DataFactory 使用手册

作者：牧心牛（QQ：2480102119）

一、概述

在项目开发测试过程中，准备测试数据一直是一个强需求，而且为了准备合适的数据，往往需要耗费大家不少的时间。DataFactory 是本人开发的一款开源工具，它致力于一劳永逸地解决这一问题以节省项目时间。它可以按需配置，生成支持所有提供了 JDBC 驱动的数据库的测试数据。

二、运行要求

DataFactory 对操作系统、内存、磁盘、数据库等基本无限制，仅要求有 JDK 或 JRE 1.8 及以上版本。

三、运行方式

1. 源代码方式

如果下载源代码，请自行编译，入口为 com.lexsoft.DataFactory ，需要传入一个参数，即：配置文件路径。

2. 可执行包方式

下载后解压缩到任意目录，在命令行运行 makedata.sh ，同样需传入配置文件路径。

3. 生成配置文件模板

命令行运行时传入三个参数，第一个参数为 new，第二个参数为数据库类型，可选：orcle,mysql,postgresql,gauss,sqlserver,db2,sqlite，或者用*代表全部数据库，第三个参数为生成的文件路径。例如：makedata.sh new mysql /tmp/mysql.xml

四、详细配置

所有的造数配置均采用 xml 方式进行配置，配置模板可参见 config.xml，下面对其中的全部配置项进行详细说明。

1. 概览

根节点为**<config>**

根节点下包含以下几类子节点：

- <options>　　参数配置
- <drivers>　　数据库驱动配置
- <connections>　　数据库连接配置
- <tasks>　　任务配置
- <tables>　　表配置

2. 参数设置

1）**配置格式**

<options>节点下为具体参数配置，它可以包含多个<param_name>标签，具体格式为：

```
<options>
    <param1 value="xxx"/>
    <param2>xxx</param2>
    <param3 value="xxx">yyy</param3>
</options>
```

在解析配置时，会自动认为上述三个节点分别配置了三个参数 param1、param2 和 param3，其值均为 xxx，即 value 属性的优先级高于节点内容。

2）**可配置参数列表**

参数名	含义	可选值及缺省值
drop	若表存在，开始时 drop 掉现有同名表	true,false

create	若表不存在，开始时 create 相应的表	**true**,false
truncate	开始时 truncate 相应表数据	**true**,false
threads	最大任务并发线程数	1
batch	批量插入时多少行记录提交一次	2000
count	单表最大造数行数	10,000
path	输出为文件时保存路径，输出文件名同表名，当文件进行了拆分了，文件名为 TableName_序号	当前目录
split_count	输出为文件时单文件最大记录行数	0(不拆分)
split_size	输出文件时单文件最大字符数，二者满足一个即拆分	0(不拆分)
field_delim	字段分隔符（部分模式使用）	英文逗号
record_delim	记录分隔符（部分模式使用）	换行符
nullas	如何记录空值（部分模式使用）	\N
autoshow	是否执行过程中自动显示进度	**true**,false
interval	自动显示进度的时间间隔，单位毫秒	30000
debug	是否调试模式，可显示较多输出信息	true,**false**

3）**参数优先级**

<options>可以出现在三个位置：

- 作为根节点<config>的子节点，以下称为 global
- 作为任务<task>配置的子节点，以下称为 task
- 作为<table>配置的子节点，以下称为 table

若同名参数出现在多个位置，则取值优先级为：task > table > global

3. 驱动类设置

<drivers>节点下为不同数据库的驱动类配置，它可以包含多个<driver>标签，具体格式为：

```
<drivers>
    <driver id="db1">oracle.jdbc.driver.OracleDriver</driver>
</drivers>
```

id 为必填内容，后续通过 id 名称被 connection 引用。

驱动类名同样为必填内容，用于连接具体的数据库

当前样例模板中已经预置了 oracle、mysql、postgresql、gauss、sqlserver、db2、sqlite 等 7 种数据库配置，如需增加其它数据库，请参见增加对新数据库支持。

4. 数据库连接设置

<connections>节点下为具体的数据库连接配置，它可以包含多个**<connection>**标签，具体格式为：

```
<connections>
    <connection id="cn1">
    <driver>db1</driver>
    <url><![CDATA[jdbc:oracle:thin:@127.0.0.1:1531:orcl]]></url>
    <username>test</username>
    <password>pass@1</password>
    <pres>
        <pre>alter session set ...</pre>
    </pres>
    <posts>
        <post>alter session set ...</post>
    </posts>
</connection >
```

