```
0:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\AbstractConnect
ion.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.nio.channels.AsynchronousChannel;
import java.nio.channels.NetworkChannel;
import java.nio.channels.SocketChannel;
import java.util.List;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.atomic.AtomicBoolean;

import com.google.common.base.Strings;

import io.mycat.backend.mysql.CharsetUtil;
import io.mycat.util.CompressUtil;
import io.mycat.util.TimeUtil;

import org.slf4j.Logger; import org.slf4j.LoggerFactory;

/**
```

```
 * @author mycat
 */
public abstract class AbstractConnection implements NIOConnection {

     protected        static        final        Logger        LOGGER        =
LoggerFactory.getLogger(AbstractConnection.class);

     protected String host;
     protected int localPort;
     protected int port;
     protected long id;
     protected volatile String charset;
     protected volatile int charsetIndex;

     protected final NetworkChannel channel;
     protected NIOProcessor processor;
     protected NIOHandler handler;

     protected int packetHeaderSize;
     protected int maxPacketSize;
     protected volatile ByteBuffer readBuffer;
     protected volatile ByteBuffer writeBuffer;

     protected   final   ConcurrentLinkedQueue<ByteBuffer>   writeQueue   =   new
ConcurrentLinkedQueue<ByteBuffer>();

     protected volatile int readBufferOffset;
     protected long lastLargeMessageTime;
     protected final AtomicBoolean isClosed;
     protected boolean isSocketClosed;
     protected long startupTime;
     protected long lastReadTime;
     protected long lastWriteTime;
     protected long netInBytes;
     protected long netOutBytes;
     protected int writeAttempts;

     protected volatile boolean isSupportCompress = false;
    protected final ConcurrentLinkedQueue<byte[]> decompressUnfinishedDataQueue =
new ConcurrentLinkedQueue<byte[]>();
    protected final ConcurrentLinkedQueue<byte[]> compressUnfinishedDataQueue = new
ConcurrentLinkedQueue<byte[]>();

     private long idleTimeout;

     private final SocketWR socketWR;
```

```
public AbstractConnection(NetworkChannel channel) {
    this.channel = channel;
    boolean isAIO = (channel instanceof AsynchronousChannel);
    if (isAIO) {
        socketWR = new AIOSocketWR(this);
    } else {
        socketWR = new NIOSocketWR(this);
    }
    this.isClosed = new AtomicBoolean(false);
    this.startupTime = TimeUtil.currentTimeMillis();
    this.lastReadTime = startupTime;
    this.lastWriteTime = startupTime;
}

public String getCharset() {
    return charset;
}

public boolean setCharset(String charset) {

    // 修复 PHP 字符集设置错误，如： set names 'utf8'
    if (charset != null) {
        charset = charset.replace("'", "");
    }

    int ci = CharsetUtil.getIndex(charset);
    if (ci > 0) {
        this.charset = charset.equalsIgnoreCase("utf8mb4") ? "utf8" : charset;
        this.charsetIndex = ci;
        return true;
    } else {
        return false;
    }
}

public boolean isSupportCompress() {
    return isSupportCompress;
}

public void setSupportCompress(boolean isSupportCompress) {
    this.isSupportCompress = isSupportCompress;
}

public int getCharsetIndex() {
    return charsetIndex;
}
```

```
    public long getIdleTimeout() {
        return idleTimeout;
    }

    public SocketWR getSocketWR() {
        return socketWR;
    }

    public void setIdleTimeout(long idleTimeout) {
        this.idleTimeout = idleTimeout;
    }

    public int getLocalPort() {
        return localPort;
    }

    public String getHost() {
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }

    public int getPort() {
        return port;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public void setLocalPort(int localPort) {
        this.localPort = localPort;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public boolean isIdleTimeout() {
        return TimeUtil.currentTimeMillis() > Math.max(lastWriteTime, lastReadTime)
+ idleTimeout;
    }

    public NetworkChannel getChannel() {
        return channel;
    }

    public int getPacketHeaderSize() {
        return packetHeaderSize;
    }

    public void setPacketHeaderSize(int packetHeaderSize) {
        this.packetHeaderSize = packetHeaderSize;
    }

    public int getMaxPacketSize() {
        return maxPacketSize;
    }

    public void setMaxPacketSize(int maxPacketSize) {
        this.maxPacketSize = maxPacketSize;
    }

    public long getStartupTime() {
        return startupTime;
    }

    public long getLastReadTime() {
        return lastReadTime;
    }

    public void setProcessor(NIOProcessor processor) {
        this.processor = processor;
        int size = processor.getBufferPool().getChunkSize();
        this.readBuffer = processor.getBufferPool().allocate(size);
    }

    public long getLastWriteTime() {
        return lastWriteTime;
    }

    public void setLastWriteTime(long lasttime){
        this.lastWriteTime = lasttime;
    }

    public long getNetInBytes() {
        return netInBytes;
    }
```

```
    public long getNetOutBytes() {
        return netOutBytes;
    }

    public int getWriteAttempts() {
        return writeAttempts;
    }

    public NIOProcessor getProcessor() {
        return processor;
    }

    public ByteBuffer getReadBuffer() {
        return readBuffer;
    }

    public ByteBuffer allocate() {
        int size = this.processor.getBufferPool().getChunkSize();
        ByteBuffer buffer = this.processor.getBufferPool().allocate(size);
        return buffer;
    }

    public final void recycle(ByteBuffer buffer) {
        this.processor.getBufferPool().recycle(buffer);
    }

    public void setHandler(NIOHandler handler) {
        this.handler = handler;
    }

    @Override
    public void handle(byte[] data) {
        if (isSupportCompress()) {
            List<byte[]>   packs   =   CompressUtil.decompressMysqlPacket(data,
decompressUnfinishedDataQueue);
            for (byte[] pack : packs) {
                if (pack.length != 0) {
                    handler.handle(pack);
                }
            }
        } else {
            handler.handle(data);
        }
    }

    @Override
    public void register() throws IOException {

    }

    public void asynRead() throws IOException {
        this.socketWR.asynRead();
    }

    public void doNextWriteCheck() throws IOException {
        this.socketWR.doNextWriteCheck();
    }

    /**
     * 读取可能的 Socket 字节流
     */
    public void onReadData(int got) throws IOException {

        if (isClosed.get()) {
            return;
        }

        lastReadTime = TimeUtil.currentTimeMillis();
        if (got < 0) {
            this.close("stream closed");
          return;
        } else if (got == 0
                && !this.channel.isOpen()) {
                this.close("socket closed");
                return;
        }
        netInBytes += got;
        processor.addNetInBytes(got);

        // 循环处理字节信息
        int offset = readBufferOffset, length = 0, position = readBuffer.position();
        for (;;) {
            length = getPacketLength(readBuffer, offset);
            if (length == -1) {
                if (offset != 0) {
                    this.readBuffer = compactReadBuffer(readBuffer, offset);
                } else if (readBuffer != null && !readBuffer.hasRemaining()) {
                    throw new RuntimeException( "invalid readbuffer capacity ,too
little buffer size "
                            + readBuffer.capacity());
                }
                break;
            }
```

```
                if (position >= offset + length && readBuffer != null) {

                    // handle this package
                    readBuffer.position(offset);
                    byte[] data = new byte[length];
                    readBuffer.get(data, 0, length);
                    handle(data);

                    // maybe handle stmt_close
                    if(isClosed()) {
                        return ;
                    }

                    // offset to next position
                    offset += length;

                    // reached end
                    if (position == offset) {
                        // if cur buffer is temper none direct byte buffer and not
                        // received large message in recent 30 seconds
                        // then change to direct buffer for performance
                        if (readBuffer != null && !readBuffer.isDirect()
                                && lastLargeMessageTime < lastReadTime - 30 * 1000L)
{  // used temp heap
                            if (LOGGER.isDebugEnabled()) {
                                LOGGER.debug("change to direct con read buffer ,cur
temp buf size :" + readBuffer.capacity());
                            }
                            recycle(readBuffer);
                            readBuffer                                              =
processor.getBufferPool().allocate(processor.getBufferPool().getConReadBuferChunk()
);
                        } else {
                            if (readBuffer != null) {
                                readBuffer.clear();
                            }
                        }
                        // no more data ,break
                        readBufferOffset = 0;
                        break;
                    } else {
                        // try next package parse
                        readBufferOffset = offset;
                        if(readBuffer != null) {
                            readBuffer.position(position);
                        }
                        continue;
                    }



                } else {
                    // not read whole message package ,so check if buffer enough and
                    // compact readbuffer
                    if (!readBuffer.hasRemaining()) {
                        readBuffer = ensureFreeSpaceOfReadBuffer(readBuffer, offset,
length);
                    }
                    break;
                }
            }
        }

        private boolean isConReadBuffer(ByteBuffer buffer) {
            return                         buffer.capacity()                        ==
processor.getBufferPool().getConReadBuferChunk() && buffer.isDirect();
        }

        private ByteBuffer ensureFreeSpaceOfReadBuffer(ByteBuffer buffer,
                int offset, final int pkgLength) {
            // need a large buffer to hold the package
            if (pkgLength > maxPacketSize) {
                throw new IllegalArgumentException("Packet size over the limit.");
            } else if (buffer.capacity() < pkgLength) {

                ByteBuffer newBuffer = processor.getBufferPool().allocate(pkgLength);
                lastLargeMessageTime = TimeUtil.currentTimeMillis();
                buffer.position(offset);
                newBuffer.put(buffer);
                readBuffer = newBuffer;

                recycle(buffer);
                readBufferOffset = 0;
                return newBuffer;

            } else {
                if (offset != 0) {
                    // compact bytebuffer only
                    return compactReadBuffer(buffer, offset);
                } else {
                    throw new RuntimeException(" not enough space");
                }
            }
```

```
    }

    private ByteBuffer compactReadBuffer(ByteBuffer buffer, int offset) {
        if(buffer == null) {
            return null;
        }
        buffer.limit(buffer.position());
        buffer.position(offset);
        buffer = buffer.compact();
        readBufferOffset = 0;
        return buffer;
    }

    public void write(byte[] data) {
        ByteBuffer buffer = allocate();
        buffer = writeToBuffer(data, buffer);
        write(buffer);

    }

    private final void writeNotSend(ByteBuffer buffer) {
        if (isSupportCompress()) {
            ByteBuffer newBuffer = CompressUtil.compressMysqlPacket(buffer, this,
compressUnfinishedDataQueue);
            writeQueue.offer(newBuffer);

        } else {
            writeQueue.offer(buffer);
        }
    }


    @Override
    public final void write(ByteBuffer buffer) {

        if (isSupportCompress()) {
            ByteBuffer newBuffer = CompressUtil.compressMysqlPacket(buffer, this,
compressUnfinishedDataQueue);
            writeQueue.offer(newBuffer);
        } else {
            writeQueue.offer(buffer);
        }

        // if ansyn write finishe event got lock before me ,then writing
        // flag is set false but not start a write request
        // so we check again
        try {
            this.socketWR.doNextWriteCheck();
        } catch (Exception e) {
            LOGGER.warn("write err:", e);
            this.close("write err:" + e);
        }
    }


    public ByteBuffer checkWriteBuffer(ByteBuffer buffer, int capacity, boolean
writeSocketIfFull) {
        if (capacity > buffer.remaining()) {
            if (writeSocketIfFull) {
                writeNotSend(buffer);
                return processor.getBufferPool().allocate(capacity);
            } else {// Relocate a larger buffer
                buffer.flip();
                ByteBuffer newBuf = processor.getBufferPool().allocate(capacity +
buffer.limit() + 1);
                newBuf.put(buffer);
                this.recycle(buffer);
                return newBuf;
            }
        } else {
            return buffer;
        }
    }

    public ByteBuffer writeToBuffer(byte[] src, ByteBuffer buffer) {
        int offset = 0;
        int length = src.length;
        int remaining = buffer.remaining();
        while (length > 0) {
            if (remaining >= length) {
                buffer.put(src, offset, length);
                break;
            } else {
                buffer.put(src, offset, remaining);
                writeNotSend(buffer);
                buffer = allocate();
                offset += remaining;
                length -= remaining;
                remaining = buffer.remaining();
                continue;
            }
        }
        return buffer;
    }
```

```
@Override
public void close(String reason) {
     if (!isClosed.get()) {
          closeSocket();
          isClosed.set(true);
          if (processor != null) {
               processor.removeConnection(this);
          }
          this.cleanup();
          isSupportCompress = false;

          // ignore null information
          if (Strings.isNullOrEmpty(reason)) {
               return;
          }
          LOGGER.info("close connection,reason:" + reason + " ," + this);
          if (reason.contains("connection,reason:java.net.ConnectException")) {
               throw new RuntimeException(" errr");
          }
     } else {
         // make sure cleanup again
         // Fix issue#1616
         this.cleanup();
     }
}

public boolean isClosed() {
     return isClosed.get();
}

public void idleCheck() {
     if (isIdleTimeout()) {
          LOGGER.info(toString() + " idle timeout");
          close(" idle ");
     }
}

/**
 * 清理资源
 */
protected void cleanup() {

     // 清理资源占用
     if (readBuffer != null) {
          this.recycle(readBuffer);
          this.readBuffer = null;
```

```
            this.readBufferOffset = 0;
        }

        if (writeBuffer != null) {
            recycle(writeBuffer);
            this.writeBuffer = null;
        }

        if (!decompressUnfinishedDataQueue.isEmpty()) {
            decompressUnfinishedDataQueue.clear();
        }

        if (!compressUnfinishedDataQueue.isEmpty()) {
            compressUnfinishedDataQueue.clear();
        }

        ByteBuffer buffer = null;
        while ((buffer = writeQueue.poll()) != null) {
            recycle(buffer);
        }
    }

    protected int getPacketLength(ByteBuffer buffer, int offset) {
        int headerSize = getPacketHeaderSize();
        if ( isSupportCompress() ) {
            headerSize = 7;
        }

        if (buffer.position() < offset + headerSize) {
            return -1;
        } else {
            int length = buffer.get(offset) & 0xff;
            length |= (buffer.get(++offset) & 0xff) << 8;
            length |= (buffer.get(++offset) & 0xff) << 16;
            return length + headerSize;
        }
    }

    public ConcurrentLinkedQueue<ByteBuffer> getWriteQueue() {
        return writeQueue;
    }

    private void closeSocket() {
        if (channel != null) {
            if (channel instanceof SocketChannel) {
                Socket socket = ((SocketChannel) channel).socket();
                if (socket != null) {
                        try {
                            socket.close();
                        } catch (IOException e) {
                          LOGGER.error("closeChannelError", e);
                        }
                    }
                }

                boolean isSocketClosed = true;
                try {
                    channel.close();
                } catch (Exception e) {
                    LOGGER.error("AbstractConnectionCloseError", e);
                }

                boolean closed = isSocketClosed && (!channel.isOpen());
                if (closed == false) {
                    LOGGER.warn("close socket of connnection failed " + this);
                }
            }
        }
        public void onConnectfinish() {
            LOGGER.debug("连接后台真正完成");
        }
}
```

1:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\AIOAcceptor.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it’s project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.StandardSocketOptions;
import java.nio.channels.AsynchronousChannelGroup;
import java.nio.channels.AsynchronousServerSocketChannel;
import java.nio.channels.AsynchronousSocketChannel;
import java.nio.channels.CompletionHandler;
import java.nio.channels.NetworkChannel;
import java.util.concurrent.atomic.AtomicLong;

import org.slf4j.Logger; import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import io.mycat.net.factory.FrontendConnectionFactory;

/**
 * @author mycat
 */
public final class AIOAcceptor implements SocketAcceptor,
        CompletionHandler<AsynchronousSocketChannel, Long> {
    private static final Logger LOGGER = LoggerFactory.getLogger(AIOAcceptor.class);
    private static final AcceptIdGenerator ID_GENERATOR = new AcceptIdGenerator();

    private final int port;
    private final AsynchronousServerSocketChannel serverChannel;
    private final FrontendConnectionFactory factory;

    private long acceptCount;
    private final String name;

    public AIOAcceptor(String name, String ip, int port,
            FrontendConnectionFactory factory, AsynchronousChannelGroup group)
            throws IOException {
        this.name = name;
        this.port = port;
        this.factory = factory;
        serverChannel = AsynchronousServerSocketChannel.open(group);
        /** 设置 TCP 属性 */
        serverChannel.setOption(StandardSocketOptions.SO_REUSEADDR, true);
        serverChannel.setOption(StandardSocketOptions.SO_RCVBUF, 1024 * 16 * 2);
        // backlog=100
        serverChannel.bind(new InetSocketAddress(ip, port), 100);
    }

    public String getName() {
        return name;
    }

    public void start() {
        this.pendingAccept();
    }

    public int getPort() {
        return port;
    }

    public long getAcceptCount() {
        return acceptCount;
    }

    private void accept(NetworkChannel channel, Long id) {
        try {
            FrontendConnection c = factory.make(channel);
            c.setAccepted(true);
            c.setId(id);
            NIOProcessor processor = MycatServer.getInstance().nextProcessor();
            c.setProcessor(processor);
            c.register();
        } catch (Exception e) {
           LOGGER.error("AioAcceptorError", e);
            closeChannel(channel);
        }
    }

    private void pendingAccept() {
        if (serverChannel.isOpen()) {
            serverChannel.accept(ID_GENERATOR.getId(), this);
        } else {
            throw new IllegalStateException(
                    "MyCAT Server Channel has been closed");
        }

    }

    @Override
    public void completed(AsynchronousSocketChannel result, Long id) {
        accept(result, id);
        // next pending waiting
        pendingAccept();

    }

    @Override
    public void failed(Throwable exc, Long id) {
        LOGGER.info("acception connect failed:" + exc);
        // next pending waiting
        pendingAccept();

    }

    private static void closeChannel(NetworkChannel channel) {
        if (channel == null) {
            return;
        }
        try {
            channel.close();
        } catch (IOException e) {
           LOGGER.error("AioAcceptorError", e);
        }
    }

    /**
     * 前端连接 ID 生成器
     *
     * @author mycat
     */
    private static class AcceptIdGenerator {

        private static final long MAX_VALUE = 0xffffffffL;

        private AtomicLong acceptId = new AtomicLong();
        private final Object lock = new Object();

        private long getId() {
            long newValue = acceptId.getAndIncrement();
            if (newValue >= MAX_VALUE) {
                synchronized (lock) {
                    newValue = acceptId.getAndIncrement();
                    if (newValue >= MAX_VALUE) {
                        acceptId.set(0);
                    }
                }
                return acceptId.getAndDecrement();
            } else {
                return newValue;
```

```
               }
          }
     }
}
```

2:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\AIOConnector.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.nio.channels.CompletionHandler;

import org.slf4j.Logger; import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import java.util.concurrent.atomic.AtomicLong;

/**
 * @author mycat
 */
public final class AIOConnector implements SocketConnector,
          CompletionHandler<Void, BackendAIOConnection> {
     private          static          final          Logger          LOGGER          =
LoggerFactory.getLogger(AIOConnector.class);
     private static final ConnectIdGenerator ID_GENERATOR = new ConnectIdGenerator();
```

```
    public AIOConnector() {

    }

    @Override
    public void completed(Void result, BackendAIOConnection attachment) {
        finishConnect(attachment);
    }

    @Override
    public void failed(Throwable exc, BackendAIOConnection conn) {
        conn.onConnectFailed(exc);
    }

    private void finishConnect(BackendAIOConnection c) {
        try {
            if (c.finishConnect()) {
                c.setId(ID_GENERATOR.getId());
                NIOProcessor processor = MycatServer.getInstance()
                        .nextProcessor();
                c.setProcessor(processor);
                c.register();
            }
        } catch (Exception e) {
            c.onConnectFailed(e);
            LOGGER.info("connect err " , e);
            c.close(e.toString());
        }
    }

    /**
     * 后端连接 ID 生成器
     *
     * @author mycat
     */
    private static class ConnectIdGenerator {

        private static final long MAX_VALUE = Long.MAX_VALUE;

        private AtomicLong connectId = new AtomicLong(0);

        private long getId() {
            return connectId.incrementAndGet();
        }
    }
}
```

```
3:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\AIOSocketWR.jav
a
package io.mycat.net;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.AsynchronousSocketChannel;
import java.nio.channels.CompletionHandler;

import java.util.concurrent.atomic.AtomicBoolean;

import io.mycat.util.TimeUtil;

public class AIOSocketWR extends SocketWR
{
    private static final AIOReadHandler aioReadHandler = new AIOReadHandler();
    private static final AIOWriteHandler aioWriteHandler = new AIOWriteHandler();
    private final AsynchronousSocketChannel channel;
    protected final AbstractConnection con;
    protected final AtomicBoolean writing = new AtomicBoolean(false);


    public AIOSocketWR(AbstractConnection conn)
    {
        channel = (AsynchronousSocketChannel) conn.getChannel();
        this.con = conn;
    }

    @Override
    public void asynRead()
    {
        ByteBuffer theBuffer = con.readBuffer;
        if (theBuffer == null)
        {
            theBuffer                                                    =
con.processor.getBufferPool().allocate(con.processor.getBufferPool().getChunkSize()
);
            con.readBuffer = theBuffer;
            channel.read(theBuffer, this, aioReadHandler);

        } else if (theBuffer.hasRemaining())
        {
            channel.read(theBuffer, this, aioReadHandler);
        } else
        {
            throw new java.lang.IllegalArgumentException("full buffer to read ");
        }

    }

    private void asynWrite(final ByteBuffer buffer)
    {

            buffer.flip();
            this.channel.write(buffer, this, aioWriteHandler);


    }

//    public  int flushChannel(final AsynchronousSocketChannel channel,
//                             final ByteBuffer bb, final long writeTimeout)
//    {
//
//        if (!bb.hasRemaining())
//        {
//            return 0;
//        }
//        int nWrite = bb.limit();
//        try
//        {
//            while (bb.hasRemaining())
//            {
//                channel.write(bb).get(writeTimeout, TimeUnit.SECONDS);
//            }
//        } catch (Exception ie)
//        {
//            con.close("write failed " + ie);
//
//        }
//        return nWrite;
//    }


    /**
     * return true ,means no more data
     *
     * @return
     */
    private boolean write0()
    {
        if (!writing.compareAndSet(false, true))
        {
            return false;
        }
        ByteBuffer theBuffer = con.writeBuffer;
        if (theBuffer == null || !theBuffer.hasRemaining())
        {// writeFinished,但要区分 bufer 是否 NULL，不 NULL，要回收
            if (theBuffer != null)
            {
                con.recycle(theBuffer);
                con.writeBuffer = null;

            }
            // poll again
            ByteBuffer buffer = con.writeQueue.poll();
            // more data
            if (buffer != null)
            {
                if (buffer.limit() == 0)
                {
                    con.recycle(buffer);
                    con.writeBuffer = null;
                    con.close("quit cmd");
                    writing.set(false);
                    return true;
                } else
                {
                    con.writeBuffer = buffer;
                    asynWrite(buffer);
                    return false;
                }
            } else
            {
                // no buffer
               writing.set(false);
                return true;
            }
        } else
        {
              theBuffer.compact();
            asynWrite(theBuffer);
            return false;
        }

    }

    protected void onWriteFinished(int result)
    {

        con.netOutBytes += result;
```

```
        con.processor.addNetOutBytes(result);
        con.lastWriteTime = TimeUtil.currentTimeMillis();
        boolean noMoreData = this.write0();
        if (noMoreData)
        {
            this.doNextWriteCheck();
        }

    }

    public void doNextWriteCheck()
    {

        boolean noMoreData = false;
        noMoreData = this.write0();
        if (noMoreData
                && !con.writeQueue.isEmpty())
        {
                this.write0();
        }


    }
}

class AIOWriteHandler implements CompletionHandler<Integer, AIOSocketWR> {

    @Override
    public void completed(final Integer result, final AIOSocketWR wr) {
        try {

            wr.writing.set(false);

            if (result >= 0) {
                wr.onWriteFinished(result);
            } else {
                wr.con.close("write erro " + result);
            }
        } catch (Exception e) {
            AbstractConnection.LOGGER.warn("caught aio process err:", e);
        }

    }

    @Override
    public void failed(Throwable exc, AIOSocketWR wr) {
        wr.writing.set(false);
        wr.con.close("write failed " + exc);
    }

}


class AIOReadHandler implements CompletionHandler<Integer, AIOSocketWR>
{
    @Override
    public void completed(final Integer i, final AIOSocketWR wr)
    {
        // con.getProcessor().getExecutor().execute(new Runnable() {
        // public void run() {
        if (i > 0)
        {
            try
            {
                wr.con.onReadData(i);
                wr.con.asynRead();
            } catch (IOException e)
            {
                wr.con.close("handle err:" + e);
            }
        } else if (i == -1)
        {
            // System.out.println("read -1 xxxxxxxxx "+con);
            wr.con.close("client closed");
        }
        // }
        // });
    }

    @Override
    public void failed(Throwable exc, AIOSocketWR wr)
    {
        wr.con.close(exc.toString());

    }
}
```

4:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\BackendAIOConnection.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.channels.NetworkChannel;

import io.mycat.backend.BackendConnection;

/**
 * @author mycat
 */
public abstract class BackendAIOConnection extends AbstractConnection implements
		BackendConnection {



	protected boolean isFinishConnect;

	public BackendAIOConnection(NetworkChannel channel) {
		super(channel);
	}

	public void register() throws IOException {
		this.asynRead();
	}


	public void setHost(String host) {
		this.host = host;
```

```
    }


    public void setPort(int port) {
        this.port = port;
    }




    public void discardClose(String reason){
        //跨节点处理,中断后端连接时关闭
    }
    public abstract void onConnectFailed(Throwable e);

    public boolean finishConnect() throws IOException {
        localPort = ((InetSocketAddress) channel.getLocalAddress()).getPort();
        isFinishConnect = true;
        return true;
    }

    public void setProcessor(NIOProcessor processor) {
        super.setProcessor(processor);
        processor.addBackend(this);
    }

    @Override
    public String toString() {
        return "BackendConnection [id=" + id + ", host=" + host + ", port="
                + port + ", localPort=" + localPort + "]";
    }
}
```

5:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\BIOConnection.java

```
package io.mycat.net;

public interface BIOConnection extends ClosableConnection{

}
```

6:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\ClosableConnection.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

public interface ClosableConnection {
	String getCharset();
	/**
	 * 关闭连接
	 */
	void close(String reason);

	boolean isClosed();

	public void idleCheck();

	long getStartupTime();

	String getHost();

	int getPort();

	int getLocalPort();

	long getNetInBytes();

	long getNetOutBytes();
}
```

7:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\ConnectionExcep
tion.java

```java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

public class ConnectionException extends RuntimeException {
     /**
      *
      */
     private static final long serialVersionUID = 1L;
     private final int code;
     private final String msg;

     public ConnectionException(int code, String msg) {
          super();
          this.code = code;
          this.msg = msg;
     }

     @Override
     public String toString() {
          return "ConnectionException [code=" + code + ", msg=" + msg + "]";
     }

}
```

8:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\factory\Backend

ConnectionFactory.java
```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.factory;

import java.io.IOException;
import java.nio.channels.AsynchronousSocketChannel;
import java.nio.channels.NetworkChannel;
import java.nio.channels.SocketChannel;

import io.mycat.MycatServer;

/**
 * @author mycat
 */
public abstract class BackendConnectionFactory {

    protected NetworkChannel openSocketChannel(boolean isAIO)
            throws IOException {
        if (isAIO) {
            return AsynchronousSocketChannel
              .open(MycatServer.getInstance().getNextAsyncChannelGroup());
        } else {
            SocketChannel channel = null;
            channel = SocketChannel.open();
            channel.configureBlocking(false);
```

```
                return channel;
            }

    }

}
```

9:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\factory\FrontendConnectionFactory.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.factory;

import java.io.IOException;
import java.net.StandardSocketOptions;
import java.nio.channels.NetworkChannel;

import io.mycat.MycatServer;
import io.mycat.net.FrontendConnection;

/**
 * @author mycat
 */
public abstract class FrontendConnectionFactory {
    protected abstract FrontendConnection getConnection(NetworkChannel channel)
            throws IOException;
    public FrontendConnection make(NetworkChannel channel) throws IOException {
        channel.setOption(StandardSocketOptions.SO_REUSEADDR, true);
        channel.setOption(StandardSocketOptions.SO_KEEPALIVE, true);

        FrontendConnection c = getConnection(channel);
        MycatServer.getInstance().getConfig().setSocketParams(c, true);
        return c;
    }



}
```

10:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\FrontendConnection.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import org.slf4j.Logger; import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import io.mycat.backend.mysql.CharsetUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.config.Capabilities;
import io.mycat.config.ErrorCode;
import io.mycat.config.Versions;
import io.mycat.net.handler.*;
import io.mycat.net.mysql.ErrorPacket;
import io.mycat.net.mysql.HandshakePacket;
import io.mycat.net.mysql.HandshakeV10Packet;
import io.mycat.net.mysql.MySQLPacket;
import io.mycat.net.mysql.OkPacket;
import io.mycat.util.CompressUtil;
import io.mycat.util.RandomUtil;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.InetSocketAddress;
import java.nio.channels.AsynchronousSocketChannel;
import java.nio.channels.NetworkChannel;
import java.nio.channels.SocketChannel;
import java.util.List;
import java.util.Set;

/**
 * @author mycat
 */
public abstract class FrontendConnection extends AbstractConnection {

    private         static         final         Logger         LOGGER         =
LoggerFactory.getLogger(FrontendConnection.class);

    protected long id;
    protected String host;
    protected int port;
    protected int localPort;
    protected long idleTimeout;
    protected byte[] seed;
    protected String user;
    protected String schema;
    protected String executeSql;

    protected FrontendPrivileges privileges;
    protected FrontendQueryHandler queryHandler;
    protected FrontendPrepareHandler prepareHandler;
    protected LoadDataInfileHandler loadDataInfileHandler;
    protected boolean isAccepted;
    protected boolean isAuthenticated;

    public FrontendConnection(NetworkChannel channel) throws IOException {
        super(channel);
        InetSocketAddress         localAddr         =         (InetSocketAddress)
channel.getLocalAddress();
        InetSocketAddress remoteAddr = null;
        if (channel instanceof SocketChannel) {
            remoteAddr       =       (InetSocketAddress)       ((SocketChannel)
channel).getRemoteAddress();

        } else if (channel instanceof AsynchronousSocketChannel) {
            remoteAddr   =   (InetSocketAddress)   ((AsynchronousSocketChannel)
channel).getRemoteAddress();
        }

        this.host = remoteAddr.getHostString();
        this.port = localAddr.getPort();
        this.localPort = remoteAddr.getPort();
        this.handler = new FrontendAuthenticator(this);
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getHost() {
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }

    public int getPort() {
        return port;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public int getLocalPort() {
        return localPort;
    }

    public void setLocalPort(int localPort) {
        this.localPort = localPort;
    }

    public void setAccepted(boolean isAccepted) {
        this.isAccepted = isAccepted;
    }

    public void setProcessor(NIOProcessor processor) {
        super.setProcessor(processor);
        processor.addFrontend(this);
    }

    public LoadDataInfileHandler getLoadDataInfileHandler() {
        return loadDataInfileHandler;
    }

    public          void          setLoadDataInfileHandler(LoadDataInfileHandler
loadDataInfileHandler) {
        this.loadDataInfileHandler = loadDataInfileHandler;
    }

    public void setQueryHandler(FrontendQueryHandler queryHandler) {
        this.queryHandler = queryHandler;
    }

    public void setPrepareHandler(FrontendPrepareHandler prepareHandler) {
        this.prepareHandler = prepareHandler;
    }

    public void setAuthenticated(boolean isAuthenticated) {
        this.isAuthenticated = isAuthenticated;
    }

    public FrontendPrivileges getPrivileges() {
        return privileges;
    }

    public void setPrivileges(FrontendPrivileges privileges) {
        this.privileges = privileges;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getSchema() {
        return schema;
    }

    public void setSchema(String schema) {
        this.schema = schema;
    }

    public String getExecuteSql() {
        return executeSql;
    }

    public void setExecuteSql(String executeSql) {
        this.executeSql = executeSql;
    }

    public byte[] getSeed() {
        return seed;
    }

    public boolean setCharsetIndex(int ci) {
        String charset = CharsetUtil.getCharset(ci);
        if (charset != null) {
            return setCharset(charset);
        } else {
            return false;
        }
    }

    public void writeErrMessage(int errno, String msg) {
        writeErrMessage((byte) 1, errno, msg);
    }

    public void writeErrMessage(byte id, int errno, String msg) {
        ErrorPacket err = new ErrorPacket();
        err.packetId = id;
        err.errno = errno;
        err.message = encodeString(msg, charset);
        err.write(this);
    }

    public void initDB(byte[] data) {

        MySQLMessage mm = new MySQLMessage(data);
        mm.position(5);
        String db = mm.readString();

        // 检查 schema 的有效性
        if (db == null || !privileges.schemaExists(db)) {
            writeErrMessage(ErrorCode.ER_BAD_DB_ERROR, "Unknown database '" + db +
"'");
            return;
        }

        if (!privileges.userExists(user, host)) {
            writeErrMessage(ErrorCode.ER_ACCESS_DENIED_ERROR, "Access denied for
user '" + user + "'");
            return;
        }

        Set<String> schemas = privileges.getUserSchemas(user);
        if (schemas == null || schemas.size() == 0 || schemas.contains(db)) {
            this.schema = db;
            write(writeToBuffer(OkPacket.OK, allocate()));
        } else {
            String s = "Access denied for user '" + user + "' to database '" + db
+ "'";
            writeErrMessage(ErrorCode.ER_DBACCESS_DENIED_ERROR, s);
        }
    }


    public void loadDataInfileStart(String sql) {
        if (loadDataInfileHandler != null) {
            try {
                loadDataInfileHandler.start(sql);
            } catch (Exception e) {
                LOGGER.error("load data error", e);
                writeErrMessage(ErrorCode.ERR_HANDLE_DATA, e.getMessage());
            }

        } else {
            writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR, "load data infile sql
is not  unsupported!");
        }
    }

    public void loadDataInfileData(byte[] data) {
        if (loadDataInfileHandler != null) {
            try {
                loadDataInfileHandler.handle(data);
            } catch (Exception e) {
                LOGGER.error("load data error", e);
                writeErrMessage(ErrorCode.ERR_HANDLE_DATA, e.getMessage());
            }
        } else {
            writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,  "load  data  infile
data is not  unsupported!");
        }

    }

    public void loadDataInfileEnd(byte packID) {
        if (loadDataInfileHandler != null) {
            try {
                loadDataInfileHandler.end(packID);
            } catch (Exception e) {
                LOGGER.error("load data error", e);
                writeErrMessage(ErrorCode.ERR_HANDLE_DATA, e.getMessage());
            }
        } else {
            writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR, "load data infile end
is not  unsupported!");
        }
    }


    public void query(String sql) {

        if (sql == null || sql.length() == 0) {
            writeErrMessage(ErrorCode.ER_NOT_ALLOWED_COMMAND, "Empty SQL");
            return;
        }

        if (LOGGER.isDebugEnabled()) {
            LOGGER.debug(new               StringBuilder().append(this).append("
").append(sql).toString());
        }

        // remove last ';'
        if (sql.endsWith(";")) {
            sql = sql.substring(0, sql.length() - 1);
        }

        // 记录 SQL
        this.setExecuteSql(sql);

        // 防火墙策略( SQL 黑名单/ 注入攻击)
        if ( !privileges.checkFirewallSQLPolicy( user, sql ) ) {
            writeErrMessage(ErrorCode.ERR_WRONG_USED,
                    "The statement is unsafe SQL, reject for user '" + user + "'");
            return;
        }

        // DML 权限检查
        try {
            boolean isPassed = privileges.checkDmlPrivilege(user, schema, sql);
            if ( !isPassed ) {
                writeErrMessage(ErrorCode.ERR_WRONG_USED,
                        "The statement DML privilege check is not passed, reject
for user '" + user + "'");
                return;
            }
         } catch( com.alibaba.druid.sql.parser.ParserException e1) {
            writeErrMessage(ErrorCode.ERR_WRONG_USED,  e1.getMessage());
            LOGGER.error("parse exception", e1 );
                return;
        }

        // 执行查询
        if (queryHandler != null) {
            queryHandler.setReadOnly(privileges.isReadOnly(user));
            queryHandler.query(sql);

        } else {
            writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,                "Query
unsupported!");
        }
    }

    public void query(byte[] data) {

        // 取得语句
        String sql = null;
        try {
            MySQLMessage mm = new MySQLMessage(data);
            mm.position(5);
            sql = mm.readString(charset);
        } catch (UnsupportedEncodingException e) {
            writeErrMessage(ErrorCode.ER_UNKNOWN_CHARACTER_SET, "Unknown charset
'" + charset + "'");
            return;
        }

        this.query( sql );
    }
```

```
     public void stmtPrepare(byte[] data) {
          if (prepareHandler != null) {
               // 取得语句
               MySQLMessage mm = new MySQLMessage(data);
               mm.position(5);
               String sql = null;
               try {
                    sql = mm.readString(charset);
               } catch (UnsupportedEncodingException e) {
                    writeErrMessage(ErrorCode.ER_UNKNOWN_CHARACTER_SET,
                              "Unknown charset '" + charset + "'");
                    return;
               }
               if (sql == null || sql.length() == 0) {
                    writeErrMessage(ErrorCode.ER_NOT_ALLOWED_COMMAND, "Empty SQL");
                    return;
               }

               // 记录 SQL
               this.setExecuteSql(sql);

               // 执行预处理
               prepareHandler.prepare(sql);
          } else {
               writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,              "Prepare
unsupported!");
          }
     }

     public void stmtSendLongData(byte[] data) {
          if(prepareHandler != null) {
               prepareHandler.sendLongData(data);
          } else {
               writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,              "Prepare
unsupported!");
          }
     }

     public void stmtReset(byte[] data) {
          if(prepareHandler != null) {
               prepareHandler.reset(data);
          } else {
               writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,              "Prepare
unsupported!");
          }
     }
```

```
    public void stmtExecute(byte[] data) {
        if (prepareHandler != null) {
            prepareHandler.execute(data);
        } else {
            writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,              "Prepare
unsupported!");
        }
    }

    public void stmtClose(byte[] data) {
        if (prepareHandler != null) {
            prepareHandler.close( data );
        } else {
            writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,              "Prepare
unsupported!");
        }
    }

    public void ping() {
        write(writeToBuffer(OkPacket.OK, allocate()));
    }

    public void heartbeat(byte[] data) {
        write(writeToBuffer(OkPacket.OK, allocate()));
    }

    public void kill(byte[] data) {
        writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR, "Unknown command");
    }

    public void unknown(byte[] data) {
        writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR, "Unknown command");
    }

    @Override
    public void register() throws IOException {
        if (!isClosed.get()) {

            // 生成认证数据
            byte[] rand1 = RandomUtil.randomBytes(8);
            byte[] rand2 = RandomUtil.randomBytes(12);

            // 保存认证数据
            byte[] seed = new byte[rand1.length + rand2.length];
            System.arraycopy(rand1, 0, seed, 0, rand1.length);
            System.arraycopy(rand2, 0, seed, rand1.length, rand2.length);
            this.seed = seed;

            // 发送握手数据包
            boolean                     useHandshakeV10                     =
MycatServer.getInstance().getConfig().getSystem().getUseHandshakeV10() == 1;
            if(useHandshakeV10) {
                HandshakeV10Packet hs = new HandshakeV10Packet();
                hs.packetId = 0;
                hs.protocolVersion = Versions.PROTOCOL_VERSION;
                hs.serverVersion = Versions.SERVER_VERSION;
                hs.threadId = id;
                hs.seed = rand1;
                hs.serverCapabilities = getServerCapabilities();
                hs.serverCharsetIndex = (byte) (charsetIndex & 0xff);
                hs.serverStatus = 2;
                hs.restOfScrambleBuff = rand2;
                hs.write(this);
            } else {
                HandshakePacket hs = new HandshakePacket();
                hs.packetId = 0;
                hs.protocolVersion = Versions.PROTOCOL_VERSION;
                hs.serverVersion = Versions.SERVER_VERSION;
                hs.threadId = id;
                hs.seed = rand1;
                hs.serverCapabilities = getServerCapabilities();
                hs.serverCharsetIndex = (byte) (charsetIndex & 0xff);
                hs.serverStatus = 2;
                hs.restOfScrambleBuff = rand2;
                hs.write(this);
            }

            // asynread response
            this.asynRead();
        }
    }

    @Override
    public void handle(final byte[] data) {

        if (isSupportCompress()) {
            List<byte[]>   packs   =   CompressUtil.decompressMysqlPacket(data,
decompressUnfinishedDataQueue);
            for (byte[] pack : packs) {
                if (pack.length != 0) {
                    rawHandle(pack);
                }
            }
        } else {
            rawHandle(data);
        }
    }

    public void rawHandle(final byte[] data) {

        //load data infile  客户端会发空包 长度为 4
        if (data.length == 4 && data[0] == 0 && data[1] == 0 && data[2] == 0) {
            // load in data 空包
            loadDataInfileEnd(data[3]);
            return;
        }
        //修改 quit 的判断,当 load data infile 分隔符为\001 时可能会出现误判断的
bug.
        if (data.length>4 && data[0] == 1 && data[1] == 0 && data[2]== 0 && data[3]
== 0 &&data[4] == MySQLPacket.COM_QUIT) {
            this.getProcessor().getCommands().doQuit();
            this.close("quit cmd");
            return;
        }
        handler.handle(data);
    }

    protected int getServerCapabilities() {
        int flag = 0;
        flag |= Capabilities.CLIENT_LONG_PASSWORD;
        flag |= Capabilities.CLIENT_FOUND_ROWS;
        flag |= Capabilities.CLIENT_LONG_FLAG;
        flag |= Capabilities.CLIENT_CONNECT_WITH_DB;
        // flag |= Capabilities.CLIENT_NO_SCHEMA;
        boolean                                                     usingCompress=
MycatServer.getInstance().getConfig().getSystem().getUseCompression()==1 ;
        if (usingCompress) {
            flag |= Capabilities.CLIENT_COMPRESS;
        }

        flag |= Capabilities.CLIENT_ODBC;
         flag |= Capabilities.CLIENT_LOCAL_FILES;
        flag |= Capabilities.CLIENT_IGNORE_SPACE;
        flag |= Capabilities.CLIENT_PROTOCOL_41;
        flag |= Capabilities.CLIENT_INTERACTIVE;
        // flag |= Capabilities.CLIENT_SSL;
        flag |= Capabilities.CLIENT_IGNORE_SIGPIPE;
        flag |= Capabilities.CLIENT_TRANSACTIONS;
        // flag |= ServerDefs.CLIENT_RESERVED;
```

```
          flag |= Capabilities.CLIENT_SECURE_CONNECTION;
         flag |= Capabilities.CLIENT_MULTI_STATEMENTS;
         flag |= Capabilities.CLIENT_MULTI_RESULTS;
         boolean                    useHandshakeV10                    =
MycatServer.getInstance().getConfig().getSystem().getUseHandshakeV10() == 1;
         if(useHandshakeV10) {
          flag |= Capabilities.CLIENT_PLUGIN_AUTH;
         }
          return flag;
     }

     protected boolean isConnectionReset(Throwable t) {
          if (t instanceof IOException) {
               String msg = t.getMessage();
               return (msg != null && msg.contains("Connection reset by peer"));
          }
          return false;
     }

     @Override
     public String toString() {
          return new StringBuilder().append("[thread=")
                    .append(Thread.currentThread().getName()).append(",class=")
                    .append(getClass().getSimpleName()).append(",id=").append(id)
                    .append(",host=").append(host).append(",port=").append(port)
                    .append(",schema=").append(schema).append(']').toString();
     }

     private final static byte[] encodeString(String src, String charset) {
          if (src == null) {
               return null;
          }
          if (charset == null) {
               return src.getBytes();
          }
          try {
               return src.getBytes(charset);
          } catch (UnsupportedEncodingException e) {
               return src.getBytes();
          }
     }

     @Override
     public void close(String reason) {
          super.close(isAuthenticated ? reason : "");
     }
}
```

11:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\BackendAsyncHandler.java

```java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

import java.util.concurrent.Executor;

import io.mycat.net.NIOHandler;

/**
 * @author mycat
 */
public abstract class BackendAsyncHandler implements NIOHandler {

    protected void offerData(byte[] data, Executor executor) {
        handleData(data);

        // if (dataQueue.offer(data)) {
        // handleQueue(executor);
        // } else {
        // offerDataError();
        // }
    }
    protected abstract void offerDataError();

    protected abstract void handleData(byte[] data);

}
```

12:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\FrontendAuthenticator.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

import java.nio.ByteBuffer;
import java.security.NoSuchAlgorithmException;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import io.mycat.backend.mysql.SecurityUtil;
import io.mycat.config.Capabilities;
import io.mycat.config.ErrorCode;
import io.mycat.config.model.UserConfig;
import io.mycat.net.FrontendConnection;
import io.mycat.net.NIOHandler;
import io.mycat.net.NIOProcessor;
import io.mycat.net.mysql.AuthPacket;
import io.mycat.net.mysql.MySQLPacket;
import io.mycat.net.mysql.QuitPacket;

/**
 * 前端认证处理器
 *
 * @author mycat
 */
public class FrontendAuthenticator implements NIOHandler {

    private          static          final          Logger          LOGGER          =
LoggerFactory.getLogger(FrontendAuthenticator.class);
    private static final byte[] AUTH_OK = new byte[] { 7, 0, 0, 2, 0, 0, 0, 2, 0, 0,
0 };

    protected final FrontendConnection source;

    public FrontendAuthenticator(FrontendConnection source) {
        this.source = source;
    }

    @Override
    public void handle(byte[] data) {
        // check quit packet
        if (data.length == QuitPacket.QUIT.length && data[4] == MySQLPacket.COM_QUIT)
{
            source.close("quit packet");
            return;
        }

        AuthPacket auth = new AuthPacket();
        auth.read(data);

        //huangyiming add
        int                              nopassWordLogin                              =
MycatServer.getInstance().getConfig().getSystem().getNonePasswordLogin();
        //如果无密码登陆则跳过密码验证这个步骤
        boolean skipPassWord = false;
        String defaultUser = "";
        if(nopassWordLogin == 1){
          skipPassWord = true;
          Map<String,              UserConfig>              userMaps              =
MycatServer.getInstance().getConfig().getUsers();
```

```
          if(!userMaps.isEmpty()){
               setDefaultAccount(auth, userMaps);
          }
        }
        // check user
        if (!checkUser(auth.user, source.getHost())) {
          failure(ErrorCode.ER_ACCESS_DENIED_ERROR, "Access denied for user '" +
auth.user + "' with host '" + source.getHost()+ "'");
          return;
        }
        // check password
        if (!skipPassWord && !checkPassword(auth.password, auth.user)) {
          failure(ErrorCode.ER_ACCESS_DENIED_ERROR, "Access denied for user '" +
auth.user + "', because password is error ");
          return;
        }

        // check degrade
        if ( isDegrade( auth.user ) ) {
           failure(ErrorCode.ER_ACCESS_DENIED_ERROR, "Access denied for user '" +
auth.user + "', because service be degraded ");
             return;
        }

        // check schema
        switch (checkSchema(auth.database, auth.user)) {
        case ErrorCode.ER_BAD_DB_ERROR:
            failure(ErrorCode.ER_BAD_DB_ERROR, "Unknown database '" + auth.database
+ "'");
            break;
        case ErrorCode.ER_DBACCESS_DENIED_ERROR:
            String s = "Access denied for user '" + auth.user + "' to database '" +
auth.database + "'";
            failure(ErrorCode.ER_DBACCESS_DENIED_ERROR, s);
            break;
        default:
            success(auth);
        }
    }

    /**
     * 设置了无密码登陆的情况下把客户端传过来的用户账号改变为默认账户
     * @param auth
     * @param userMaps
     */
     private void setDefaultAccount(AuthPacket auth, Map<String, UserConfig> userMaps)
{
        String defaultUser;
        Iterator<UserConfig> items = userMaps.values().iterator();
        while(items.hasNext()){
            UserConfig userConfig = items.next();
            if(userConfig.isDefaultAccount()){
                defaultUser = userConfig.getName();
                auth.user = defaultUser;
            }
        }
    }

//TODO: add by zhuam
//前端 connection 达到该用户设定的阀值后, 立马降级拒绝连接
protected boolean isDegrade(String user) {

    int benchmark = source.getPrivileges().getBenchmark(user);
    if ( benchmark > 0 ) {

        int forntedsLength = 0;
        NIOProcessor[] processors = MycatServer.getInstance().getProcessors();
            for (NIOProcessor p : processors) {
                forntedsLength += p.getForntedsLength();
            }

            if ( forntedsLength >= benchmark ) {
                return true;
            }
    }

        return false;
}

protected boolean checkUser(String user, String host) {
    return source.getPrivileges().userExists(user, host);
}

protected boolean checkPassword(byte[] password, String user) {
    String pass = source.getPrivileges().getPassword(user);

    // check null
    if (pass == null || pass.length() == 0) {
        if (password == null || password.length == 0) {
            return true;
        } else {
            return false;
        }
    }
```

```
        if (password == null || password.length == 0) {
            return false;
        }

        // encrypt
        byte[] encryptPass = null;
        try {
            encryptPass          =          SecurityUtil.scramble411(pass.getBytes(),
source.getSeed());
        } catch (NoSuchAlgorithmException e) {
            LOGGER.warn(source.toString(), e);
            return false;
        }
        if (encryptPass != null && (encryptPass.length == password.length)) {
            int i = encryptPass.length;
            while (i-- != 0) {
                if (encryptPass[i] != password[i]) {
                    return false;
                }
            }
        } else {
            return false;
        }

        return true;
    }

    protected int checkSchema(String schema, String user) {
        if (schema == null) {
            return 0;
        }
        FrontendPrivileges privileges = source.getPrivileges();
        if (!privileges.schemaExists(schema)) {
            return ErrorCode.ER_BAD_DB_ERROR;
        }
        Set<String> schemas = privileges.getUserSchemas(user);
        if (schemas == null || schemas.size() == 0 || schemas.contains(schema)) {
            return 0;
        } else {
            return ErrorCode.ER_DBACCESS_DENIED_ERROR;
        }
    }

    protected void success(AuthPacket auth) {
        source.setAuthenticated(true);
        source.setUser(auth.user);
        source.setSchema(auth.database);
        source.setCharsetIndex(auth.charsetIndex);
        source.setHandler(new FrontendCommandHandler(source));

        if (LOGGER.isInfoEnabled()) {
            StringBuilder s = new StringBuilder();
            s.append(source).append('\'').append(auth.user).append("'          login
success");
            byte[] extra = auth.extra;
            if (extra != null && extra.length > 0) {
                s.append(",extra:").append(new String(extra));
            }
            LOGGER.info(s.toString());
        }

        ByteBuffer buffer = source.allocate();
        source.write(source.writeToBuffer(AUTH_OK, buffer));
        boolean                          clientCompress                          =
Capabilities.CLIENT_COMPRESS==(Capabilities.CLIENT_COMPRESS & auth.clientFlags);
        boolean                                                     usingCompress=
MycatServer.getInstance().getConfig().getSystem().getUseCompression()==1 ;
        if(clientCompress&&usingCompress)
        {
            source.setSupportCompress(true);
        }
    }

    protected void failure(int errno, String info) {
        LOGGER.error(source.toString() + info);
        source.writeErrMessage((byte) 2, errno, info);
    }

}
```

13:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\FrontendCommandHandler.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.config.ErrorCode;
import io.mycat.net.FrontendConnection;
import io.mycat.net.NIOHandler;
import io.mycat.net.mysql.MySQLPacket;
import io.mycat.statistic.CommandCount;

/**
 * 前端命令处理器
 *
 * @author mycat
 */
public class FrontendCommandHandler implements NIOHandler
{

    protected final FrontendConnection source;
    protected final CommandCount commands;

    public FrontendCommandHandler(FrontendConnection source)
    {
        this.source = source;
        this.commands = source.getProcessor().getCommands();
    }

    @Override
    public void handle(byte[] data)
    {

if(source.getLoadDataInfileHandler()!=null&&source.getLoadDataInfileHandler().isSt
artLoadData())
        {
            MySQLMessage mm = new MySQLMessage(data);
            int  packetLength = mm.readUB3();
            if(packetLength+4==data.length)
            {
                source.loadDataInfileData(data);
            }
            return;
        }
        switch (data[4])
        {
            case MySQLPacket.COM_INIT_DB:
                commands.doInitDB();
                source.initDB(data);
                break;
            case MySQLPacket.COM_QUERY:
                commands.doQuery();
                source.query(data);
                break;
            case MySQLPacket.COM_PING:
                commands.doPing();
                source.ping();
                break;
            case MySQLPacket.COM_QUIT:
                commands.doQuit();
                source.close("quit cmd");
                break;
            case MySQLPacket.COM_PROCESS_KILL:
                commands.doKill();
                source.kill(data);
                break;
            case MySQLPacket.COM_STMT_PREPARE:
                commands.doStmtPrepare();
                source.stmtPrepare(data);
                break;
            case MySQLPacket.COM_STMT_SEND_LONG_DATA:
              commands.doStmtSendLongData();
              source.stmtSendLongData(data);
              break;
            case MySQLPacket.COM_STMT_RESET:
              commands.doStmtReset();
              source.stmtReset(data);
              break;
            case MySQLPacket.COM_STMT_EXECUTE:
                commands.doStmtExecute();
                source.stmtExecute(data);
                break;
            case MySQLPacket.COM_STMT_CLOSE:
                commands.doStmtClose();
                source.stmtClose(data);
                break;
            case MySQLPacket.COM_HEARTBEAT:
                commands.doHeartbeat();
                source.heartbeat(data);
                break;
            default:
                     commands.doOther();
                     source.writeErrMessage(ErrorCode.ER_UNKNOWN_COM_ERROR,
                             "Unknown command");

        }
    }

}
```

14:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\FrontendPrepareHandler.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

/**
 * SQL 预处理处理器
 *
 * @author mycat, CrazyPig
 */
public interface FrontendPrepareHandler {
    void prepare(String sql);

    void sendLongData(byte[] data);

    void reset(byte[] data);

    void execute(byte[] data);

    void close(byte[] data);

    void clear();

}
```

15:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\FrontendPrivileges.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

import java.util.Set;

/**
 * 权限提供者
 *
 * @author mycat
 */
public interface FrontendPrivileges {

    /**
     * 检查 schema 是否存在
     */
    boolean schemaExists(String schema);

    /**
     * 检查用户是否存在，并且可以使用 host 实行隔离策略。
     */
    boolean userExists(String user, String host);

    /**
     * 提供用户的服务器端密码
     */
    String getPassword(String user);

    /**
     * 提供有效的用户 schema 集合
     */
    Set<String> getUserSchemas(String user);

    /**
     * 检查用户是否为只读权限
     * @param user
     * @return
     */
    Boolean isReadOnly(String user);

    /**
     * 获取设定的系统最大连接数的降级阀值
     * @param user
     * @return
     */
    int getBenchmark(String user);


    /**
     * 检查防火墙策略
     * （白名单策略）
     * @param user
     * @param host
     * @return
     */
    boolean checkFirewallWhiteHostPolicy(String user, String host);
    /**
     * 检查防火墙策略
     * (SQL 黑名单及注入策略)
     * @param sql
     * @return
     */
    boolean checkFirewallSQLPolicy(String user, String sql);


    /**
     * 检查 SQL 语句的 DML 权限
     * @return
     */
    boolean checkDmlPrivilege(String user, String schema, String sql);

}
```

16:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\FrontendQueryHandler.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

/**
 * 查询处理器
 *
 * @author mycat
 */
public interface FrontendQueryHandler {

     void query(String sql);

     void setReadOnly(Boolean readOnly);
}
```

17:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\handler\LoadDataInfileHandler.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.handler;

/**
 * load data infile
 *
 * @author magicdoom
 */
public interface LoadDataInfileHandler
{

    void start(String sql);
    void handle(byte[] data);

    void end(byte packID);

    void clear();

    byte getLastPackId();

    boolean isStartLoadData();

}
```

18:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\AuthPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.io.IOException;
import java.io.OutputStream;
import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.backend.mysql.StreamUtil;
import io.mycat.config.Capabilities;
import io.mycat.net.BackendAIOConnection;

/**
 * From client to server during initial handshake.
 *
 * <pre>
 * Bytes                        Name
 * -----                        ----
 * 4                            client_flags
 * 4                            max_packet_size
 * 1                            charset_number
 * 23                           (filler) always 0x00...
 * n (Null-Terminated String)   user
 * n (Length Coded Binary)      scramble_buff (1 + x bytes)
 * n (Null-Terminated String)   databasename (optional)
 *
 *                                                                              @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Client_Authentic
ation_Packet
 * </pre>
 *
 * @author mycat
 */
public class AuthPacket extends MySQLPacket {
    private static final byte[] FILLER = new byte[23];

    public long clientFlags;
    public long maxPacketSize;
    public int charsetIndex;
    public byte[] extra;// from FILLER(23)
    public String user;
    public byte[] password;
    public String database;

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        clientFlags = mm.readUB4();
        maxPacketSize = mm.readUB4();
        charsetIndex = (mm.read() & 0xff);
        // read extra
        int current = mm.position();
        int len = (int) mm.readLength();
        if (len > 0 && len < FILLER.length) {
            byte[] ab = new byte[len];
            System.arraycopy(mm.bytes(), mm.position(), ab, 0, len);
            this.extra = ab;
        }
        mm.position(current + FILLER.length);
        user = mm.readStringWithNull();
        password = mm.readBytesWithLength();
        if  (((clientFlags  &  Capabilities.CLIENT_CONNECT_WITH_DB)  !=  0)  &&
mm.hasRemaining()) {
            database = mm.readStringWithNull();
        }
    }

    public void write(OutputStream out) throws IOException {
        StreamUtil.writeUB3(out, calcPacketSize());
        StreamUtil.write(out, packetId);
        StreamUtil.writeUB4(out, clientFlags);
        StreamUtil.writeUB4(out, maxPacketSize);
        StreamUtil.write(out, (byte) charsetIndex);
        out.write(FILLER);
        if (user == null) {
            StreamUtil.write(out, (byte) 0);
        } else {
            StreamUtil.writeWithNull(out, user.getBytes());
        }
        if (password == null) {
            StreamUtil.write(out, (byte) 0);
        } else {
            StreamUtil.writeWithLength(out, password);
        }
        if (database == null) {
            StreamUtil.write(out, (byte) 0);
        } else {
            StreamUtil.writeWithNull(out, database.getBytes());
        }
    }

    @Override
    public void write(BackendAIOConnection c) {
        ByteBuffer buffer = c.allocate();
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        BufferUtil.writeUB4(buffer, clientFlags);
        BufferUtil.writeUB4(buffer, maxPacketSize);
        buffer.put((byte) charsetIndex);
        buffer = c.writeToBuffer(FILLER, buffer);
        if (user == null) {
            buffer = c.checkWriteBuffer(buffer, 1,true);
            buffer.put((byte) 0);
        } else {
            byte[] userData = user.getBytes();
            buffer = c.checkWriteBuffer(buffer, userData.length + 1,true);
            BufferUtil.writeWithNull(buffer, userData);
        }
        if (password == null) {
            buffer = c.checkWriteBuffer(buffer, 1,true);
            buffer.put((byte) 0);
        } else {
            buffer                    =                   c.checkWriteBuffer(buffer,
BufferUtil.getLength(password),true);
            BufferUtil.writeWithLength(buffer, password);
        }
        if (database == null) {
            buffer = c.checkWriteBuffer(buffer, 1,true);
            buffer.put((byte) 0);
        } else {
            byte[] databaseData = database.getBytes();
            buffer = c.checkWriteBuffer(buffer, databaseData.length + 1,true);
            BufferUtil.writeWithNull(buffer, databaseData);
        }
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        int size = 32;// 4+4+1+23;
        size += (user == null) ? 1 : user.length() + 1;
        size += (password == null) ? 1 : BufferUtil.getLength(password);
        size += (database == null) ? 1 : database.length() + 1;
        return size;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Authentication Packet";
    }

