DB Stat User Guide

DB Stat 用户使用手册

北京东方联合投资管理有限公司

作者：甘强

修订历史

日期	版本	描述	作者
2013 年 8 月 16 日	1.0.0	草拟	甘强
2013 年 9 月 12 日	1.0.0	定稿	甘强
2013 年 9 月 27 日	1.1.0	新增“多实例化定时器”功能	甘强

目录

第 1 章 简介

随着“大数据”时代的来临，给传统的统计数据的生产方式带来很大的挑战。而当今企业中的重要数据基本存储在商业或开源的关系型数据库当中，如何将原始数据库中的数据统计到目标数据库中去，以方便前端应用查询已成为一个主流需求。

谁更适合阅读此文档？

如果你想使用 DB Stat，本文档更加适合懂 SQL 语法跟 Java 编程的人员阅读。

DB Stat 是一款通用的半自动化的数据库统计工具。它可以为开发者减少至少 50%的编码工作，让开发者更加关注统计工作的业务逻辑。何谓半自动化？就是不能直接使用，需要人工配置或编程。

> *“好的架构必须使每个关注点相互分离，也就是说系统中的一部分发生了改变，不会影响其他部分。即使需要改变，也能够清晰地识别出哪些部分需要改变。如果需要扩展架构，影响将会最小化。已经可以工作的每个部分都将继续工作。”*
>
> *--- Ivar Jacobson*

第 2 章 功能特点

DB Stat 功能流程图

其中，统计规则具体表示为 SQL 语句。

DB Stat 具体特点

- 跨数据源。
- 支持不同数据库类型。
- 默认多线程、异步。
- 支持可配置。
- 支持可编程接口。
- 插件式功能扩展。

“Given enough eyeballs, all bugs are shallow.”

--- Eric Steven Raymond

第 3 章 如何使用

使用步骤

3.1 准备

- 操作系统：Unix、Linux。
- JDK：6.0 或者以上
- 数据库：Mysql 或者 Oracle
- 内存：2G 或者以上
- 硬盘：1G 或者以上

3.2 安装

注意：首先要安装 JDK 1.6 环境

解压缩 db-stat-[version].tar.gz

```
#tar zxvf db-stat-[version].tar.gz
```

3.3 配置

主要是针对 db-stat-[version]/conf/db-stat.xml 文件进行配置，具体配置说明将会在下一章进行详细说明。

3.4 启动

```
#cd db-stat-[version].tar.gz

#./bin/startup.sh
```

3.5 停止

```
#cd db-stat-[version].tar.gz

#./bin/shutdown.sh
```

> *“Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other,with no structural integrity， but just done by brute force and thousands of slaves.”.*
>
> *--- Alan Kay*

第 4 章 配置

当你执行 db-stat 启动命令之前需要进行人工配置，即：主要是通过对 dbstat-[version]/conf/db-stat.xml 文件进行配置。配置文件主要分为三大部分：定时器(Timer)、读取端(Reader)、写入端(Writer)。

4.1 定时器

定时器主要功能是定时执行程序一次或者 N 次。以下配置为一个实际配置的例子：

```
<timer>
  <instance id=”t1”>
     <interval>24*60*60*1000</interval>
     <starttime>2013-09-26 12:33:00</starttime>
  </instance>
  <instance id=”t2”>
     <interval>30*24*60*60*1000</interval>
     <starttime>2013-09-26 12:33:00</starttime>
  </instance>
</timer>
```

具体 XML 配置参数

元素名称	类型	说明	是否必须
instance	标签类型	代表一个 timer 实例，实例化个数范围是 1---N。reader 会共享使用实例化出的 timer	是
id	标签类型	instance 标签的属性，其值必须保证唯一。	是

interval	Long 类型	每次执行的时间间隔。其值为一个正整数，并且支持数字的加、减、乘、除操作。 它主要用于需要**重复**统计的案例中。例如：可以统计日、周、月、年等时间维度。 单位为：毫秒 注意：当值为 0 时或者没有配置该属性时，表示只是统计一次。	否
starttime	Date 类型	开始执行统计的时间。当此值小于当前服务器的时间会报错。 格式： yyyy-MM-dd HH:mm:ss	是

