# 使用篇

目录

[使用篇 1](#_Toc303932502)

[1. 推荐版本 2](#_Toc303932503)

[2. 新用TDDL分库分表 2](#_Toc303932504)

[1.1 使用动态数据源(具有动态调节ip,密码,读写权重变换,增库,主备切换功能) 2](#_Toc303932505)

[1.2使用本地数据源(不具有动态调节ip,密码,读写权重变换,增库,主备切换功能) 4](#_Toc303932506)

[1.3 2.3.3规则详解 5](#_Toc303932507)

[1.4 2.4.4新规则详解 8](#_Toc303932508)

[2. 不使用分库分表,将一个DBCP数据升级为一个Atom DataSource 9](#_Toc303932509)

[3. 不使用分库分表,将一个DBCP数据升级为一个 Group DataSource 9](#_Toc303932510)

[4. 使用tddl-sequence 10](#_Toc303932511)

[4.1 原理 10](#_Toc303932512)

[4.2 使用 11](#_Toc303932513)

## 推荐版本

TDDL推荐版本依赖

<dependency>

<groupId>com.taobao.tddl</groupId>

<artifactId>tddl-client</artifactId>

<version>3.0.1.1</version>

</dependency>

TDDL-SEQUENCE推荐版本依赖(3.0.0 9月低release)

<dependency>

<groupId>com.taobao.tddl</groupId>

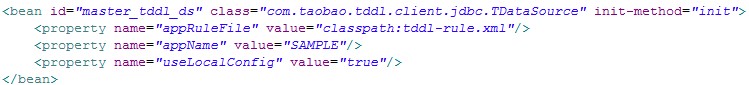
<artifactId>tddl-sequence</artifactId>

<version>3.0.0-SNAPSHOT</version>

</dependency>

## 新用TDDL分库分表

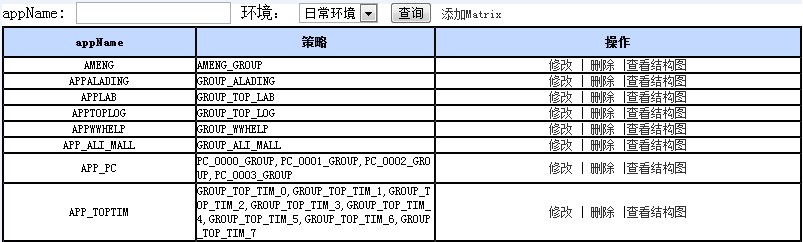
### 1.1 使用动态数据源(具有动态调节ip,密码,读写权重变换,增库,主备切换功能)



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

使用动态数据源,配置很简单,如上图所示. 唯一需要注意的是 appName一般配成你的应用名称.主要目的是不与其他应用重名. 因为这个appName属性是向我们持久配置中心diamond请求数据的一个凭证,如果和其他应用重名,那么你的应用将有可能拿不到你期望的数据源配置了.

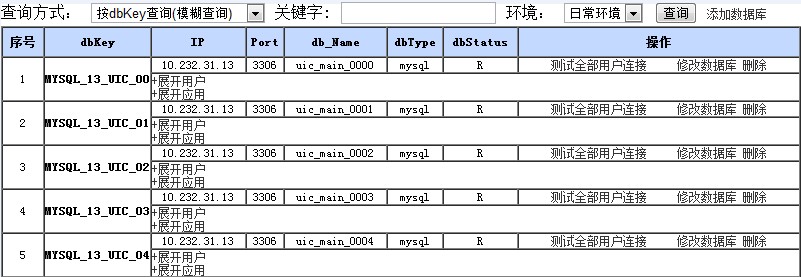
这时,你可能有疑问了.我的数据库配置在哪呢? 这个时候,你应该要找到你们的开发DBA,让他将你们应用使用数据源配置到持久配置中心diamond上去,包括数据库ip,port等信息,数据库主备权重信息,而这些信息都是不用你操心的,dba会通过一个脚本将其配置上去,你如果需要知道具体的权重配置,可以登录<http://ops.jm.taobao.net/rtools/jade/jade_global.htm>(日常)去查看.具体如下几图:



Matrix管理(主要说明你应用使用了哪些group)



Group管理(主要说明group数据源里面各个原子数据源的读写权重关系)



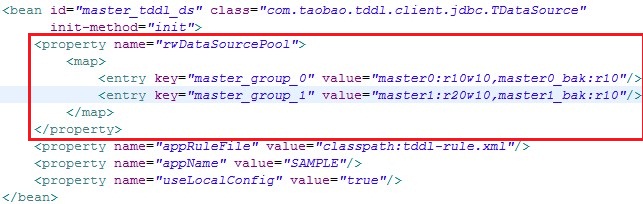
Global管理(主要说明数据源的ip,port,用户信息,使用这个数据源的应用配置等信息)

做完以上的事情(实际上就2步),那么你的程序应该就能run起来了,哦,对了,我们还没有配置规则.那么我们可以跳到第三节,看下我们最常用的配置的是如果配置的.

