

Experimental Study on Heterogeneous data storage and exchange abstract storage middleware

Weihaio Liu, Linhai Jiang, Xitong Luo, Shiwei Guo

School of Information Technology Engineering Tianjin University of Technology and Education Tianjin 300222, China

Abstract: Some Internet companies on the market do not have rich experience in database development, and there are certain problems in the face of sudden, urgent and high concurrent requirements like epidemic health code. Hope that through the interface provided by this middleware, the data can be automatically split according to certain rules. Using the advantages of different architecture databases, the data were stored in different architecture databases to achieve data heterogeneity. And it can extract, combine and return data from different databases, abstract the process of database access and access. The decoupling of database and business code is realized to solve the pain point of high-performance product development difficulty.

Keywords: Database; Exchange; Middleware.

1. Introduction

As a heterogeneous database system is a collection of related multiple database systems, which can realize data sharing and transparent access, each database system already exists and has its own DBMS before joining the heterogeneous database system, so we decided to use mysql, redis, and hbase databases to form a heterogeneous database system.

The main research content is according to the user corresponding constraint configuration, the data will be divided into this product. Different fields are scattered into different databases, and the data has its own business characteristics. For example, some data are used frequently, accessed frequently, and modified frequently, which are suitable for storing in redis database. Data that is written and deleted frequently and read infrequently is a good place to store in an hbase database. Other trivial non-obvious features are stored in MySQL. It can also be used to concatenate data fields stored in different databases into a single table for users to use. Users do not need to know the details of different databases and can write good performance products.

2. Mechanical Analysis

This product (Figure 1) is configured according to the corresponding constraints of the user, and the data incoming to this product is split. Different fields are scattered into different databases, and the data has its own business characteristics.

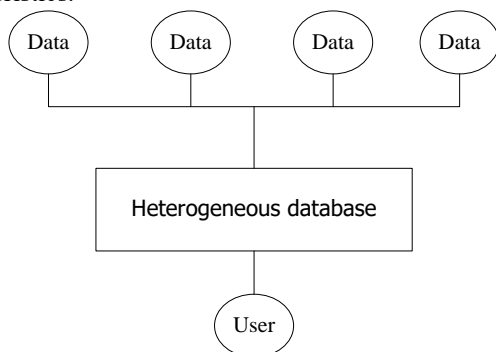


Fig. 1 Heterogeneous database overall structure

Study the underlying principle of mysql MySQL has many storage engines, understand the most common innodb storage engine, because innodb storage engine supports transactions, can be mainly for online transaction processing applications, and supports row-level locking, because mysql is stored on a physical disk, processing data in memory execution. The disk read and write speed is very slow, frequent read and write, the performance will be poor, innodb can divide the data into several pages, so that at least one page of data is read into the memory or one page of data is written to the disk, which can reduce the number of interactions, thus improving performance.

MySQL indexes use B+ trees, and B+ tree nodes store indexes. When the storage capacity of a single node is limited, a single node can store a large number of indexes, which reduces the height of the entire B+ tree and reduces disk IO. Secondly, the leaf nodes of the B+ tree are where the real data is stored, and the leaf nodes are connected with a linked list, which is itself ordered and more efficient when searching the data range. InnoDB is aggregated indexing, so both data and indexes are stored in the same file. First of all, InnoDB will build an index B+ tree based on the primary KEY ID as the key, and the leaf nodes of the B+ tree store the primary key ID and the corresponding data. InnoDB needs to save storage space. There may be many indexes in a table, and InnoDB will generate an index tree for each indexed field. If the index tree for each field stores specific data, the index data file of the table becomes very large (data is extremely redundant). From the perspective of saving disk space, it is really not necessary to store specific data for each field index tree, and through this seemingly "unnecessary" step, a huge amount of disk space is saved at the expense of the performance of fewer queries.

There are three types of data constraints, namely high frequency data, high write and low read data, and common data. Data with high frequency constraints is stored in redis, data with high write constraints is stored in hbase database, and other data is stored in MySQL database.