4.2 读取端

该部分配置包括源端数据源信息、统计的 SQL 语句。以下为一个实际配置的例子：

```
<reader>
  <datasource>
      <driverclassname>
         com.mysql.jdbc.Driver
       </driverclassname>
       <url>jdbc:mysql://localhost:3306/test</url>
       <username>root</username>
       <password>root</password>
       <initpoolsize>10</initpoolsize>
  </datasource>
```

```
  <select>
      <sql id="1" tid=”t1”>
        <value>
          select count(*) as UV,name from AAA where
          date=#{yesterday}
         </value>
      </sql>
      <sql id="2" tid=”t2”>

<class>com.ncfgroup.dbstat.cases.OrderSelectStat</class>
      </sql>
       <sql refid="3">
        <value>
              Select name from AAA where id=1
         </value>
      </sql>
     <sql refid="4">
        <value>
           Select id from AAA where code='0001'
         </value>
      </sql>
   </select>
</reader>
```

具体 XML 配置参数

元素名称	类型	说明	是否必须
reader	标签类型	该标签只包含一个 datasource 子标签跟一个 select 子标签。	是
datasource	标签类型	表示源端数据源	是
driverclassname	String 类型	JDBC 驱动名称。 注意：当前版本只支持	是

		mysql 和 oracle。	
url	String 类型	数据库地址。	是
username	String 类型	数据库用户名。	是
password	String 类型	数据库密码。	是
initpoolsize	Int 类型	数据库连接池大小，如果不填，默认为 10	否
select	标签类型	其子标签<sql>中的内容只能写 select 语句。	是
sql	标签类型	sql 子标签可为任意多个且只能写 Select sql。	是
id	String 类型	sql 子标签的属性，其值必须保证唯一。	是
tid	String 类型	sql 子标签的属性，其值必须保证是 timer 标签中的 instance 的 id 值。代表该查询 sql 执行的具体 timer。	是
value	String 类型	里面写纯 SQL、扩展 SQL。 注意：使用该标签时不能使用 class 标签。	否
class	String 类型	里面写扩展的 java 类路径。当纯 SQL 配置、扩展 SQL 配置(内置函数)不够用时使用它。	否

		注意：使用该标签时不能使用 value 标签。	

4.3 写入端

该部分配置包括目标端数据源信息、统计的 SQL 语句。以下为一个实际配置的例子：

```
<writer>
  <datasource>
       <driverclassname>
        com.mysql.jdbc.Driver
       </driverclassname>
       <url>jdbc:mysql://localhost:3306/test</url>
       <username>root</username>
       <password>root</password>
       <initpoolsize>10</initpoolsize>
  </datasource>
  <insert>
      <sql refid="1">
        <value>
          insert into HOUR_DASHBOARD(id,pv,uv)
          values ('#{uuid}', '${pv}', '${uv}')
        </value>
      </sql>
      <sql refid="2">
     <class>
        com.ncfgroup.dbstat.cases.OrderInsertStat
    </class>
      </sql>
   </insert>
   <update>
      <sql refid="3">
        <value>
        UPDATE Person SET Address = 'Zhongshan 23'
```

```
        WHERE LastName = '${name}'
        </value>
      </sql>
    </update>
    <delete>
      <sql refid="4">
        <value>
           DELETE FROM Person WHERE id= '${id}'
        </value>
      </sql>
    </ delete>
</writer>
```

