使用 Oracle Logminer 同步 Demo

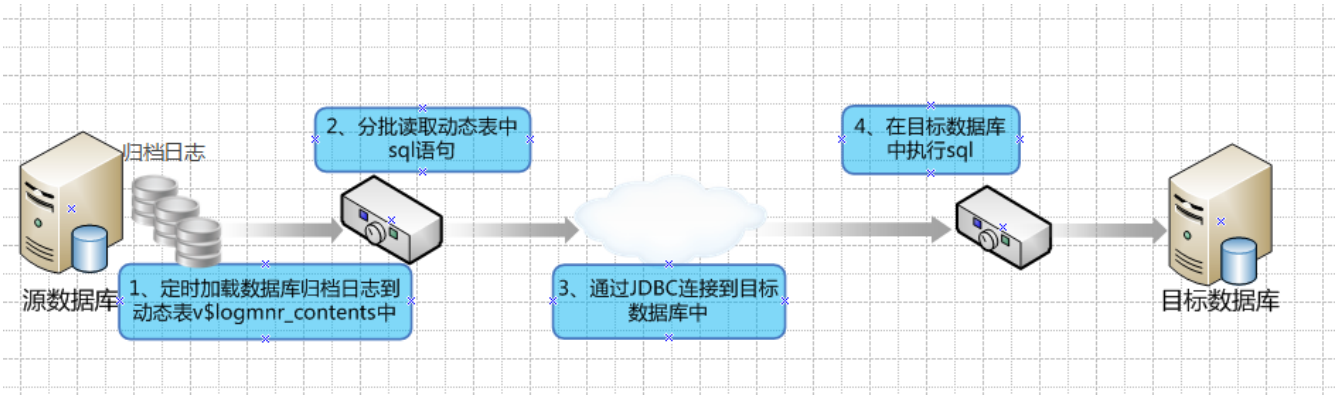
1 Demo 介绍

1.1 Demo 设想

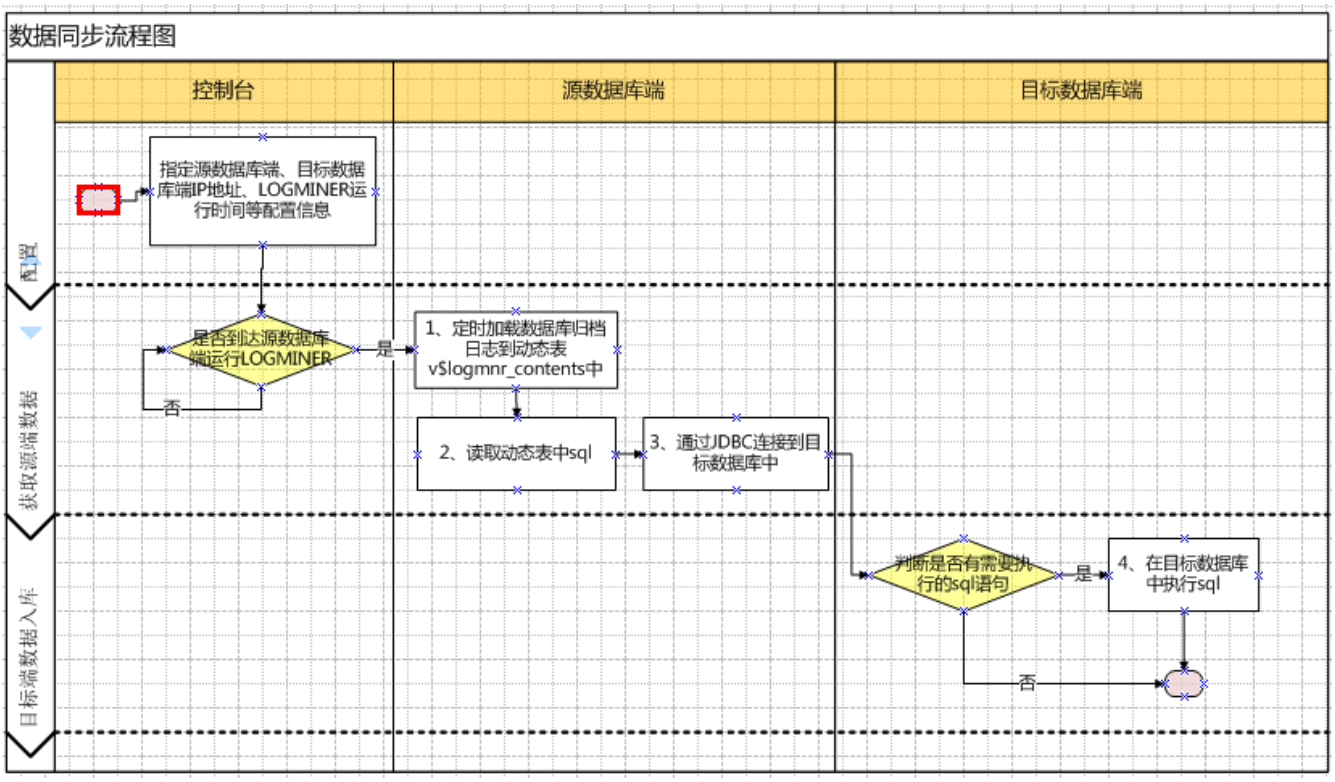
前面介绍了 Oracle LogMiner 配置使用以及使用 LogMiner 进行解析日志文件性能，在这篇文章中将利用 LogMiner 进行数据同步，实现从源目标数据库到目标数据库之间的数据同步。由于 LogMiner 支持的版本是 8.1 及以上 ,所以进行数据同步的 Oracle 数据库版本也必须是 8.1 及以上。