For example, some data are used frequently, accessed frequently, and modified frequently, which are suitable for storing in redis database. Data that is written and deleted frequently and read infrequently is a good place to store in an hbase database. Other trivial non-obvious features are stored in MySQL (Figure 2).

The data types are int, float, string, and bool (Table 1). Choosing the right type is very important to optimize your database. In the real world, different data can be abstracted, and different data have different types. Different types mean that different data types have different operations and different binary representations. Therefore, it is necessary to design different data types. Data capture is the basis of

database synchronization and capture of changing data. Therefore, we refer to some methods for data capture, such as snapshot-based method and log-based method. Snapshot-based method is relatively inefficient and not feasible as a synchronous database, so log-based method can be effectively synchronized.

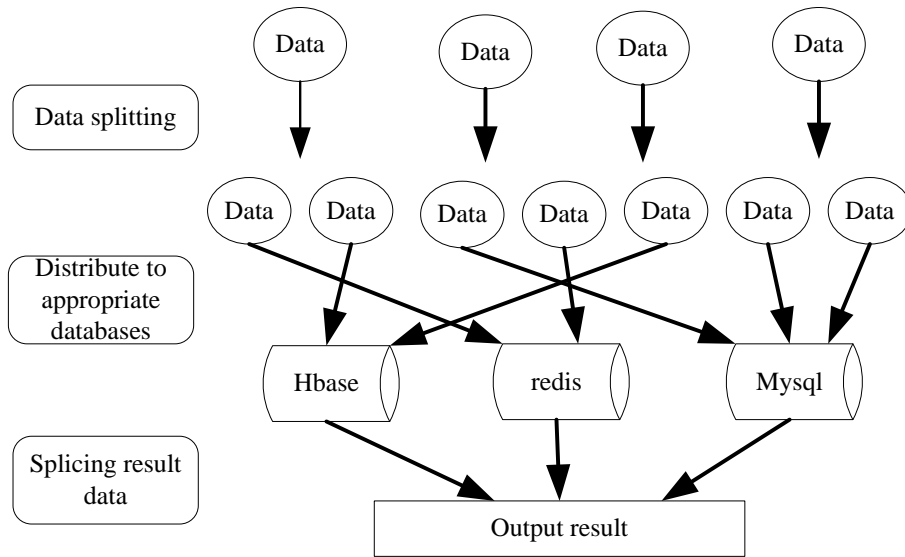


Fig. 2 Data stream diagram

You can define standard log formats, including key information such as time stamps, user ids, data ids, and operation types (read, write, and delete). The log collector monitors the modules involved in data access in the system to ensure that logs are generated for each data operation. Implement an access frequency counter according to the log, and count the access frequency of each data item according to the log record. An algorithm is designed to calculate the weights of access frequency features for each data item based on the results of access frequency statistics. In addition, real-time processing module is introduced to monitor data access events and update the characteristics of access frequency in real time. Asynchronous processing mechanisms are used to ensure that real-time processing does not affect primary system functions. Combined with real-time data analysis, the algorithm is designed to dynamically adjust the weights at run time to adapt to changes in access patterns.

The design of the rules engine in this heterogeneous database project is critical, which is responsible for intelligent data classification based on the corresponding constraints and data characteristics of the user. We define the following three types of rules:

Rule 1. If the data access frequency is higher than 35% of the threshold and the data is of the frequently modified type, the data is stored in Redis.

Rule 2: If the data is frequently written but infrequently read, the data is stored in HBase.

Rule 3: If the data is regular business data and does not belong to the type of high-frequency access, high-frequency modification, or frequent writing, the data is stored in MySQL.

The rule engine will analyze the incoming data. If the access frequency of a certain data is higher than 35% of the threshold and it belongs to the type of frequent modification, the rule engine will match rule 1 and perform the corresponding action to store the data in Redis. If the data is frequently written but infrequently read, rule 2 is matched and

the data is stored in HBase. If a piece of data belongs to a common service and does not meet the requirements for storing it in Redis or HBase, the rule engine matches rule 3 and performs corresponding actions to store the data in MySQL.

