海数科技 Hadoop 大数据处理文献

Hadoop 数据迁移:从 Oracle 向 Hadoop 数据迁移

说明：

通过使用MapReduce的方式，使Hadoop可以直接访问Oracle，并将相关的数据写入到HDFS文件当中。从而可以顺利地将Oracle中的数据迁移到Hadoop文件系统中。

1、定义一个数据库信息类DBInfo

```
public class DBInfo {
    private String driverClass = "oracle.jdbc.driver.OracleDriver";
    private String ip;
    private String port;
    private String sid;
    private String username;
    private String password;
    private String tabName;
    private String condition;
    private String orderBy;
    private String[] fields;

    public String[] getFields() {
        return fields;
    }
    public void setFields(String[] fields) {
        this.fields = fields;
    }
    public String getDriverClass() {
        return driverClass;
```

```
}
public String getUrl() {
     StringBuffer url = new StringBuffer("jdbc:oracle:thin:@");
     url.append(this.ip).append(":");
     url.append(this.port).append(":");
     url.append(this.sid);
     return url.toString();
}
public String getIp() {
     return ip;
}
public void setIp(String ip) {
     this.ip = ip;
}
public String getPort() {
     return port;
}
public void setPort(String port) {
     this.port = port;
}
public String getSid() {
     return sid;
}
public void setSid(String sid) {
     this.sid = sid;
}
public String getUsername() {
     return username;
}
public void setUsername(String username) {
```

```
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getTabName() {
        return tabName;
    }
    public void setTabName(String tabName) {
        this.tabName = tabName;
    }
    public String getCondition() {
        return condition;
    }
    public void setCondition(String condition) {
        this.condition = condition;
    }
    public String getOrderBy() {
        return orderBy;
    }
    public void setOrderBy(String orderBy) {
        this.orderBy = orderBy;
    }
}
```

该类主要用于数据库访问对象的配置。

2、定义一个Recorder对象，该对象对应需要迁移的数据结构

```
import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Timestamp;

import java.text.ParseException;

import java.text.SimpleDateFormat;


import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.mapreduce.lib.db.DBWritable;


public class GPSRecorder implements DBWritable, WritableComparable {


    @Override

    public int compareTo(Object obj) {

        return -1;

    }


    private int id;

    private Timestamp revtime;

    private float longitude; // 经度

    private float latitude; // 纬度

    private String gpskey;

    private float direction;

    private float speed;

    private String data_serial;

    private int gps_mileage;
```

```
public int getId() {
    return id;
}

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

@Override
public void write(PreparedStatement stat) throws SQLException {
    stat.setInt(1, this.id);
    stat.setTimestamp(2, this.revtime);
    stat.setFloat(3, this.longitude);
    stat.setFloat(4, this.latitude);
    stat.setString(5, this.gpskey);
    stat.setFloat(6, this.direction);
    stat.setFloat(7, this.speed);
    stat.setString(8, this.data_serial);
    stat.setInt(9, this.gps_mileage);
}

@Override
public void readFields(ResultSet rs) throws SQLException {
    this.id = rs.getInt(1);
    this.revtime = rs.getTimestamp(2);
    this.longitude = rs.getFloat(3);
    this.latitude = rs.getFloat(4);
    this.gpskey = rs.getString(5);
    this.direction = rs.getFloat(6);
    this.speed = rs.getFloat(7);
```

```
        this.data_serial = rs.getString(8);
        this.gps_mileage = rs.getInt(9);
    }

    @Override
    public void write(DataOutput out) throws IOException {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        out.writeInt(this.id);
        out.writeUTF(format.format(this.revtime));
        out.writeFloat(this.longitude);
        out.writeFloat(this.latitude);
        out.writeUTF(this.gpskey);
        out.writeFloat(this.direction);
        out.writeFloat(this.speed);
        out.writeUTF(this.data_serial);
        out.writeInt(this.gps_mileage);
    }

    @Override
    public void readFields(DataInput in) throws IOException {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        this.id = in.readInt();
        this.revtime = Timestamp.valueOf(in.readUTF());
        this.longitude = in.readFloat();
        this.latitude = in.readFloat();
        this.gpskey = in.readUTF();
        this.direction = in.readFloat();
        this.speed = in.readFloat();
        this.data_serial = in.readUTF();
        this.gps_mileage = in.readInt();
```

```
    }

    public String values() {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        StringBuffer sb = new StringBuffer();
        sb.append(this.id).append(",");
        sb.append(format.format(this.revtime)).append(",");
        sb.append(this.longitude).append(",");
        sb.append(this.latitude).append(",");
        sb.append(this.gpskey).append(",");
        sb.append(this.direction).append(",");
        sb.append(this.speed).append(",");
        sb.append(this.data_serial).append(",");
        sb.append(this.gps_mileage);
        return sb.toString();
    }
}
```

