# ObMySQL模块设计文档

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **文档版本** | **修订章节** | **修订原因** | **修订日期** | **修订人** |
| **1** | 0.1 |  | 创建 | 2012.7.5 | 方戟 |
| **2** |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

OceanBase在内部开始支持SQL语句的查询，为了降低业务部门使用Oceanbase的门槛，需要在OB内部实现一个兼容MySQL协议的模块。 使OB的使用与MySQL的SQL无异。ObMySQL获取从MySQL客户端发来的请求，解析该请求为OB支持的格式交给MergeServer来处理，并转化MergeServer传回来的数据打包成MySQL认识的数据包格式返回给MySQL客户端。从而达到使用MySQL客户端访问OB数据库的目的。

## ObMySQL模块结构



如上图ObMySQl作为客户和Oceanbase的中间代理模块，实现MySQL协议到Oceanbase内部协议的转换。

### 1.1访问方式

用户首先建立一个到ObMySQL的链接，而后所有的请求都是在这个连接上一同步的方式来访问，直到用户退出连接。

### 1.2功能划分

整个ObMySQL模块分为三个部分

1. MySQL协议解析

负责MySQL协议数据包的encode/decode

1. 连接管理

负责连接的建立，以及连接状态的管理，比如在某个连接上是否有预处理过的SQL语句

1. 数据集转换

负责将Oceanbase返回的结果数据集，转换成MySQL

## 连接初始化

这部分主要是处理MySQL客户端的连接过程。描述需要Oceanbase提供哪些状态来保证客户端能够成功的与服务端建立连接。

在MySQL服务器中，对于每一个客户端会新建一个线程来服务(sql/sql\_connect.cc中的handle\_one\_connection)

在Ob的实现中，先忽略掉用户密码认证，相信所有发起连接的客户

### MySQL认证过程

这个过程分为三个部分首先客户端建立一个到服务端的连接，并且从服务端获取一个handshake包，然后根据这个包进行处理发送给服务端一个数据包，最后服务端发回一个成功或者失败的包

1. 握手包 以及涉及的MySQL内部状态 OB需要提供的状态

长度 说明

1 协议的版本号 (0x0A)

n 以0结尾的字符串。描述服务器版本

4 thread id

8 scramble的前8个字节

1 0x00。也就是说让scramble看起来是一个以0结尾的字符串

2 server capabilities的低两个字节。

1 server character set

2 server status

2 server capabilities的高两个字节。

1 scramble的总长度

10 保留。必须以0填充。

n (至少12） scramble的剩余部分。（不包含'\0'）

1 0x00。也就是说让scramble看起来是一个以0结尾的字符串

在这个包里thread id scramble字段在MySQL的连接中比较重要，但是对于到Oceanbase的连接我们可以不做任何认证，直接让他连接上，所以对这些字段信息可以忽略掉，而对于

server capacity字段需要特别的关注。

在MySQL的文档中定义了这个4字节的字段的每一位代表的意思

名字 从右往左数第几位 说明

CLIENT\_LONG\_PASSWORD 1 new more secure passwords

CLIENT\_FOUND\_ROWS 2 Found instead of affected rows

CLIENT\_LONG\_FLAG 3 Get all column flags

CLIENT\_CONNECT\_WITH\_DB 4 One can specify db on connect

CLIENT\_NO\_SCHEMA 5 Don't allow database.table.column

CLIENT\_COMPRESS 6 Can use compression protocol

CLIENT\_ODBC 7 Odbc client

CLIENT\_LOCAL\_FILES 8 Can use LOAD DATA LOCAL

CLIENT\_IGNORE\_SPACE 9 Ignore spaces before '('

