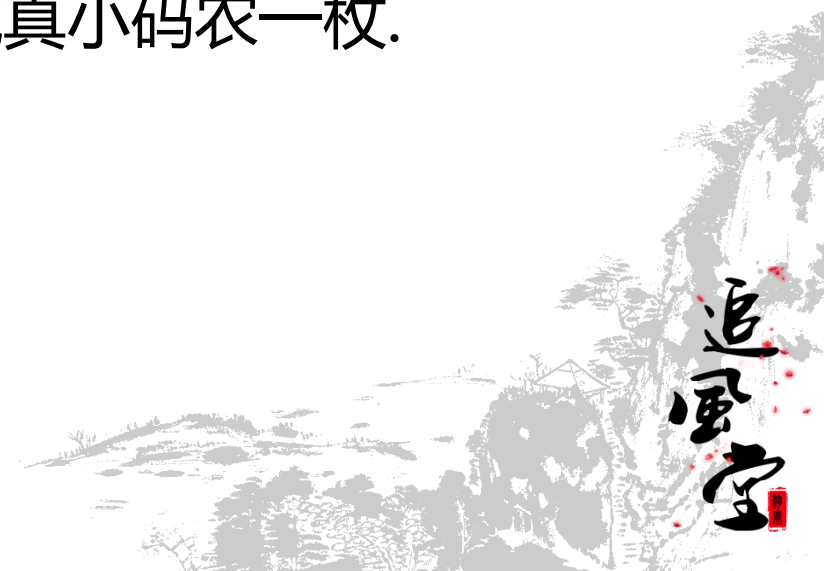
TDDL的使用与配置

部门：产品技术部-java中间件

君 瑜

讲师介绍

- **产品技术部-**(太多层级,省略)-励强 (花名：君瑜)**
- 2010年加入淘宝,之后一直参与数据层相关产品的开发和维护工作,包括TDDL,愚公数据在线迁移套件,大禹管理平台等系统和产品的开发和维护.纯真小码农一枚.



系列课程介绍

- **关于中间件的系列课程**
  - 《消息中间件notify的使用》
  - 《消息中间件notify的概念和原理》
  - 《消息中间件metamorphosis简介和使用》
  - 《服务框架HSF体系和架构》
  - 《服务框架HSF使用和配置》
  - 《分布式数据层TDDL原理》
  - 《分布式数据层TDDL使用和配置》

