Tddl\_c的配置主要分为两部分：数据源配置和规则配置。这两部分配置都存放在diamond服务器上，前端由rtools web控制台(<http://ops.jm.taobao.net/rtools/index/index.htm>)控制动态数据源配置和路由规则的推送。

## 数据源配置

总体拓扑图

Matrix(appName)

Atom1

Group1

Atom1\_bak

Group2

Atom2

Atom2\_bak

(1). Matrix中的配置主要是在分库分表下，一个appName下使用的group-ds

dataId组成规范:**"com.taobao.tddl.v1\_"+appName+"\_dbgroups"**

配置内容(示例): **sample\_group\_0,sample\_group\_1**

其中sample\_group\_0,sample\_group\_1即下一层需要使用的dbGroupKey

(2).Group中的配置主要是配置一组对等的数据的读写权重

dataId组成规范:**"com.taobao.tddl.jdbc.group\_V2.4.1\_"+dbGroupKey**

配置内容(示例):**tddl\_sample\_0:r10w10p0,tddl\_sample\_0\_bac:r10w0p0**

其中tddl\_sample\_0和tddl\_sample\_0\_bak就是下一层需要的dbKey,后面r为读权重，w为写权重

(3).atom ds中的配置分为了3部分（global,app,user）,配置内容全部为java的properties格式

**Global**

dataId组成规范: **"com.taobao.tddl.atom.global."+dbKey**

配置内容：

|  |  |
| --- | --- |
| 属性key | 说明 |
| ip | 数据实例的ip |
| port | 数据实例的端口 |
| dbname | 数据库名称 |
| dbType | MYSQL,ORACLE |
| dbStatus | RW,NA |

**App**

dataId组成规范: **"com.taobao.tddl.atom.app."+appName+"."+dbKey**

配置内容：

|  |  |
| --- | --- |
| 属性key | 说明 |
| username | 该应用使用的用户名 |
| oracleConType | oci,thin,如果db为mysql,则不用理会 |
| minPoolSize | 最小连接池 |
| maxPoolSize | 最大连接池 |
| idleTimeout | 连接的最大空闲时间 |
| blockingTimeout | 等待连接的最大时间 |
| preparedStatementCacheSize | Oracle专用 |
| writeRestrictTimes | 单位timeSliceInMillis写限制，默认空不限制 |
| readRestrictTimes | 单位timeSliceInMillis读限制，默认空不限制 |
| threadCountRestrict | 并发线程限制，默认空不限制 |
| timeSliceInMillis | 限制的时间单位 |
| connectionProperties | 连接参数 |

**User**

dataId组成规范: **"com.taobao.tddl.atom.passwd."+dbName+"."+dbType+"."+userName**

配置内容：

|  |  |
| --- | --- |
| 属性key | 说明 |
| encPasswd | 密码 |
| encKey | 密钥 |

## 路由规则配置：

Tddl\_cl的路由规则参考tddl的规则抽象，可能需要一些适应性修改，但是其本身抽象从实际经验来看是比较比较简练与适合是。

**样例1(格式化模式)：**



**样例1说明：**

1. Vtabroot是总控的节点,dbType可能值为mysql或者oracle,tableRule是需要分库分表的逻辑表列表，就像注册表一样，其中key默认为这套表的前缀(如实际表order\_0001逻辑表为order).defaultDbIndex指向的group ds是为了在用户进来一条不分库不分表的sql，比如示例中的 select \* from picture,那么这个sql的需要执行的表picture并不在分库分表的表列表里面，那么直接在defaultDbIndex指向的group ds执行掉。
2. order\_bean,user\_bean,item\_bean分别是order,user,item 3套表的分库分表规则。其中dbNamePattern是库名的格式化，因为从库规则dbRules里面算出来的只是一个值，所以需要格式化，tbNamePattern也道理相同，tbRules和dbRules分别对应表的计算规则和库的计算规则，里面的计算脚本我们可以用lua脚本。

allowFullTableScan对应那种完全没有分库分表字段的sql,也就是意味这需要扫描全表，默认禁止，因为全表扫性能可能会比较低，并且一旦疏忽，执行delete操作，会造成严重的事故，这个以前也碰到过。

最后解释下#item\_id,1\_number,2#这种表达式，实际上我们在做范围查询的时候，类似select \* from item where item\_id>0 and item\_id<4 and gmt\_create>’2012-2-28’ 这种范围查询条件对于规则计算是无法进行的，我们只能得到具体的item\_id和gmt\_create的值才能做计算，那么得到具体值的过程我们叫枚举，比如item\_id枚举出来的值为1,2,3

而gmt\_create实际上是一个开区间范围，所以我们必须知道它的最大枚举范围是多少，

从配置中得到，gmt\_create的最大枚举数是12，单位是1\_month, 那么得到的枚举值为

’2012-3-28’, ’2012-4-28’, ’2012-5-28’, ’2012-6-28’, ’2012-7-28’, ’2012-8-28’, ’2012-9-28’, ’2012-10-28’, ’2012-11-28’, ’2012-12-28’, ’2013-1-28’, ’2013-2-28’ , 这样我们得到了3×12组值，逐一带入即可。那么现在我们知道，#item\_id,1\_number,2#这种字符串第一段表示字段名称，第二段表示自增的单位值和单位(number,hour,date,month,year,默认不写类型即number),第三段表示最大枚举的数量。

**样例2(格式化模式)：**



**样例2说明：**

1. 出现的字段所表达的意思和样例1相同
2. 主要不同点是格式化属性dbNamePattern和tbNamePattern去除了，tbRules和dbRules算出来的值直接对应groupds的名称和具体的表名。即所见即所得，并且可以任意拼写表名和库名。但是从实际使用来说，规范化的库表命名无论对于运维还是后续可能进行的规范化处理都是有好处的，所以建议使用样例1.