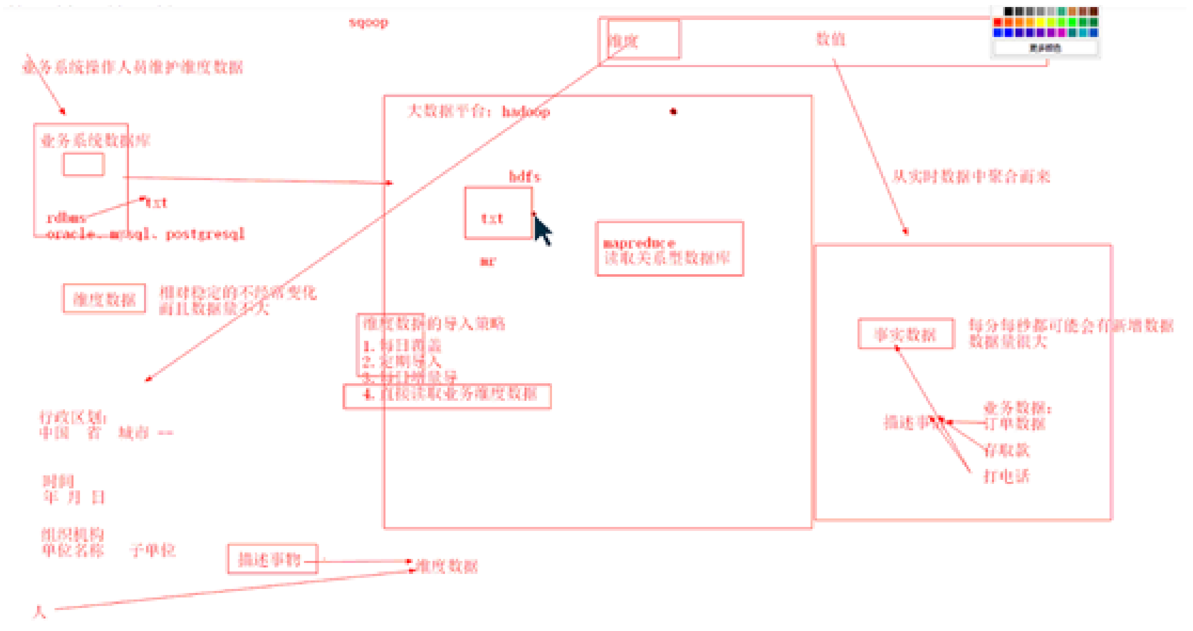
Day09_Hadoop读写关系型数据库

大数据-张军锋 Day09 Hadoop读写关系型数据库

事实数据和维度数据

> 维度数据相对稳定的，不经常变化，而且数据量不大，描述事物本身的数据.例如行政区划，时间(年月日)等等
>
> 事实数据是每分每秒都在变化，并且数据量很大描述事情的数据，例如：业务数据，订单数据，存取款等等



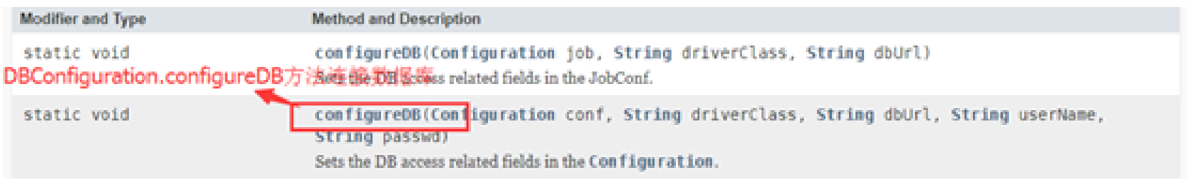
设置mysql远程调用

```
1.GRANT ALL PRIVILEGES ON *.* TO 'root'@'%'IDENTIFIED BY 'root' WITH GRANT
OPTION;
2.Flush PRIVILEGES;
```