课程目标与目标学员

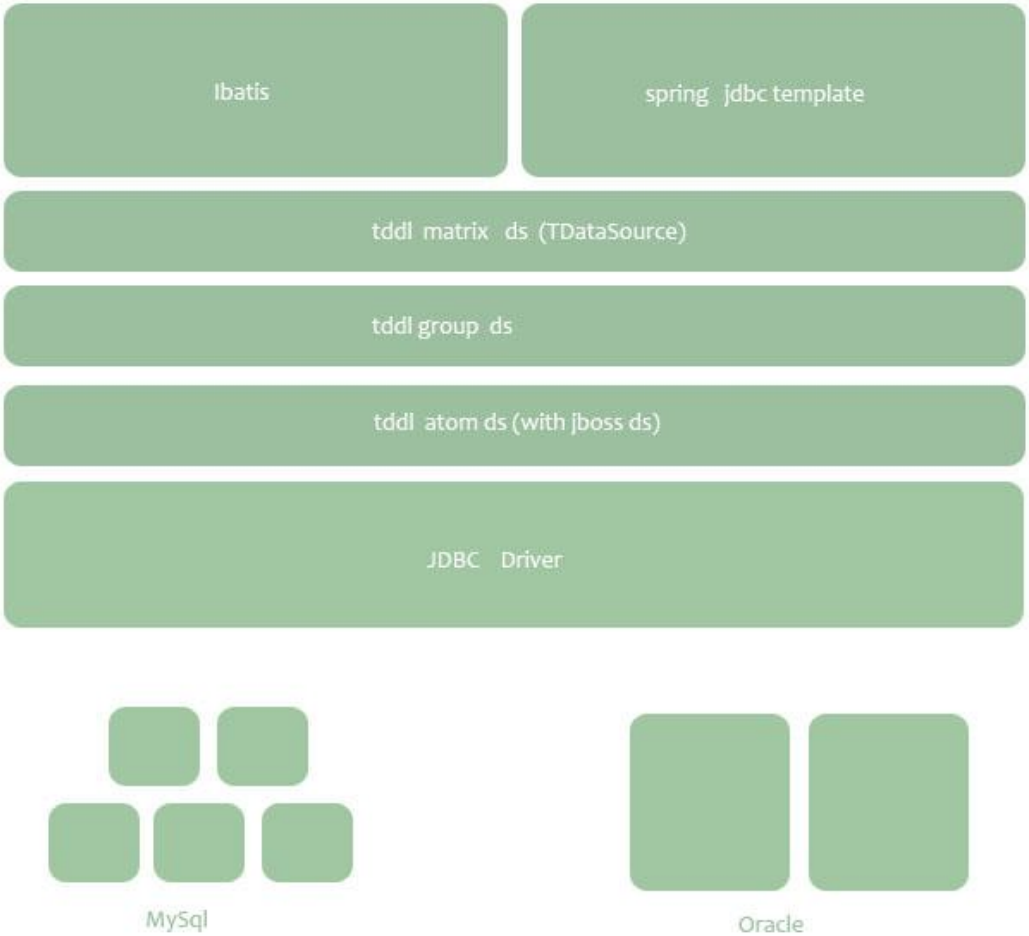
- **面向学员：JAVA开发同学,DBA,对TDDL以及在线数据库自由扩容感兴趣的同学**
- **课程目标：通过本课程，学员能够：**
  - 使用TDDL这个中间件
  - 了解TDDL的实现原理和特性
  - 了解在线扩容的原理和实现

课程大纲页

- TDDL 简单原理与架构
- TDDL 使用
- 愚公数据迁移系统和精卫数据同步系统

TDDL简单原理和架构

架构(1)


Ibatis
spring jdbc template
tddl matrix ds (TDataSource)
tddl group ds
tddl atom ds (with jboss ds)
JDBC Driver
MySql
Oracle

架构(2)

1. 三层数据源每层都按JDBC规范实现

2. Matrix层(TDataSource)实现分库分表逻辑,底下持有多个GroupDs实例

3. Group层(TGroupDataSource) 实现数据库的主备/读写分离逻辑,底下持有多个AtomDs实例

4. Atom层(TAtomDataSource)实现数据库ip,port,password,connectionProperties等信息的动态推送,持有原子的数据源(分离的JBOSS数据源).

架构(3)


sql和参数
sql解析
规则计算
表名替换
选择GroupDs
执行sql
根据权重选
AtomDs
具有重试策
略地在AtomDs
执行sql
读写数控制
、线程并发数
控制
执行sql,
返回结果集
合并处理
多个结果集
查询或
更新结果

为何要做SQL解析

1. 分库分表条件,order by, group by, limit m,n , join信息, SUM,MAX,MIN等聚合函数信息,DISTINCT信息

2. TDDL行复制需要重新拼写SQL,带上sync_version字段.

怎么做SQL解析

1.解析流程

SQL->词法分析+语法分析->抽象语法树(AST)->SQL对象

2.实现方式(antlr工具)

(1).编写 *.g 词法分析和语法分析文件

(2).使用antlr进行编译，生成java代码

(3).运行时调用生成的java代码，生成sql对象

3.其他方式

(1) javacc(开源的JSQLParser)

(2) asm(完全不依靠工具来分析，自己解析sql，最后生成java字节码)

