Multi SQL dialect framework in GaussDB

Danny Chen, Yang Sun, Jun Tang

Cloud BU
Huawei Canada
danny.chen@huawei.com,yang.sun5@huawei.com,jun.tang1@huawei.com

1 Abstract

Although there are ANSI/ISO standards for SQL, each database vendor may choose to implement its own dialect. That may include grammar, data types, function implementation, system table/views, protocols. Database SQL compatibility allows applications written for database A to run on another database B without modification, if database B is compatible with database A. With SQL compatibility feature, database application developers can quickly adopt a new database system with previous experiences; End users can have the freedom to migrate their existing applications to a different database for cost saving purpose; DBAs can even consolidate their multiple database systems into the same system for easy of maintenance. Through this talk, we will describe the challenges during the development and showcase how we built a framework to support multiple SQL dialects without sacrificing performance, while providing high degree of compatibility and maintaining good isolation between the SQL dialects. Using this framework, GaussDB already supported Oracle and MySQL compatibility mode, and can easily extend to support other modes when needed.

2 Challenges of supporting multiple SQL dialects

Unlike developing a new database, supporting multiple SQL compatibility modes requires the database to be backward compatible to existing behavior, while supporting new SQL dialects. The work focused on four major areas: protocol, syntax and semantics, meta data and error handling. On top of these changes, compatibility also affects deployment/migration process and utilities such as monitoring and backup/restore. Each of these areas has its own challenges, here are some examples:

Each SQL dialects has its own set of keywords, bringing challenges on identifier behavior, including reserved objects and object case sensitiveness.

Each databases has its own typing system and type conversion mechanism. They affect function lookup as well. It is impractical to implement a whole new mechanism for each compatibility mode.

Although SQL standard defined the guidance on SQLSTATE, but it's up to each database vendor to define the error code which is closely tied to individual error scenarios. It's almost impossible to map all errors from one database to another implementation.

Intrusive changes to existing code may cause performance degradation and duplicating too much logic will make the maintenance difficult. Finding a good balance between extensibility and good performance is not trivial.

3 Introducing a framework to support Multi-SQL dialect in GaussDB

The framework includes an extension, a templated database, a set of function hooks and the methodology for error handling.

We contained all new logic, including grammar definition, new protocol, new data types, new operator and function logic implementation, in its own extension, which is similar to a plugin, and compiled into a shared library. We carefully chose the new logic entry point and define them as function hooks. The function hooks are high enough to guarantees isolation between different SQL dialects. Within GaussDB's typing system, we used a generic type ANY to simulate weak type system like MySQL and inherit CAST table for strong type systems. All the objects including system catalogs, tables, views, types, functions are grouped together and placed in a template database. During compatibility mode database creation, the template database is used to speedup the creation process. The template database and extension design also greatly simplified online migration process as well as hot patch application logic.