当然在本文中介绍的是 LogMiner 进行数据同步例子 ,也可以利用 LogMiner 进行数据审计、数据操作追踪等功能，由于这些从操作原理来说是一致，在本文不做讨论。

1.2 框架图



1.3 流程图



- 配置阶段

  1、 控制端：指定源端、目标端数据库信息、LOGMINER 同步时间等配置信息;

- 获取源端同步数据

  2、 控制台：通过定时轮询的方式检测是否到达数据同步时间，如果是则进行数据同步，否则继续进行轮询；

  3、 源数据库：定时加载数据库归档日志文件到动态表 v$logmnr_contents 中；

  4、 源数据库：根据条件读取指定 sql 语句；

- 目标端数据入库

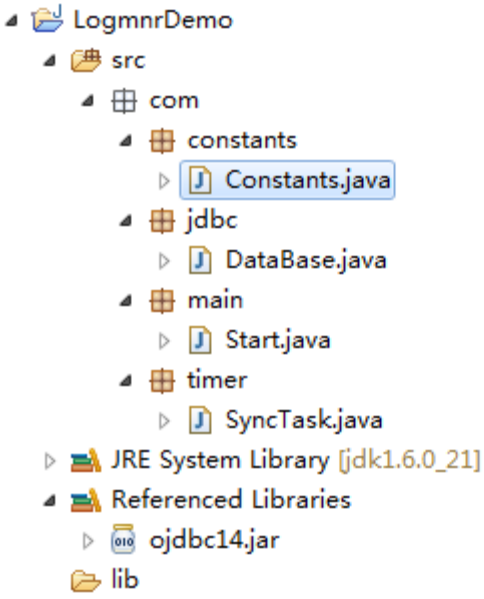
  5、 源数据库：执行 sql 语句。

2 代码分析

2.1 目录及环境配置

在该 Demo 项目中需要引入 Oracle JDBC 驱动包，具体项目分为四个类：

1. Start.java：程序入口方法；
2. SyncTask.java：数据同步 Demo 核心，生成字典文件和读取日志文件、目标数据库执行 SQL 语句等；
3. DataBase.java：数据库操作基础类；
4. Constants.java：源数据库、目标数据库配置、字典文件和归档文件路径。



2.2 代码分析

2.2.1 Constants.java

在该类中设置了数据同步开始 SCN 号、源数据库配置、目标数据库配置以及字典文件/日志文件路径。需要注意的是在源数据库配置中有两个用户：一个是调用 LogMiner 用户，该用户需要拥有 dbms_logmnr、dbms_logmnr_d 两个过程权限，在该 Demo 中该用为为 sync；另外一个为 LogMiner 读取该用户操作 SQL 语句，在该 Demo 中该用为为 LOGMINER。

```
package com.constants;

/**
 * [Constants]|描述:Logminer配置参数
 * @作者: ***
 * @日期: 2013-1-15 下午01:53:57
 * @修改历史:
 */
public class Constants {

    /** 上次数据同步最后SCN号 */
    public static String LAST_SCN = "0";
```

```
    /** 源数据库配置 */
    public static String DATABASE_DRIVER="oracle.jdbc.driver.OracleDriver";
    public static String
SOURCE_DATABASE_URL="jdbc:oracle:thin:@127.0.0.1:1521:practice";
    public static String SOURCE_DATABASE_USERNAME="sync";
    public static String SOURCE_DATABASE_PASSWORD="sync";
    public static String SOURCE_CLIENT_USERNAME = "LOGMINER";

    /** 目标数据库配置 */
    public static String SOURCE_TARGET_URL="jdbc:oracle:thin:@127.0.0.1:1521:target";
    public static String SOURCE_TARGET_USERNAME="target";
    public static String SOURCE_TARGET_PASSWORD="target";

    /** 日志文件路径 */
    public static String LOG_PATH = "D:\\oracle\\oradata\\practice";

    /** 数据字典路径 */
    public static String DATA_DICTIONARY = "D:\\oracle\\oradata\\practice\\LOGMNR";
}
```