CLIENT\_PROTOCOL\_41 10 New 4.1 protocol

CLIENT\_INTERACTIVE 11 This is an interactive client

CLIENT\_SSL 12 Switch to SSL after handshake

CLIENT\_IGNORE\_SIGPIPE 13 IGNORE sigpipes

CLIENT\_TRANSACTIONS 14 Client knows about transactions

CLIENT\_RESERVED 15 Old flag for 4.1 protocol

CLIENT\_SECURE\_CONNECTION 16 New 4.1 authentication

CLIENT\_MULTI\_STATEMENTS 17 Enable/disable multi-stmt support

CLIENT\_MULTI\_RESULTS 18 Enable/disable multi-results

CLIENT\_PS\_MULTI\_RESULTS 19 Multi-results in PS-protocol

CLIENT\_PLUGIN\_AUTH 20 Client supports plugin authentication

CLIENT\_SSL\_VERIFY\_SERVER\_CERT 31

CLIENT\_REMEMBER\_OPTIONS 32

最新的客户端使用的4.1版本的协议，在Ob中capacity这个字段取值2^15+2^9使最新的客户端能够连上

1. 用户发送认证包

客户端根据服务器发给的scramble对原始密码进行散列，然后和其它参数一起发给服务器

1. 通过认证 ok/error包

在Ob的实现中 在收到客户端发来的认证包后，不做处理直接发送OK包给客户端完成连接的建立

伪代码

给libeasy的handler结构实现下面这个回调函数

int (\*on\_connect) (easy\_connection\_t \*c)

hander->on\_connect = ObMySQLConn.on\_connect;

由于MySQL的认证包和命令包没有pcode来区分 所以在建立连接的时候把认证这部分的包通过原始socket全部处理完，以后在这个连接上的包就去不是命令包，在主循环中根据Command code来调用不同的处理函数

**int** ObMySQLConn::on\_connect(easy\_connection\_t\* c)

{

// get server status && capacilities

**ObServerCap cap = ObSql.getCap();**

**// send handshake packet**

**send\_handShake\_packet（cap）；**

**// get auth packet**

**get\_auth\_packet();**

**// send ok packet**

**dend\_ok\_packet();**

**// start io event on connection**

**easy\_connection\_evio\_start(c);**

}

## 连接管理

一个ObMySQL需要服务多个Client，每个Client也可以向ObMySQL发起多个连接，这里描述一下ObMySQL处理多个连接的方式。

使用libeasy来实现可以有下面两种模式

1. libeasy里面的IO线程直接收包并且处理

优点：没有线程间的切换

缺点：IO线程的个数在启动的时候就确定。如果有多于IO线程个的连接连过来，有可能所有的线程正在处理请求，导致某些连接上的请求得不到及时的响应。

1. libeasy里面的IO线程负责收包，然后根据连接去唤醒一个特定的线程，这个线程负责处理这个请求

优点：可以通过一个线程池来保证一个线程处理一个连接上的数据

确定：每次请求都需要从IO线程切换到工作线程

为了保证可以及时处理多个连接上的请求，采用第二种方式来处理连接上的请求。

这与MySQL的策略是一样，使用一个线程来服务一个连接，ObMySQL中实现中一个线程池，当有连接进来的时候，先从线程池分配，如果线程池没有可用的线程则创建一个新的线程来服务当前的连接，连接断开后相应的工作线程还给线程池。

这部分内容需要实现一个线程池，该线程池能够的线程数需要能够动态的增加和减少，还需要实现一个从connection到线程池中线程的映射， 并且这个映射关系在连接没有断开之前需要一直保持。

## MySQL命令

截至到Mysql 5.5.9，mysql server一共支持30种command具体如下

COM\_SLEEP

COM\_QUIT

COM\_INIT\_DB

COM\_QUERY

COM\_FIELD\_LIST

COM\_CREATE\_DB

COM\_DROP\_DB

COM\_REFRESH

COM\_SHUTDOWN

COM\_STATISTICS

COM\_PROCESS\_INFO

COM\_CONNECT

COM\_PROCESS\_KILL

COM\_DEBUG

COM\_PING

COM\_TIME

COM\_DELAYED\_INSERT

COM\_CHANGE\_USER

COM\_BINLOG\_DUMP

COM\_TABLE\_DUMP

COM\_CONNECT\_OUT

COM\_REGISTER\_SLAVE

COM\_STMT\_PREPARE

COM\_STMT\_EXECUTE

COM\_STMT\_SEND\_LONG\_DATA

COM\_STMT\_CLOSE

COM\_STMT\_RESET

COM\_SET\_OPTION

COM\_STMT\_FETCH

COM\_DAEMON

这其中大多数命令使用的都比较少，对于目前Oceanbase，可以支持最普通的SQL语句查询以及变量绑定的功能。对于其它不支持的命令，可以提醒用户不要使用，并且在ObMySQL中做些简单的错误处理。