### 1.2使用本地数据源(不具有动态调节ip,密码,读写权重变换,增库,主备切换功能)



在你的spring文件中会配有DBCP数据源



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

rwDataSourcePool中的key是要在规则中使用的(具体讲到规则会说明).value指向本地配置原子数据源,并且标明了他们之间的读写关系. 一组数据源即数据相同(通过数据复制),只是存在着读写权重关系.

### 1.3 2.3.3规则详解

规则我们一般会配置成一个独立的spring配置,比如tddl-rule.xml,然后在TDataSouce的bean中进行property引用,如下图:

**tddl-ruleFile.jpg**

下图是一个标准的配置文件.对于图中的(1)(2)(3)(4)点,这里我具体来说明下

1. 中表示我有几张表是需要分库分表的,key一般表示逻辑表(也就是不带后缀的),value-ref指向接下来具体定义的规则bean.
2. 中表示的是一些特殊的开关,**一般这几个property都不需要写**,completeDistinct表示是否支持多库多表的distinct(true即为支持).needIdInGroup表示是否需要进行id in形式的优化, 这个在有大量参数在in()中的场景下,大大提高性能(true即为支持).defaultDbIndex表示如果一条sql进来,我们发现from, into,update后面跟的第一张表不分库分表,那么默认将这条sql在defaultDbIndex指向的数据库上直接执行掉. 这个对于不分库分表的sql也走TDataSource是一个很好的解决方案.
3. 中dbIndexs表示这个逻辑表所分布的库的id, 使用动态数据源,这些id即matrix管理中你对应appName下的group列表中所有或者部分group dsKey,

使用本地数据源,这些id即TDataSource中rwDataSourcePool中所有或者部分的key.

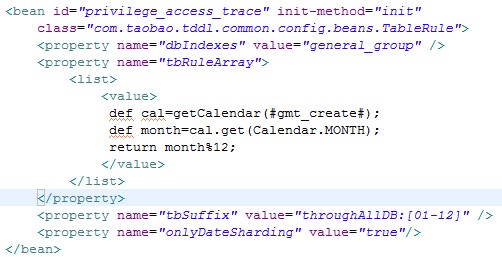
1. 中 disableFullTableScan表示禁止全表扫描. 默认true,即禁止全表扫描成立.一般应用中的场景就是sql中不带分库分表条件, TDDL实际上就不知道如何分库分表了.如果带有这个property并且设置为false,那么也就是放开禁止全表扫描,那么这条sql会在所有这张逻辑表的分表上执行这条sql. 一般来说,不建议配置的时候加上这句配置,除非有全表扫描的需求.



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

dbRuleArray表示分库条件,tbRuleArray表示分表条件,他们算出来的结果只是一个序号,比如上图中soccerplayer dbRuleArray算出值为0,那么实际dbKey就是tddl-group-0, tbRuleArray也是一样,如果算出值为10,那么实际表就是 soccerplayer\_0010(从0000开始).

dbRuleArray和tbRuleArray实际都是一个List我们可以配置成这样,并且在规则中配上一段代码,以方便应用对分库分表键的一些特殊处理要求,但是需要注意的是,一定要返回一个序号.也就是选db或者选table的序号,并且代码的格式为groovy或者java.如下图



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

规则可以配多个(因为是List),tddl处理逻辑就是看你sql带的分库分表条件符合哪一个规则的分库分表键(##中表示).一次只会选一个,现在最前面的优先级越高.

Sample规则中我们常常会看到类似**#gmt\_create, 1\_month,12#**或者**#seller\_id,1,512#**各位可能会疑惑了,这个1\_month和12以及1和512代表什么意思呢? 其实我们的应用常常会做一些范围查询. 比如where gmt\_create> now()或者seller\_id>0, 对于TDDL来说>这种符号对它来说是毫无意义的,它需要具体的值根据规则去算,那怎么办呢. 通过规则去枚举这些值, 比如gmt\_create>now()

1. 首先根据规则知道2个值之间跨度是1\_month,也就是1个月
2. 再者知道最大枚举数是12. 那么根据跨度和最大枚举数,然后起始值是现在,比如7月, 那么最终枚举出来的值是 7,8,9,10,11,12,1,2,3,4,5,6 ,然后tddl就可以根据这些值去算路由结果了.

对于where gmt\_create>now() and gmt\_create<sometime, 那么这个时候12的最大枚举数就没用了,但是 1\_month的跨度还是有效,因为依然要根据这个跨度枚举值.

默认跨度值为实数,也就是不是按时间来枚举的, 如果如上按时间来枚举,那么就需要类似1\_date ,1\_month,1\_year这类的跨度值.