}
```

19:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\BinaryPa
cket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.io.IOException;
import java.io.InputStream;
import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.StreamUtil;
import io.mycat.net.BackendAIOConnection;
import io.mycat.net.FrontendConnection;

/**
 * @author mycat
 */
public class BinaryPacket extends MySQLPacket {
    public static final byte OK = 1;
    public static final byte ERROR = 2;
    public static final byte HEADER = 3;
    public static final byte FIELD = 4;
    public static final byte FIELD_EOF = 5;
    public static final byte ROW = 6;
    public static final byte PACKET_EOF = 7;

    public byte[] data;

    public void read(InputStream in) throws IOException {
        packetLength = StreamUtil.readUB3(in);
        packetId = StreamUtil.read(in);
        byte[] ab = new byte[packetLength];
        StreamUtil.read(in, ab, 0, ab.length);
        data = ab;
    }

    @Override
    public  ByteBuffer  write(ByteBuffer  buffer,  FrontendConnection  c,boolean
writeSocketIfFull) {
        buffer                    =                    c.checkWriteBuffer(buffer,
c.getPacketHeaderSize(),writeSocketIfFull);
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer = c.writeToBuffer(data, buffer);
        return buffer;
    }
    @Override
    public void write(BackendAIOConnection c) {
        ByteBuffer buffer = c.allocate();
        buffer=
c.checkWriteBuffer(buffer,c.getPacketHeaderSize()+calcPacketSize(),false);
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer.put(data);
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        return data == null ? 0 : data.length;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Binary Packet";
    }

}

20:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\BinaryRo
wDataPacket.java
package io.mycat.net.mysql;


import java.nio.ByteBuffer;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import io.mycat.backend.mysql.BufferUtil;
import io.mycat.config.Fields;
import io.mycat.memory.unsafe.row.UnsafeRow;
import io.mycat.net.FrontendConnection;
import io.mycat.util.ByteUtil;
import io.mycat.util.DateUtil;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * ProtocolBinary::ResultsetRow:
 * row of a binary resultset (COM_STMT_EXECUTE)

 * Payload
 * 1              packet header [00]
 * string[$len]   NULL-bitmap, length: (column_count + 7 + 2) / 8
 * string[$len]   values
 *
 * A Binary Protocol Resultset Row is made up of the NULL bitmap
 * containing as many bits as we have columns in the resultset + 2
 * and the values for columns that are not NULL in the Binary Protocol Value format.
 *
 *                                                                              @see
@http://dev.mysql.com/doc/internals/en/binary-protocol-resultset-row.html#packet-P
rotocolBinary::ResultsetRow
 * @see @http://dev.mysql.com/doc/internals/en/binary-protocol-value.html
 * @author CrazyPig
 *
 */
public class BinaryRowDataPacket extends MySQLPacket {
     private         static         final         Logger         LOGGER         =
LoggerFactory.getLogger(BinaryRowDataPacket.class);
     public int fieldCount;
     public List<byte[]> fieldValues;
     public byte packetHeader = (byte) 0;
     public byte[] nullBitMap;

     public List<FieldPacket> fieldPackets;

     public BinaryRowDataPacket() {}

     /**
      * 从 UnsafeRow 转换成 BinaryRowDataPacket
      *
      * 说明: 当开启<b>isOffHeapuseOffHeapForMerge</b>参数时,会使用UnsafeRow封装数
```

```
据，
     * 因此需要从这个对象里面将数据封装成 BinaryRowDataPacket
     *
     * @param fieldPackets
     * @param unsafeRow
     */
    public void read(List<FieldPacket> fieldPackets, UnsafeRow unsafeRow) {
        this.fieldPackets = fieldPackets;
        this.fieldCount = unsafeRow.numFields();
        this.fieldValues = new ArrayList<byte[]>(fieldCount);
        this.packetId = unsafeRow.packetId;
        this.nullBitMap = new byte[(fieldCount + 7 + 2) / 8];

        for(int i = 0; i < this.fieldCount; i++) {
            byte[] fv = unsafeRow.getBinary(i);
            FieldPacket fieldPk = fieldPackets.get(i);
            if(fv == null) {
                storeNullBitMap(i);
                this.fieldValues.add(fv);
            } else {
                convert(fv, fieldPk);
            }
        }
    }

    /**
     * 从 RowDataPacket 转换成 BinaryRowDataPacket
     * @param fieldPackets 字段包集合
     * @param rowDataPk 文本协议行数据包
     */
    public void read(List<FieldPacket> fieldPackets, RowDataPacket rowDataPk) {
        this.fieldPackets = fieldPackets;
        this.fieldCount = rowDataPk.fieldCount;
        this.fieldValues = new ArrayList<byte[]>(fieldCount);
        this.packetId = rowDataPk.packetId;
        this.nullBitMap = new byte[(fieldCount + 7 + 2) / 8];

        List<byte[]> _fieldValues = rowDataPk.fieldValues;
        for (int i = 0; i < fieldCount; i++) {
            byte[] fv = _fieldValues.get(i);
            FieldPacket fieldPk = fieldPackets.get(i);
            if (fv == null) { // 字段值为 null,根据协议规定存储 nullBitMap
                storeNullBitMap(i);
                this.fieldValues.add(fv);
            } else {
                convert(fv, fieldPk);
            }
        }
    }

    private void storeNullBitMap(int i) {
        int bitMapPos = (i + 2) / 8;
        int bitPos = (i + 2) % 8;
        this.nullBitMap[bitMapPos] |= (byte) (1 << bitPos);
    }

    /**
     * 从 RowDataPacket 的 fieldValue 的数据转化成 BinaryRowDataPacket 的 fieldValue
数据
     * @param fv
     * @param fieldPk
     */
    private void convert(byte[] fv, FieldPacket fieldPk) {

        int fieldType = fieldPk.type;
        switch (fieldType) {
        case Fields.FIELD_TYPE_STRING:
        case Fields.FIELD_TYPE_VARCHAR:
        case Fields.FIELD_TYPE_VAR_STRING:
        case Fields.FIELD_TYPE_ENUM:
        case Fields.FIELD_TYPE_SET:
        case Fields.FIELD_TYPE_LONG_BLOB:
        case Fields.FIELD_TYPE_MEDIUM_BLOB:
        case Fields.FIELD_TYPE_BLOB:
        case Fields.FIELD_TYPE_TINY_BLOB:
        case Fields.FIELD_TYPE_GEOMETRY:
        case Fields.FIELD_TYPE_BIT:
        case Fields.FIELD_TYPE_DECIMAL:
        case Fields.FIELD_TYPE_NEW_DECIMAL:
            // Fields
            // value (lenenc_str) -- string

            // Example
            // 03 66 6f 6f -- string = "foo"
            this.fieldValues.add(fv);
            break;
        case Fields.FIELD_TYPE_LONGLONG:
            // Fields
            // value (8) -- integer

            // Example
            // 01 00 00 00 00 00 00 00 -- int64 = 1
            long longVar = ByteUtil.getLong(fv);
            this.fieldValues.add(ByteUtil.getBytes(longVar));
            break;
        case Fields.FIELD_TYPE_LONG:
        case Fields.FIELD_TYPE_INT24:
            // Fields
            // value (4) -- integer

            // Example
            // 01 00 00 00 -- int32 = 1
            int intVar = ByteUtil.getInt(fv);
            this.fieldValues.add(ByteUtil.getBytes(intVar));
            break;
        case Fields.FIELD_TYPE_SHORT:
        case Fields.FIELD_TYPE_YEAR:
            // Fields
            // value (2) -- integer

            // Example
            // 01 00 -- int16 = 1
            short shortVar = ByteUtil.getShort(fv);
            this.fieldValues.add(ByteUtil.getBytes(shortVar));
            break;
        case Fields.FIELD_TYPE_TINY:
            // Fields
            // value (1) -- integer

            // Example
            // 01 -- int8 = 1
            int tinyVar = ByteUtil.getInt(fv);
            byte[] bytes = new byte[1];
            bytes[0] = (byte)tinyVar;
            this.fieldValues.add(bytes);
            break;
        case Fields.FIELD_TYPE_DOUBLE:
            // Fields
            // value (string.fix_len) -- (len=8) double

            // Example
            // 66 66 66 66 66 66 24 40 -- double = 10.2
            double doubleVar = ByteUtil.getDouble(fv);
            this.fieldValues.add(ByteUtil.getBytes(doubleVar));
            break;
        case Fields.FIELD_TYPE_FLOAT:
            // Fields
            // value (string.fix_len) -- (len=4) float

            // Example
            // 33 33 23 41 -- float = 10.2
```

```
                float floatVar = ByteUtil.getFloat(fv);
                this.fieldValues.add(ByteUtil.getBytes(floatVar));
                break;
            case Fields.FIELD_TYPE_DATE:
                try {
                    Date    dateVar    =    DateUtil.parseDate(ByteUtil.getDate(fv),
DateUtil.DATE_PATTERN_ONLY_DATE);
                    this.fieldValues.add(ByteUtil.getBytes(dateVar, false));

                } catch(org.joda.time.IllegalFieldValueException e1) {
                    // 当时间为 0000-00-00 00:00:00 的时候, 默认返回 1970-01-01
08:00:00.0
                    this.fieldValues.add(ByteUtil.getBytes(new Date(0L), false));
                } catch (ParseException e) {
                    LOGGER.error("error",e);
                }
                break;
            case Fields.FIELD_TYPE_DATETIME:
            case Fields.FIELD_TYPE_TIMESTAMP:
                String dateStr = ByteUtil.getDate(fv);
                Date dateTimeVar = null;
                try {
                    if (dateStr.indexOf(".") > 0) {
                        dateTimeVar          =          DateUtil.parseDate(dateStr,
DateUtil.DATE_PATTERN_FULL);
                        this.fieldValues.add(ByteUtil.getBytes(dateTimeVar,
false));
                    } else {
                        dateTimeVar          =          DateUtil.parseDate(dateStr,
DateUtil.DEFAULT_DATE_PATTERN);
                        this.fieldValues.add(ByteUtil.getBytes(dateTimeVar,
false));
                    }
                } catch(org.joda.time.IllegalFieldValueException e1) {
                    // 当时间为 0000-00-00 00:00:00 的时候, 默认返回 1970-01-01
08:00:00.0
                    this.fieldValues.add(ByteUtil.getBytes(new Date(0L), false));

                } catch (ParseException e) {
                    LOGGER.error("error",e);
                }
                break;
            case Fields.FIELD_TYPE_TIME:
                String timeStr = ByteUtil.getTime(fv);
                Date timeVar = null;
                try {
                    if (timeStr.indexOf(".") > 0) {
```

```
                    timeVar           =           DateUtil.parseDate(timeStr,
DateUtil.TIME_PATTERN_FULL);
                    this.fieldValues.add(ByteUtil.getBytes(timeVar, true));
                } else {
                    timeVar           =           DateUtil.parseDate(timeStr,
DateUtil.DEFAULT_TIME_PATTERN);
                    this.fieldValues.add(ByteUtil.getBytes(timeVar, true));
                }

            } catch(org.joda.time.IllegalFieldValueException e1) {
                // 当时间为 0000-00-00 00:00:00 的时候, 默认返回 1970-01-01
08:00:00.0
                this.fieldValues.add(ByteUtil.getBytes(new Date(0L), true));

            } catch (ParseException e) {
                LOGGER.error("error",e);
            }
            break;
        }

    }

    public void write(FrontendConnection conn) {

        int size = calcPacketSize();
        int packetHeaderSize = conn.getPacketHeaderSize();
        int totalSize = size + packetHeaderSize;
        ByteBuffer bb = null;

        bb = conn.getProcessor().getBufferPool().allocate(totalSize);

        BufferUtil.writeUB3(bb, calcPacketSize());
        bb.put(packetId);
        bb.put(packetHeader); // packet header [00]
        bb.put(nullBitMap); // NULL-Bitmap
        for(int i = 0; i < fieldCount; i++) { // values
            byte[] fv = fieldValues.get(i);
            if(fv != null) {
                FieldPacket fieldPk = this.fieldPackets.get(i);
                int fieldType = fieldPk.type;
                switch(fieldType) {
                case Fields.FIELD_TYPE_STRING:
                case Fields.FIELD_TYPE_VARCHAR:
                case Fields.FIELD_TYPE_VAR_STRING:
                case Fields.FIELD_TYPE_ENUM:
                case Fields.FIELD_TYPE_SET:
                case Fields.FIELD_TYPE_LONG_BLOB:
```

```
                case Fields.FIELD_TYPE_MEDIUM_BLOB:
                case Fields.FIELD_TYPE_BLOB:
                case Fields.FIELD_TYPE_TINY_BLOB:
                case Fields.FIELD_TYPE_GEOMETRY:
                case Fields.FIELD_TYPE_BIT:
                case Fields.FIELD_TYPE_DECIMAL:
                case Fields.FIELD_TYPE_NEW_DECIMAL:
                    // 长度编码的字符串需要一个字节来存储长度(0 表示空字符串)
                    BufferUtil.writeLength(bb, fv.length);
                    break;
                    default:
                        break;
                }
                if(fv.length > 0) {
                    bb.put(fv);
                }
            }
        }
        conn.write(bb);

    }

    @Override
    public ByteBuffer write(ByteBuffer bb, FrontendConnection c,
            boolean writeSocketIfFull) {
        int size = calcPacketSize();
        int packetHeaderSize = c.getPacketHeaderSize();
        int totalSize = size + packetHeaderSize;
        bb = c.checkWriteBuffer(bb, totalSize, writeSocketIfFull);
        BufferUtil.writeUB3(bb, size);
        bb.put(packetId);
        bb.put(packetHeader); // packet header [00]
        bb.put(nullBitMap); // NULL-Bitmap
        for(int i = 0; i < fieldCount; i++) { // values
            byte[] fv = fieldValues.get(i);
            if(fv != null) {
                FieldPacket fieldPk = this.fieldPackets.get(i);
                int fieldType = fieldPk.type;
                switch(fieldType) {
                case Fields.FIELD_TYPE_STRING:
                case Fields.FIELD_TYPE_VARCHAR:
                case Fields.FIELD_TYPE_VAR_STRING:
                case Fields.FIELD_TYPE_ENUM:
                case Fields.FIELD_TYPE_SET:
                case Fields.FIELD_TYPE_LONG_BLOB:
                case Fields.FIELD_TYPE_MEDIUM_BLOB:
                case Fields.FIELD_TYPE_BLOB:
                    case Fields.FIELD_TYPE_TINY_BLOB:
                    case Fields.FIELD_TYPE_GEOMETRY:
                    case Fields.FIELD_TYPE_BIT:
                    case Fields.FIELD_TYPE_DECIMAL:
                    case Fields.FIELD_TYPE_NEW_DECIMAL:
                        // 长度编码的字符串需要一个字节来存储长度(0 表示空字符串)
                        BufferUtil.writeLength(bb, fv.length);
                        break;
                        default:
                            break;
                    }
                    if(fv.length > 0) {
                        bb.put(fv);
                    }
                }
            }
            return bb;
        }

        @Override
        public int calcPacketSize() {
            int size = 0;
            size = size + 1 + nullBitMap.length;
            for(int i = 0, n = fieldValues.size(); i < n; i++) {
                byte[] value = fieldValues.get(i);
                if(value != null) {
                    FieldPacket fieldPk = this.fieldPackets.get(i);
                    int fieldType = fieldPk.type;
                    switch(fieldType) {
                    case Fields.FIELD_TYPE_STRING:
                    case Fields.FIELD_TYPE_VARCHAR:
                    case Fields.FIELD_TYPE_VAR_STRING:
                    case Fields.FIELD_TYPE_ENUM:
                    case Fields.FIELD_TYPE_SET:
                    case Fields.FIELD_TYPE_LONG_BLOB:
                    case Fields.FIELD_TYPE_MEDIUM_BLOB:
                    case Fields.FIELD_TYPE_BLOB:
                    case Fields.FIELD_TYPE_TINY_BLOB:
                    case Fields.FIELD_TYPE_GEOMETRY:
                    case Fields.FIELD_TYPE_BIT:
                    case Fields.FIELD_TYPE_DECIMAL:
                    case Fields.FIELD_TYPE_NEW_DECIMAL:
                        /*
                         * 长度编码的字符串需要计算存储长度, 根据 mysql 协议文档描
述
                         * To convert a length-encoded integer into its numeric value,
check the first byte:
```

```
                         * If it is < 0xfb, treat it as a 1-byte integer.
                     * If it is 0xfc, it is followed by a 2-byte integer.
                     * If it is 0xfd, it is followed by a 3-byte integer.
                     * If it is 0xfe, it is followed by a 8-byte integer.
                         *
                         */
                        if(value.length != 0) {
                            /*
                             * 长度编码的字符串需要计算存储长度,不能简单默认只有 1
个字节是表示长度,当数据足够长,占用的就不止 1 个字节
                             */
//                          size = size + 1 + value.length;
                            size = size + BufferUtil.getLength(value);
                        } else {
                            size = size + 1; // 处理空字符串,只计算长度 1 个字节
                        }
                        break;
                        default:
                            size = size + value.length;
                            break;
                    }
                }
            }
            return size;
        }

        @Override
        protected String getPacketInfo() {
            return "MySQL Binary RowData Packet";
        }
}

21:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\CommandP
acket.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.io.IOException;
import java.io.OutputStream;
import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.backend.mysql.StreamUtil;
import io.mycat.net.BackendAIOConnection;

/**
 * From client to server whenever the client wants the server to do something.
 *
 * <pre>
 * Bytes         Name
 * -----         ----
 * 1             command
 * n             arg
 *
 * command:      The most common value is 03 COM_QUERY, because
 *               INSERT UPDATE DELETE SELECT etc. have this code.
 *               The possible values at time of writing (taken
 *               from /include/mysql_com.h for enum_server_command) are:
 *
 *               #      Name                Associated client function
 *               -      ----                --------------------------
 *               0x00   COM_SLEEP           (none, this is an internal thread state)
 *               0x01   COM_QUIT            mysql_close
 *               0x02   COM_INIT_DB         mysql_select_db
 *               0x03   COM_QUERY           mysql_real_query
 *               0x04   COM_FIELD_LIST      mysql_list_fields
 *               0x05   COM_CREATE_DB       mysql_create_db (deprecated)
 *               0x06   COM_DROP_DB         mysql_drop_db (deprecated)
 *               0x07   COM_REFRESH         mysql_refresh
 *               0x08   COM_SHUTDOWN        mysql_shutdown
 *               0x09   COM_STATISTICS      mysql_stat
 *              0x0a   COM_PROCESS_INFO    mysql_list_processes
 *              0x0b   COM_CONNECT         (none, this is an internal thread state)
 *              0x0c   COM_PROCESS_KILL    mysql_kill
 *              0x0d   COM_DEBUG           mysql_dump_debug_info
 *              0x0e   COM_PING            mysql_ping
 *              0x0f   COM_TIME            (none, this is an internal thread state)
 *              0x10   COM_DELAYED_INSERT  (none, this is an internal thread state)
 *              0x11   COM_CHANGE_USER     mysql_change_user
 *              0x12   COM_BINLOG_DUMP     sent by the slave IO thread to request a
binlog
 *              0x13   COM_TABLE_DUMP      LOAD TABLE ... FROM MASTER (deprecated)
 *              0x14   COM_CONNECT_OUT     (none, this is an internal thread state)
 *              0x15   COM_REGISTER_SLAVE  sent by the slave to register with the
master (optional)
 *              0x16   COM_STMT_PREPARE    mysql_stmt_prepare
 *              0x17   COM_STMT_EXECUTE    mysql_stmt_execute
 *              0x18   COM_STMT_SEND_LONG_DATA mysql_stmt_send_long_data
 *              0x19   COM_STMT_CLOSE      mysql_stmt_close
 *              0x1a   COM_STMT_RESET      mysql_stmt_reset
 *              0x1b   COM_SET_OPTION      mysql_set_server_option
 *              0x1c   COM_STMT_FETCH      mysql_stmt_fetch
 * 
 * arg:         The text of the command is just the way the user typed it, there is
no processing
 *              by the client (except removal of the final ';').
 *              This field is not a null-terminated string; however,
 *              the size can be calculated from the packet size,
 *              and the MySQL client appends '\0' when receiving.
 * 
 *                                                                          @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Command_Packet_.
28Overview.29
 * </pre>
 * 
 * @author mycat
 */
public class CommandPacket extends MySQLPacket {

    public byte command;
    public byte[] arg;

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        command = mm.read();
        arg = mm.readBytes();
    }



    public void write(OutputStream out) throws IOException {
        StreamUtil.writeUB3(out, calcPacketSize());
        StreamUtil.write(out, packetId);
        StreamUtil.write(out, command);
        out.write(arg);
    }

    @Override
    public void write(BackendAIOConnection c) {
        ByteBuffer buffer = c.allocate();
        try {
             BufferUtil.writeUB3(buffer, calcPacketSize());
             buffer.put(packetId);
             buffer.put(command);
             buffer = c.writeToBuffer(arg, buffer);
             c.write(buffer);
        } catch(java.nio.BufferOverflowException e1) {
          //fixed issues #98 #1072
          buffer   =      c.checkWriteBuffer(buffer,   c.getPacketHeaderSize()   +
calcPacketSize(), false);
             BufferUtil.writeUB3(buffer, calcPacketSize());
             buffer.put(packetId);
             buffer.put(command);
             buffer = c.writeToBuffer(arg, buffer);
             c.write(buffer);
        }
    }

    @Override
    public int calcPacketSize() {
        return 1 + arg.length;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Command Packet";
    }


}
```

22:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\EmptyPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

/**
 * @author mycat 暂时只发现在 load data infile 时用到
 */
public class EmptyPacket extends MySQLPacket {
    public static final byte[] EMPTY = new byte[] { 0, 0, 0,3 };

    @Override
    public int calcPacketSize() {
        return 0;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Empty Packet";
    }

}
```

23:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\EOFPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.buffer.BufferArray;
import io.mycat.net.FrontendConnection;

/**
 * From Server To Client, at the end of a series of Field Packets, and at the
 * end of a series of Data Packets.With prepared statements, EOF Packet can also
 * end parameter information, which we'll describe later.
 *
 * <pre>
 * Bytes                 Name
 * -----                 ----
 * 1                     field_count, always = 0xfe
 * 2                     warning_count
 * 2                     Status Flags
 *
 *                                                                           @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#EOF_Packet
 * </pre>
 *
 * @author mycat
 */
public class EOFPacket extends MySQLPacket {
    public static final byte FIELD_COUNT = (byte) 0xfe;

    public byte fieldCount = FIELD_COUNT;
    public int warningCount;
    public int status = 2;

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        fieldCount = mm.read();
        warningCount = mm.readUB2();
        status = mm.readUB2();
    }

    @Override
    public  ByteBuffer  write(ByteBuffer  buffer,  FrontendConnection  c,boolean
writeSocketIfFull) {
        int size = calcPacketSize();
        buffer    =    c.checkWriteBuffer(buffer,    c.getPacketHeaderSize()    +
size,writeSocketIfFull);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        buffer.put(fieldCount);
        BufferUtil.writeUB2(buffer, warningCount);
        BufferUtil.writeUB2(buffer, status);
        return buffer;
    }

    @Override
    public int calcPacketSize() {
        return 5;// 1+2+2;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL EOF Packet";
    }

     public void write(BufferArray bufferArray) {
          int size = calcPacketSize();
          ByteBuffer buffer = bufferArray.checkWriteBuffer(packetHeaderSize
                    + size);
          BufferUtil.writeUB3(buffer, size);
          buffer.put(packetId);
          buffer.put(fieldCount);
          BufferUtil.writeUB2(buffer, warningCount);
          BufferUtil.writeUB2(buffer, status);
     }

}
```

24:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\ErrorPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.io.ByteArrayOutputStream;
import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.net.FrontendConnection;

/**
 * From server to client in response to command, if error.
 *
 * <pre>
 * Bytes                       Name
 * -----                       ----
 * 1                            field_count, always = 0xff
 * 2                            errno
 * 1                            (sqlstate marker), always '#'
 * 5                            sqlstate (5 characters)
 * n                            message
 *
 *                                                                               @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Error_Packet
 * </pre>
 *
 * @author mycat
 */
public class ErrorPacket extends MySQLPacket {
    public static final byte FIELD_COUNT = (byte) 0xff;
    private static final byte SQLSTATE_MARKER = (byte) '#';
    private static final byte[] DEFAULT_SQLSTATE = "HY000".getBytes();

    public byte fieldCount = FIELD_COUNT;
    public int errno;
    public byte mark = SQLSTATE_MARKER;
    public byte[] sqlState = DEFAULT_SQLSTATE;
    public byte[] message;

    public void read(BinaryPacket bin) {
        packetLength = bin.packetLength;
        packetId = bin.packetId;
        MySQLMessage mm = new MySQLMessage(bin.data);
        fieldCount = mm.read();
        errno = mm.readUB2();
        if (mm.hasRemaining() && (mm.read(mm.position()) == SQLSTATE_MARKER)) {
            mm.read();
            sqlState = mm.readBytes(5);
        }
        message = mm.readBytes();
    }

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        fieldCount = mm.read();
        errno = mm.readUB2();
        if (mm.hasRemaining() && (mm.read(mm.position()) == SQLSTATE_MARKER)) {
            mm.read();
            sqlState = mm.readBytes(5);
        }
        message = mm.readBytes();
    }

    public byte[] writeToBytes(FrontendConnection c) {
        ByteBuffer buffer = c.allocate();
        buffer = write(buffer, c, false);
        buffer.flip();
        byte[] data = new byte[buffer.limit()];
        buffer.get(data);
        c.recycle(buffer);
        return data;
    }
    public byte[] writeToBytes() {
        ByteBuffer buffer = ByteBuffer.allocate(calcPacketSize()+4);
        int size = calcPacketSize();
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        buffer.put(fieldCount);
        BufferUtil.writeUB2(buffer, errno);
        buffer.put(mark);
        buffer.put(sqlState);
        if (message != null) {
            buffer.put(message);
        }
        buffer.flip();
        byte[] data = new byte[buffer.limit()];
        buffer.get(data);

        return data;
    }
    @Override
    public ByteBuffer write(ByteBuffer buffer, FrontendConnection c,
            boolean writeSocketIfFull) {
        int size = calcPacketSize();
        buffer = c.checkWriteBuffer(buffer, c.getPacketHeaderSize() + size,
                  writeSocketIfFull);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        buffer.put(fieldCount);
        BufferUtil.writeUB2(buffer, errno);
        buffer.put(mark);
        buffer.put(sqlState);
        if (message != null) {
            buffer = c.writeToBuffer(message, buffer);
        }
        return buffer;
    }
```

```
    public void write(FrontendConnection c) {
        ByteBuffer buffer = c.allocate();
        buffer = this.write(buffer, c, true);
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        int size = 9;// 1 + 2 + 1 + 5
        if (message != null) {
            size += message.length;
        }
        return size;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Error Packet";
    }

}
```

25:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\ExecutePacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.io.UnsupportedEncodingException;

import io.mycat.backend.mysql.BindValue;
import io.mycat.backend.mysql.BindValueUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.backend.mysql.PreparedStatement;

/**
 * <pre>
 *  Bytes                      Name
 *  -----                      ----
 *  1                          code
 *  4                          statement_id
 *  1                          flags
 *  4                          iteration_count
 *  (param_count+7)/8          null_bit_map
 *  1                          new_parameter_bound_flag (if new_params_bound == 1:)
 *  n*2                        type of parameters
 *  n                          values for the parameters
 *
--------------------------------------------------------------------------------
 *  code:                      always COM_EXECUTE
 *
 *  statement_id:              statement identifier
 *
 *  flags:                     reserved for future use. In MySQL 4.0, always 0.
 *                             In MySQL 5.0:
 *                               0: CURSOR_TYPE_NO_CURSOR
 *                               1: CURSOR_TYPE_READ_ONLY
 *                               2: CURSOR_TYPE_FOR_UPDATE
 *                               4: CURSOR_TYPE_SCROLLABLE
 *
 *  iteration_count:           reserved for future use. Currently always 1.
 *
 *  null_bit_map:              A bitmap indicating parameters that are NULL.
 *                             Bits are counted from LSB, using as many bytes
 *                             as necessary ((param_count+7)/8)
 *                             i.e. if the first parameter (parameter 0) is NULL, then
 *                             the least significant bit in the first byte will be 1.
 *
 *  new_parameter_bound_flag:  Contains 1 if this is the first time
 *                             that "execute" has been called, or if
 *                             the parameters have been rebound.
 *
 *  type:                        Occurs once for each parameter;
 *                               The highest significant bit of this 16-bit value
 *                               encodes the unsigned property. The other 15 bits
 *                               are reserved for the type (only 8 currently used).
 *                               This block is sent when parameters have been rebound
 *                               or when a prepared statement is executed for the
 *                               first time.
 *
 *  values:                      for all non-NULL values, each parameters appends its
value
 *                               as described in Row Data Packet: Binary (column values)
 * @see https://dev.mysql.com/doc/internals/en/com-stmt-execute.html
 * </pre>
 *
 * @author mycat, CrazyPig
 */
public class ExecutePacket extends MySQLPacket {

    public byte code;
    public long statementId;
    public byte flags;
    public long iterationCount;
    public byte[] nullBitMap;
    public byte newParameterBoundFlag;
    public BindValue[] values;
    protected PreparedStatement pstmt;

    public ExecutePacket(PreparedStatement pstmt) {
        this.pstmt = pstmt;
        this.values = new BindValue[pstmt.getParametersNumber()];
    }

    public void read(byte[] data, String charset) throws UnsupportedEncodingException
{
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        code = mm.read();
        statementId = mm.readUB4();
        flags = mm.read();
        iterationCount = mm.readUB4();

        // 读取 NULL 指示器数据
        int parameterCount = values.length;
        if(parameterCount > 0) {
             nullBitMap = new byte[(parameterCount + 7) / 8];
```

```
                for (int i = 0; i < nullBitMap.length; i++) {
                    nullBitMap[i] = mm.read();
                }

                // 当 newParameterBoundFlag==1 时，更新参数类型。
                newParameterBoundFlag = mm.read();
            }
            if (newParameterBoundFlag == (byte) 1) {
                for (int i = 0; i < parameterCount; i++) {
                    pstmt.getParametersType()[i] = mm.readUB2();
                }
            }

            // 设置参数类型和读取参数值
            byte[] nullBitMap = this.nullBitMap;
            for (int i = 0; i < parameterCount; i++) {
                BindValue bv = new BindValue();
                bv.type = pstmt.getParametersType()[i];
                if ((nullBitMap[i / 8] & (1 << (i & 7))) != 0) {
                    bv.isNull = true;
                } else {
                    BindValueUtil.read(mm, bv, charset);
                    if(bv.isLongData) {
                       bv.value = pstmt.getLongData(i);
                    }
                }
                values[i] = bv;
            }
        }

        @Override
        public int calcPacketSize() {

            return 0;
        }

        @Override
        protected String getPacketInfo() {
            return "MySQL Execute Packet";
        }

    }
```

26:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\FieldPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.buffer.BufferArray;
import io.mycat.net.FrontendConnection;

/**
 * From Server To Client, part of Result Set Packets. One for each column in the
 * result set. Thus, if the value of field_columns in the Result Set Header
 * Packet is 3, then the Field Packet occurs 3 times.
 *
 * <pre>
 * Bytes                      Name
 * -----                      ----
 * n (Length Coded String)    catalog
 * n (Length Coded String)    db
 * n (Length Coded String)    table
 * n (Length Coded String)    org_table
 * n (Length Coded String)    name
 * n (Length Coded String)    org_name
 * 1                          (filler)
 * 2                          charsetNumber
 * 4                          length
 * 1                          type
 * 2                          flags
 * 1                          decimals
 * 2                          (filler), always 0x00
 * n (Length Coded Binary)    default
 *
 *                                                                             @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Field_Packet
 * </pre>
 *
 * @author mycat
 */
public class FieldPacket extends MySQLPacket {
     private static final byte[] DEFAULT_CATALOG = "def".getBytes();
     private static final byte[] FILLER = new byte[2];

     public byte[] catalog = DEFAULT_CATALOG;
     public byte[] db;
     public byte[] table;
     public byte[] orgTable;
     public byte[] name;
     public byte[] orgName;
     public int charsetIndex;
     public long length;
     public int type;
     public int flags;
     public byte decimals;
     public byte[] definition;

     /**
      * 把字节数组转变成 FieldPacket
      */
     public void read(byte[] data) {
          MySQLMessage mm = new MySQLMessage(data);
          this.packetLength = mm.readUB3();
          this.packetId = mm.read();
          readBody(mm);
     }

     /**
      * 把 BinaryPacket 转变成 FieldPacket
      */
     public void read(BinaryPacket bin) {
          this.packetLength = bin.packetLength;
          this.packetId = bin.packetId;
          readBody(new MySQLMessage(bin.data));
     }
```

```
@Override
public ByteBuffer write(ByteBuffer buffer, FrontendConnection c,
        boolean writeSocketIfFull) {
    int size = calcPacketSize();
    buffer = c.checkWriteBuffer(buffer, c.getPacketHeaderSize() + size,
            writeSocketIfFull);
    BufferUtil.writeUB3(buffer, size);
    buffer.put(packetId);
    writeBody(buffer);
    return buffer;
}

@Override
public int calcPacketSize() {
    int size = (catalog == null ? 1 : BufferUtil.getLength(catalog));
    size += (db == null ? 1 : BufferUtil.getLength(db));
    size += (table == null ? 1 : BufferUtil.getLength(table));
    size += (orgTable == null ? 1 : BufferUtil.getLength(orgTable));
    size += (name == null ? 1 : BufferUtil.getLength(name));
    size += (orgName == null ? 1 : BufferUtil.getLength(orgName));
    size += 13;// 1+2+4+1+2+1+2
    if (definition != null) {
        size += BufferUtil.getLength(definition);
    }
    return size;
}

@Override
protected String getPacketInfo() {
    return "MySQL Field Packet";
}

private void readBody(MySQLMessage mm) {
    this.catalog = mm.readBytesWithLength();
    this.db = mm.readBytesWithLength();
    this.table = mm.readBytesWithLength();
    this.orgTable = mm.readBytesWithLength();
    this.name = mm.readBytesWithLength();
    this.orgName = mm.readBytesWithLength();
    mm.move(1);
    this.charsetIndex = mm.readUB2();
    this.length = mm.readUB4();
    this.type = mm.read() & 0xff;
    this.flags = mm.readUB2();
    this.decimals = mm.read();
    mm.move(FILLER.length);
```

```
        if (mm.hasRemaining()) {
            this.definition = mm.readBytesWithLength();
        }
    }

    private void writeBody(ByteBuffer buffer) {
        byte nullVal = 0;
        BufferUtil.writeWithLength(buffer, catalog, nullVal);
        BufferUtil.writeWithLength(buffer, db, nullVal);
        BufferUtil.writeWithLength(buffer, table, nullVal);
        BufferUtil.writeWithLength(buffer, orgTable, nullVal);
        BufferUtil.writeWithLength(buffer, name, nullVal);
        BufferUtil.writeWithLength(buffer, orgName, nullVal);
        buffer.put((byte) 0x0C);
        BufferUtil.writeUB2(buffer, charsetIndex);
        BufferUtil.writeUB4(buffer, length);
        buffer.put((byte) (type & 0xff));
        BufferUtil.writeUB2(buffer, flags);
        buffer.put(decimals);
       buffer.put((byte)0x00);
       buffer.put((byte)0x00);
        //buffer.position(buffer.position() + FILLER.length);
        if (definition != null) {
            BufferUtil.writeWithLength(buffer, definition);
        }
    }

    public  void write(BufferArray bufferArray) {
        int size = calcPacketSize();
        ByteBuffer buffer = bufferArray.checkWriteBuffer(packetHeaderSize + size);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        writeBody(buffer);
    }

}
```

27:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\HandshakePacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.net.FrontendConnection;

/**
 * From server to client during initial handshake.
 *
 * <pre>
 * Bytes                        Name
 * -----                        ----
 * 1                            protocol_version
 * n (Null-Terminated String)   server_version
 * 4                            thread_id
 * 8                            scramble_buff
 * 1                            (filler) always 0x00
 * 2                            server_capabilities
 * 1                            server_language
 * 2                            server_status
 * 13                           (filler) always 0x00 ...
 * 13                           rest of scramble_buff (4.1)
 *
 *                                                                              @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Handshake_Initia
lization_Packet
 * </pre>
 *
 * @author mycat
 */
```

```
public class HandshakePacket extends MySQLPacket {
    private static final byte[] FILLER_13 = new byte[] { 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0 };

    public byte protocolVersion;
    public byte[] serverVersion;
    public long threadId;
    public byte[] seed;
    public int serverCapabilities;
    public byte serverCharsetIndex;
    public int serverStatus;
    public byte[] restOfScrambleBuff;

    public void read(BinaryPacket bin) {
        packetLength = bin.packetLength;
        packetId = bin.packetId;
        MySQLMessage mm = new MySQLMessage(bin.data);
        protocolVersion = mm.read();
        serverVersion = mm.readBytesWithNull();
        threadId = mm.readUB4();
        seed = mm.readBytesWithNull();
        serverCapabilities = mm.readUB2();
        serverCharsetIndex = mm.read();
        serverStatus = mm.readUB2();
        mm.move(13);
        restOfScrambleBuff = mm.readBytesWithNull();
    }

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        protocolVersion = mm.read();
        serverVersion = mm.readBytesWithNull();
        threadId = mm.readUB4();
        seed = mm.readBytesWithNull();
        serverCapabilities = mm.readUB2();
        serverCharsetIndex = mm.read();
        serverStatus = mm.readUB2();
        mm.move(13);
        restOfScrambleBuff = mm.readBytesWithNull();
    }

    public void write(FrontendConnection c) {
        ByteBuffer buffer = c.allocate();
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer.put(protocolVersion);
        BufferUtil.writeWithNull(buffer, serverVersion);
        BufferUtil.writeUB4(buffer, threadId);
        BufferUtil.writeWithNull(buffer, seed);
        BufferUtil.writeUB2(buffer, serverCapabilities);
        buffer.put(serverCharsetIndex);
        BufferUtil.writeUB2(buffer, serverStatus);
        buffer.put(FILLER_13);
        //        buffer.position(buffer.position() + 13);
        BufferUtil.writeWithNull(buffer, restOfScrambleBuff);
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        int size = 1;
        size += serverVersion.length;// n
        size += 5;// 1+4
        size += seed.length;// 8
        size += 19;// 1+2+1+2+13
        size += restOfScrambleBuff.length;// 12
        size += 1;// 1
        return size;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Handshake Packet";
    }

}
```

28:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\HandshakeV10Packet.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.config.Capabilities;
import io.mycat.backend.mysql.BufferUtil;
import io.mycat.net.FrontendConnection;

/**
 * From mycat server to client during initial handshake.
 *
 * <pre>
 * Bytes                        Name
 * -----                        ----
 * 1                            protocol_version (always 0x0a)
 * n (string[NULL])             server_version
 * 4                            thread_id
 * 8 (string[8])                auth-plugin-data-part-1
 * 1                            (filler) always 0x00
 * 2                            capability flags (lower 2 bytes)
 *   if more data in the packet:
 * 1                            character set
 * 2                            status flags
 * 2                            capability flags (upper 2 bytes)
 *   if capabilities & CLIENT_PLUGIN_AUTH {
 * 1                            length of auth-plugin-data
 *   } else {
 * 1                            0x00
 *   }
 * 10 (string[10])              reserved (all 0x00)
 *   if capabilities & CLIENT_SECURE_CONNECTION {
 * string[$len]   auth-plugin-data-part-2 ($len=MAX(13, length of auth-plugin-data -
8))
 *   }
 *   if capabilities & CLIENT_PLUGIN_AUTH {
 * string[NUL]    auth-plugin name
 * }
 *
 *                                                                              @see
http://dev.mysql.com/doc/internals/en/connection-phase-packets.html#Protocol::Hand
shakeV10
 * </pre>
 *
 * @author CrazyPig
 * @since 2016-11-13
 *
 */
public class HandshakeV10Packet extends MySQLPacket {
    private static final byte[] FILLER_10 = new byte[] { 0, 0, 0, 0, 0, 0, 0, 0, 0,
0 };
    private      static      final      byte[]      DEFAULT_AUTH_PLUGIN_NAME      =
"mysql_native_password".getBytes();

    public byte protocolVersion;
    public byte[] serverVersion;
    public long threadId;
    public byte[] seed; // auth-plugin-data-part-1
    public int serverCapabilities;
    public byte serverCharsetIndex;
    public int serverStatus;
    public byte[] restOfScrambleBuff; // auth-plugin-data-part-2
    public byte[] authPluginName = DEFAULT_AUTH_PLUGIN_NAME;

    public void write(FrontendConnection c) {

     ByteBuffer buffer = c.allocate();
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer.put(protocolVersion);
        BufferUtil.writeWithNull(buffer, serverVersion);
        BufferUtil.writeUB4(buffer, threadId);
        buffer.put(seed);
        buffer.put((byte)0); // [00] filler
        BufferUtil.writeUB2(buffer, serverCapabilities); // capability flags (lower
2 bytes)
        buffer.put(serverCharsetIndex);
        BufferUtil.writeUB2(buffer, serverStatus);
        BufferUtil.writeUB2(buffer, (serverCapabilities >> 16)); // capability flags
(upper 2 bytes)
        if((serverCapabilities & Capabilities.CLIENT_PLUGIN_AUTH) != 0) {
          if(restOfScrambleBuff.length <= 13) {
              buffer.put((byte) (seed.length + 13));
          } else {
              buffer.put((byte) (seed.length + restOfScrambleBuff.length));
```

```
            }
        } else {
            buffer.put((byte) 0);
        }
        buffer.put(FILLER_10);
        if((serverCapabilities & Capabilities.CLIENT_SECURE_CONNECTION) != 0) {
            buffer.put(restOfScrambleBuff);
            // restOfScrambleBuff.length always to be 12
            if(restOfScrambleBuff.length < 13) {
                for(int i = 13 - restOfScrambleBuff.length; i > 0; i--) {
                    buffer.put((byte)0);
                }
            }
        }
        if((serverCapabilities & Capabilities.CLIENT_PLUGIN_AUTH) != 0) {
            BufferUtil.writeWithNull(buffer, authPluginName);
        }
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        int size = 1; // protocol version
        size += (serverVersion.length + 1); // server version
        size += 4; // connection id
        size += seed.length;
        size += 1; // [00] filler
        size += 2; // capability flags (lower 2 bytes)
        size += 1; // character set
        size += 2; // status flags
        size += 2; // capability flags (upper 2 bytes)
        size += 1;
        size += 10; // reserved (all [00])
        if((serverCapabilities & Capabilities.CLIENT_SECURE_CONNECTION) != 0) {
            // restOfScrambleBuff.length always to be 12
            if(restOfScrambleBuff.length <= 13) {
                size += 13;
            } else {
                size += restOfScrambleBuff.length;
            }
        }
        if((serverCapabilities & Capabilities.CLIENT_PLUGIN_AUTH) != 0) {
            size += (authPluginName.length + 1); // auth-plugin name
        }
        return size;
    }
    @Override
    protected String getPacketInfo() {
        return "MySQL HandshakeV10 Packet";
    }

}
```

29:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\HeartbeatPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.net.BackendAIOConnection;

/**
 * From client to server when the client do heartbeat between mycat cluster.
 *
 * <pre>
 * Bytes         Name
 * -----         ----
 * 1             command
 * n             id
 *
 * @author mycat
 */
public class HeartbeatPacket extends MySQLPacket {

    public byte command;
    public long id;

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        command = mm.read();
        id = mm.readLength();
    }

    @Override
    public void write(BackendAIOConnection c) {
        ByteBuffer buffer = c.allocate();
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer.put(command);
        BufferUtil.writeLength(buffer, id);
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        return 1 + BufferUtil.getLength(id);
    }

    @Override
    protected String getPacketInfo() {
        return "Mycat Heartbeat Packet";
    }

}
```

30:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\LongDataPacket.java

```
package io.mycat.net.mysql;

import io.mycat.backend.mysql.MySQLMessage;

/**
 *
 * <pre>
 *
 * COM_STMT_SEND_LONG_DATA sends the data for a column. Repeating to send it, appends
the data to the parameter.
 * No response is sent back to the client.

 * COM_STMT_SEND_LONG_DATA:
 * COM_STMT_SEND_LONG_DATA
 * direction: client -> server
 * response: none

 * payload:
 *   1              [18] COM_STMT_SEND_LONG_DATA
 *   4              statement-id
 *   2              param-id
 *   n              data
 *
 * </pre>
 *
 * @see https://dev.mysql.com/doc/internals/en/com-stmt-send-long-data.html
 *
 * @author CrazyPig
 * @since 2016-09-08
 *
 */
public class LongDataPacket extends MySQLPacket {

     private static final byte PACKET_FALG = (byte) 24;
     private long pstmtId;
     private long paramId;
     private byte[] longData = new byte[0];

     public void read(byte[] data) {
          MySQLMessage mm = new MySQLMessage(data);
          packetLength = mm.readUB3();
          packetId = mm.read();
          byte code = mm.read();
          assert code == PACKET_FALG;
          pstmtId = mm.readUB4();
          paramId = mm.readUB2();
          this.longData = mm.readBytes(packetLength - (1 + 4 + 2));
     }

     @Override
     public int calcPacketSize() {
          return 1 + 4 + 2 + this.longData.length;
     }
```

```
    @Override
    protected String getPacketInfo() {
        return "MySQL Long Data Packet";
    }

    public long getPstmtId() {
        return pstmtId;
    }

    public long getParamId() {
        return paramId;
    }

    public byte[] getLongData() {
        return longData;
    }


}
```

31:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\MySQLPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;
```

```
import java.nio.ByteBuffer;

import io.mycat.net.BackendAIOConnection;
import io.mycat.net.FrontendConnection;

/**
 * @author mycat
 */
public abstract class MySQLPacket {
    /**
     * none, this is an internal thread state
     */
    public static final byte COM_SLEEP = 0;

    /**
     * mysql_close
     */
    public static final byte COM_QUIT = 1;

    /**
     * mysql_select_db
     */
    public static final byte COM_INIT_DB = 2;

    /**
     * mysql_real_query
     */
    public static final byte COM_QUERY = 3;

    /**
     * mysql_list_fields
     */
    public static final byte COM_FIELD_LIST = 4;

    /**
     * mysql_create_db (deprecated)
     */
    public static final byte COM_CREATE_DB = 5;

    /**
     * mysql_drop_db (deprecated)
     */
    public static final byte COM_DROP_DB = 6;

    /**
     * mysql_refresh
     */
    public static final byte COM_REFRESH = 7;

    /**
     * mysql_shutdown
     */
    public static final byte COM_SHUTDOWN = 8;

    /**
     * mysql_stat
     */
    public static final byte COM_STATISTICS = 9;

    /**
     * mysql_list_processes
     */
    public static final byte COM_PROCESS_INFO = 10;

    /**
     * none, this is an internal thread state
     */
    public static final byte COM_CONNECT = 11;

    /**
     * mysql_kill
     */
    public static final byte COM_PROCESS_KILL = 12;

    /**
     * mysql_dump_debug_info
     */
    public static final byte COM_DEBUG = 13;

    /**
     * mysql_ping
     */
    public static final byte COM_PING = 14;

    /**
     * none, this is an internal thread state
     */
    public static final byte COM_TIME = 15;

    /**
     * none, this is an internal thread state
     */
    public static final byte COM_DELAYED_INSERT = 16;
```

```
/**
 * mysql_change_user
 */
public static final byte COM_CHANGE_USER = 17;

/**
 * used by slave server mysqlbinlog
 */
public static final byte COM_BINLOG_DUMP = 18;

/**
 * used by slave server to get master table
 */
public static final byte COM_TABLE_DUMP = 19;

/**
 * used by slave to log connection to master
 */
public static final byte COM_CONNECT_OUT = 20;

/**
 * used by slave to register to master
 */
public static final byte COM_REGISTER_SLAVE = 21;

/**
 * mysql_stmt_prepare
 */
public static final byte COM_STMT_PREPARE = 22;

/**
 * mysql_stmt_execute
 */
public static final byte COM_STMT_EXECUTE = 23;

/**
 * mysql_stmt_send_long_data
 */
public static final byte COM_STMT_SEND_LONG_DATA = 24;

/**
 * mysql_stmt_close
 */
public static final byte COM_STMT_CLOSE = 25;

/**
```

```
     * mysql_stmt_reset
     */
    public static final byte COM_STMT_RESET = 26;

    /**
     * mysql_set_server_option
     */
    public static final byte COM_SET_OPTION = 27;

    /**
     * mysql_stmt_fetch
     */
    public static final byte COM_STMT_FETCH = 28;

    /**
     * Mycat heartbeat
     */
    public static final byte COM_HEARTBEAT = 64;

    //包头大小
    public static final int packetHeaderSize = 4;


    public int packetLength;
    public byte packetId;

    /**
     * 把数据包写到 buffer 中，如果 buffer 满了就把 buffer 通过前端连接写出
(writeSocketIfFull=true)。
     */
    public  ByteBuffer  write(ByteBuffer  buffer,  FrontendConnection  c,boolean
writeSocketIfFull) {
        throw new UnsupportedOperationException();
    }

    /**
     * 把数据包通过后端连接写出，一般使用 buffer 机制来提高写的吞吐量。
     */
    public void write(BackendAIOConnection c) {
        throw new UnsupportedOperationException();
    }

    /**
     * 计算数据包大小，不包含包头长度。
     */
    public abstract int calcPacketSize();
    /**
     * 取得数据包信息
     */
    protected abstract String getPacketInfo();

    @Override
    public String toString() {
        return                                                                      new
StringBuilder().append(getPacketInfo()).append("{length=").append(packetLength).ap
pend(",id=")
                .append(packetId).append('}').toString();
    }