怎么做规则计算

1.规则语言：GROOVY

2.初始化的时候 将规则文件中的dbRules,tbRules片段分别拼成

一个groovy类文件，然后用groovy的ClassLoader加载，生成

可以在jvm上运行的字节码，缓存计算Method,运行时反射调用

怎么做表名替换

1.方式1：正则表达式

2.方式2：根据sql解析生成的sql对象反向输出，tddl的异构复制必须要通过这种方式生成sql,因为需要加版本信息。

怎么做结果集合并

1.分库分表下可能返回多个结果集，那么需要根据sql的具体情况做结果集合并

(1).order by
(2).limit m,n (分页查询)
(3).sum
(4).max
(5).min
(6).distinct

2. Jdbc结果集的游标移动逐条取数据

```
while(rs.next()){
    rs.getString(x);
    …
}
```

TGroupDataSource的原理

1.GroupDs由一组原子的数据源组成，也就是通常说的主备结构，实际上它可以配置成多主多备，主备通常是通过读写比重来区分。

2.权重信息一般放在diamond上，当然也可以本地使用，权重的几种形式：

3.其他特性：失败快速抛出和单线程重试

MYSQL_ICDB_00	MYSQL_CM4_ICDB0:r10w10p0,MYSQL_CM3_ICDB0:r0w0p0,MYSQL_CM5_ICDB0:r0w0p0
SYNC_LOG_CART_GROUP	my160104_cm6_sync_log_cart:r10w10p0,my160105_cm6_sync_log_cart:r10w10p0

TAtomDataSource的原理（1）

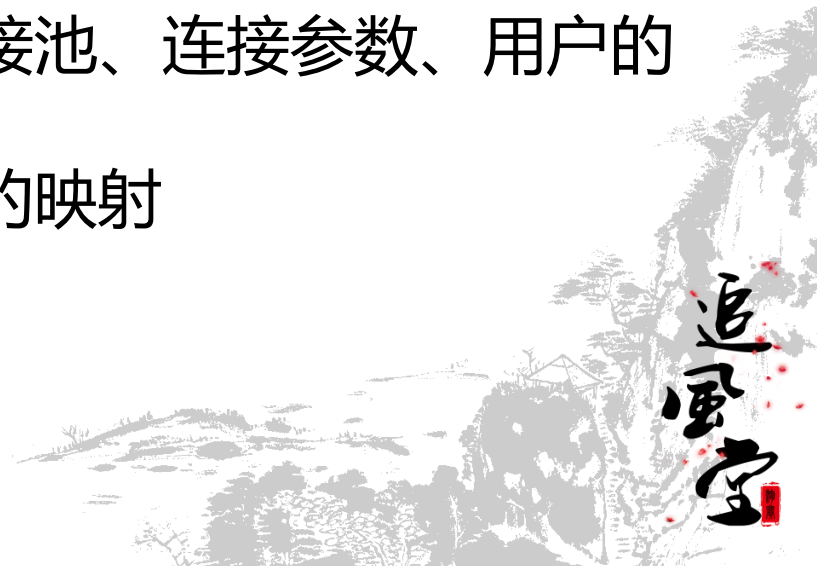
1.AtomDs是一个物理数据库的代理，它对单库的sql操作做了jdbc的浅包装，但是控制了真实数据库的数据源创建和链接分配，这也是他能够同时创建oracle和mysql数据源的原因。

2.AtomDs创建数据源和连接时需要用到的参数全部放在diamond上，参数包括3部分：

(1).global:数据库的ip,port,dbname信息

(2).app:appName和数据源连接池、连接参数、用户的映射

(3).passwd:数据库用户和密码的映射



TAtomDataSource的原理（2）

db160_cm2_lake	172.19.68.160	1521	lake	oracle	可用	测试全部用户连接	修改数据库 删除

+展开用户

用户1:itb (测试连接)
用户2:water (测试连接)

+展开应用

应用1:

- appName:dbaccesslayer
- blockingTimeout:5000
- connectionProperties:SetBigStringTryClob=true;defaultRowPrefetch=50
- maxPoolSize:10
- minPoolSize:2
- idleTimeout:20
- dbKey:db160_cm2_lake
- userName:water
- preparedStatementCacheSize:10
- oracleConType:thin
- 修改