### 1.4 2.4.4新规则详解

新规则需要使用,请具体咨询TDDL相关同学,TDDL-RULE-LE以及动态迁移只支持2.4.4新规则.这里只贴下一个具体的使用案例.基本思想就是规则算出结果就是路由结果,所见即所得 **注意 : 新规则格式只有tddl-2.4.4之后才会有.**

**第一种风格(有结果格式化)**

****

**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

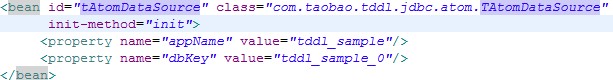
**第二种风格(完全所见所得)**

****

**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

## 2. 不使用分库分表,将一个DBCP数据升级为一个Atom DataSource

1. **Example**

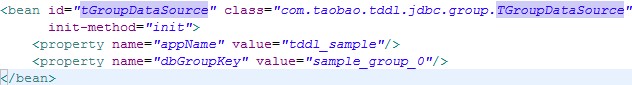


**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

appName即应用的名称,主要和其他应用区别开,dbKey即atom ds在持久配置中心的key(让dba协同配置)

## 3. 不使用分库分表,将一个DBCP数据升级为一个 Group DataSource

1. **Example**



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

appName即应用的名称,主要和其他应用区别开,dbGroupKey就是group ds在持久配置中心的key(让dba协同配置)

## 4. 使用tddl-sequence

### 4.1 原理

Tddl sequence有两种实现,一个是DefaultSequenceDao,另外一个是GroupSequenceDao. 两者的区别在与前者是sequence表只支持单库排布,后者支持多库排布,具有更好的容灾策略以及默认使用动态数据源.所以这边只介绍GroupSequenceDao, 如需要使用DefaultSequenceDao,请找我们,当然我们还是会推荐您使用GroupSequenceDao.

GroupSequenceDao原理在于我们将sequence的数据段放在多个机器上,并且按照规则中的步长相互错开,确保不重合. 举个例子,我需要一个 sample\_sequence, 然后sequence数据放在 sample\_group\_0,sample\_group\_1,sample\_group\_2, sample\_group\_3 四组机器上.并且配置文件中规定步长为1000,也就是客户端每次拿1000.那么起始的时候,4个group上的sequence初始值为

|  |  |
| --- | --- |
| sample\_group\_0 | 0 |
| sample\_group\_1 | 1000 |
| sample\_group\_2 | 2000 |
| sample\_group\_3 | 3000 |

那么应用真正启动的时候,可能某一台机器上去取sequence,随机取到了sample\_group\_1,那么这台机器上的应用会拿到1000-1999这一千个id,而这个时候4个group上sequence会变成下图所示,你也许注意到了,下次从sample\_group\_1上取得的id就变成了5000-5999. 那么也就是这样,完全避免了多机上取sequence的重复.比如sample\_group\_1他会永远只会取到1000-1999,5000-5999,9000-9999,13000-13999…其他group也一样,相互不会重叠.

|  |  |
| --- | --- |
| sample\_group\_0 | 0 |
| sample\_group\_1 | 5000 |
| sample\_group\_2 | 2000 |
| sample\_group\_3 | 3000 |

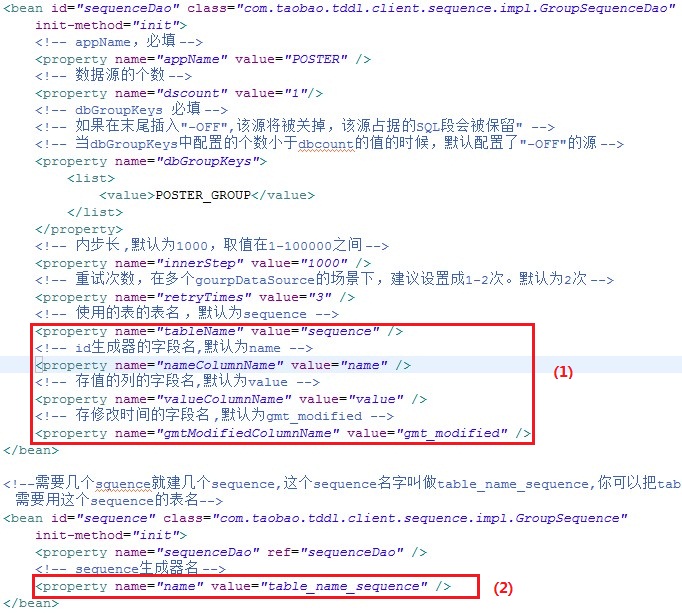
### 4.2 使用

**第一步**:定义一个tddl-sequence.xml(当然你和别的配置混合在一起也没事,但我们建议分开).如下图.

**(1) 这些property指定需要建的表名,以及表中的一些字段**

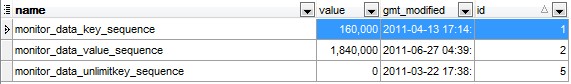
**(2) 这个property指定需要从sequence表中的哪条记录读取.因为可能一张sequence表中有多个表的sequence生成数据,所以采用name区分**

**以上两点可以结合接下来第二步中的sequence表截图进行对比.**



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

**第二步**:在对应的数据库中建sequence表.如下图



**注意!:文档中的参数都属于参考值. 线上发布参数配置请具体询问TDDL的运维同学**

上一步中标红的name property中的value即表中name字段对应值. value对应的值即sequence的当前分配值