|  |
| --- |
| package com.huawei.bigdata.hive.example;  import java.io.File;  import java.io.FileInputStream;  import java.io.IOException;  import java.io.InputStream;  import java.sql.SQLException;  import java.util.Properties;  import com.huawei.bigdata.security.LoginUtil;  import org.apache.commons.logging.Log;  import org.apache.commons.logging.LogFactory;  import org.apache.hadoop.conf.Configuration;  import static org.apache.hadoop.fs.CommonConfigurationKeysPublic.HADOOP\_SECURITY\_AUTHENTICATION;  import static org.apache.hadoop.fs.CommonConfigurationKeysPublic.HADOOP\_SECURITY\_AUTHORIZATION;  /\*\*  \* This class is providing simple example code for using hive  \*/  public class ExampleMain {  private final static Log LOG = LogFactory.getLog(ExampleMain.class.getName());  private static String CONF\_DIR = System.getProperty("user.dir") + File.separator + "conf" + File.separator;  private static final String ZOOKEEPER\_DEFAULT\_LOGIN\_CONTEXT\_NAME = "Client";  private static final String ZOOKEEPER\_DEFAULT\_SERVER\_PRINCIPAL = "zookeeper/hadoop";  //Hive client configuration, it is located in $HIVE\_CLIENT/config  public static final String HIVE\_CLIENT\_PROPERTIES = "hiveclient.properties";  private static String userName;  private static String userKeytabFile;  private static String krb5File;  private static Configuration conf;  public static void main(String[] args) {  ClientInfo clientInfo;  boolean isSecurityMode;  try {  clientInfo = new ClientInfo(CONF\_DIR + HIVE\_CLIENT\_PROPERTIES);  isSecurityMode = "KERBEROS".equalsIgnoreCase(clientInfo.getAuth());  init(isSecurityMode);  } catch (IOException e) {  LOG.error("Failed to login because ", e);  return;  }  JDBCExample jdbcExample = new JDBCExample(clientInfo, isSecurityMode);  try {  jdbcExample.run();  } catch (Exception e) {  LOG.error("failed to run jdbcExample, ", e);  }  }  private static void init(boolean isSecurityMode) throws IOException {  conf = new Configuration();  /\*\*  \* Other way to set conf for zk. If use this way,  \* can ignore the way in the 'login' method  \*/  if (isSecurityMode) {  userName = "hiveuser";  userKeytabFile = CONF\_DIR + "user.keytab";  krb5File = CONF\_DIR + "krb5.conf";  conf.set(HADOOP\_SECURITY\_AUTHENTICATION, "kerberos");  conf.set(HADOOP\_SECURITY\_AUTHORIZATION, "true");  /\*\*  \* One way for connect zk, Note: if this process  \* will connect more than one zk cluster, this way may be not proper. you  \* can contact us for more help  \*/  LoginUtil.setJaasConf(ZOOKEEPER\_DEFAULT\_LOGIN\_CONTEXT\_NAME, userName, userKeytabFile);  LoginUtil.setZookeeperServerPrincipal(ZOOKEEPER\_DEFAULT\_SERVER\_PRINCIPAL);  LoginUtil.login(userName, userKeytabFile, krb5File, conf);  }  }  }  /\*\*  \* Hive client info.  \*/  class ClientInfo {  //The zk quorum info, format like: ip1:port,ip2:port...  private String zkQuorum = null;  private String auth = null;  private String saslQop = null;  private String zooKeeperNamespace = null;  private String serviceDiscoveryMode = null;  private String principal = null;  private Properties clientInfo = null;  public ClientInfo(String hiveclientFile) throws IOException {  InputStream fileInputStream = null;  try {  clientInfo = new Properties();  File propertiesFile = new File(hiveclientFile);  fileInputStream = new FileInputStream(propertiesFile);  clientInfo.load(fileInputStream);  } catch (Exception e) {  throw new IOException(e);  } finally {  if (fileInputStream != null) {  fileInputStream.close();  fileInputStream = null;  }  }  initialize();  }  private void initialize() {  zkQuorum = clientInfo.getProperty("zk.quorum");  auth = clientInfo.getProperty("auth");  saslQop = clientInfo.getProperty("sasl.qop");  zooKeeperNamespace = clientInfo.getProperty("zooKeeperNamespace");  serviceDiscoveryMode = clientInfo.getProperty("serviceDiscoveryMode");  principal = clientInfo.getProperty("principal");  }  public String getZkQuorum() {  return zkQuorum;  }  public String getSaslQop() {  return saslQop;  }  public String getAuth() {  return auth;  }  public String getZooKeeperNamespace() {  return zooKeeperNamespace;  }  public String getServiceDiscoveryMode() {  return serviceDiscoveryMode;  }  public String getPrincipal() {  return principal;  }  } |