应用2:

- appName:APP_LAKE_TOPS
- blockingTimeout:5000
- connectionProperties:characterEncoding=gbk;autoReconnect=true
- maxPoolSize:3
- minPoolSize:1
- idleTimeout:60
- dbKey:db160_cm2_lake
- userName:water
- 修改

TDDL SEQUENCE原理

sample_group_0	0
sample_group_1	1000
sample_group_2	2000
sample_group_3	3000

随机取到sample_group_1,取

sample_group_0	0
sample_group_1	5000
sample_group_2	2000
sample_group_3	3000

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.

TDDL使用

TDataSource使用

```
<bean id="tddlDataSource" class="com.taobao.tddl.client.jdbc.TDataSource"
    init-method="init">
    <property name="useLocalConfig" value="true"></property>
    <property name="appName" value="tddl_sample"></property>
    <property name="appRuleFile" value="classpath:tddl-rule.xml"></property>
</bean>
```

appName	策略	操作
tddl_exchange_sample	tddl_exchange_sample_group	修改 | 删除 | 查看结构图
tddl_sample	sample_group_0, sample_group_1	修改 | 删除 | 查看结构图
tddl_sample_real	tddl_group_0, tddl_group_1	修改 | 删除 | 查看结构图

TGroupDataSource使用

```
<bean id="tGroupDataSource" class="com.taobao.tddl.jdbc.group.TGroupDataSource"
      init-method="init">
    <property name="appName" value="tddl_sample"/>
    <property name="dbGroupKey" value="sample_group_0"/>
</bean>
```

groupKey	策略	操作
sample_group_0	tddl_sample_0:r10w10p0,tddl_sample_0_bac:r10w0p0	修改 删除

TAtomDataSource使用

```
<bean id="tAtomDataSource" class="com.taobao.tddl.jdbc.atom.TAtomDataSource"
      init-method="init">
    <property name="appName" value="tddl_sample"/>
    <property name="dbKey" value="tddl_sample_0"/>
</bean>
```

序号	dbKey	IP	Port	db_Name	dbType	dbStatus	操作
1	tddl_sample_0	10.13.42.67	3306	tddl_sample_0	mysql	RW	测试全部用户连接 修改数据库 删除
		+展开用户 用户1:tddl (测试连接) +展开应用 应用1: • appName:tddl_sample • maxPoolSize:100 • minPoolSize:1 • dbKey:tddl_sample_0 • userName:tddl • 修改 应用2: • appName:tddl_sample_real • blockingTimeout:2000 • maxPoolSize:20 • minPoolSize:1 • idleTimeout:10 • dbKey:tddl_sample_0 • userName:tddl • 修改					

TDDL 规则配置(1)

TDDL 3.0.0版本开始提供规则的统一管理和动态推送(2.4.4新规则支持)

```
<bean id="tddl_ds" class="com.taobao.tddl.client.jdbc.TDataSource" init-method="init">
    <property name="dynamicRule" value="true" />
    <property name="appName" value="SNSBLOG" />
</bean>
```

序号	appName	使用版本	规则库	操作
1	SNSBLOG	V1	V1	查看 | 修改 | 增加 | 删除

TDDL 规则配置(2)

http://baike.corp.taobao.com/index.php/TDDL_Intro

TDDL 规则配置(3)

#gmt_create, 1_month,12#

#seller_id,1,512#

SELECT * FROM logic_table WHERE seller_id > 10

SELECT * FROM logic_table WHERE gmt_create > now()

SELECT * FROM logic_table WHERE seller_id >10 and seller_id<40

…

ThreadLocal使用

```
SimpleCondition sc=new SimpleCondition();
sc.setVirtualTableName("moddbtab");
sc.putParamIndexForBatch("pk",0);
sc.setRouteType(ROUTE_TYPE.FLUSH_ON_CLOSECONNECTION);
ThreadLocalMap.put(ThreadLocalString.ROUTE_CONDITION, sc);
```

SimpleCondition可以设定包括Orderby,Groupby,
分库分表键,limit m,n信息

把该段代码放在目标sql执行前.