}
```

32:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\OkPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.MycatServer;
import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.net.FrontendConnection;

/**
 * From server to client in response to command, if no error and no result set.
 *
 * <pre>
 * Bytes                        Name
 * -----                        ----
 * 1                            field_count, always = 0
 * 1-9 (Length Coded Binary)    affected_rows
 * 1-9 (Length Coded Binary)    insert_id
 * 2                            server_status
 * 2                            warning_count
 * n   (until end of packet)    message fix:(Length Coded String)
 *
 * @see http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#OK_Packet
 * </pre>
 *
 * @author mycat
 */
public class OkPacket extends MySQLPacket {
    public static final byte FIELD_COUNT = 0x00;
    public static final byte[] OK = new byte[] { 7, 0, 0, 1, 0, 0, 0, 2, 0, 0,
            0 };

    public byte fieldCount = FIELD_COUNT;
    public long affectedRows;
    public long insertId;
    public int serverStatus;
    public int warningCount;
    public byte[] message;

    public void read(BinaryPacket bin) {
        packetLength = bin.packetLength;
        packetId = bin.packetId;
        MySQLMessage mm = new MySQLMessage(bin.data);
        fieldCount = mm.read();
        affectedRows = mm.readLength();
        insertId = mm.readLength();
        serverStatus = mm.readUB2();
        warningCount = mm.readUB2();
        if (mm.hasRemaining()) {
            this.message = mm.readBytesWithLength();
        }
    }

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        fieldCount = mm.read();
        affectedRows = mm.readLength();
        insertId = mm.readLength();
        serverStatus = mm.readUB2();
        warningCount = mm.readUB2();
        if (mm.hasRemaining()) {
            this.message = mm.readBytesWithLength();
        }
    }

    public byte[] writeToBytes(FrontendConnection c) {
        ByteBuffer buffer = c.allocate();
        this.write(buffer, c);
        buffer.flip();
        byte[] data = new byte[buffer.limit()];
        buffer.get(data);
        c.recycle(buffer);
        return data;
    }

    private ByteBuffer write(ByteBuffer buffer, FrontendConnection c) {

        int size = calcPacketSize();
        buffer = c.checkWriteBuffer(buffer, c.getPacketHeaderSize() + size,
                true);
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer.put(fieldCount);
        BufferUtil.writeLength(buffer, affectedRows);
        BufferUtil.writeLength(buffer, insertId);
        BufferUtil.writeUB2(buffer, serverStatus);
        BufferUtil.writeUB2(buffer, warningCount);
        if (message != null) {
            BufferUtil.writeWithLength(buffer, message);
        }

        return buffer;

    }

    public void write(FrontendConnection c) {
        ByteBuffer buffer = write(c.allocate(), c);
        c.write(buffer);
    }
    @Override
    public int calcPacketSize() {
        int i = 1;
        i += BufferUtil.getLength(affectedRows);
        i += BufferUtil.getLength(insertId);
        i += 4;
        if (message != null) {
            i += BufferUtil.getLength(message);
        }
        return i;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL OK Packet";
    }

     public byte[] writeToBytes() {

        int totalSize = calcPacketSize() + packetHeaderSize;
        ByteBuffer
buffer=MycatServer.getInstance().getBufferPool().allocate(totalSize);
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        buffer.put(fieldCount);
        BufferUtil.writeLength(buffer, affectedRows);
        BufferUtil.writeLength(buffer, insertId);
        BufferUtil.writeUB2(buffer, serverStatus);
        BufferUtil.writeUB2(buffer, warningCount);
        if (message != null) {
            BufferUtil.writeWithLength(buffer, message);
        }
        buffer.flip();
        byte[] data = new byte[buffer.limit()];
        buffer.get(data);
          MycatServer.getInstance().getBufferPool().recycle(buffer);
          return data;
    }

}
```

33:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\PingPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

/**
 * @author mycat
 */
public class PingPacket extends MySQLPacket {
    public static final byte[] PING = new byte[] { 1, 0, 0, 0, 14 };

    @Override
    public int calcPacketSize() {
        return 1;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Ping Packet";
    }

}
```

34:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\PreparedOkPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.net.FrontendConnection;

/**
 * <pre>
 * From server to client, in response to prepared statement initialization packet.
 * It is made up of:
 *   1.a PREPARE_OK packet
 *   2.if "number of parameters" > 0
 *       (field packets) as in a Result Set Header Packet
 *       (EOF packet)
 *   3.if "number of columns" > 0
 *       (field packets) as in a Result Set Header Packet
 *       (EOF packet)
 *
 *
---------------------------------------------------------------------------------
-------
 *
 *  Bytes              Name
 *  -----              ----
 *  1                  0 - marker for OK packet
 *  4                  statement_handler_id
 *  2                  number of columns in result set
 *  2                  number of parameters in query
 *  1                  filler (always 0)
 *  2                  warning count
 *
 *                                                                              @see
http://dev.mysql.com/doc/internals/en/prepared-statement-initialization-packet.htm
l
 * </pre>
 *
 * @author mycat
 */
public class PreparedOkPacket extends MySQLPacket {

    public byte flag;
    public long statementId;
    public int columnsNumber;
    public int parametersNumber;
    public byte filler;
    public int warningCount;

    public PreparedOkPacket() {
        this.flag = 0;
        this.filler = 0;
    }

    @Override
    public   ByteBuffer   write(ByteBuffer   buffer,   FrontendConnection   c,boolean
writeSocketIfFull) {
        int size = calcPacketSize();
        buffer    =    c.checkWriteBuffer(buffer,     c.getPacketHeaderSize()     +
size,writeSocketIfFull);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        buffer.put(flag);
        BufferUtil.writeUB4(buffer, statementId);
        BufferUtil.writeUB2(buffer, columnsNumber);
        BufferUtil.writeUB2(buffer, parametersNumber);
        buffer.put(filler);
        BufferUtil.writeUB2(buffer, warningCount);
        return buffer;
    }

    @Override
    public int calcPacketSize() {
        return 12;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL PreparedOk Packet";
    }

}
```

35:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\QuitPack
et.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

/**
 * @author mycat
 */
public class QuitPacket extends MySQLPacket {
    public static final byte[] QUIT = new byte[] { 1, 0, 0, 0, 1 };

    @Override
    public int calcPacketSize() {
        return 1;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Quit Packet";
    }

}
```

36:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\Reply323Packet.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.io.IOException;
import java.io.OutputStream;
import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.StreamUtil;
import io.mycat.net.BackendAIOConnection;

/**
 * @author mycat
 */
public class Reply323Packet extends MySQLPacket {

    public byte[] seed;

    public void write(OutputStream out) throws IOException {
        StreamUtil.writeUB3(out, calcPacketSize());
        StreamUtil.write(out, packetId);
        if (seed == null) {
            StreamUtil.write(out, (byte) 0);
        } else {
            StreamUtil.writeWithNull(out, seed);
        }
    }

    @Override
    public void write(BackendAIOConnection c) {
        ByteBuffer buffer = c.allocate();
        BufferUtil.writeUB3(buffer, calcPacketSize());
        buffer.put(packetId);
        if (seed == null) {
            buffer.put((byte) 0);
        } else {
            BufferUtil.writeWithNull(buffer, seed);
        }
        c.write(buffer);
    }

    @Override
    public int calcPacketSize() {
        return seed == null ? 1 : seed.length + 1;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL Auth323 Packet";
    }

}
```

37:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\RequestFilePacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.net.FrontendConnection;

/**
 * load data local infile 向客户端请求发送文件用
 */
public class RequestFilePacket extends MySQLPacket
{
    public static final byte FIELD_COUNT = (byte) 251;
    public byte command = FIELD_COUNT;
    public byte[] fileName;



    @Override
    public  ByteBuffer  write(ByteBuffer  buffer,  FrontendConnection  c,  boolean
writeSocketIfFull)
    {
        int size = calcPacketSize();
        buffer  =  c.checkWriteBuffer(buffer,  c.getPacketHeaderSize()  +  size,
writeSocketIfFull);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        buffer.put(command);
        if (fileName != null)
        {

            buffer.put(fileName);

        }

        c.write(buffer);
        return buffer;
    }

    @Override
    public int calcPacketSize()
    {
        return fileName == null ? 1 : 1 + fileName.length;
    }

    @Override
    protected String getPacketInfo()
    {
        return "MySQL Request File Packet";
    }


}
```

38:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\ResetPacket.java

```
package io.mycat.net.mysql;

import io.mycat.backend.mysql.MySQLMessage;

/**
 * <pre>
 *
 * COM_STMT_RESET resets the data of a prepared statement which was accumulated with COM_STMT_SEND_LONG_DATA commands and closes the cursor if it was opened with COM_STMT_EXECUTE

 * The server will send a OK_Packet if the statement could be reset, a ERR_Packet if not.
 *
 * COM_STMT_RESET:
 * COM_STMT_RESET
 * direction: client -> server
 * response: OK or ERR

 * payload:
 *   1              [1a] COM_STMT_RESET
 *   4              statement-id
 *
 * </pre>
 *
 * @author CrazyPig
 * @since 2016-09-08
 *
 */
public class ResetPacket extends MySQLPacket {

     private static final byte PACKET_FALG = (byte) 26;
     private long pstmtId;

     public void read(byte[] data) {
          MySQLMessage mm = new MySQLMessage(data);
          packetLength = mm.readUB3();
          packetId = mm.read();
          byte code = mm.read();
          assert code == PACKET_FALG;
          pstmtId = mm.readUB4();
     }

     @Override
     public int calcPacketSize() {
          return 1 + 4;
     }

     @Override
     protected String getPacketInfo() {
          return "MySQL Reset Packet";
     }

     public long getPstmtId() {
          return pstmtId;
     }

}
```

39:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\ResultSetHeaderPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.buffer.BufferArray;
import io.mycat.net.FrontendConnection;

/**
 * From server to client after command, if no error and result set -- that is,
 * if the command was a query which returned a result set. The Result Set Header
 * Packet is the first of several, possibly many, packets that the server sends
 * for result sets. The order of packets for a result set is:
 *
 * <pre>
 * (Result Set Header Packet)   the number of columns
 * (Field Packets)              column descriptors
 * (EOF Packet)                 marker: end of Field Packets
 * (Row Data Packets)           row contents
 * (EOF Packet)                 marker: end of Data Packets
 *
 * Bytes                        Name
 * -----                        ----
 * 1-9   (Length-Coded-Binary)  field_count
 * 1-9   (Length-Coded-Binary)  extra
 *
 *                                                                                 @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Result_Set_Heade
r_Packet
 * </pre>
 *
 * @author mycat
 */
public class ResultSetHeaderPacket extends MySQLPacket {
    public int fieldCount;
    public long extra;

    public void read(byte[] data) {
        MySQLMessage mm = new MySQLMessage(data);
        this.packetLength = mm.readUB3();
        this.packetId = mm.read();
        this.fieldCount = (int) mm.readLength();
        if (mm.hasRemaining()) {
            this.extra = mm.readLength();
        }
    }

    @Override
    public  ByteBuffer  write(ByteBuffer  buffer,  FrontendConnection  c,boolean
writeSocketIfFull) {
        int size = calcPacketSize();
        buffer    =    c.checkWriteBuffer(buffer,    c.getPacketHeaderSize()    +
size,writeSocketIfFull);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        BufferUtil.writeLength(buffer, fieldCount);
        if (extra > 0) {
            BufferUtil.writeLength(buffer, extra);
        }
        return buffer;
    }

    public void write(BufferArray bufferArray) {
         int size = calcPacketSize();
         ByteBuffer buffer = bufferArray
                   .checkWriteBuffer(packetHeaderSize + size);
         BufferUtil.writeUB3(buffer, size);
         buffer.put(packetId);
         BufferUtil.writeLength(buffer, fieldCount);
         if (extra > 0) {
              BufferUtil.writeLength(buffer, extra);
         }
     }

    @Override
    public int calcPacketSize() {
        int size = BufferUtil.getLength(fieldCount);
        if (extra > 0) {
            size += BufferUtil.getLength(extra);
        }
        return size;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL ResultSetHeader Packet";
    }



}
```

40:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\mysql\RowDataPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.mysql;

import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;

import io.mycat.backend.mysql.BufferUtil;
import io.mycat.backend.mysql.MySQLMessage;
import io.mycat.buffer.BufferArray;
import io.mycat.net.FrontendConnection;
```

```
/**
 * From server to client. One packet for each row in the result set.
 *
 * <pre>
 * Bytes                   Name
 * -----                   ----
 * n (Length Coded String) (column value)
 * ...
 *
 * (column value):         The data in the column, as a character string.
 *                         If a column is defined as non-character, the
 *                         server converts the value into a character
 *                         before sending it. Since the value is a Length
 *                         Coded String, a NULL can be represented with a
 *                         single byte containing 251(see the description
 *                         of Length Coded Strings in section "Elements" above).
 *
 *                                                                      @see
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Protocol#Row_Data_Packet
 * </pre>
 *
 * @author mycat
 */
public class RowDataPacket extends MySQLPacket {
     private static final byte NULL_MARK = (byte) 251;
    private static final byte EMPTY_MARK = (byte) 0;

     public byte[] value;
     public int fieldCount;
     public final List<byte[]> fieldValues;

     public RowDataPacket(int fieldCount) {
          this.fieldCount = fieldCount;
          this.fieldValues = new ArrayList<byte[]>(fieldCount);
     }

     public void add(byte[] value) {
          //这里应该修改value
          fieldValues.add(value);
     }
     public void addFieldCount(int add) {
          //这里应该修改field
          fieldCount=fieldCount+add;
     }

     public void read(byte[] data) {
          value = data;
```

```
        MySQLMessage mm = new MySQLMessage(data);
        packetLength = mm.readUB3();
        packetId = mm.read();
        for (int i = 0; i < fieldCount; i++) {
            fieldValues.add(mm.readBytesWithLength());
        }
    }

    @Override
    public ByteBuffer write(ByteBuffer bb, FrontendConnection c,
            boolean writeSocketIfFull) {
        bb = c.checkWriteBuffer(bb, c.getPacketHeaderSize(), writeSocketIfFull);
        BufferUtil.writeUB3(bb, calcPacketSize());
        bb.put(packetId);
        for (int i = 0; i < fieldCount; i++) {
            byte[] fv = fieldValues.get(i);
            if (fv == null ) {
                bb = c.checkWriteBuffer(bb, 1, writeSocketIfFull);
                bb.put(RowDataPacket.NULL_MARK);
            }else if (fv.length == 0) {
              bb = c.checkWriteBuffer(bb, 1, writeSocketIfFull);
              bb.put(RowDataPacket.EMPTY_MARK);
          }
          else {
                bb = c.checkWriteBuffer(bb, BufferUtil.getLength(fv),
                        writeSocketIfFull);
                BufferUtil.writeLength(bb, fv.length);
                bb = c.writeToBuffer(fv, bb);
            }
        }
        return bb;
    }

    @Override
    public int calcPacketSize() {
        int size = 0;
        for (int i = 0; i < fieldCount; i++) {
            byte[] v = fieldValues.get(i);
            size += (v == null || v.length == 0) ? 1 : BufferUtil.getLength(v);
        }
        return size;
    }

    @Override
    protected String getPacketInfo() {
        return "MySQL RowData Packet";
    }
    public void write(BufferArray bufferArray) {
        int size = calcPacketSize();
        ByteBuffer buffer = bufferArray.checkWriteBuffer(packetHeaderSize + size);
        BufferUtil.writeUB3(buffer, size);
        buffer.put(packetId);
        for (int i = 0; i < fieldCount; i++) {
            byte[] fv = fieldValues.get(i);
            if (fv == null) {
                buffer = bufferArray.checkWriteBuffer(1);
                buffer.put(RowDataPacket.NULL_MARK);
            } else if (fv.length == 0) {
                buffer = bufferArray.checkWriteBuffer(1);
                buffer.put(RowDataPacket.EMPTY_MARK);
            } else {
                buffer = bufferArray.checkWriteBuffer(BufferUtil
                        .getLength(fv.length));
                BufferUtil.writeLength(buffer, fv.length);
                bufferArray.write(fv);
            }
        }
    }

}
```

41:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOAcceptor.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.StandardSocketOptions;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.util.Set;

import io.mycat.util.SelectorUtil;
import org.slf4j.Logger; import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import io.mycat.net.factory.FrontendConnectionFactory;

/**
 * @author mycat
 */
public final class NIOAcceptor extends Thread implements SocketAcceptor{
     private static final Logger LOGGER = LoggerFactory.getLogger(NIOAcceptor.class);
     private static final AcceptIdGenerator ID_GENERATOR = new AcceptIdGenerator();

     private final int port;
     private volatile Selector selector;
     private final ServerSocketChannel serverChannel;
     private final FrontendConnectionFactory factory;
     private long acceptCount;
     private final NIOReactorPool reactorPool;

     public NIOAcceptor(String name, String bindIp,int port,
               FrontendConnectionFactory factory, NIOReactorPool reactorPool)
               throws IOException {
          super.setName(name);
          this.port = port;
          this.selector = Selector.open();
          this.serverChannel = ServerSocketChannel.open();
          this.serverChannel.configureBlocking(false);
          /** 设置 TCP 属性 */
          serverChannel.setOption(StandardSocketOptions.SO_REUSEADDR, true);
          serverChannel.setOption(StandardSocketOptions.SO_RCVBUF, 1024 * 16 * 2);
          // backlog=100
```

```
        serverChannel.bind(new InetSocketAddress(bindIp, port), 100);
        this.serverChannel.register(selector, SelectionKey.OP_ACCEPT);
        this.factory = factory;
        this.reactorPool = reactorPool;
    }

    public int getPort() {
        return port;
    }

    public long getAcceptCount() {
        return acceptCount;
    }

    @Override
    public void run() {
        int invalidSelectCount = 0;
        for (;;) {
            final Selector tSelector = this.selector;
            ++acceptCount;
            try {
                long start = System.nanoTime();
               tSelector.select(1000L);
                long end = System.nanoTime();
                Set<SelectionKey> keys = tSelector.selectedKeys();
                if    (keys.size()    ==    0    &&    (end    -    start)    <
SelectorUtil.MIN_SELECT_TIME_IN_NANO_SECONDS )
                {
                    invalidSelectCount++;
                }
                else
              {
                    try {
                        for (SelectionKey key : keys) {
                            if (key.isValid() && key.isAcceptable()) {
                                accept();
                            } else {
                                key.cancel();
                            }
                        }
                    } finally {
                        keys.clear();
                        invalidSelectCount = 0;
                    }
                }
                if (invalidSelectCount > SelectorUtil.REBUILD_COUNT_THRESHOLD)
                {
                        final          Selector          rebuildSelector          =
SelectorUtil.rebuildSelector(this.selector);
                        if (rebuildSelector != null)
                        {
                             this.selector = rebuildSelector;
                        }
                        invalidSelectCount = 0;
                   }
              } catch (Exception e) {
                   LOGGER.warn(getName(), e);
              }
         }
    }

    private void accept() {
         SocketChannel channel = null;
         try {
              channel = serverChannel.accept();
              channel.configureBlocking(false);
              FrontendConnection c = factory.make(channel);
              c.setAccepted(true);
              c.setId(ID_GENERATOR.getId());
              NIOProcessor processor = (NIOProcessor) MycatServer.getInstance()
                        .nextProcessor();
              c.setProcessor(processor);

              NIOReactor reactor = reactorPool.getNextReactor();
              reactor.postRegister(c);

         } catch (Exception e) {
            LOGGER.warn(getName(), e);
              closeChannel(channel);
         }
    }

    private static void closeChannel(SocketChannel channel) {
         if (channel == null) {
              return;
         }
         Socket socket = channel.socket();
         if (socket != null) {
              try {
                   socket.close();
              } catch (IOException e) {
                LOGGER.error("closeChannelError", e);
              }
         }
```

```
        try {
            channel.close();
        } catch (IOException e) {
          LOGGER.error("closeChannelError", e);
        }
    }

    /**
     * 前端连接 ID 生成器
     *
     * @author mycat
     */
    private static class AcceptIdGenerator {

        private static final long MAX_VALUE = 0xffffffffL;

        private long acceptId = 0L;
        private final Object lock = new Object();

        private long getId() {
            synchronized (lock) {
                if (acceptId >= MAX_VALUE) {
                    acceptId = 0L;
                }
                return ++acceptId;
            }
        }
    }

}
```

42:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOConnection.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * @author mycat
 */
public interface NIOConnection extends ClosableConnection{

    /**
     * connected
     */
    void register() throws IOException;

    /**
     * 处理数据
     */
    void handle(byte[] data);

    /**
     * 写出一块缓存数据
     */
    void write(ByteBuffer buffer);



}
```

43:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOConnector.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.SocketChannel;
import java.util.Set;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import java.util.concurrent.atomic.AtomicLong;

import io.mycat.util.SelectorUtil;

/**
 * @author mycat
 */
public final class NIOConnector extends Thread implements SocketConnector {
	private static final Logger LOGGER = LoggerFactory.getLogger(NIOConnector.class);
	public static final ConnectIdGenerator ID_GENERATOR = new ConnectIdGenerator();

	private final String name;
	private volatile Selector selector;
	private final BlockingQueue<AbstractConnection> connectQueue;
	private long connectCount;
```

```
    private final NIOReactorPool reactorPool;

    public NIOConnector(String name, NIOReactorPool reactorPool)
            throws IOException {
        super.setName(name);
        this.name = name;
        this.selector = Selector.open();
        this.reactorPool = reactorPool;
        this.connectQueue = new LinkedBlockingQueue<AbstractConnection>();
    }

    public long getConnectCount() {
        return connectCount;
    }

    public void postConnect(AbstractConnection c) {
        connectQueue.offer(c);
        selector.wakeup();
    }

    @Override
    public void run() {
        int invalidSelectCount = 0;
        for (;;) {
            final Selector tSelector = this.selector;
            ++connectCount;
            try {
                long start = System.nanoTime();
                tSelector.select(1000L);
                long end = System.nanoTime();
                connect(tSelector);
                Set<SelectionKey> keys = tSelector.selectedKeys();
                if    (keys.size()    ==    0    &&    (end    -    start)    <
SelectorUtil.MIN_SELECT_TIME_IN_NANO_SECONDS )
                {
                    invalidSelectCount++;
                }
                else
                {
                    try {
                        for (SelectionKey key : keys)
                        {
                            Object att = key.attachment();
                            if    (att    !=    null    &&    key.isValid()    &&
key.isConnectable())
                            {
                                finishConnect(key, att);
```

```
                                    } else
                                    {
                                        key.cancel();
                                    }
                                }
                            } finally
                            {
                                invalidSelectCount = 0;
                                keys.clear();
                            }
                        }
                        if (invalidSelectCount > SelectorUtil.REBUILD_COUNT_THRESHOLD)
                        {
                            final          Selector          rebuildSelector          =
SelectorUtil.rebuildSelector(this.selector);
                            if (rebuildSelector != null)
                            {
                                this.selector = rebuildSelector;
                            }
                            invalidSelectCount = 0;
                        }
                    } catch (Exception e) {
                        LOGGER.warn(name, e);
                    }
                }
            }

            private void connect(Selector selector) {
                AbstractConnection c = null;
                while ((c = connectQueue.poll()) != null) {
                    try {
                        SocketChannel channel = (SocketChannel) c.getChannel();
                        channel.register(selector, SelectionKey.OP_CONNECT, c);
                        channel.connect(new InetSocketAddress(c.host, c.port));

                    } catch (Exception e) {
                        LOGGER.error("error:",e);
                        c.close(e.toString());
                    }
                }
            }

            private void finishConnect(SelectionKey key, Object att) {
                BackendAIOConnection c = (BackendAIOConnection) att;
                try {
                    if (finishConnect(c, (SocketChannel) c.channel)) {
                        clearSelectionKey(key);
                        c.setId(ID_GENERATOR.getId());
                        NIOProcessor processor = MycatServer.getInstance()
                                .nextProcessor();
                        c.setProcessor(processor);
                        NIOReactor reactor = reactorPool.getNextReactor();
                        reactor.postRegister(c);
                        c.onConnectfinish();
                    }
                } catch (Exception e) {
                    clearSelectionKey(key);
                    LOGGER.error("error:",e);
                    c.close(e.toString());
                    c.onConnectFailed(e);

                }
            }

            private boolean finishConnect(AbstractConnection c, SocketChannel channel)
                    throws IOException {
                if (channel.isConnectionPending()) {
                    channel.finishConnect();

                    c.setLocalPort(channel.socket().getLocalPort());
                    return true;
                } else {
                    return false;
                }
            }

            private void clearSelectionKey(SelectionKey key) {
                if (key.isValid()) {
                    key.attach(null);
                    key.cancel();
                }
            }

            /**
             * 后端连接 ID 生成器
             *
             * @author mycat
             */
            public static class ConnectIdGenerator {

                private static final long MAX_VALUE = Long.MAX_VALUE;
                private AtomicLong connectId = new AtomicLong(0);

                public long getId() {
                    return connectId.incrementAndGet();
                }
            }

    }
```

44:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOHandler.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

/**
 * @author mycat
 */
public interface NIOHandler {

    void handle(byte[] data);

}
```

45:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOProcessor.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.util.Iterator;
import java.util.Map.Entry;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.ConcurrentMap;
import java.util.concurrent.atomic.AtomicInteger;

import io.mycat.buffer.BufferPool;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import io.mycat.MycatServer;
import io.mycat.backend.BackendConnection;
import io.mycat.statistic.CommandCount;
import io.mycat.util.NameableExecutor;
import io.mycat.util.TimeUtil;

/**
 * @author mycat
 */
public final class NIOProcessor {

    private static final Logger LOGGER = LoggerFactory.getLogger("NIOProcessor");
    private final String name;
    private final BufferPool bufferPool;
    private final NameableExecutor executor;
    private final ConcurrentMap<Long, FrontendConnection> frontends;
    private final ConcurrentMap<Long, BackendConnection> backends;
    private final CommandCount commands;
    private long netInBytes;
    private long netOutBytes;

    // TODO: add by zhuam
    // reload @@config_all 后，老的 backends  全部移往 backends_old，待检测任务进行
销毁
    public final static ConcurrentLinkedQueue<BackendConnection> backends_old = new
ConcurrentLinkedQueue<BackendConnection>();

    //前端已连接数
    private AtomicInteger frontendsLength = new AtomicInteger(0);

    public NIOProcessor(String name, BufferPool bufferPool,
            NameableExecutor executor) throws IOException {
        this.name = name;
        this.bufferPool = bufferPool;
        this.executor = executor;
        this.frontends = new ConcurrentHashMap<Long, FrontendConnection>();
        this.backends = new ConcurrentHashMap<Long, BackendConnection>();
        this.commands = new CommandCount();
    }

    public String getName() {
        return name;
    }

    public BufferPool getBufferPool() {
        return bufferPool;
    }

    public int getWriteQueueSize() {
        int total = 0;
        for (FrontendConnection fron : frontends.values()) {
            total += fron.getWriteQueue().size();
        }
        for (BackendConnection back : backends.values()) {
            if (back instanceof BackendAIOConnection) {
                total += ((BackendAIOConnection) back).getWriteQueue().size();
            }
        }
```

```
		return total;

	}

	public NameableExecutor getExecutor() {
		return this.executor;
	}

	public CommandCount getCommands() {
		return this.commands;
	}

	public long getNetInBytes() {
		return this.netInBytes;
	}

	public void addNetInBytes(long bytes) {
		this.netInBytes += bytes;
	}

	public long getNetOutBytes() {
		return this.netOutBytes;
	}

	public void addNetOutBytes(long bytes) {
		this.netOutBytes += bytes;
	}

	public void addFrontend(FrontendConnection c) {
		this.frontends.put(c.getId(), c);
		this.frontendsLength.incrementAndGet();
	}

	public ConcurrentMap<Long, FrontendConnection> getFrontends() {
		return this.frontends;
	}

	public int getForntedsLength(){
		return this.frontendsLength.get();
	}

	public void addBackend(BackendConnection c) {
		this.backends.put(c.getId(), c);
	}

	public ConcurrentMap<Long, BackendConnection> getBackends() {
		return this.backends;
```

```
    }

    /**
     * 定时执行该方法，回收部分资源。
     */
    public void checkBackendCons() {
        backendCheck();
    }

    /**
     * 定时执行该方法，回收部分资源。
     */
    public void checkFrontCons() {
        frontendCheck();
    }

    // 前端连接检查
    private void frontendCheck() {
        Iterator<Entry<Long, FrontendConnection>> it = frontends.entrySet()
                .iterator();
        while (it.hasNext()) {
            FrontendConnection c = it.next().getValue();

            // 删除空连接
            if (c == null) {
                it.remove();
                this.frontendsLength.decrementAndGet();
                continue;
            }

            // 清理已关闭连接，否则空闲检查。
            if (c.isClosed()) {
                // 此处在高并发情况下会存在并发问题, fixed #1072  极有可能解决了
#700
                //c.cleanup();
                it.remove();
                this.frontendsLength.decrementAndGet();
            } else {
                // very important ,for some data maybe not sent
                checkConSendQueue(c);
                c.idleCheck();
            }
        }
    }

    private void checkConSendQueue(AbstractConnection c) {
        // very important ,for some data maybe not sent
```

```
          if (!c.writeQueue.isEmpty()) {
               c.getSocketWR().doNextWriteCheck();
          }
     }

     // 后端连接检查
     private void backendCheck() {
          long                              sqlTimeout                              =
MycatServer.getInstance().getConfig().getSystem().getSqlExecuteTimeout() * 1000L;
          Iterator<Entry<Long,          BackendConnection>>          it          =
backends.entrySet().iterator();
          while (it.hasNext()) {
               BackendConnection c = it.next().getValue();

               // 删除空连接
               if (c == null) {
                    it.remove();
                    continue;
               }
               // SQL 执行超时的连接关闭
               if (c.isBorrowed() && c.getLastTime() < TimeUtil.currentTimeMillis() -
sqlTimeout) {
                    LOGGER.warn("found backend connection SQL timeout ,close it " + c);
                    c.close("sql timeout");
               }

               // 清理已关闭连接，否则空闲检查。
               if (c.isClosed()) {
                    it.remove();

               } else {
                    // very important ,for some data maybe not sent
                    if (c instanceof AbstractConnection) {
                         checkConSendQueue((AbstractConnection) c);
                    }
                    c.idleCheck();
               }
          }
     }

     public void removeConnection(AbstractConnection con) {
          if (con instanceof BackendConnection) {
               this.backends.remove(con.getId());
          } else {
               this.frontends.remove(con.getId());
               this.frontendsLength.decrementAndGet();
          }
```

```
	}
	//jdbc 连接用这个释放
	public void removeConnection(BackendConnection con){
		this.backends.remove(con.getId());
	}

}
```

46:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOReactor.jav
a

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net;

import java.io.IOException;
import java.nio.channels.CancelledKeyException;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.util.Queue;
import java.util.Set;
import java.util.concurrent.ConcurrentLinkedQueue;

import io.mycat.util.SelectorUtil;
import org.slf4j.Logger; import org.slf4j.LoggerFactory;
```

```java
/**
 * 网络事件反应器
 *
 * <p>
 * Catch exceptions such as OOM so that the reactor can keep running for response client!
 * </p>
 * @since 2016-03-30
 *
 * @author mycat, Uncle-pan
 *
 */
public final class NIOReactor {
    private static final Logger LOGGER = LoggerFactory.getLogger(NIOReactor.class);
    private final String name;
    private final RW reactorR;

    public NIOReactor(String name) throws IOException {
        this.name = name;
        this.reactorR = new RW();
    }

    final void startup() {
        new Thread(reactorR, name + "-RW").start();
    }

    final void postRegister(AbstractConnection c) {
        reactorR.registerQueue.offer(c);
        reactorR.selector.wakeup();
    }

    final Queue<AbstractConnection> getRegisterQueue() {
        return reactorR.registerQueue;
    }

    final long getReactCount() {
        return reactorR.reactCount;
    }

    private final class RW implements Runnable {
        private volatile Selector selector;
        private final ConcurrentLinkedQueue<AbstractConnection> registerQueue;
        private long reactCount;

        private RW() throws IOException {
            this.selector = Selector.open();
            this.registerQueue = new ConcurrentLinkedQueue<AbstractConnection>();
        }
	@Override
	public void run() {
		int invalidSelectCount = 0;
		Set<SelectionKey> keys = null;
		for (;;) {
			++reactCount;
			try {
				final Selector tSelector = this.selector;
				long start = System.nanoTime();
				tSelector.select(500L);
				long end = System.nanoTime();
				register(tSelector);
				keys = tSelector.selectedKeys();
				if    (keys.size()    ==    0    &&    (end    -    start)    <
SelectorUtil.MIN_SELECT_TIME_IN_NANO_SECONDS )
				{
					invalidSelectCount++;
				}
				else
				{
					invalidSelectCount = 0;
					for (SelectionKey key : keys) {
						AbstractConnection con = null;
						try {
							Object att = key.attachment();
							if (att != null) {
								con = (AbstractConnection) att;
								if (key.isValid() && key.isReadable()) {
									try {
										con.asynRead();
									} catch (IOException e) {
										con.close("program   err:"   +
e.toString());

										continue;
									} catch (Exception e) {
										LOGGER.warn("caught err:", e);
										con.close("program   err:"   +
e.toString());

										continue;
									}
								}
								if (key.isValid() && key.isWritable()) {
									con.doNextWriteCheck();
								}
							} else {
								key.cancel();
```

```
                                }
                            } catch (CancelledKeyException e) {
                                if (LOGGER.isDebugEnabled()) {
                                    LOGGER.debug(con    +    "    socket    key
canceled");
                                }
                            } catch (Exception e) {
                                LOGGER.warn(con + " " + e);
                            } catch (final Throwable e) {
                                // Catch exceptions such as OOM and close
connection if exists
                                //so that the reactor can keep running!
                                // @author Uncle-pan
                                // @since 2016-03-30
                                if (con != null) {
                                    con.close("Bad: " + e);
                                }
                                LOGGER.error("caught err: ", e);
                                continue;
                            }
                        }
                    }
                    if                  (invalidSelectCount                   >
SelectorUtil.REBUILD_COUNT_THRESHOLD)
                    {
                        final        Selector        rebuildSelector        =
SelectorUtil.rebuildSelector(this.selector);
                        if (rebuildSelector != null)
                        {
                            this.selector = rebuildSelector;
                        }
                        invalidSelectCount = 0;
                    }
                } catch (Exception e) {
                    LOGGER.warn(name, e);
                } catch (final Throwable e){
                    // Catch exceptions such as OOM so that the reactor can keep
running!
                // @author Uncle-pan
                // @since 2016-03-30
                    LOGGER.error("caught err: ", e);
                } finally {
                    if (keys != null) {
                        keys.clear();
                    }

                }
```

```
            }
        }

        private void register(Selector selector) {
            AbstractConnection c = null;
            if (registerQueue.isEmpty()) {
                return;
            }
            while ((c = registerQueue.poll()) != null) {
                try {
                    ((NIOSocketWR) c.getSocketWR()).register(selector);
                    c.register();
                } catch (Exception e) {
                    c.close("register err" + e.toString());
                }
            }
        }

    }

}
```

47:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOReactorPool.java

```
package io.mycat.net;

import java.io.IOException;

public class NIOReactorPool {
    private final NIOReactor[] reactors;
    private volatile int nextReactor;

    public NIOReactorPool(String name, int poolSize) throws IOException {
        reactors = new NIOReactor[poolSize];
        for (int i = 0; i < poolSize; i++) {
            NIOReactor reactor = new NIOReactor(name + "-" + i);
            reactors[i] = reactor;
            reactor.startup();
        }
    }

    public NIOReactor getNextReactor() {
//        if (++nextReactor == reactors.length) {
//            nextReactor = 0;
//        }
//        return reactors[nextReactor];
```

```
        int i = ++nextReactor;
        if (i >= reactors.length) {
            i=nextReactor = 0;
        }
        return reactors[i];
    }
}
```

48:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\NIOSocketWR.ja
va

```
package io.mycat.net;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.SocketChannel;
import java.util.concurrent.atomic.AtomicBoolean;

import io.mycat.util.TimeUtil;

public class NIOSocketWR extends SocketWR {
     private SelectionKey processKey;
     private static final int OP_NOT_READ = ~SelectionKey.OP_READ;
     private static final int OP_NOT_WRITE = ~SelectionKey.OP_WRITE;
     private final AbstractConnection con;
     private final SocketChannel channel;
     private final AtomicBoolean writing = new AtomicBoolean(false);

     public NIOSocketWR(AbstractConnection con) {
          this.con = con;
          this.channel = (SocketChannel) con.channel;
     }

     public void register(Selector selector) throws IOException {
          try {
               processKey = channel.register(selector, SelectionKey.OP_READ, con);
          } finally {
               if (con.isClosed.get()) {
                    clearSelectionKey();
               }
          }
     }

     public void doNextWriteCheck() {

          if (!writing.compareAndSet(false, true)) {
                return;
            }

            try {
                boolean noMoreData = write0();
                writing.set(false);
                if (noMoreData && con.writeQueue.isEmpty()) {
                    if   ((processKey.isValid()   &&   (processKey.interestOps()   &
SelectionKey.OP_WRITE) != 0)) {
                        disableWrite();
                    }

                } else {

                    if   ((processKey.isValid()   &&   (processKey.interestOps()   &
SelectionKey.OP_WRITE) == 0)) {
                        enableWrite(false);
                    }
                }

            } catch (IOException e) {
                if (AbstractConnection.LOGGER.isDebugEnabled()) {
                    AbstractConnection.LOGGER.debug("caught err:", e);
                }
                con.close("err:" + e);
            }

        }

        private boolean write0() throws IOException {

            int written = 0;
            ByteBuffer buffer = con.writeBuffer;
            if (buffer != null) {
                while (buffer.hasRemaining()) {
                    written = channel.write(buffer);
                    if (written > 0) {
                        con.netOutBytes += written;
                        con.processor.addNetOutBytes(written);
                        con.lastWriteTime = TimeUtil.currentTimeMillis();
                    } else {
                        break;
                    }
                }

                if (buffer.hasRemaining()) {
                    con.writeAttempts++;
                    return false;
                } else {
                    con.writeBuffer = null;
                    con.recycle(buffer);
                }
            }
            while ((buffer = con.writeQueue.poll()) != null) {
                if (buffer.limit() == 0) {
                    con.recycle(buffer);
                    con.close("quit send");
                    return true;
                }

                buffer.flip();
                try {
                    while (buffer.hasRemaining()) {
                        written = channel.write(buffer);// java.io.IOException:
                                        // Connection reset by peer
                        if (written > 0) {
                            con.lastWriteTime = TimeUtil.currentTimeMillis();
                            con.netOutBytes += written;
                            con.processor.addNetOutBytes(written);
                            con.lastWriteTime = TimeUtil.currentTimeMillis();
                        } else {
                            break;
                        }
                    }
                } catch (IOException e) {
                    con.recycle(buffer);
                    throw e;
                }
                if (buffer.hasRemaining()) {
                    con.writeBuffer = buffer;
                    con.writeAttempts++;
                    return false;
                } else {
                    con.recycle(buffer);
                }
            }
            return true;
        }

        private void disableWrite() {
            try {
                SelectionKey key = this.processKey;
                key.interestOps(key.interestOps() & OP_NOT_WRITE);
            } catch (Exception e) {
                AbstractConnection.LOGGER.warn("can't disable write " + e + " con "
                        + con);
            }

        }

        private void enableWrite(boolean wakeup) {
            boolean needWakeup = false;
            try {
                SelectionKey key = this.processKey;
                key.interestOps(key.interestOps() | SelectionKey.OP_WRITE);
                needWakeup = true;
            } catch (Exception e) {
                AbstractConnection.LOGGER.warn("can't enable write " + e);

            }
            if (needWakeup && wakeup) {
                processKey.selector().wakeup();
            }
        }

        public void disableRead() {

            SelectionKey key = this.processKey;
            key.interestOps(key.interestOps() & OP_NOT_READ);
        }

        public void enableRead() {

            boolean needWakeup = false;
            try {
                SelectionKey key = this.processKey;
                key.interestOps(key.interestOps() | SelectionKey.OP_READ);
                needWakeup = true;
            } catch (Exception e) {
                AbstractConnection.LOGGER.warn("enable read fail " + e);
            }
            if (needWakeup) {
                processKey.selector().wakeup();
            }
        }

        private void clearSelectionKey() {
            try {
                SelectionKey key = this.processKey;
                if (key != null && key.isValid()) {
                    key.attach(null);
                    key.cancel();
                }
            } catch (Exception e) {
                AbstractConnection.LOGGER.warn("clear selector keys err:" + e);
            }
        }

        @Override
        public void asynRead() throws IOException {
            ByteBuffer theBuffer = con.readBuffer;
            if (theBuffer == null) {

                theBuffer                                                              =
    con.processor.getBufferPool().allocate(con.processor.getBufferPool().getChunkSize()
    );

                con.readBuffer = theBuffer;
            }

            int got = channel.read(theBuffer);

            con.onReadData(got);
        }

    }
```

49:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\AuthenticationCleartextPassword.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationCleartextPassword (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(3) Specifies that a clear-text password is required.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationCleartextPassword extends PostgresPacket {

}
```

50:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\AuthenticationGSS.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;
```

```
/**
 * <pre>
 * AuthenticationGSS (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(7) Specifies that GSSAPI authentication is required.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationGSS extends PostgresPacket {

}
```

51:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\AuthenticationGSSContinue.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationGSSContinue (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32 Length of message contents in bytes, including self.
 * Int32(8) Specifies that this message contains GSSAPI or SSPI data.
 * Byten GSSAPI or SSPI authentication data.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationGSSContinue extends PostgresPacket {

}
```

52:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\AuthenticationKerberosV5.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationKerberosV5 (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(2) Specifies that Kerberos V5 authentication is required.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationKerberosV5 extends PostgresPacket {
}

53:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Authe
nticationMD5Password.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationMD5Password (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(12) Length of message contents in bytes, including self.
 * Int32(5) Specifies that an MD5-encrypted password is required.
 * Byte4 The salt to use when encrypting the password.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationMD5Password extends PostgresPacket {

}

54:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Authe
nticationOk.java
```

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationOk (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(0) Specifies that the authentication was successful.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationOk extends PostgresPacket {

}
```

55:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\AuthenticationSCMCredential.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationSCMCredential (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(6) Specifies that an SCM credentials message is required.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationSCMCredential extends PostgresPacket {

}

56:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Authe
nticationSSPI.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * AuthenticationSSPI (B)
 * Byte1('R') Identifies the message as an authentication request.
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(9) Specifies that SSPI authentication is required.
 * </pre>
 *
 * @author mycat
 */
public class AuthenticationSSPI extends PostgresPacket {

}
```

57:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\BackendKeyData.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * BackendKeyData (B)
 * Byte1('K') Identifies the message as cancellation key data.
 *            The frontend must save these values if it wishes to be able to
 *            issue CancelRequest messages later.
 * Int32(12) Length of message contents in bytes, including self.
 * Int32 The process ID of this backend.
 * Int32 The secret key of this backend.
 * </pre>
 *
 * @author mycat
 */
public class BackendKeyData extends PostgresPacket {

}
```

58:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Bind.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Bind (F)
 * Byte1('B') Identifies the message as a Bind command.
 * Int32 Length of message contents in bytes, including self.
 * String The name of the destination portal (an empty string selects the unnamed
portal).
 * String The name of the source prepared statement (an empty string selects the unnamed
 *        prepared statement).
 * Int16 The number of parameter format codes that follow (denoted C below).
 *       This can be zero to indicate that there are no parameters or that the parameters
 *       all use the default format(text); or one, in which case the specified format
code
 *       is applied to all parameters; or it can equal the actual number of parameters.
 * Int16[C] The parameter format codes. Each must presently be zero (text) or
one(binary).
 * Int16 The number of parameter values that follow (possibly zero). This must match
the
 *       number of parameters needed by the query. Next, the following pair of fields
appear
 *       for each parameter:
 * Int32 The length of the parameter value, in bytes (this count does not include
 *       itself). Can be zero. As a special case, -1 indicates a NULL parameter
 *       value. No value bytes follow in the NULL case.
 * Byten The value of the parameter, in the format indicated by the associated format
code.
 *       n is the above length. After the last parameter, the following fields appear:
 * Int16 The number of result-column format codes that follow (denoted R
 *       below). This can be zero to indicate that there are no result columns or
 *       that the result columns should all use the default format (text); or one,
 *       in which case the specified format code is applied to all result columns
 *       (if any); or it can equal the actual number of result columns of the query.
 * Int16[R] The result-column format codes. Each must presently be zero (text) or one
(binary).
 * </pre>
 *
 * @author mycat
 */
public class Bind extends PostgresPacket {

}

59:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\BindC
```

omplete.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * BindComplete (B)
 * Byte1('2') Identifies the message as a Bind-complete indicator.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class BindComplete extends PostgresPacket {

}
```

60:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CancelRequest.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CancelRequest (F)
 * Int32(16) Length of message contents in bytes,including self.
 * Int32(80877102) The cancel request code. The value is chosen to
 *                 contain 1234 in the most significant 16 bits, and
 *                 5678 in the least 16 significant bits. (To avoid
 *                 confusion, this code must not be the same as any
 *                 protocol version number.)
 * Int32 The process ID of the target backend.
 * Int32 The secret key for the target backend.
 * </pre>
 *
 * @author mycat
 */
public class CancelRequest extends PostgresPacket {

}

61:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Close.
java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Close (F)
 * Byte1('C') Identifies the message as a Close command.
 * Int32 Length of message contents in bytes, including self.
 * Byte1 'S' to close a prepared statement; or 'P' to close a portal.
 * String The name of the prepared statement or portal to close (an
 *        empty string selects the unnamed prepared statement or portal).
 * </pre>
 *
 * @author mycat
 */
public class Close extends PostgresPacket {