2.2.2 SyncTask.java

在该类中有两个方法，第一个方法为 createDictionary，作用为生成数据字典文件，另外一个是 startLogmur，该方法是 LogMiner 分析同步方法。

```
/**
 * <p>方法名称: createDictionary|描述: 调用logminer生成数据字典文件</p>
 * @param sourceConn 源数据库连接
 * @throws Exception 异常信息
 */
public void createDictionary(Connection sourceConn) throws Exception{
    String createDictSql = "BEGIN dbms_logmnr_d.build(dictionary_filename =>
'dictionary.ora', dictionary_location =>'"+Constants.DATA_DICTIONARY+"'); END;";
    CallableStatement callableStatement = sourceConn.prepareCall(createDictSql);
    callableStatement.execute();
}


/**
 * <p>方法名称: startLogmur|描述:启动logminer分析 </p>
 * @throws Exception
 */
public void startLogmur() throws Exception{
```

```
    Connection sourceConn = null;
    Connection targetConn = null;
    try {
        ResultSet resultSet = null;

        // 获取源数据库连接
        sourceConn = DataBase.getSourceDataBase();
        Statement statement = sourceConn.createStatement();

        // 添加所有日志文件，本代码仅分析联机日志
        StringBuffer sbSQL = new StringBuffer();
        sbSQL.append(" BEGIN");
        sbSQL.append("
dbms_logmnr.add_logfile(logfilename=>'"+Constants.LOG_PATH+"\\REDO01.LOG',
options=>dbms_logmnr.NEW);");
        sbSQL.append("
dbms_logmnr.add_logfile(logfilename=>'"+Constants.LOG_PATH+"\\REDO02.LOG',
options=>dbms_logmnr.ADDFILE);");
        sbSQL.append("
dbms_logmnr.add_logfile(logfilename=>'"+Constants.LOG_PATH+"\\REDO03.LOG',
options=>dbms_logmnr.ADDFILE);");
        sbSQL.append(" END;");
        CallableStatement callableStatement = sourceConn.prepareCall(sbSQL+"");
        callableStatement.execute();

        // 打印获分析日志文件信息
        resultSet = statement.executeQuery("SELECT db_name, thread_sqn, filename FROM
v$logmnr_logs");
        while(resultSet.next()){
            System.out.println("已添加日志文件==>"+resultSet.getObject(3));
        }

        System.out.println("开始分析日志文件,起始scn号:"+Constants.LAST_SCN);
        callableStatement = sourceConn.prepareCall("BEGIN
dbms_logmnr.start_logmnr(startScn=>'"+Constants.LAST_SCN+"',dictfilename=>'"+Consta
nts.DATA_DICTIONARY+"\\dictionary.ora',OPTIONS
=>DBMS_LOGMNR.COMMITTED_DATA_ONLY+dbms_logmnr.NO_ROWID_IN_STMT);END;");
        callableStatement.execute();
        System.out.println("完成分析日志文件");

        // 查询获取分析结果
        System.out.println("查询分析结果");
        resultSet = statement.executeQuery("SELECT
scn,operation,timestamp,status,sql_redo FROM v$logmnr_contents WHERE
seg_owner='"+Constants.SOURCE_CLIENT_USERNAME+"' AND seg_type_name='TABLE' AND
operation !='SELECT_FOR_UPDATE'");
```

```
    // 连接到目标数据库，在目标数据库执行redo语句
    targetConn = DataBase.getTargetDataBase();
    Statement targetStatement = targetConn.createStatement();

    String lastScn = Constants.LAST_SCN;
    String operation = null;
    String sql = null;
    boolean isCreateDictionary = false;
    while(resultSet.next()){
        lastScn = resultSet.getObject(1)+"";
        if( lastScn.equals(Constants.LAST_SCN) ){
            continue;
        }

        operation = resultSet.getObject(2)+"";
        if( "DDL".equalsIgnoreCase(operation) ){
            isCreateDictionary = true;
        }

        sql = resultSet.getObject(5)+"";

        // 替换用户
        sql = sql.replace("\""+Constants.SOURCE_CLIENT_USERNAME+"\".", "");
        System.out.println("scn="+lastScn+",自动执行sql=="+sql+"");

        try {
            targetStatement.executeUpdate(sql.substring(0, sql.length()-1));
        } catch (Exception e) {
            System.out.println("测试一下,已经执行过了");
        }
    }

    // 更新scn
    Constants.LAST_SCN = (Integer.parseInt(lastScn))+"";

    // DDL发生变化，更新数据字典
    if( isCreateDictionary ){
        System.out.println("DDL发生变化，更新数据字典");
        createDictionary(sourceConn);
        System.out.println("完成更新数据字典");
        isCreateDictionary = false;
    }

    System.out.println("完成一个工作单元");

}
finally{
```

