UDC 004[005]

DATABASES ARCHITECTURE OPTIMIZATION FOR IMPROVING RELIABILITY AND SPEED OF WORK

KAREN GRIGORIAN, DMITRY PIATKIN
Polotsk State University, Belarus

This article discusses the technologies and techniques for implementing the fault tolerance of the server part of the application and increasing the response speed of the database.

The main problem when working with a large amount of data is often a poorly organized part of the application that is responsible for data storage. Modern technologies often require increased attention to the back-end part of the application. And the development of the back-end begins with a database. Databases, their device and architecture, play a key role in the appropriate usage of the application and the integrity of user data.

There are two fundamental approaches to the organization of database interaction architecture in the application: shading, replication.

Let's consider the principle of replication. Replication allows you to create a complete database duplicate. So, instead of one server, there will be several servers in the product being developed. Most often use the master-slave scheme:

1. Master is the main database server, where all data is being received. All changes to the data (addition, update, deletion) must occur on this server.

2. Slave is an auxiliary database server that copies all data from the master. From this server should read the data. There can be several servers of this type.

Replication allows you to use two or more identical servers instead of one. Data read operations (SELECT) are often larger than data modification operations (INSERT / UPDATE). Therefore, replication allows you to unload the main server by transferring read operations to the Slave.

To organize replication technology, two database connections must be declared in the product code of the product. One for the master and one for the Slave. An example implementation is provided in Listing 1.

Listing 1

```
<?
$master = mysql_connect('10.10.0.1', 'root',);
$slave = mysql_connect('10.10.0.2', 'root',);
$q = mysql_query('INSERT INTO users ...', $master);
$q = mysql_query('SELECT * FROM users WHERE...', $slave);
```

Replication is usually supported by the DBMS itself (for example, MariaDb) and is configurable regardless of the application.

It should be noted that replication itself is not a very convenient scaling mechanism. The reason for this is the data dissynchronization and delays in copying from Master to Slave. But this is an excellent tool for providing fault tolerance. You can always switch to a slave if the master breaks and vice versa. Most often, replication is used in conjunction with shading, for reasons of reliability.

The second approach to organization of the structure is shading. The essence of the shading is in dividing the database into separate parts so that each of them can be carried to a separate server. There are several types of shading - vertical and horizontal.

Horizontal shading is the separation of one table into different servers. This should be used for tables that store impressive amounts of information and do not fit on a single server. The division of the table into pieces is done according to this principle:

1. Multiple servers create the same table (only the structure, without data).

2. The application selects the condition by which the desired connection will be determined (for example, even for one server, and odd ones for another).

3. Before each access to the table, the desired connection is selected.

Suppose that there is an application that works with a highly loaded table that stores user's photos. There are also two servers (usually called shards) for this table. A possible usage scenario is for odd users to store and retrieve information using the first server, and even the second. Thus, on each of the servers there will be only a part of all data about the photos. An example is shown in Listing 2.

Listing 2.

```
<?
$image_connections = [
  '1' => '10.10.0.1',
  '2' => '10.10.0.2',
];

$user_id = $_SESSION['user_id'];
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection =
mysql_connect($image_connections[$connection_num],'root', '');
$q = mysql_query('SELECT * FROM images WHREE user_id = ' . intval($user_id), $connection);
```

The result of the operation $ user_id% 2 is the remainder of the division by 2. That is, for even numbers - 0, and for odd ones - 1.

Horizontal shading is a very powerful tool for scaling data.

Vertical shading is the allocation of a table or a group of tables to a separate server.

In order to apply such a kind of shading, as vertical, it is necessary to use the appropriate connection to work with each table. An example is shown in Listing 3.

Listing 3.

```
<?
$users_connection = mysql_connect('10.10.0.1', 'root', '');
$image_connection = mysql_connect('10.10.0.2', 'root', '');

$q = mysql_query('SELECT * FROM users WHERE ...', $users_connection);
$q = mysql_query('SELECT * FROM images WHERE...', $image_connection);

$q = mysql_query('SELECT * FROM albums WHERE...', $image_connection);
```

Unlike replication, we use different connections for any operations, but with certain tables.

Do not apply the technique of shading to all tables. The right approach is a step-by-step process of splitting up growing tables. You should think about horizontal shading, when the number of records in one table is a significant factor in the performance of the application. Shading and replication are often used together. An example of such an implementation is presented in Listing 4 below.

Listing 4.

```
<?
$image_connections = [
  '1' => [
            'master' => '10.10.0.10',
            'slave' => '10.10.0.11',
  ],
  '2' => [
            'master' => '10.10.0.20',
            'slave' => '10.10.0.21',
  ],
];

$user_id = $_SESSION['user_id'];

$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection = mysql_connect($image_connections[$connection_num]['slave'], 'root', '');
$q = mysql_query('SELECT * FROM image WHREE user_id = ' . intval($user_id), $connection);

$image_id = 7;
```

```
$connection_num = $user_id % 2 == 0 ? 1 : 2;
$connection = mysql_connect($image_connections[$connection_num]['master'], 'root', '');
$q = mysql_query('UPDATE images SET views = views + 1 WHREE image_id = ' . intval($image_id), $connection);
  '2' => '10.10.0.2',
```

This scheme is often used not only for scaling, but also for providing fault tolerance. So, if one of the shard servers fails, there will always be a spare one.

In this paper, we can highlight the fact that we have considered some possibilities for the competent reorganization of databases in order to improve the optimization and increase the fault tolerance.

REFERENCES

1. [Electronic resource] //habrahabr.ru/company/oleg-bunin/blog/309330/. – Дата доступа: 16.02.2018.
2. [Electronic resource] // https://en.wikipedia.org/wiki/Shard_(database_architecture). – Дата доступа: 16.02.2018.