具体 XML 配置参数

元素名称	类型	说明	是否必须
writer	标签类型	该标签只包含一个 datasource 子标签跟一个 insert/update/delete 子标签。	是
datasource	标签类型	表示目标端数据源	是
driverclassname	String 类型	JDBC 驱动名称。 注意：当前版本只支持 mysql 和 oracle。	是
url	String 类型	数据库地址。	是
username	String 类型	数据库用户名。	是
password	String 类型	数据库密码。	是
initpoolsize	Int 类型	数据库连接池大小，如果不填，默认为 10	否

insert	标签类型	其子标签<sql>中的内容只能写 insert 语句。其中插入的值即 where 条件中的值来自 reader 中的 select sql 结果。	否
update	标签类型	其子标签<sql>中的内容只能写 update 语句。其中插入的值即 where 条件中的值来自 reader 中的 select sql 结果。注意：当 select sql 的结果集只有一条记录时才可使用用此 update 功能。	否
delete	标签类型	其子标签<sql>中的内容只能写 delete 语句。其中插入的值即 where 条件中的值来自 reader 中的 select sql 结果。	否
sql	标签类型	sql 子标签可为任意多个。	是
sql 标签的 refid 属性	String 类型	其值来源于 reader 标签中的 sql 子标签的 id 属性值，且与其一一对应。	是
value	String 类型	里面写纯 SQL、扩展 SQL。 注意：使用该标签时不能同时使用 class 标签。	否

class	String 类型	里面写扩展的 java 类路径。 注意：使用该标签时不能同时使用 value 标签。	否

4.4 补充说明

说明一

在<insert>或<update>或<delete>标签中的<sql>自标签中的 sql 语句可以使用与其对应(即：refid 与<select>中的<sql>的属性 id 对应)的值，语法表示为：**${Select 的字段名称}**，即：select 查询出的值，可以当成 insert 或 update 或 delete 的 where 条件的值来使用。

说明二

DB Stat 支持三种 SQL 配置：纯 SQL 配置、扩展 SQL 配置(使用 db-stat 的内置 sql 函数)、可编程式配置(支持 JAVA 代码去拼 SQL 语句)：

第一，纯 SQL：支持嵌套 SQL、SQL 函数。

第二，扩展 SQL：为了更好地支持大家直接拼 SQL，尽量减少编码工作，DB Stat 提供了一些内置的 SQL 函数，使用时需要用**#{函数名称}**来标注。以下为内置 SQL 函数的详细说明：

函数名称	说明
yesterday	代表昨天日期。 默认格式：yyyy-MM-dd
today	代表今天日期。 默认格式：yyyy-MM-dd

lasthour	代表上一个小时时间。 默认格式：yyyy-MM-dd HH
hour	代表当前小时时间。 默认格式：yyyy-MM-dd HH
lastweek	代表上一周时间。 默认格式：yyyy-WW（WW 代表一年中的第几周）
lastweekday	代表上一周的某天时间。即：往前推 7 天。 默认格式：yyyy-MM-dd
lastweekofyear	代表上一周在一年当中是第几周。 类型：int 类型
lastweekfirstday	代表上一周的第一天。 默认格式：yyyy-MM-dd
lastweekendday	代表上一周的最后一天。 默认格式：yyyy-MM-dd
month	代表当前月份。 默认格式：yyyy-MM
lastmonth	代表上一月份。 默认格式：yyyy-MM
lastmonthofyear	代表上一个月份。 类型：int 类型
lastyear	代表去年。 默认格式：yyyy
year	代表今年。