RouteHelper使用

```
RouteHelper.executeByDB("tddl_group_0");
String deleteSql="delete from moddbtab_0000 where pk=?";
Object[] objs=new Object[]{id3};
tddlJT.update(deleteSql,objs);
```

到指定库上执行

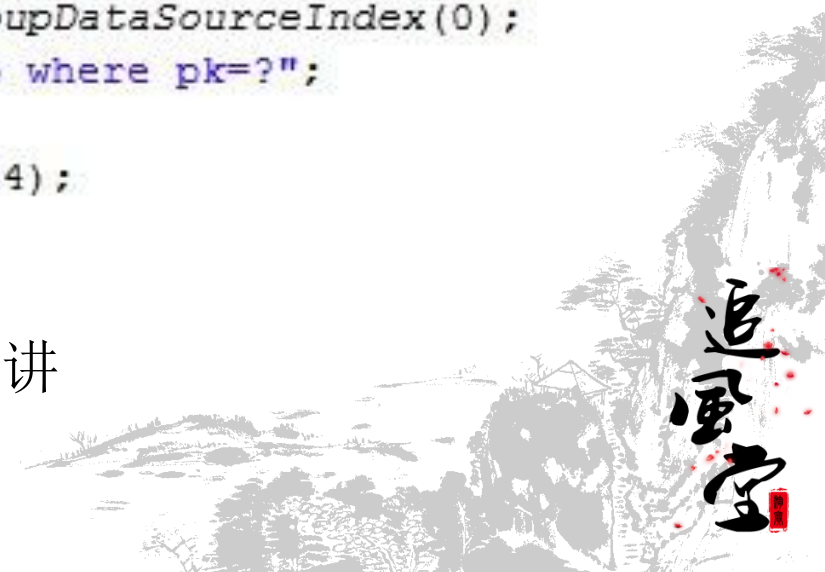
```
RouteHelper.executeByCondition("moddbtab","pk",id4);
String deleteSql="delete from moddbtab where pk=?";
Object[] objs=new Object[]{id3};
tddlJT.update(deleteSql,objs);
```

指定分库分表条件和值,绕开sql解析

```
GroupDataSourceRouteHelper.executeByGroupDataSourceIndex(0);
String querySql="select * from moddbtab where pk=?";
Object[] objs4=new Object[]{id3};
Map re=tddlJT.queryForMap(querySql,objs4);
```

选择主备

RouteHelper是对ThreadLocal的封装,讲ThreadLocal的使用简化到一条语句.
放在目标sql执行前.



TDDL Hint使用

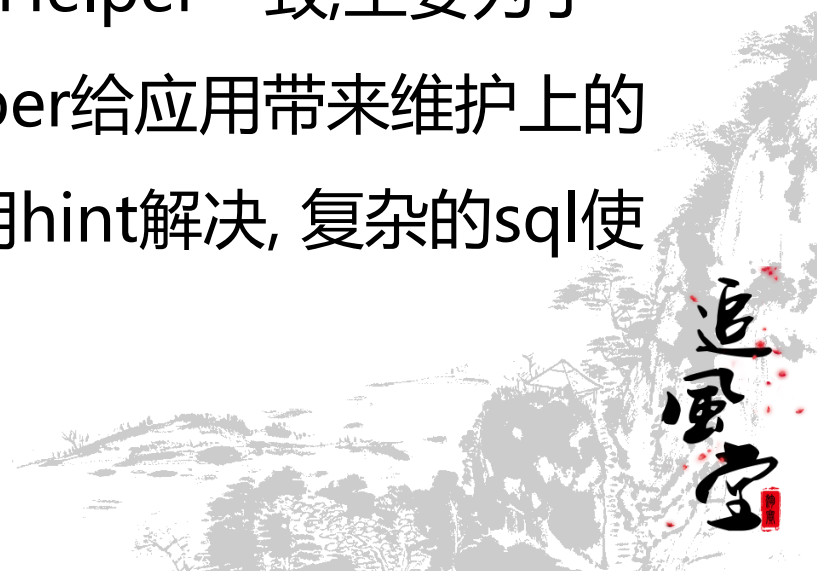
```
/*+TDDL({type:executeByDB,dbId:tddl_group_0})*/delete from moddbtab_0000 where pk=?'

/*+TDDL({type:executeByCondition,parameters:[\"pk=?;i\"],virtualTableName:moddbtab})*/

/*+TDDL({type:?,parameters:[\"pk>=?;i and pk<=?;i\"],virtualTableName:moddbtab})*/sel

/*+TDDL({type:?,parameters:[\"pk>=?;i and pk<=?;i\"],skip:0,max:2,virtualTableName:moddbtab})*/
```