定义一个与Oracle中表结构一致的对象，该Redorder用于进行数据的转换。

3、定义一个进行数据转换的Mapper

```
import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class GPSMapper extends
        Mapper<LongWritable, GPSRecorder, Text, NullWritable> {
    public void map(LongWritable key, GPSRecorder value, Context c)
```

```
            throws IOException, InterruptedException {
        Text KeyValue = new Text(value.values());
        c.write(KeyValue, NullWritable.get());
    }
}
```

定义一个Mapper类，该类从DataDrivenDBInputFormat中读入自定义的GPSRecorder数据。

通过该类型，可以转化为MapperReduce模型，将该数据转换为存储在HDFS的目标格式。

只有Mapper，没有Reduce。

4、定义一个访问驱动

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.db.DBConfiguration;
import org.apache.hadoop.mapreduce.lib.db.DataDrivenDBInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class GPSQueryData {
    public static void main(String[] args) throws Exception{
        DBInfo bi = new DBInfo();
        bi.setIp("192.168.1.19");  //Oracle服务器的IP
        bi.setPort("1521");   //Oracle服务的端口
        bi.setSid("oradb");  //服务器的SID
        bi.setUsername("demo");  //Schema的名称
        bi.setPassword("demo");  //用户密码
        String query = "select id,
revtime,longitude,latitude,gpskey,direction,speed,data_serial,"+
```

```
                    "gps_mileage from gps_hadoop partition(DY_GPS_HADOOP_20121213) where
rownum < 800 "+
                    "order by revtime";
        String inBound = "select count(*) from gps_hadoop partition(GPS_HADOOP_20121213) where
rownum < 800";
        //定义访问数据的信息，第一个SQL是数据查询，第二个SQL是数据计量，为Task的数量分配做
准备。

        Configuration conf = new Configuration();
        Job job = new Job(conf);
        job.setJarByClass(GPSQueryData.class);
        job.setMapperClass(GPSMapper.class);
        job.setReducerClass(Reducer.class);
        //定义Mapper及Reducer的类名称

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(NullWritable.class);
        job.setNumReduceTasks(1);
        //定义Maper的输出Key,Value

        FileOutputFormat.setOutputPath(job, new Path("/user/hadoop/outgps"));
        FileSystem hdfs;
        hdfs = FileSystem.get(conf);
        Path path = new Path("/user/hadoop/outgps");
        if (hdfs.exists(path)) {
            hdfs.delete(path, true);
        }
        //自动删除存在的文件夹

        DBConfiguration.configureDB(job.getConfiguration(),
```

```
                bi.getDriverClass(), bi.getUrl(), bi.getUsername(),
                bi.getPassword());
        DataDrivenDBInputFormat.setInput(job, GPSRecorder.class,query, inBound);
        //自定义SQL的数据查询
        job.waitForCompletion(true);
    }
}
```

关于Oracle驱动表：

可以使用ojdbc6.jar作为Oracle JDBC的驱动表，但在部署到Hadoop时需要注意。

当第一次运行时，即时该包已经部署到了${HADOOP_HOME}/lib文件夹下面，仍会出现找不到驱动包的错误。

有两种方式可以解决：

1、在每个节点下的${HADOOP_HOME}/lib下添加该包，重启集群。

2.a、把包传到集群上： hadoop fs -put mysql-connector-java-5.1.0- bin.jar /lib

2.b、在mr程序提交job前，添加语句：DistributedCache.addFileToClassPath(new Path("/lib/mysql-connector-java- 5.1.0-bin.jar"), conf);