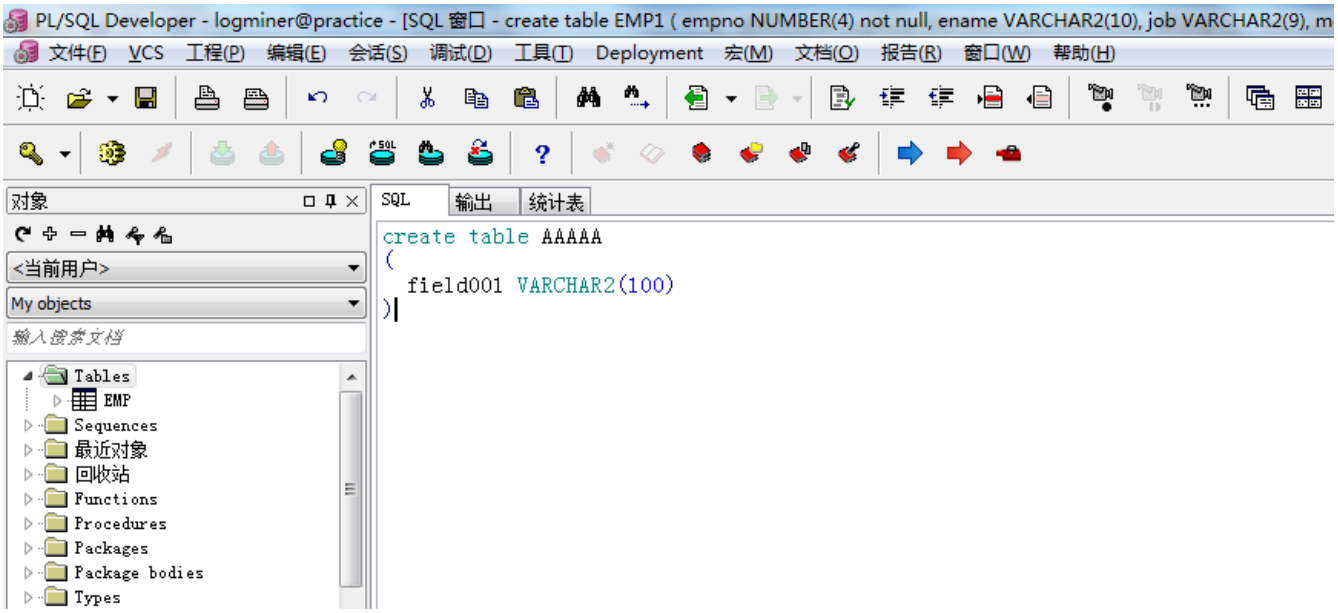
```
        if( null != sourceConn ){
            sourceConn.close();
        }
        if( null != targetConn ){
            targetConn.close();
        }

        sourceConn = null;
        targetConn = null;
    }
}
```