A data abstraction layer is introduced: data models in different databases are abstracted and unified into a common data model. This abstraction layer hides the differences in the underlying database, allowing users to access the data through a uniform interface. Assume that there is a user table. The fields stored in MySQL include user_id and username, the fields stored in Redis include user_id and last_login_time, and the fields stored in HBase include user_id and total_orders. Through the data abstraction layer, these fields can be mapped to a generic user object, and the user only needs to worry about the generic object without worrying about which database it is stored in.

Data synchronization and mapping: Establish a data synchronization mechanism to ensure data consistency in different databases. This includes real-time or periodic synchronization, synchronizing data from one database to another, and performing necessary mapping transformations. Assume that a user has changed the user name in MySQL. The system needs to synchronize the change to Redis and HBase. In addition, for a field mapping, for example, username corresponds to nickname in Redis, you need to map the field during synchronization.

Query optimization: During query execution, the system needs to optimize the query plan, select the appropriate database according to the query requirements of users, and merge the results. If a user needs to query detailed information about a user, the system can query username in MySQL, last_login_time in Redis, and total_orders in HBase based on query conditions, and then combine these results into a complete user information and return it to the user.

Table 1. Data types

type	size	scope	purpose
int	4 Bytes	(-2 147 483 648, 2 147 483 647)	Integer
float	4 Bytes	(-3.402 823 466 E+38, -1.175 494 351 E-38), 0, (1.175 494 351 E-38, 3.402 823 466 351 E+38)	floating-point
string	0-65535 bytes		string
bool	1 Bytes	(-128, 127)	bool

3. Innovation and characteristics

Through the analysis of the corresponding constraints of the user, the system is able to intelligently split and disperse the incoming data to be stored in different databases. This intelligent data allocation can optimize the data storage structure and improve the overall system

The project has flexible configuration options, allowing users to adjust the data storage strategy according to business needs. According to the characteristics of the data, users can flexibly configure which fields are stored in Redis, HBase or MySQL to achieve personalized data management. By storing data in different databases according to their business characteristics, the system is able to optimize the operation of different types of data. For example, the data that is read and written frequently is stored in Redis, the data that is written and deleted frequently but not read frequently is stored in HBase, and the general data is stored in MySQL to improve the overall performance. The project realizes the seamless splicing of data fields stored in different databases into a table, provides users with a unified data interface, so that users do not need to understand the details of the underlying database, reduces the threshold of use, and improves the development efficiency. Selecting the appropriate database storage according to the business characteristics of the data makes the selection of the database more suitable for the actual business needs, so as to find a balance between performance and cost. Users can easily use this heterogeneous database system and focus on the development of business logic.

4. Conclusion

By digging into the essence of the problem, combined with reasonable algorithms and technical means, the problems of intelligent data allocation, dynamic configuration, performance optimization, and unaware splicing are successfully solved, and an efficient, flexible and easy-to-use data storage solution is provided for users.

Acknowledgment

Supported by College Student Innovation and Entrepreneurship Training Project of Tianjin University of Technology and Education (No. 202210066162)

References

- [1] X Zhang. Reliable mass data storage based on heterogeneous part of repeat code [J]. Journal of electroacoustic technology, 2023,47 (9) : 70-72. The DOI: 10.16311 / j.a udioe. 2023.09.021.
- [2] Q Yu. Research on Water Chiller Air Conditioning Management System and Data Security Based on Multi-source Heterogeneous Data [D]. Shandong construction university, 2023. DOI: 10.27273 / , dc nki. Gsajc. 2023.000079.
- [3] C Liu. Application of Heterogeneous Storage Data Migration Technology in Enterprise [J]. Digital Communication World, 2023, (03):124-126.
- [4] C Luo, X Jin, Y Zhang et al. For heterogeneous data and the location of the DHT storage decoupling algorithm [J]. Journal of software, 2023, (10): 4930-4940. The DOI: 10.13328 / j.carol carroll nki jos. 006663.