下面逐一进行说明：

1）**<driver>** 数据库驱动，其值为对应的数据库驱动配置项 id

2）**<url>** 数据库连接串

3）**<username>** 数据库用户名

4）**<password>** 数据库密码

5）**<pres>** 可包含多个**<pre>**子节点，每一个<pre>内容为一条可执行的 SQL

语句，在建立数据库连接后，会依次执行每一条<pre>里面指定的 SQL 语句

6）**<posts>** 可包含多个**<post>**子节点，每一个<post>内容为一条可执行的 SQL 语句，在造数完成后，会依次执行每一条<post>里面指定的 SQL 语句

5. 任务设置

<tasks>节点下为具体的待执行任务，它可以包含多个**<task>**标签，具体格式为：

```
<tasks>
    <task id="t1" exec="true" connection="cn1" method="insert">
        <tables>
            <table>t1</table>
            <table>t2</table>
        </tables>
        <options>
            <count>10000</count>
        </options>
    </task>
</tasks>
```

下面逐一进行说明：

1）**id** 任务 ID，用以输出日志

2）**exec** 是否执行该任务，必填，可选值：true，false

3）**connection** 任务使用的数据库连接 id，除 method="file"之外必填。

4）**method** 数据插入方式，详细请参加数据插入方式

5）**<tables>** 节点，可包含多个**<table>**节点，每个<table>节点指定一个要造数的表

6）**<options>** 节点，即任务级别的参数配置

6. 数据插入方式

在每一个具体的 task 中，基于性能或其它考虑，可选择不同的数据插入方式，目前支持的方式有：

1) insert

支持所有数据库类型，通过标准 jdbc 的 PreparedStatement，以批量插入的方式进行数据插入，批量大小由参数 batch 指定。

2) values

对于 mysql 数据库或任意基于 mysql 的数据库，可采用 insert values 多值的方式插入以提高效率，多值由参数 batch 指定。

对于 postgresql 或 gauss，也可以采用该方式，但实测该方式效率反而不如 insert。

3) load

对于 mysql 数据库或任意基于 mysql 且支持 load data 的数据库，可以用此方式指定用 load data 加载以提高效率，每次 load 数据大小由参数 batch 指定。

4) copy

对于 postgresql、gauss 或任何基于 postgresql 且支持 copy 的数据库，可采用此模式进行加载以提高效率。每次 copy 数据大小由参数 batch 指定。

5) bulk

对于 SQL Server 数据库，可采用此方式，以 bulk copy 的模式进行数据插入以提高效率，单次 bulk copy 大小由参数 batch 指定。

6) file

仅希望将数据输出到文件时，可以通过此方式实现，此时会用到五个参数：path、split_count、split_size、field_delim 和 record_delim。

7. 表设置

<tables>节点下为具体每一张表的配置，它可以包含多个**<table>**标签，每一

个对应一张表，具体格式为：

```
<tables>
    <table name="tab1">
        <options>
            <count>10000</count>
        </options>
        <column name="col1" type="long">
            ...
        </column>
        ...
        <keys>
            <key>primary key(col1)</key>
        </keys>
        <posts>
            <post>alter table tab1 add index idx_col2 (col2)</post>
        </posts>
    </table>
</tables>
```

下面逐一进行说明：

1）**name** 表名，可以是 table 名称，也可以使用 schema.table 格式

2）**options** 表级别的参数配置

3）**<column>** 表的每一字段配置，详细请参见字段设置，字段的配置顺序即建表后的字段顺序

4）**<keys>** 表的主键、外键等设置，格式为 create table 中的 key 子句，如果无需工具帮忙创建表，则此部分可以忽略

5）**<posts>** 包含多个**<post>**子句，每一个为一条完整的 SQL 语句，会自动在 create table 成功后依次执行，如果无需工具帮忙创建表，则此部分可以忽略

8. 字段设置

<column>节点为表中每一个字段的配置，简单示例如下：

```
<column name="col1" type="long" nullable="0.1">
    <dbtype>bigint not null</dbtype>
    <increment>
```

```
        <min>1</min>
        <max>100000000</max>
        <step>2</step>
    </increment>
</column>
```

下面我们以创建表 test 为例进行详细说明：

```
create table test (
    id bigint not null,
    name varchar(20) default 'nn',
    gender char(1) default 'M',
    phone varchar(11),
    primary key (id),
    unique key key_phone (phone)
)
```