Hadoop读写关系型数据库

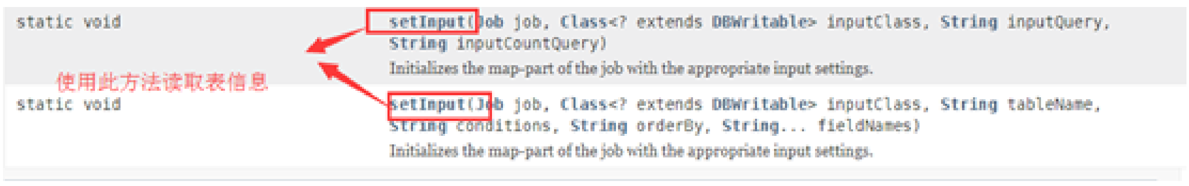
读数据库

1. 连接数据库



Modifier and Type	Method and Description
static void	configureDB(Configuration job, String driverClass, String dbUrl) Sets the DB access related fields in the JobConf.
static void	configureDB(Configuration conf, String driverClass, String dbUrl, String userName, String passwd) Sets the DB access related fields in the Configuration.

2. 获取表信息



static void	setInput(Job job, Class<? extends DBWritable> inputClass, String inputQuery, String inputCountQuery) Initializes the map-part of the job with the appropriate input settings.
static void	setInput(Job job, Class<? extends DBWritable> inputClass, String tableName, String conditions, String orderBy, String... fieldNames) Initializes the map-part of the job with the appropriate input settings.

3. 设置inputFormat为 DBinputWritable
4. Map的输入类型key：longWritable，value：DBWritable

不管是读取还是写入都需要我们自定义数据类型的，下面定义一个实现类实现DBWritable接口来从rdbms中获取的数据进行对接

```java
// 对应表word_count create table word_count(wc_word varchar(255),w
c_count integer)
public static class WordCountDBWritable implements DBWritable, Writ
able {
    private String word;
    private int count;
    public String getWord() {
        return word;
    }
    public void setWord(String word) {
        this.word = word;
    }
    public int getCount() {
        return count;
    }
    public void setCount(int count) {
        this.count = count;
    }

    @Override
    public String toString() {
        return "WordCountDBWritable [word=" + word + ", count=" + c
ount + "]";
    }

    // 将数据写入到数据库
    // insert into word_count(wc_word,wc_count) values(?,?)
    @Override
    public void write(PreparedStatement statement) throws SQLExcept
ion {
        statement.setString(1, this.word);
        statement.setInt(2, this.count);
    }

    // 从数据库中读数据
    @Override
    public void readFields(ResultSet resultSet) throws SQLException
{
        this.word = resultSet.getString("wc_word");
        this.count = resultSet.getInt("wc_count");
    }
    @Override
    public void write(DataOutput out) throws IOException {
        out.writeUTF(this.word);
        out.writeInt(this.count);
    }
    @Override
    public void readFields(DataInput in) throws IOException {
        this.word = in.readUTF();
        this.count = in.readInt();
    }
}
```

定义mapper

```
public static class WriteToDBMap extends Mapper<LongWritable, Text,
Text, IntWritable> {
    private final IntWritable ONE = new IntWritable(1);
    private String[] infos;
    private Text outKey = new Text();

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWri
table, Text, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        infos = value.toString().split("\\s");
        for (String word : infos) {
            outKey.set(word);
            context.write(outKey, ONE);
        }
    }
}
```

定义reducer

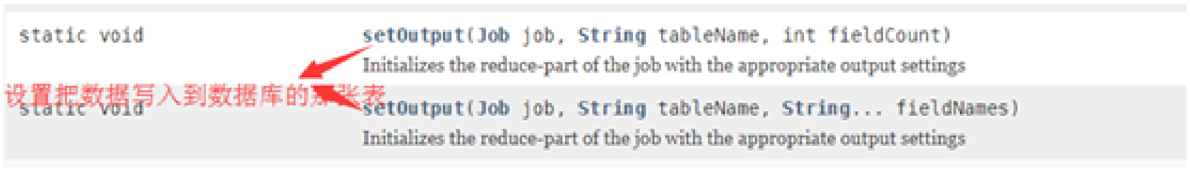
```
public static class WriteToDBReducer extends Reducer<Text, IntWrita
ble, WordCountDBWritable, NullWritable> {
    private WordCountDBWritable outKey = new WordCountDBWritable();
    private NullWritable outValue = NullWritable.get();
    private int sum;
    @Override
    protected void reduce(Text key, Iterable<IntWritable> valuess,
            Reducer<Text, IntWritable, WordCountDBWritable, NullWri
table>.Context context)
            throws IOException, InterruptedException {
        sum = 0;
        for (IntWritable value : valuess) {
            sum += value.get();
        }
        // 设置输出数据到数据库
        outKey.setWord(key.toString());
        outKey.setCount(sum);
        context.write(outKey, outValue);
    }
}
```