Hint的功用和ThreadLocal和RouteHelper一致,主要为了避免滥用ThreadLocal和RouteHelper给应用带来维护上的困难,一般简单sql绕开sql解析可以用hint解决, 复杂的sql使用ThreadLocal方式解决.



TDDL SEQUENCE使用(1)

```
<bean id="sequenceDao" class="com.taobao.tddl.client.sequence.impl.GroupSequenceDao"
    init-method="init">
    <!-- appName, 必填 -->
    <property name="appName" value="POSTER" />
    <!-- 数据源的个数 -->
    <property name="dscount" value="1"/>
    <!-- dbGroupKeys 必填 -->
    <!-- 如果在末尾插入"-OFF",该源将被关掉，该源占据的SQL段会被保留" -->
    <!-- 当dbGroupKeys中配置的个数小于dbcount的值的时候，默认配置了"-OFF"的源 -->
    <property name="dbGroupKeys">
        <list>
            <value>POSTER_GROUP</value>
        </list>
    </property>
    <!-- 内步长,默认为1000，取值在1-100000之间 -->
    <property name="innerStep" value="100" />
    <!-- 重试次数，在多个gourpDataSource的场景下，建议设置成1-2次。默认为2次 -->
    <property name="retryTimes" value="2" />
    <!-- 使用的表的表名，默认为sequence -->
    <property name="tableName" value="sequence" />
    <!-- id生成器的字段名,默认为name -->
    <property name="nameColumnName" value="name" />
    <!-- 存值的列的字段名,默认为value -->
    <property name="valueColumnName" value="value" />
    <!-- 存修改时间的字段名,默认为gmt_modified -->
    <property name="gmtModifiedColumnName" value="gmt_modified" />
</bean>
```

TDDL SEQUENCE使用(2)

```
<bean id="sequence" class="com.taobao.tddl.client.sequence.impl.GroupSequence"
    init-method="init">
    <property name="sequenceDao" ref="sequenceDao" />
    <!-- sequence生成器名 -->
    <property name="name" value="table_name_sequence" />
</bean>
```

有几张逻辑表,就声明几个sequence

name	value	gmt_modified	id
monitor_data_key_sequence	160,000	2011-04-13 17:14:	1
monitor_data_value_sequence	1,840,000	2011-06-27 04:39:	2
monitor_data_unlimitkey_sequence	0	2011-03-22 17:38:	5

愚公数据迁移系统和精卫数据同步系统

愚公-特性

1. 支持在线的mysql扩容
2. 支持在线去oracle
3. 支持在线mysql,oracle整库迁移
4. 支持不对等表结构迁移(目标库多字段,少字段,复杂数据过滤逻辑等)
5. 目标存储支持mysql,oracle,并可扩展为其他存储

愚公-mysql扩容(1)

1. 分增量复制,全量复制和数据校验3部分
2. 增量复制使用mysql binlog(row模式)
3. 全量通过全表扫描
4. 数据重新散列采用tddl
5. 使用业务自定义转换类(DataTranslator)进行不对等变换和数据过滤