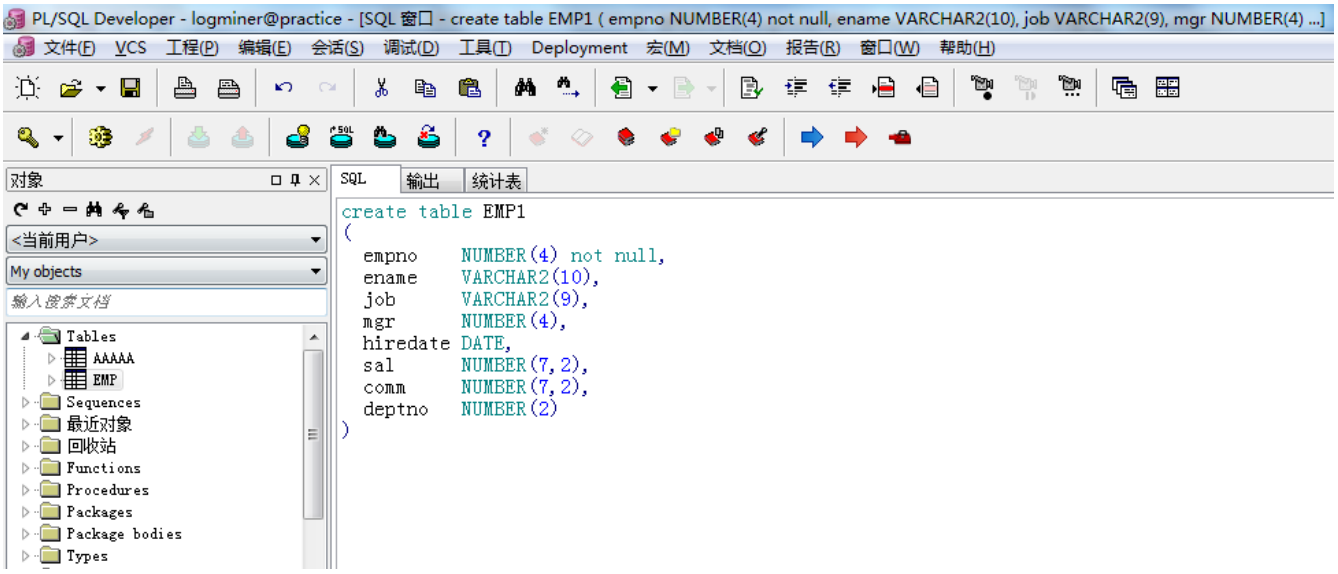
3 运行结果

3.1 源数据库操作

1、创建 AAAAA 表，并插入数据

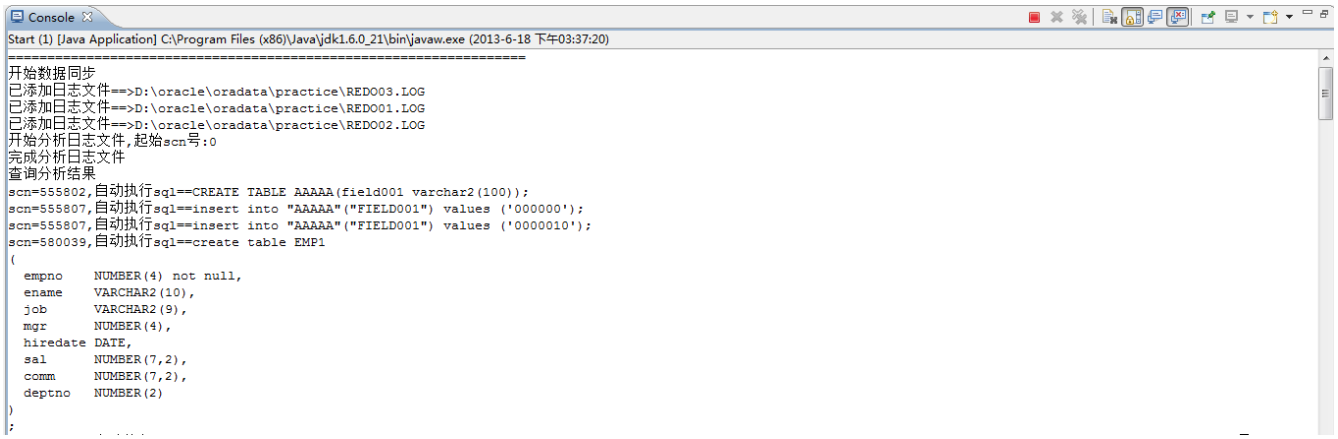


2、创建 EMP1 表



3.2 运行 Demo

在控制台中输出如下日志



3.3 目标数据库结果

创建 AAAAA 和 EMP1 表，并在 AAAAA 插入了数据