing technology developed by SaaS, expands the shared database and sharing mode, and the program development, deployment and implementation are similar to Hibernate, At present also has certain limitation.**

Keywords- SaaS; hibernate shards; database fragmentation

I. INTRODUCTION

Compared to common enterprise applications, SaaS applications are multi-tenant, which is a key feature. Multi-Tenant is the ability of an application to host multiple tenants and share databases in a single code base. A relatively cost-effective strategy for implementing multi-tenant is to provide a shared database and shared-mode strategy for all tenants. But this approach poses a huge challenge to database expansion because the database is shared among all the tenants of the SaaS application. Another simple method is to extend the shared database through database fragmentation based on the tenant ID.

A database fragment is a partition of a database, and each partition is called a fragment. In Hibernate, a sub frame hibernate shards is extended, which adds horizontal partitioning support, has fragmentation capabilities, and effectively resolves database expansion problems for SaaS applications.

II. DATABASE FRAGMENTATION TECHNOLOGY FOR SAAS APPLICATIONS

The SaaS application uses a shared database and shared schema approach that addresses the problem of reduced performance when too many users are accessing the database or database at the same time. Database fragmentation is the most efficient way to extend a database horizontally, because the rows in a shared schema are different for different tenants depending on the tenant ID. The database can be partitioned horizontally based on the tenant ID, which then moves the data belonging to each tenant to a single partition. Database fragmentation offers many advantages, such as faster read and write to the database, improved search responses, smaller table sizes, and on-demand for tables.

Hibernate Database Fragmentation the application code that is connected to multiple databases is extracted based on the tenant environment by providing metadata configuration and API partitioning, that is, the operation of inserting data according to the tenant environment and the operation of reading data from multiple databases.

III. IMPLEMENTING PARTITIONS USING HIBERNATE SHARDS

Hibernate shards minimizes the complexity of the implementation of fragmentation data, and its primary goal is to enable applications to query and process shared datasets using standard Hibernate Core APIs. The advantage is to provide a non-invasive solution to support database fragmentation, while using Hibernate to build existing SaaS applications and to allow solutions for SaaS applications that already use Hibernate but do not yet require fragmentation.

Hibernate Core provides fragmentation-aware interface extensions, so the code does not need to know that it is interacting with a fragmented dataset. The fragmentation-aware extension that acts as a shard engine is as follows:

- org.hibernate.shards.ShardedSessionFactory
- org.hibernate.shards.criteria.ShardedCriteria
- org.hibernate.shards.session.ShardedSession
- org.hibernate.shards.query.ShardedQuery
- org.hibernate.shards.strategy.access.ShardAccessStrategy , Hibernate decides how to apply database operations across multiple fragments. This policy is referenced every time a query is executed. The two default implementations of Sequentialshardaccessstrategy and Parallelshardaccessstrategy have been provided.
- org.hibernate.shards.strategy.resolution.ShardResolutionStrategy , which is used to determine the fragment set where the object of the given ID is located. Fragment resolution is associated with ID generation, and Hibernate provides multiple policies for ID generation, such as local, application-level UUID generation, and distributed Hilo generation.
- org.hibernate.shards.strategy.selection.ShardSelectionStrategy, which allows developers to decide where to create the fragments of a new object. Although Hibernate initially provides an open box-used loop implementation, it is necessary to provide a tenant ID based implementation for SaaS-based applications.

Copyright © 2018, the Authors. Published by Atlantis Press.
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

Adding a database is necessary when a SaaS application's dataset exceeds the database capacity originally allocated to the application. The ideal solution is to distribute data across multiple fragments. Re-fragmentation is a complex issue that, if not considered in the design process, can cause serious concurrency when managing production applications.

Virtual fragmentation can be used to point multiple virtual fragments to physical fragments in a hibernate configuration file. Allows developers to define multiple virtual fragments and map them to two or three of physical fragments. Developers should create virtual fragments for each tenant that can be mapped to an expected physical database. Once you have provided a new tenant, you can then create a virtual fragment and map it to the same expected physical fragment or more physical fragment.

IV. Essentials of Development, Deployment and Implementation of SaaS Program Based on Hibernate Shard

The main libraries and tools used for deployment and implementation are:

- hibernate-shards.jar
- hibernate3.jar
- hsqldb.jar
- Eclipse IDE

Initially, in a shared database, shared schema method, only one hibernate.cfg.xml is configured. As the database instance grows, add hibernate.cfg.xml to each database instance used in the fragment. These Hibernate profiles differ in connection, session factory, and shared ID details

Hibernate shards configuration is similar to Hibernate configuration. The main difference is that the hibernate.connection.shard_id and Hibernate.shard.enable_cross_shard_relationship_checks properties are added. Also note that the session factory and the shared ID must represent fragmentation uniquely.

Define the fragmentation policy to use in your application. You need to create a custom shardselectionstrategy called Customershardselectionstrategy, which implements Shardselectionstrategy, which determines the fragmentation of rows stored based on the tenant ID. The fragment ID is mapped relative to the virtual fragment ID map configured in the subsequent steps. If you do not provide a virtual fragment ID mapping (optional), the physical Fragment ID mapping is the default.

V. Limitations of the Hibernate shards Framework

Some of the main limitations of the Hibernate shards framework are as follows:

ShardedSessionImpl, ShardedCriteriaImpl and ShardedQueryImpl provide some of the less-than-used methods that haven't been fully implemented.

So far, the fragmentation framework cannot be used for HQL, and it can only be used for SQL.

Hibernate Shards does not provide support for distributed transactions in an unmanaged environment. If the application requires distributed transactions, we need to insert transaction management implementations that support distributed transactions.

The Hibernate shards Framework currently does not support cross shared object graphs.

The current configuration mechanism is not feasible when Sessionfactory is configured with JPA.

VI. Conclusions

Because of its innate feature and the requirement of database fragmentation, the SaaS application Hibernate shards Framework provides a strong feasibility for database fragmentation. However, there are some limitations in the hibernate shards Framework. Through the late refinement and leveraging of the Hibernate shards Framework in the data tier, it will help build and design a truly multi-tenant SaaS application based on Java EE.

Acknowledgment

I thank reviewers for constructive reviews and suggestions that improved the quality of this paper. This work was supported by the guidance project of science and technology research plan of Hubei Provincial Department of Education, project name is research on the construction of content management system in education industry based on SharePoint.

References

[1] S Navathe,S Ceri,G Wiederhold,J Dou. Vertical Partitioning Algorithms for Database Design . 1984,9(04).

[2] Zhang Zhenyou,Chang Shiguang,Ding Tiefan. Research on the Dynamic Integrating with Heterogeneous Database System based XML and Hibernate. International Conference on Information Technology and Management Innovation.2013.

[3] Wei-Wen Wu.Developing an explorative model for SaaS adoption.Expert Systems with Application.2011, 38(12).