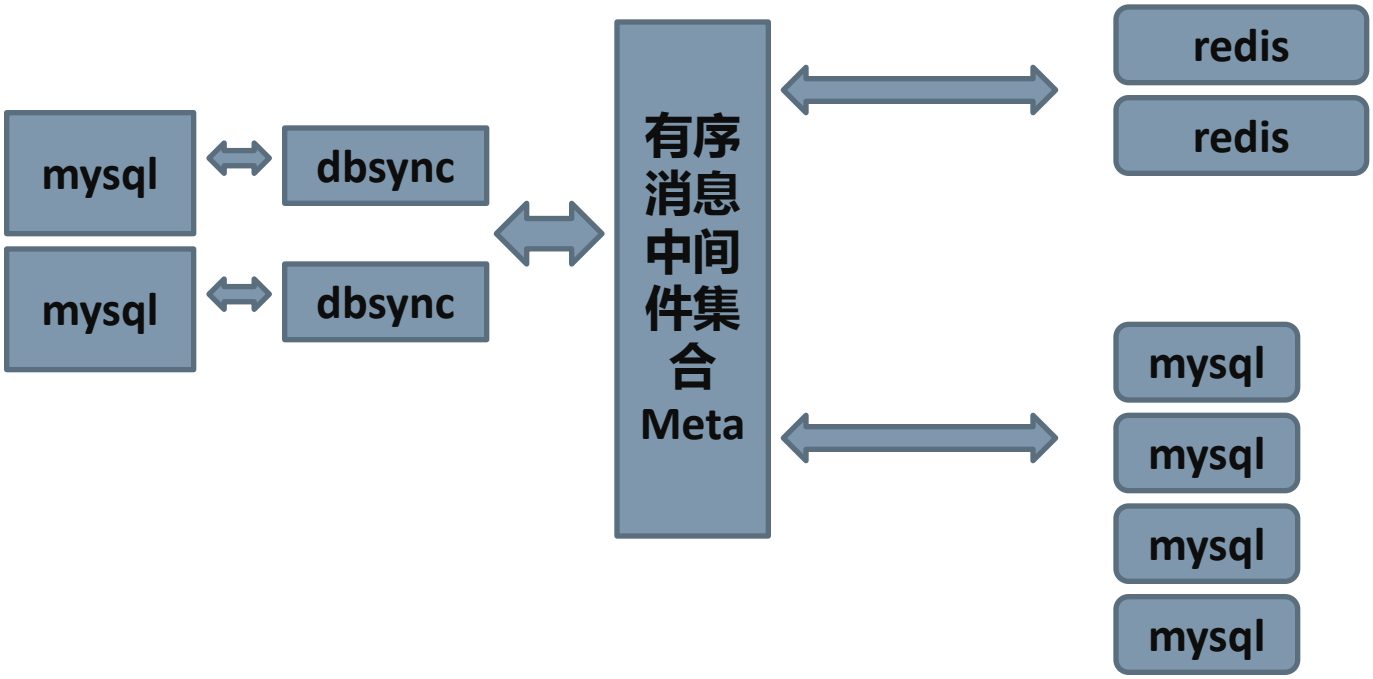
愚公-mysql扩容(2)