}
```

62:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Close
Complete.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CloseComplete (B)
 * Byte1('3') Identifies the message as a Close-complete indicator.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class CloseComplete extends PostgresPacket {

}
```

63:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CommandComplete.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CommandComplete (B)
 * Byte1('C') Identifies the message as a command-completed response.
 * Int32 Length of message contents in bytes, including self.
 * String The command tag. This is usually a single word that identifies which SQL
command was completed.
 *        For an INSERT command, the tag is INSERT oid rows, where rows is the number
of rows inserted.
 *        oid is the object ID of the inserted row if rows is 1 and the target table
has OIDs; otherwise oid is 0.
 *        For a DELETE command, the tag is DELETE rows where rows is the number of rows
deleted.
 *        For an UPDATE command, the tag is UPDATE rows where rows is the number of
rows updated.
 *        For a SELECT or CREATE TABLE AS command, the tag is SELECT rows where rows
is the number of rows retrieved.
 *        For a MOVE command, the tag is MOVE rows where rows is the number of rows
the cursor's position has been changed by.
 *        For a FETCH command, the tag is FETCH rows where rows is the number of rows
that have been retrieved from the cursor.
 *        For a COPY command, the tag is COPY rows where rows is the number of rows
copied.
 *        (Note: the row count appears only in PostgreSQL 8.2 and later.)
 * </pre>
 *
 * @author mycat
 */
public class CommandComplete extends PostgresPacket {

}

64:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CopyB
othResponse.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CopyBothResponse (B)
 * Byte1('W') Identifies the message as a Start Copy Both response.
 *            This message is used only for Streaming Replication.
 * Int32 Length of message contents in bytes, including self.
 * Int8 0 indicates the overall COPY format is textual (rows separated
 *      by newlines, columns separated by separator characters, etc).
 *      1 indicates the overall copy format is binary (similar to DataRow
 *      format). See COPY for more information.
 * Int16 The number of columns in the data to be copied(denoted N below).
 * Int16[N] The format codes to be used for each column. Each must presently
 *          be zero (text) or one (binary). All must be zero if the overall
 *          copy format is textual.
 * </pre>
 *
 * @author mycat
 */
public class CopyBothResponse extends PostgresPacket {

}

65:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CopyD
ata.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CopyData (F & B)
 * Byte1('d') Identifies the message as COPY data.
 * Int32 Length of message contents in bytes, including self.
 * Byten Data that forms part of a COPY data stream. Messages sent from the backend
will
 *       always correspond to single data rows, but messages sent by frontends
 *       might divide the data stream arbitrarily.
 * </pre>
 *
 * @author mycat
 */
public class CopyData extends PostgresPacket {

}

66:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CopyD
one.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CopyDone (F & B)
 * Byte1('c') Identifies the message as a COPY-complete indicator.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class CopyDone extends PostgresPacket {

}
```

67:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CopyFail.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CopyFail (F)
 * Byte1('f') Identifies the message as a COPY-failure indicator.
 * Int32 Length of message contents in bytes, including self.
 * String An error message to report as the cause of failure.
 * </pre>
 *
 * @author mycat
 */
public class CopyFail extends PostgresPacket {

}
```

68:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CopyInResponse.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CopyInResponse (B)
 * Byte1('G') Identifies the message as a Start Copy In response.
 *            The frontend must now send copy-in data (if not prepared
 *            to do so, send a CopyFail message).
 * Int32 Length of message contents in bytes, including self.
 * Int8 0 indicates the overall COPY format is textual (rows separated
 *      by newlines, columns separated by separator characters, etc).
 *      1 indicates the overall copy format is binary (similar to DataRow
 *      format). See COPY for more information.
 * Int16 The number of columns in the data to be copied (denoted N below).
 * Int16[N] The format codes to be used for each column. Each must presently
 *          be zero (text) or one (binary). All must be zero if the overall
 *          copy format is textual.
 * </pre>
 *
 * @author mycat
 */
public class CopyInResponse extends PostgresPacket {

}

69:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\CopyO
utResponse.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * CopyOutResponse (B)
 * Byte1('H') Identifies the message as a Start Copy Out response.
 *            This message will be followed by copy-out data. Int32 Length of
 *            message contents in bytes, including self.
 * Int8 0 indicates the overall COPY format is textual (rows separated by
 *      newlines, columns separated by separator characters, etc). 1 indicates
 *      the overall copy format is binary(similar to DataRow format).
 *      See COPY for more information.
 * Int16 The number of columns in the data to be copied (denoted N below).
 * Int16[N] The format codes to be used for each column. Each must presently
 *          be zero(text) or one (binary). All must be zero if the overall
 *          copy format is textual.
 * </pre>
 *
 * @author mycat
 */
public class CopyOutResponse extends PostgresPacket {

}

70:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\DataR
ow.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * DataRow (B)
 * Byte1('D') Identifies the message as a data row.
 * Int32 Length of message contents in bytes, including self.
 * Int16 The number of column values that follow (possibly zero).
 *       Next, the following pair of fields appear for each column:
 * Int32 The length of the column value, in bytes(this count does not
 *       include itself). Can be zero. As a special case, -1 indicates
 *       a NULL column value. No value bytes follow in the NULL case.
 * Byten The value of the column, in the format indicated by the associated
 *       format code. n is the above length.
 * </pre>
 *
 * @author mycat
 */
public class DataRow extends PostgresPacket {

}

71:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Descr
ibe.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Describe (F)
 * Byte1('D') Identifies the message as a Describe command.
 * Int32 Length of message contents in bytes, including self.
 * Byte1 'S' to describe a prepared statement; or 'P' to describe a portal.
 * String The name of the prepared statement or portal to describe (an empty
 *        string selects the unnamed prepared statement or portal).
 * </pre>
 *
 * @author mycat
 */
public class Describe extends PostgresPacket {

}

72:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Empty
QueryResponse.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * EmptyQueryResponse (B)
 * Byte1('I') Identifies the message as a response to an empty query
 *            string. (This substitutes for CommandComplete.)
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class EmptyQueryResponse extends PostgresPacket {

}
```

73:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\ErrorResponse.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;
```

```
/**
 * <pre>
 * ErrorResponse (B)
 * Byte1('E') Identifies the message as an error.
 * Int32 Length of message contents in bytes, including self.
 *       The message body consists of one or more identified fields,
 *       followed by a zero byte as a terminator. Fields can appear
 *       in any order. For each field there is the following:
 * Byte1 A code identifying the field type; if zero, this is the
 *       message terminator and no string follows. The presently defined
 *       field types are listed in Section 46.6. Since more field types
 *       might be added in future, frontends should silently ignore
 *       fields of unrecognized type.
 * String The field value.
 * </pre>
 *
 * @author mycat
 */
public class ErrorResponse extends PostgresPacket {

}
```

74:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Execute.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
```

```
package io.mycat.net.postgres;

/**
 * <pre>
 * Execute (F)
 * Byte1('E') Identifies the message as an Execute command.
 * Int32 Length of message contents in bytes, including self.
 * String The name of the portal to execute (an empty string
 *        selects the unnamed portal).
 * Int32 Maximum number of rows to return, if portal contains a
 *       query that returns rows (ignored otherwise).
 *       Zero denotes "no limit".
 * </pre>
 *
 * @author mycat
 */
public class Execute extends PostgresPacket {

}
```

75:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Flush.java

```
package io.mycat.net.postgres;
```

```
/**
 * <pre>
 * Flush (F)
 * Byte1('H') Identifies the message as a Flush command.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class Flush extends PostgresPacket {

}
```

76:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\FunctionCall.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * FunctionCall (F)
 * Byte1('F') Identifies the message as a function call.
 * Int32 Length of message contents in bytes, including self.
 * Int32 Specifies the object ID of the function to call.
 * Int16 The number of argument format codes that follow (denoted C below).
 *       This can be zero to indicate that there are no arguments or that
 *       the arguments all use the default format (text); or one, in which
 *       case the specified format code is applied to all arguments; or it
 *       can equal the actual number of arguments.
 * Int16[C] The argument format codes. Each must presently be zero (text) or
 *          one (binary).
 * Int16 Specifies the number of arguments being supplied to the function.
 *       Next, the following pair of fields appear for each argument:
 * Int32 The length of the argument value, in bytes (this count does not include
 *       itself). Can be zero. As a special case, -1 indicates a NULL argument
 *       value. No value bytes follow in the NULL case.
 * Byten The value of the argument, in the format indicated by the associated
 *       format code. n is the above length. After the last argument, the
 *       following field appears:
 * Int16 The format code for the function result. Must presently be zero (text)
 *       or one (binary).
 * </pre>
 *
 * @author mycat
 */
public class FunctionCall extends PostgresPacket {

}
```

77:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\FunctionCallResponse.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * FunctionCallResponse (B)
 * Byte1('V') Identifies the message as a function call result.
 * Int32 Length of message contents in bytes, including self.
 * Int32 The length of the function result value, in bytes (this count does
 *       not include itself). Can be zero. As a special case, -1 indicates a
 *       NULL function result. No value bytes follow in the NULL case.
 * Byten The value of the function result, in the format indicated by the
 *       associated format code. n is the above length.
 * </pre>
 *
 * @author mycat
 */
public class FunctionCallResponse extends PostgresPacket {

}
```

78:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\NoData.java

```
package io.mycat.net.postgres;

/**
 * <pre>
 * NoData (B)
 * Byte1('n') Identifies the message as a no-data indicator.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class NoData extends PostgresPacket {

}
```

79:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\NoticeResponse.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * NoticeResponse (B)
 * Byte1('N') Identifies the message as a notice.
 * Int32 Length of message contents in bytes, including self. The message
 *       body consists of one or more identified fields, followed by a zero
 *       byte as a terminator. Fields can appear in any order. For each
 *       field there is the following:
 * Byte1 A code identifying the field type; if zero, this is the message
 *       terminator and no string follows. The presently defined field types
 *       are listed in Section 46.6. Since more field types might be added
 *       in future, frontends should silently ignore fields of unrecognized
 *       type.
 * String The field value.
 * </pre>
 *
 * @author mycat
 */
public class NoticeResponse extends PostgresPacket {

}
```

80:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\NotificationResponse.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * NotificationResponse (B)
 * Byte1('A') Identifies the message as a notification response.
 * Int32 Length of message contents in bytes,including self.
 * Int32 The process ID of the notifying backend process.
 * String The name of the channel that the notify has been raised on.
 * String The "payload" string passed from the notifying process.
 * </pre>
 *
 * @author mycat
 */
public class NotificationResponse extends PostgresPacket {

}
```

81:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\ParameterDescription.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * ParameterDescription (B)
 * Byte1('t') Identifies the message as a parameter description.
 * Int32 Length of message contents in bytes, including self.
 * Int16 The number of parameters used by the statement (can be zero).
 *       Then,for each parameter, there is the following:
```

```
 * Int32 Specifies the object ID of the parameter data type.
 * </pre>
 *
 * @author mycat
 */
public class ParameterDescription extends PostgresPacket {

}
```

82:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\ParameterStatus.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * ParameterStatus (B)
 * Byte1('S') Identifies the message as a run-time parameter status report.
 * Int32 Length of message contents in bytes,including self.
 * String The name of the run-time parameter being reported.
 * String The current value of the parameter.
 * </pre>
 *
 * @author mycat
 */
```

```
public class ParameterStatus extends PostgresPacket {

}
```

83:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Parse.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Parse (F)
 * Byte1('P') Identifies the message as a Parse command.
 * Int32 Length of message contents in bytes, including self.
 * String The name of the destination prepared statement (an empty string
 *        selects the unnamed prepared statement).
 * String The query string to be parsed.
 * Int16 The number of parameter data types specified (can be zero). Note
 *       that this is not an indication of the number of parameters that
 *       might appear in the query string, only the number that the frontend
 *       wants to prespecify types for. Then, for each parameter, there is
 *       the following:
 * Int32 Specifies the object ID of the parameter data type. Placing a zero
 *       here is equivalent to leaving the type unspecified.
 * </pre>
 *
 * @author mycat
 */
public class Parse extends PostgresPacket {

}
```

84:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\ParseComplete.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * ParseComplete (B)
 * Byte1('1') Identifies the message as a Parse-complete indicator.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class ParseComplete extends PostgresPacket {

}
```

85:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\PasswordMessage.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * PasswordMessage (F)
 * Byte1('p') Identifies the message as a password response. Note that this is
 *            also used for GSSAPI and SSPI response messages (which is really
 *            a design error, since the contained data is not a null-terminated
 *            string in that case, but can be arbitrary binary data).
 * Int32 Length of message contents in bytes, including self.
 * String The password(encrypted, if requested).
 * </pre>
 *
 * @author mycat
 */
public class PasswordMessage extends PostgresPacket {

}
```

86:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\PortalSuspended.java

```
/*
```

```
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * PortalSuspended (B)
 * Byte1('s') Identifies the message as a portal-suspended indicator. Note this
 *            only appears if an Execute message's row-count limit was reached.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class PortalSuspended extends PostgresPacket {

}
```

87:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\PostgresPacket.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * @see http://www.postgresql.org/docs/9.1/interactive/protocol.html
 * @author mycat
 */
public abstract class PostgresPacket {
    /**
     * <pre>
     * AuthenticationOk (B)
     * AuthenticationKerberosV5 (B)
     * AuthenticationCleartextPassword (B)
     * AuthenticationMD5Password (B)
     * AuthenticationSCMCredential (B)
     * AuthenticationGSS (B)
     * AuthenticationSSPI (B)
     * AuthenticationGSSContinue (B)
     * </pre>
     */
    public static final byte AUTHENTICATION = (byte) 'R';

    /**
     * BackendKeyData (B)
     */
    public static final byte BACKEND_KEY_DATA = (byte) 'K';

    /**
     * Bind (F)
     */
    public static final byte BIND = (byte) 'B';
```

```
/**
 * BindComplete (B)
 */
public static final byte BIND_COMPLETE = (byte) '2';

/**
 * CancelRequest (F)
 */

/**
 * Close (F)
 */
public static final byte CLOSE = (byte) 'C';

/**
 * CloseComplete (B)
 */
public static final byte CLOSE_COMPLETE = (byte) '3';

/**
 * CommandComplete (B)
 */
public static final byte COMMAND_COMPLETE = (byte) 'C';

/**
 * CopyData (F & B)
 */
public static final byte COPY_DATA = (byte) 'd';

/**
 * CopyDone (F & B)
 */
public static final byte COPY_DONE = (byte) 'c';

/**
 * CopyFail (F)
 */
public static final byte COPY_FAIL = (byte) 'f';

/**
 * CopyInResponse (B)
 */
public static final byte COPY_IN_RESPONSE = (byte) 'G';

/**
 * CopyOutResponse (B)
 */
```

```
public static final byte COPY_OUT_RESPONSE = (byte) 'H';

/**
 * CopyBothResponse (B)
 */
public static final byte COPY_BOTH_RESPONSE = (byte) 'W';

/**
 * DataRow (B)
 */
public static final byte DATA_ROW = (byte) 'D';

/**
 * Describe (F)
 */
public static final byte DESCRIBE = (byte) 'D';

/**
 * EmptyQueryResponse (B)
 */
public static final byte EMPTY_QUERY_RESPONSE = (byte) 'I';

/**
 * ErrorResponse (B)
 */
public static final byte ERROR_RESPONSE = (byte) 'E';

/**
 * Execute (F)
 */
public static final byte EXECUTE = (byte) 'E';

/**
 * Flush (F)
 */
public static final byte FLUSH = (byte) 'H';

/**
 * FunctionCall (F)
 */
public static final byte FUNCTION_CALL = (byte) 'F';

/**
 * FunctionCallResponse (B)
 */
public static final byte FUNCTION_CALL_RESPONSE = (byte) 'V';
```

```
/**
 * NoData (B)
 */
public static final byte NO_DATA = (byte) 'n';

/**
 * NoticeResponse (B)
 */
public static final byte NOTICE_RESPONSE = (byte) 'N';

/**
 * NotificationResponse (B)
 */
public static final byte NOTIFICATION_RESPONSE = (byte) 'A';

/**
 * ParameterDescription (B)
 */
public static final byte PARAMETER_DESCRIPTION = (byte) 't';

/**
 * ParameterStatus (B)
 */
public static final byte PARAMETER_STATUS = (byte) 'S';

/**
 * Parse (F)
 */
public static final byte PARSE = (byte) 'P';

/**
 * ParseComplete (B)
 */
public static final byte PARSE_COMPLETE = (byte) '1';

/**
 * PasswordMessage (F)
 */
public static final byte PASSWORD_MESSAGE = (byte) 'p';

/**
 * PortalSuspended (B)
 */
public static final byte PORTAL_SUSPENDED = (byte) 's';

/**
 * Query (F)
 */
public static final byte QUERY = (byte) 'Q';

/**
 * ReadyForQuery (B)
 */
public static final byte READY_FOR_QUERY = (byte) 'Z';

/**
 * RowDescription (B)
 */
public static final byte ROW_DESCRIPTION = (byte) 'T';

/**
 * SSLRequest (F)
 */

/**
 * StartupMessage (F)
 */

/**
 * Sync (F)
 */
public static final byte SYNC = (byte) 'S';

/**
 * Terminate (F)
 */
public static final byte TERMINATE = (byte) 'X';

private byte             type;
private int              length;

public byte getType() {
    return type;
}

public void setType(byte type) {
    this.type = type;
}

public int getLength() {
    return length;
}

public void setLength(int length) {
        this.length = length;
    }



}
```

88:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Query.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Query (F)
 * Byte1('Q') Identifies the message as a simple query.
 * Int32 Length of message contents in bytes, including self.
 * String The query string itself.
 * </pre>
 *
 * @author mycat
 */
public class Query extends PostgresPacket {

}
```

89:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\ReadyForQuery.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * ReadyForQuery (B)
 * Byte1('Z') Identifies the message type. ReadyForQuery is sent whenever the
 *            backend is ready for a new query cycle.
 * Int32(5) Length of message contents in bytes, including self.
 * Byte1 Current backend transaction status indicator. Possible values are 'I'
 *       if idle(not in a transaction block); 'T' if in a transaction block;
 *       or 'E' if in a failed transaction block (queries will be rejected until
 *       block is ended).
 * </pre>
 *
 * @author mycat
 */
public class ReadyForQuery extends PostgresPacket {

}
```

90:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\RowDe

```
scription.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * RowDescription (B)
 * Byte1('T') Identifies the message as a row description.
 * Int32 Length of message contents in bytes, including self.
 * Int16 Specifies the number of fields in a row (can be zero). Then, for
 *       each field,there is the following: String The field name.
 * Int32 If the field can be identified as a column of a specific table,
 *       the object ID of the table; otherwise zero.
 * Int16 If the field can be identified as a column of a specific table, the
 *       attribute number of the column; otherwise zero.
 * Int32 The object ID of the field's data type.
 * Int16 The data type size (see pg_type.typlen). Note that negative values
 *       denote variable-width types.
 * Int32 The type modifier (see pg_attribute.atttypmod). The meaning of the
 *       modifier is type-specific.
 * Int16 The format code being used for the field. Currently will be zero
 *       (text) or one (binary). In a RowDescription returned from the
 *       statement variant of Describe, the format code is not yet known and
 *       will always be zero.
 * </pre>
 *
 * @author mycat
 */
public class RowDescription extends PostgresPacket {

}
```

91:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\SSLRequest.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * SSLRequest (F)
 * Int32(8) Length of message contents in bytes, including self.
 * Int32(80877103) The SSL request code. The value is chosen to contain 1234 in
 *                 the most significant 16 bits, and 5679 in the least 16 significant
 *                 bits. (To avoid confusion, this code must not be the same as any
 *                 protocol version number.)
 * </pre>
 *
 * @author mycat
 */
public class SSLRequest extends PostgresPacket {
}

92:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Start
upMessage.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * StartupMessage (F)
 * Int32 Length of message contents in bytes, including self.
 * Int32(196608) The protocol version number. The most significant 16 bits are
 *               the major version number (3 for the protocol described here).
 *               The least significant 16 bits are the minor version number (0
 *               for the protocol described here). The protocol version number
 *               is followed by one or more pairs of parameter name and value
 *               strings. A zero byte is required as a terminator after the
 *               last name/value pair. Parameters can appear in any order. user
 *               is required, others are optional. Each parameter is specified as:
 * String The parameter name. Currently recognized names are:
 *        user The database user name to connect as. Required; there is no default.
 *        database The database to connect to. Defaults to the user name.
 *        options Command-line arguments for the backend. (This is deprecated in
 *                favor of setting individual run-time parameters.) In addition to
 *                 the above, any run-time parameter that can be set at backend start
 *                 time might be listed. Such settings will be applied during backend
 *                 start (after parsing the command-line options if any). The values
 *                 will act as session defaults.
 * String The parameter value.
 * </pre>
 *
 * @author mycat
 */
public class StartupMessage extends PostgresPacket {

}
```

93:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Sync.java

```
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Sync (F)
 * Byte1('S') Identifies the message as a Sync command.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class Sync extends PostgresPacket {

}

94:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\postgres\Termi
nate.java
/*
 * Copyright (c) 2013, OpenCloudDB/MyCAT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is free software;Designed and Developed mainly by many Chinese
 * opensource volunteers. you can redistribute it and/or modify it under the
 * terms of the GNU General Public License version 2 only, as published by the
 * Free Software Foundation.
 *
 * This code is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
 * version 2 for more details (a copy is included in the LICENSE file that
 * accompanied this code).
 *
 * You should have received a copy of the GNU General Public License version
 * 2 along with this work; if not, write to the Free Software Foundation,
 * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Any questions about this component can be directed to it's project Web address
 * https://code.google.com/p/opencloudb/.
 *
 */
package io.mycat.net.postgres;

/**
 * <pre>
 * Terminate (F)
 * Byte1('X') Identifies the message as a termination.
 * Int32(4) Length of message contents in bytes, including self.
 * </pre>
 *
 * @author mycat
 */
public class Terminate extends PostgresPacket {

}

95:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\SocketAcceptor.
```

```
java
package io.mycat.net;

public interface SocketAcceptor {

	void start();

	String getName();

	int getPort();

}
```

96:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\SocketConnector.java

```
package io.mycat.net;

public interface SocketConnector {

}
```

97:F:\git\java\shardingjdbc\Mycat-Server\src\main\java\io\mycat\net\SocketWR.java

```
package io.mycat.net;

import java.io.IOException;


public abstract class SocketWR {
	public abstract void asynRead() throws IOException;
	public abstract void doNextWriteCheck() ;
}
```