|  |
| --- |
| package com.huawei.bigdata.hive.example;  import java.io.IOException;  import java.util.Iterator;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.conf.Configured;  import org.apache.hadoop.hive.conf.HiveConf;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.WritableComparable;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.Mapper;  import org.apache.hadoop.mapreduce.Reducer;  import org.apache.hadoop.util.Tool;  import org.apache.hadoop.util.ToolRunner;  import org.apache.hive.hcatalog.data.DefaultHCatRecord;  import org.apache.hive.hcatalog.data.HCatRecord;  import org.apache.hive.hcatalog.data.schema.HCatSchema;  import org.apache.hive.hcatalog.mapreduce.HCatInputFormat;  import org.apache.hive.hcatalog.mapreduce.HCatOutputFormat;  import org.apache.hive.hcatalog.mapreduce.OutputJobInfo;  /\*\*  \* 概述：  \* 1)本示例演示了如何使用HCatalog提供的HCatInputFormat和HCatOutputFormat接口。  \* 2)本示例将演示从Hive表t1读取数据，并进行group by后再 load到表t2的详细实现细节,相当于以下HQL:  \* select col1, count(\*) from t1 group by col1;  \*  \* 使用前准备：  \* 1)创建源头表t1:  \* create table t1(col1 int);  \* 2)创建目的表t2:  \* create table t2(col1 int,col2 int);  \* 3)将工程打成jar包后上传。  \* 4)配置环境变量：  \*  \* export HADOOP\_HOME=<path\_to\_hadoop\_install>  \* export HCAT\_HOME=<path\_to\_hcat\_install>  \* export HIVE\_HOME=<path\_to\_hive\_install>  \* export LIB\_JARS=$HCAT\_HOME/share/hcatalog/hive-hcatalog-core-1.3.0.jar,  \* $HCAT\_HOME/lib/hive-metastore-1.3.0.jar,  \* $HCAT\_HOME/lib/hive-exec-1.3.0.jar,  \* $HCAT\_HOME/lib/libfb303-0.9.2.jar,  \* $HCAT\_HOME/lib/slf4j-api-1.7.5.jar  \* export HADOOP\_CLASSPATH=$HCAT\_HOME/share/hcatalog/hive-hcatalog-core-1.3.0.jar:  \* $HCAT\_HOME/lib/hive-metastore-1.3.0.jar:$HIVE\_HOME/lib/hive-exec-1.3.0.jar:  \* $HCAT\_HOME/lib/libfb303-0.9.2.jar:  \* $HADOOP\_HOME/etc/hadoop:$HIVE\_HOME/conf:  \* $HCAT\_HOME/lib/slf4j-api-1.7.5.jar  \*  \* 提交任务：  \* yarn --config $HADOOP\_HOME/etc/hadoop jar jar <path\_to\_jar> <main\_class> -libjars $LIB\_JARS t1 t2  \*  \* 参考资料：  \* 详细接口说明请参考：https://cwiki.apache.org/confluence/display/Hive/HCatalog+InputOutput  \*  \* \*/  public class HCatalogExample extends Configured implements Tool {  public static class Map extends  Mapper<LongWritable, HCatRecord, IntWritable, IntWritable> {  int age;  @Override  protected void map(  LongWritable key,  HCatRecord value,  Context context)  throws IOException, InterruptedException {  age = (Integer) value.get(0);  context.write(new IntWritable(age), new IntWritable(1));  }  }  public static class Reduce extends Reducer<IntWritable, IntWritable,  IntWritable, HCatRecord> {  @Override  protected void reduce(  IntWritable key,  Iterable<IntWritable> values,  Context context)  throws IOException, InterruptedException {  int sum = 0;  Iterator<IntWritable> iter = values.iterator();  while (iter.hasNext()) {  sum++;  iter.next();  }  HCatRecord record = new DefaultHCatRecord(2);  record.set(0, key.get());  record.set(1, sum);  context.write(null, record);  }  }  public int run(String[] args) throws Exception {  HiveConf.setLoadMetastoreConfig(true);  Configuration conf = getConf();  String[] otherArgs = args;  String inputTableName = otherArgs[0];  String outputTableName = otherArgs[1];  String dbName = "default";  @SuppressWarnings("deprecation")  Job job = new Job(conf, "GroupByDemo");  HCatInputFormat.setInput(job, dbName, inputTableName);  job.setInputFormatClass(HCatInputFormat.class);  job.setJarByClass(HCatalogExample.class);  job.setMapperClass(Map.class);  job.setReducerClass(Reduce.class);  job.setMapOutputKeyClass(IntWritable.class);  job.setMapOutputValueClass(IntWritable.class);  job.setOutputKeyClass(WritableComparable.class);  job.setOutputValueClass(DefaultHCatRecord.class);  OutputJobInfo outputjobInfo = OutputJobInfo.create(dbName,outputTableName, null);  HCatOutputFormat.setOutput(job, outputjobInfo);  HCatSchema schema = outputjobInfo.getOutputSchema();  HCatOutputFormat.setSchema(job, schema);  job.setOutputFormatClass(HCatOutputFormat.class);  return (job.waitForCompletion(true) ? 0 : 1);  }  public static void main(String[] args) throws Exception {  int exitCode = ToolRunner.run(new HCatalogExample(), args);  System.exit(exitCode);  }  } |