binlog复制(不消费日志)
binlog复制(消费日志)
t1
t2
全量复制
数据校验
停冲突数据写,不一致数据处理
使用新规则
删除旧库数据

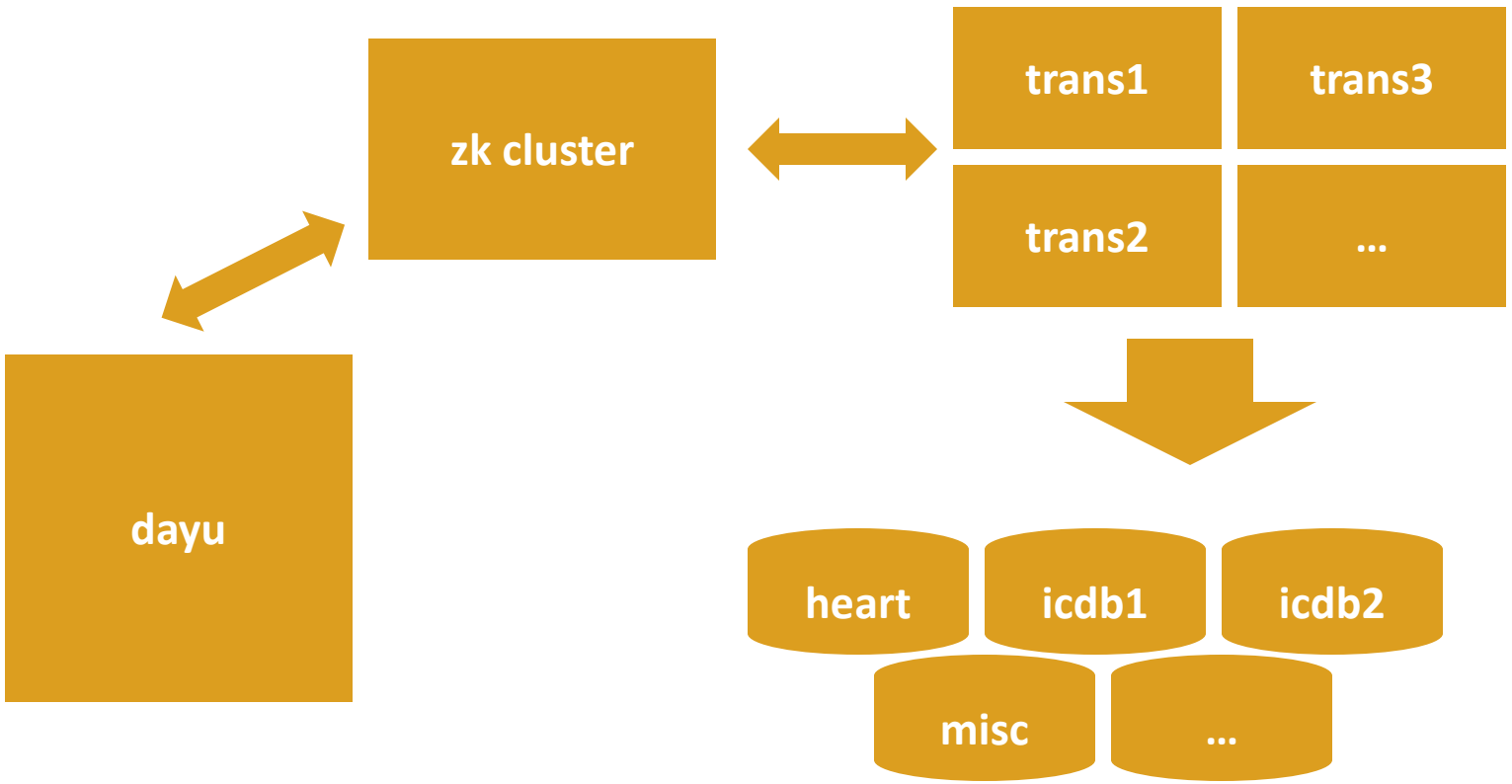
愚公-去Oracle

1. 过程类似mysql的扩容
2. 增量日志使用oracle自带的物化视图
3. 全量通过全表扫描
4. 数据重新散列使用tddl
5. 使用业务自定义转换类(DataTranslator)进行不对等变换和数据过滤

精卫-数据同步


mysql
mysql
dbsync
dbsync
有序消息中间件集合
Meta
redis
redis
mysql
mysql
mysql
mysql

大禹-控制平台



http://ops.jm.taobao.net/dayu/

课程回顾

1、 TDDL 3层数据和sequence的原理

2、 TDDL 的使用

3、 愚公迁移系统,精卫同步系统,大禹控制平台

FAQ