1）**name**

表中字段名称，例如 id

2）**type**

字段类型，此处的类型仅为告诉造数程序制造何种类型数据，并不代表数据库中实际类型，支持的字段类型为：

a) **short** 短整数，对应数据库中 smallint 类数据

b) **int** 整数数据，可对应数据库中 int 类数据

c) **long** 长整数数据，可对应数据库中 bigint 类数据

d) **float** 单精度浮点数据，可对应数据库中 float/real 类数据

e) **double** 双精度浮点数据，可对应数据库中 double 类数据

f) **decimal** 完整精度数据，对应数据库中的 decimal 类数据

g) **char** 字符串数据，可对应数据库中 char/nchar/bpchar 类数据

h) **varchar** 字符串数据，可对应数据库中 varchar/text/clob 类数据

i) **blob** 字节类数据，可对应数据库中 blob/binary/bytea 类数据

j) **date** 日期数据，可对应数据库中 date 类数据（仅日期）

k) **time** 时间数据，可对应数据库中 time 类数据（仅时间）

l) **timestamp** 时间戳数据，可对应数据库中 timestamp 类数据

例如：id 对应的 type 为 long；gender 对应为 char

3）**nullable**

生成该字段数据为 null 的概率，可以指定为一个小于等于 1 的小数，如果指定值<=0 即代表不生成 null 值，缺省为不生成 null 数据。

4）**dbtype**

建表时使用的数据库类型及设置，例如:

id 对应的为 `bigint not null`

name 对应的为 `varchar(20) default 'nn'`

即对应各字段在 create table 语句中除去列名称的剩余内容。

如果无需工具创建表，则此部分可以忽略。

5）数据生成方式

针对每个字段，均支持 skip、fixed、random、increment、list、roulette 和 ratio 共七种不同的数据生成方式，不过，对于不同的字段类型，配置各有区别，详细请参见后文。

9. 数据生成方式详解

1）skip

即不生成该字段数据，仅 create table 时使用该字段。例如：

```
<column name="auto_id" type="long">
    <dbtype>bigint not null auto_increment</dbtype>
    <skip/>
</column>
```

2）**fixed**

永远返回一个固定值，下例中列出了各类型数据的格式，此处统一说明，后文不再赘述，同样为简化内容，后文举例时也不再写<column>和<dbtype>标签，仅写数据生成标签内容：

```
<column name="col1" type="long">
    <dbtype>bigint not null</dbtype>
    <fixed>1</fixed>
</column>
<column name="col2" type="char">
    <dbtype>char(4)</dbtype>
    <fixed>abcd</fixed>
</column>
<column name="col3" type="char">
    <dbtype>varchar(20)</dbtype>
    <fixed>0x303031ab</fixed>
</column>
<column name="col4" type="date">
    <dbtype>date</dbtype>
    <fixed>2020-01-01</fixed>
</column>
<column name="col5" type="time">
    <dbtype>time</dbtype>
    <fixed>01:02:03</fixed>
</column>
<column name="col6" type="timestamp">
    <dbtype>timestamp(6)</dbtype>
    <fixed>2020-01-01 01:02:03.123456</fixed>
</column>
<column name="col7" type="blob">
    <dbtype>bytea</dbtype>
    <fixed>abcd</fixed>
</column>
<column name="col8" type="blob">
    <dbtype>bytea</dbtype>
    <fixed>0x4142434445</fixed>
</column>
```

- char/blob 类型支持两种输入方式，一种是直接输入字符串，另一种是以

0x 开头的以 16 进制表示的字节值。工具会根据具体情况用 getBytes 或 new String 进行转换。

- 日期类型的格式固定为 yyyy-MM-dd
- 时间类型格式固定为 HH:mm:ss（24 小时制）
- 时间戳类型格式固定为 yyyy-MM-dd HH:mm:ss.S

3）**random**

在指定范围内随机生成。

对于 short、int、long、float、double、decimal 等数值类数据，具体配置为：

```
<random>
    <min>1</min>
    <max>1000</max>
    <candidates></candidates>
</random>
```

其中： `min` 为最小值，`max` 为最大值，`candidates` 此时无意义可无需指定。

对于 date、time、timestamp 等时间类数据，min 和 max 必须为对应的格式，同样不使用 candidates，例如对于一个 timestamp 字段：

```
<random>
    <min>2020-02-01 00:00:00</min>
    <max>2020-04-01 00:00:00</max>
    <candidates></candidates>
</random>
```

对于 char、varchar、blob 等字段，`min` 代表生成字符串或字节数组的最小长度，`max` 代表最大长度，`candidates` 代表从那些字符（或字节）中随机选取数据。此时可增加两个参数，`prefix` 代表生成字符串的前缀，`suffix` 代表生成字符串的后缀。因为生成随机长度字符串，因此不能保证一定会有前缀和后缀，当随机长度不足前缀或后缀长度时，优先保证前缀或后缀的生成。

```
<random>
    <min>1</min>
```

```
    <max>6</max>
    <prefix>HD</prefix>
    <suffix>S</suffix>
    <candidates>abcd</candidates>
</random>
```

