**Mycat 新一代Mysql分布式集群，大数据处理中间件，中国第一开源软件，欢迎志愿者参与：**

[**http://code.google.com/p/opencloudb/**](http://code.google.com/p/opencloudb/)

本文档由Mycat志愿者团队提供，转载请注明，谢谢。

**Checkout项目**

可以用eclipse的svn插件来进行项目检出，也可以用Tortoise SVN等工具检出，由于maven(M2)中的buildnumber-maven-plugin 中的SVNkit最高支持1.7的SVN仓库，因此当你用Tortoise SVN 1.8的工具或版本高于1.7的SVNKIT下载时，此插件无法获取本地仓库的SVN reversion，导致编译失败，此时你可以下载并安装svn命令行工具（安装完成后，在命令行运行svn，确认已经能执行，若无法执行，请手工在系统PATH路径中增加SVN的命令所在路径）并重启Eclipse，SVN命令行工具下载地址： <http://www.sliksvn.com/pub/> ，然后修改POM.XML文件，采用SVN命令来代替SVNKIT来获取版本号，将下面**2处**注释，将采用默认的svn命令。

 

**Eclispe中启动**

**入口程序是org.opencloudb.MycatStartup ，右键run as ，出现下面的界面，需要设置MYCAT\_HOME目录，为你当前的工程所在的目录（src\main）**



若启动报错，DirectBuffer内存不够，则可以再加 JVM 系统参数，-**XX:MaxDirectMemorySize=128M**

# 一：Mycat分片函数的扩展

下面说明分片函数的定义和配置，以Mycat中的例子为例，长整形范围自动分片规则**auto-sharding-long**的定义如下：

<tableRule name="**auto-sharding-long**">

 <rule>

 <columns>**id**</columns>

 <algorithm>**rang-long**</algorithm>

 </rule>

 </tableRule>

<function name="**rang-long**" class="**org.opencloudb.route.function.AutoPartitionByLong**">

 <property name="**mapFile**">autopartition-long.txt</property>

 </function>

上述表明：用字段id作为分片字段，并采用命名为**rang-long**的分片函数**： org.opencloudb.route.function.AutoPartitionByLong** 来完成分片映射，此分片函数有一个属性**mapFile**，配置为autopartition-long.txt，意思是从这个文件中加载分片定义：

# range start-end ,data node index

0-2000000=0

2000001-4000000=1

4000001-6000000=2

6000001-8000000=3

8000001-10000000=4

0到2000000的id分片到节点0，2000001到4000000的id分片到节点1，依次类推。

Mycat的分片函数的的目标只有一个：根据某个字段的值，返回其所对应的分片节点DataNode的index，index从零开始；若字段的值为非法，则返回NULL。

用户自定义扩展的分片函数要实现org.opencloudb.config.model.rule.RuleAlgorithm接口：

public interface RuleAlgorithm {

 void init();

 Integer calculate(String columnValue);

}

分片函数可以定义类的属性，并生成对应的set方法，当此函数被加载的时候，会从 rule.xml文件中找相关的属性，并注入（调用set方法），最后调用init()接口，完成初始化过程，这个过程可以实现资源文件的加载和数据解析。

 Calculate方法接受字段的值，并计算返回一个分片节点的index。

附：AutoPartitionByLong的例子

public class AutoPartitionByLong implements RuleAlgorithm {

 private String mapFile;

 private LongRange[] longRongs;

 @Override

 public void init() {

 initialize();

 }

 public void setMapFile(String mapFile) {

 this.mapFile = mapFile;

 }

 @Override

 public Integer calculate(String columnValue) {

 long value = Long.valueOf(columnValue);

 Integer rst = null;

 for (LongRange longRang : this.longRongs) {

 if (value <= longRang.valueEnd && value >= longRang.valueStart) {

 return longRang.nodeIndx;

 }

 }

 return rst;

 }

 private void initialize() {

 BufferedReader in = null;

 try {

 // FileInputStream fin = new FileInputStream(new File(fileMapPath));

 InputStream fin = this.getClass().getClassLoader()

 .getResourceAsStream(mapFile);

 if (fin == null) {

 throw new RuntimeException("can't find class resource file "

 + mapFile);

 }

 in = new BufferedReader(new InputStreamReader(fin));

 LinkedList<LongRange> longRangeList = new LinkedList<LongRange>();

 for (String line = null; (line = in.readLine()) != null;) {

 line = line.trim();

 if (line.startsWith("#") || line.startsWith("//"))

 continue;

 int ind = line.indexOf('=');

 if (ind < 0) {

 System.out.println(" warn: bad line int "+mapFile+ " :"+line);

 continue;

 }

 try {

 String pairs[] = line.substring(0, ind).trim().split("-");

 long longStart = Long.parseLong(pairs[0].trim());

 long longEnd = Long.parseLong(pairs[1].trim());

 int nodeId = Integer.parseInt(line.substring(ind + 1)

 .trim());

 longRangeList

 .add(new LongRange(nodeId, longStart, longEnd));

 } catch (Exception e) {

 }

 }

 longRongs = longRangeList.toArray(new LongRange[longRangeList

 .size()]);

 } catch (Exception e) {

 if (e instanceof RuntimeException) {

 throw (RuntimeException) e;

 } else {

 throw new RuntimeException(e);

 }

 } finally {

 try {

 in.close();

 } catch (Exception e2) {

 }

 }

 }

 static class LongRange {

 public final int nodeIndx;

 public final long valueStart;

 public final long valueEnd;

 public LongRange(int nodeIndx, long valueStart, long valueEnd) {

 super();

 this.nodeIndx = nodeIndx;

 this.valueStart = valueStart;

 this.valueEnd = valueEnd;

 }

 }

}

单元测试代码如下：

public class AutoPartitionByLongTest {

 @Test

 public void test()

 {

 AutoPartitionByLong autoPartition=new AutoPartitionByLong();

 autoPartition.setMapFile("autopartition-long.txt");

 autoPartition.init();

 String idVal="0";

 Assert.assertEquals(true, 0==autoPartition.calculate(idVal));

 idVal="2000000";

 Assert.assertEquals(true, 0==autoPartition.calculate(idVal));

 idVal="2000001";

 Assert.assertEquals(true, 1==autoPartition.calculate(idVal));

 idVal="4000000";

 Assert.assertEquals(true, 1==autoPartition.calculate(idVal));

 idVal="4000001";

 Assert.assertEquals(true, 2==autoPartition.calculate(idVal));

 }

}