> 定义job，执行程序

```
public static void main(String[] args) throws Exception {
    Configuration configuration = new Configuration();
    // 设置数据库连接
    DBConfiguration.configureDB(configuration, "com.mysql.jdbc.Driv
er","jdbc:mysql://192.168.6.170:3306/xs","root","root");

    Job job = Job.getInstance(configuration);
    job.setJarByClass(WriteToDB.class);
    job.setJobName("将数据写入到mysql数据库");
    job.addFileToClassPath(new Path("/mysql-connector-java-5.1.39.j
ar"));


    job.setMapperClass(WriteToDBMap.class);
    job.setReducerClass(WriteToDBReducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setOutputKeyClass(WordCountDBWritable.class);
    job.setOutputValueClass(NullWritable.class);

    // 设置输入
    FileInputFormat.addInputPath(job, new Path("/README.txt"));
    // 设置输出
    DBOutputFormat.setOutput(job, "word_count", 2);
    System.exit(job.waitForCompletion(true) ? 0 : 1);

}
```

将数据写入到数据库

1. DBoutputFormat设置为job的输出格式
2. Reduce的key：DBWritable，value：随便写,不会写入数据库(一般使用NullWritable)
3. DBOutputFormat.setOutPut(),设置把数据写入到数据库的那张表
4. DBConfiguration.configureDB(),设置输出数据库的连接


static void setOutput(Job job, String tableName, int fieldCount)
Initializes the reduce-part of the job with the appropriate output settings

static void setOutput(Job job, String tableName, String... fieldNames)
Initializes the reduce-part of the job with the appropriate output settings

不管是读还是写，都是需要定义数据类型的，上面已经定义过了，在这里就不在啰嗦了

> 定义mapper

```
public static class ReadDBMap extends Mapper<LongWritable, WordCoun
tDBWritable, Text, NullWritable> {
    private Text outKey = new Text();
    private NullWritable outValue = NullWritable.get();

    @Override
    protected void map(LongWritable key, WordCountDBWritable value,
            Mapper<LongWritable, WordCountDBWritable, Text, NullWri
table>.Context context)
            throws IOException, InterruptedException {
        outKey.set(value.toString());
        context.write(outKey, outValue);
    }
}
```

> 定义reduce
> 我们是读取数据库中的数据，并不需要我们做什么处理，所以reducer就显的多与了，因为我们就不使用。mr中提供了 `job.setNumReduceTasks(0);` 来设置不使用reducer进行数据分析
>
> 定义job,执行程序

```
public static void main(String[] args) throws Exception {
    Configuration configuration = new Configuration();
    DBConfiguration.configureDB(configuration, "com.mysql.jdbc.Driv
er", "jdbc:mysql://192.168.6.170:3306/xs", "root",
            "root");

    Job job = Job.getInstance(configuration);
    job.setJarByClass(ReadDB.class);
    job.setJobName("读数据库");

    job.setMapperClass(ReadDBMap.class);
    // 因为我们不使用reduce，将reduce设置为0
    job.setNumReduceTasks(0);

//      job.addFileToClassPath(new Path("/mysql-connector-java-
5.1.39.jar"));
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    DBInputFormat.setInput(job, WordCountDBWritable.class, "select
* from word_count", "SELECT COUNT(*) FROM word_count");
//      DBInputFormat.setInput(job, WordCountDBWritable.class, "wor
d_count", "", "wc_count", "wc_word", "wc_count");
    Path outputDir = new Path("/bd14/ReadDB");
    outputDir.getFileSystem(configuration).delete(outputDir, true);
    FileOutputFormat.setOutputPath(job, outputDir);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```