其含义为随机生成最短 1 字节，最长 6 字节的字符串，可选字符为 abcd，前缀为 HD，后缀为 S，例如生成的数据可能为：

- 空字符串、S （随机长度为 0 或 1 时，不满足前缀长度但满足后缀长度）
- HD（随机长度为 2 时，满足前缀和后缀长度但不满足二者之和，优先前缀）
- HDS、HDacS（随机长度满足二者和时，优先前缀和后缀）

再如：

```
<random>
    <min>1</min>
    <max>4</max>
    <candidates>0x3031323234</candidates>
</random>
```

我们知道 0x30~0x39 分别对应字符 0~9，因此生成的字符串可能为：0，00，123，40 等等。

特别的，candidates 还支持一些预设的字符集合（为了 SQL 语句及数据库兼容性，0x00，0x0d，0x0a，0x2c，0x27 和 0x5c 这几个字符没有使用），它们是：

candidate 取值	字符集合
??id	生成满足校验规范的随机身份证号，此时 min 和 max 必须指定为一个日期格式，代表生日的取值范围，例如 min=1980-01-01。缺省为（1950-01-01 至 2020-01-01） 仅仅指定 ??id 则生成以 110000 开头的身份证号，亦可以用 ??id:110101,110102 这样的格式指定多个 6 位前缀，以生成指定地区的身份证号

??mobile	生成随机手机号，此时不使用 min 和 max。 仅仅指定??mobie 则生成以 138、133、185 或 165 开头的手机号，亦可以用 ??mobile:138,136 这样的格式指定多个 3 位前缀，以生成指定运营商号段的手机号
??uuid	生成随机的 UUID，此时不适用 min 和 max。
??ascii	仅使用可显示的 ascii 字符
??latin1	仅使用 latin1 字符集中的字符
??chn	仅使用简体中文
??gbk	仅使用中文（包括繁体）
??jpn	仅使用日文
??kor	仅使用韩文
??upper	仅使用大写字母
??lower	仅使用小写字母
??alpha	仅使用大小写字母
??number	仅使用数字
??word	仅使用大小写字母及数字

4）**increment**

在指定范围内按步长递增

对于 short、int、long、float、double、decimal 等数值类数据，具体配置为：

```
<increment>
    <start>1</start>
    <end>1000</end>
    <step>2</step>
</increment>
```

其中： `start` 为最小值，`end` 为最大值，`step` 为递增步长。

对于 date、time、timestamp 等时间类数据，start 和 end 必须为对应的格式，

例如对于一个 timestamp 字段：

```
 ncrement>
    <start>2020-02-01 00:00:00</start>
    <end>2020-04-01 00:00:00</end>
    <step>1d2h3m4s555</step>
</increment>
```

此处还需注意步长的指定，使用 d 代表天、h 代表小时、m 代表分钟、s 代表秒，最后不带单位的代表毫秒，此例中的步长即为：1 天 2 小时 3 分钟 4 秒 555 毫秒，再如 1d3m 代表 1 天 3 分钟，500 代表 500 毫秒，以此类推。

对于 char、blob 等无递增含义的字段，为了兼容性，当指定该种生成方式时，将会返回一个固定值。

5）**list**

在指定列表中顺序选择，具体格式如下：

```
<list>
    <item>1</item>
    <item>2</item>
    <item>3</item>
</list>
```

逐行生成数据时，会依次选择第一个元素、第二个元素、第 N 个元素，然后周而复始。

6）**roulette**

在指定列表中随机选择，具体格式如下：

```
<roulette>
    <item>1</item>
    <item>2</item>
    <item>3</item>
</roulette>
```

逐行生成数据时，每次会随机选择其中一个元素，相当于平均分布。不过由于省去了额外的计算，效率会略高于指定平均分布的 ratio 方式。

7）**ratio**

在指定列表中按概率选择，具体格式如下：

```
<ratio>
    <item ratio="30">Male</item>
    <item ratio="70">Female</item>
</ratio>
```

每个 item 都具有 ratio 属性，该属性指定了生成这个 item 的概率，首先在生成数据之前对所有元素的概率归一化，如上例中 Male 的概率为 30/(30+70)即 30%，同样计算的 Female 概率为 70%，然后生成每一行数据时按照每个元素的概率分布进行随机取值。

五、扩展应用

1. 增加对新数据库支持

    1) 引入目标数据的 jdbc 包，如果是下载的执行码，则将 jar 包放到 libs 目录下即可。

    2) 在 config 中配置 driver 标签，设置 jdbc 驱动类

    3) 在 connection 中配置 url 连接等参数

    4) 使用 insert 方式进行插入数据

2. 增加对新的字段类型支持

    扩展点为 ColumnType、Column 和 Config

3. 增加新的加载方式

    扩展点为 Job

4. 增加新的数据生成方式

    扩展点为 DataGenerator 以及相应的 Column 和 Config。