	默认格式：yyyy
uuid	代表 36 位随机 UUID 字符串。

第三，可编程式 SQL：如果上述两种方法仍然无法满足要求，此时可以动手写代码去拼凑 SQL 语句。db-stat 会通过反射查找扩展的 java 类。当然扩展类的时候需要遵循一定规则，即：需要继承特定的抽象类，实现其特定的方法。如果在<select>标签中的 sql 扩展时需要继承

com.ncfgroup.dbstat.api.SelectBuilder 类，实现其 buildSelectSql 方法；如果是在<insert>标签中的 sql 扩展时需要继承 com.ncfgroup.dbstat.api.InsertBuilder，实现其 buildInsertSqls 方法；如果是在<update>标签中的 sql 扩展时需要继承

com.ncfgroup.dbstat.api.UpdateBuilder，实现其 buildUpdateSqls 方法；如果是在<delete>标签中的 sql 扩展时需要继承

com.ncfgroup.dbstat.api.DeleteBuilder，实现其 buildDeleteSqls 方法。

Select 查询接口实例

```
public class TestSelectBuilder extends SelectBuilder{
  @Override
  public SqlConf buildSelectSql(){
    // 创建一个 SqlConf 对象
    SqlConf sc = new SqlConf();
    sc.setValue(“select * from A ”);
    return sc;
  }
}
```

Insert 插入接口实例

```
public class TestInsertBuilder extends InsertBuilder{
  @Override
  public List<String> buildInsertSqls(
              List<Map<String,Object>> selectResult){
    // selectResult 为对应其查询的结果集
    List<String> sqls = new ArrayList<String>();
    for (Map<String, Object> map : selectResult) {
      StringBuffer sql = new StringBuffer("insert into USER
(id,name) values ( ");
      // 注意字段名称(ID,NAME)统一要大写才能获取到值
      sql.append("'"+map.get("ID")+"',");
      sql.append("'"+map.get("NAME")+"' ) ");
      sqls.add(sql.toString());
    }
    return sqls;
  }
}
```

Update 修改接口实例

```
public class TestUpdateBuilder extends UpdateBuilder{
  @Override
  public List<String> buildUpdateSqls(
                  List<Map<String,Object>> selectResult){
    // selectResult 为对应其查询的结果集
    List<String> sqls = new ArrayList<String>();
    for (Map<String, Object> map : selectResult) {
```

```
        StringBuffer sql = new StringBuffer("UPDATE Person SET
name='Fred' WHERE id=");
        // 注意字段名称(ID)统一要大写才能获取到值
        sql.append("'"+map.get("ID")+"'");
        sqls.add(sql.toString());
    }
    return sqls;
  }
}
```

Delete 删除接口实例

```
public class TestDeleteBuilder extends DeleteBuilder{
  @Override
  public List<String> buildDeleteSqls(
                List<Map<String,Object>> selectResult){
    // selectResult 为对应其查询的结果集
    List<String> sqls = new ArrayList<String>();
    for (Map<String, Object> map : selectResult) {
        StringBuffer sql = new StringBuffer("DELETE FROM Person
WHERE LastName=");
        // 注意字段名称(NAME)统一要大写才能获取到值
        sql.append("'"+map.get("NAME")+"' ");
        sqls.add(sql.toString());
    }
    return sqls;
  }
```

```
}
```

> *“一个缺陷充斥的系统，将始终是一个缺陷充斥的系统。”*
>
> *--- Timothy C.Lethbridge*

第 5 章 应用案例

应用案例最关键的是灵活定义统计规则，即 SQL 语句。以下列举出一些常用的场景：

5.1 统计历史数据

为了更好的呈现出有用的、一定规则的历史数据，可使用 DB Stat 将原始数据统计到目标库当中。

5.2 数据迁移

异构数据库之间如果要做数据迁移也是件不是特别容易的事情，针对这种情况也可以通过 DB Stat 进定时进行数据迁移。

5.3 数据初始化

当应用系统需要一大批有规律的数据支持时，这时可以先录入一些数据，当作数据模版，然后通过 DB Stat 统计往同一张表里插入更多的数据。

5.4 数据清洗

当一张表中的冗余数据太多，想将一些没用的数据清洗掉，并将有用的数据导入到一张新表中，这时可使用 DB Stat 来完成此功能。

> *“九个人不能让一个孩子在一个月内出生。”*
>
> *--- Fred Brooks*