|  |
| --- |
| package com.huawei.bigdata.hive.example;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.ResultSetMetaData;  import java.sql.SQLException;  /\*\*  \* Simple example for hive jdbc.  \*/  public class JDBCExample {  private static final String HIVE\_DRIVER = "org.apache.hive.jdbc.HiveDriver";  private ClientInfo clientInfo;  private boolean isSecurityMode;  public JDBCExample(ClientInfo clientInfo, boolean isSecurityMode){  this.clientInfo = clientInfo;  this.isSecurityMode = isSecurityMode;  }  /\*\*  \*  \* @throws ClassNotFoundException  \* @throws SQLException  \*/  public void run() throws ClassNotFoundException, SQLException {  //Define hive sql, the sql can not include ";"  String[] sqls = {"CREATE TABLE IF NOT EXISTS employees\_info(id INT,name STRING)",  "SELECT COUNT(\*) FROM employees\_info", "DROP TABLE employees\_info"};  StringBuilder sBuilder = new StringBuilder(  "jdbc:hive2://").append(clientInfo.getZkQuorum()).append("/");  if (isSecurityMode) {  sBuilder.append(";serviceDiscoveryMode=")  .append(clientInfo.getServiceDiscoveryMode())  .append(";zooKeeperNamespace=")  .append(clientInfo.getZooKeeperNamespace())  .append(";sasl.qop=")  .append(clientInfo.getSaslQop())  .append(";auth=")  .append(clientInfo.getAuth())  .append(";principal=")  .append(clientInfo.getPrincipal())  .append(";");  } else {  sBuilder.append(";serviceDiscoveryMode=")  .append(clientInfo.getServiceDiscoveryMode())  .append(";zooKeeperNamespace=")  .append(clientInfo.getZooKeeperNamespace())  .append(";auth=none");  }  String url = sBuilder.toString();  Class.forName(HIVE\_DRIVER);  Connection connection = null;  try {  /\*\*  \* Get JDBC connection, If not use security mode, need input correct username,  \* otherwise, wil login as "anonymous" user  \*/  connection = DriverManager.getConnection(url, "", "");  /\*\*  \* Run the create table sql, then can load the data if needed. eg.  \* "load data inpath '/tmp/employees.txt' overwrite into table employees\_info;"  \*/  execDDL(connection,sqls[0]);  System.out.println("Create table success!");  execDML(connection,sqls[1]);  execDDL(connection,sqls[2]);  System.out.println("Delete table success!");  }  finally {  if (null != connection) {  connection.close();  }  }  }  public static void execDDL(Connection connection, String sql)  throws SQLException {  PreparedStatement statement = null;  try {  statement = connection.prepareStatement(sql);  statement.execute();  }  finally {  if (null != statement) {  statement.close();  }  }  }  public static void execDML(Connection connection, String sql) throws SQLException {  PreparedStatement statement = null;  ResultSet resultSet = null;  ResultSetMetaData resultMetaData = null;  try {  statement = connection.prepareStatement(sql);  resultSet = statement.executeQuery();  /\*\*  \* Print the column name to console  \*/  resultMetaData = resultSet.getMetaData();  int columnCount = resultMetaData.getColumnCount();  for (int i = 1; i <= columnCount; i++) {  System.out.print(resultMetaData.getColumnLabel(i) + '\t');  }  System.out.println();  /\*\*  \* Print the query result to console  \*/  while (resultSet.next()) {  for (int i = 1; i <= columnCount; i++) {  System.out.print(resultSet.getString(i) + '\t');  }  System.out.println();  }  }  finally {  if (null != resultSet) {  resultSet.close();  }  if (null != statement) {  statement.close();  }  }  }  } |