在OB中需要支持的命令可以简化如下

COM\_QUERY 查询请求

COM\_STMT\_PREPARE 绑定变量

COM\_STMT\_EXECUTE 执行绑定变量

COM\_STMT\_CLOSE 结束绑定变量

COM\_STMT\_FETCH 获取执行后的结果

COM\_QUIT 客户端退出

变量绑定功能的基本流程如下

1. mysql\_stmt\_init 获取绑定变量的句柄
2. mysql\_stmt\_prepare 向服务器发送带参数的SQL语句
3. mysql\_stmt\_param\_bind 绑定变量
4. mysql\_stmt\_execute 发送变量到服务器执行
5. mysql\_stmt\_store\_result 对于有结果集的获取结果集(可选)
6. mysql\_stmt\_fetch 按行获取结果集中的数据

绑定变量和普通的SQL语句执行在结果集的处理上是有区别的，具体区别在MySQL协议章节

## MySQL协议

关于MySQL的包格式，可以参考

<http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol>

这个文档详细的说明了MySQL 客户端服务器之间各种数据包的格式及其含义

要实现这个协议在ObMySQL中需要有一个encode/decode基础类型的工具

对于每一个数据包有一个Header

Bytes Name

----- ----

3 Packet Length

1 Packet Number

3个字节表示数据包的长度，最大的数据2^24-1约为16MB, 1个字节表示包序列号，序列号在每次执行一个新语句时被重置为0

### 普通Query

最常用的交互方式，使用的命令是COM\_QUERY，所有INSERT，UPDATE，SELECT，DELETE都是通过这个命令来发起请求的。

客户端通过发送COMMAND\_PACKET把请求传给服务端

Bytes Name

----- ----

1 command 这里是COM\_QUERY

n arg 可以是任意的SQL语句 发送的时候会去掉最后的分号

对于没有结果集的请求，服务器处理完后返回OK或者ERROR

VERSION 4.1

Bytes Name

----- ----

1 (Length Coded Binary) field\_count, always = 0 //0表示是OK

1-9 (Length Coded Binary) affected\_rows

1-9 (Length Coded Binary) insert\_id //

2 server\_status //表示服务器的状态 是不是在一个事务中

2 warning\_count

n (until end of packet) message

VERSION 4.1

Bytes Name

----- ----

1 field\_count, always = 0xff //全1表示是ERROR

2 errno

1 (sqlstate marker), always '#'

5 sqlstate (5 characters) //与错误码响应的sqlstate

n message

如果一个请求是有结果集的，那么返回多个数据包

(Result Set Header Packet) the number of columns

(Field Packets) column descriptors

(EOF Packet) marker: end of Field Packets

(Row Data Packets) row contents //这个可以有多个

(EOF Packet) marker: end of Data Packets

1. Result Set Header Packet

返回结果集的每一行有多少个列

1. Field Packets

描述列的信息 类型、长度、表名、库名等等。有多少列就返回多少个这样的包，一个包描述一个列

1. EOF Packet

标识列描述返回完成

1. Row Data Packet

返回行数据每个字段被序列化到一个连续的buffer中。一个包只含有一行数据

1. EOF Packet

标识行数据返回完成

### 绑定变量

上一节中叙述过MySQL绑定变量的基本流程

1. mysql\_stmt\_init

获取绑定变量的句柄

1. mysql\_stmt\_prepare

向服务器发送带”?”的SQL语句

服务器处理成功返回

Bytes Name

----- ----

1 0

4 statement\_handler\_id //这个是statment唯一标识，以后客户端发送execute请求都会带上这个id

2 number of columns in result set

2 number of parameters in query

1 filler (always 0)

2 warning count

参数个数n 那么后续会发送n给参数描述的数据包如下

Bytes Name

----- ----

2 type

2 flags

1 decimals

4 length

1. mysql\_stmt\_param\_bind

绑定变量，这部分工作在客户端完成 把变量拷贝到statement内部的存储区，以后多次execute不需要每次都bind param，只有在参数改变的情况下需要重新绑定

1. mysql\_stmt\_execute

发送变量到服务器执行

客户端向服务端发送如下数据包

Bytes Name

----- ----

1 code

4 statement\_id //之前服务端返回的id

1 flags

4 iteration\_count

if param\_count > 0:

(param\_count+7)/8 null\_bit\_map

1 new\_parameter\_bound\_flag

if new\_params\_bound == 1:

n\*2 type of parameters //一个参数用2个字节来表示类型， 16bits中的最高位表示unsinged/singed 低15位表示具体的类型

n values for the parameters //参数的值按顺寻序列化到这里

如果execute执行成功，那么会像简单QUERY返回结果集那样将数据集返回给客户端

1. mysql\_stmt\_store\_result

对于有结果集的获取结果集(可选) 把execute返回的结果集全部缓存在stmt中

1. mysql\_stmt\_fetch

按行获取结果集中的数据

## SQL执行结果处理

在ObMySQL中，只需要将客户端传来的SQL语句传给ObSQL去执行，如下

int direct\_execute(const ObString &stmt, ObResultSet &result);

返回参数result是Oceanbase SQL执行引擎返回给ObMySQL的结果数据集。

根据这个函数的执行情况按如下的方式来处理结果集



根据上图，在这部分有连个映射关系需要处理

1. 从OB的error code转换成MySQL的errno 以及对应的sqlstate

在MySQL中sqlstate在 /usr/include/mysql/sql\_state.h

errno到sqlstate的转换函数如下 sql/sql\_state.c

const char \*mysql\_errno\_to\_sqlstate(uint mysql\_errno)

{

uint first=0, end= array\_elements(sqlstate\_map)-1;

struct st\_map\_errno\_to\_sqlstate \*map;

/\* Do binary search in the sorted array \*/

while (first != end)

{

uint mid= (first+end)/2;

map= sqlstate\_map+mid;

if (map->mysql\_errno < mysql\_errno)

first= mid+1;

else

end= mid;

}

map= sqlstate\_map+first;

if (map->mysql\_errno == mysql\_errno)

return map->odbc\_state;

return "HY000"; /\* General error \*/

}

ObMySQL中也定义一个类似struct st\_map\_errno\_to\_sqlstate的数组来完成OB error code到sqlstate的转换

1. 从OB的data type 到MySQL的data type

|  |  |
| --- | --- |
| Oceanbase Data Type | MySQL Data Type |
| ObMinType |  |
| ObNullType | MYSQL\_TYPE\_NULL |
| ObIntType | MYSQL\_TYPE\_LONGLONG 64bits |
| ObFloatType | MYSQL\_TYPE\_FLOAT 32bits |
| ObDoubleType | MYSQL\_TYPE\_DOUBLE 64bits |
| ObDateTimeType | MYSQL\_TYPE\_TIMESTAMP 精确到微妙 |
| ObPreciseDateTimeType | MYSQL\_TYPE\_TIMESTAMP |
| ObVarcharType | MYSQL\_TYPE\_VAR\_STRING //mysql 有几种string不知道哪种比较对 |
| ObSeqType |  |
| ObCreateTimeType | MYSQL\_TYPE\_TIMESTAMP |
| ObModifyTimeType | MYSQL\_TYPE\_TIMESTAMP |
| ObExtendType |  |
| ObBoolType |  |
| ObDecimalType |  |
| ObMaxType |  |

附MySQL支持的数据类型

MYSQL\_TYPE\_DECIMAL

MYSQL\_TYPE\_TINY

MYSQL\_TYPE\_SHORT

MYSQL\_TYPE\_LONG

MYSQL\_TYPE\_FLOAT

MYSQL\_TYPE\_DOUBLE

MYSQL\_TYPE\_NULL

MYSQL\_TYPE\_TIMESTAMP

MYSQL\_TYPE\_LONGLONG

MYSQL\_TYPE\_INT24

MYSQL\_TYPE\_DATE

MYSQL\_TYPE\_TIME

MYSQL\_TYPE\_DATETIME

MYSQL\_TYPE\_YEAR

MYSQL\_TYPE\_NEWDATE

MYSQL\_TYPE\_VARCHAR (new in MySQL 5.0)

MYSQL\_TYPE\_BIT (new in MySQL 5.0)

MYSQL\_TYPE\_NEWDECIMAL (new in MYSQL 5.0)

MYSQL\_TYPE\_ENUM

MYSQL\_TYPE\_SET

MYSQL\_TYPE\_TINY\_BLOB

MYSQL\_TYPE\_MEDIUM\_BLOB

MYSQL\_TYPE\_LONG\_BLOB

MYSQL\_TYPE\_BLOB

MYSQL\_TYPE\_VAR\_STRING

MYSQL\_TYPE\_STRING

MYSQL\_TYPE\_GEOMETRY