轻松掌握MyCat2，从入门到实战教程

-- 楼兰：你的神秘Java宝藏

一、MyCat2入门概述

1、MyCat2是什么

关于MyCat，不用做过多介绍，曾经大名鼎鼎的分库分表中间件。诞生于2013年，从MyCat1.6版本之后，陷入了一段时间的沉寂。从2021年11月低开始，重新推出新版本的MyCat2，官网地址：http://www.mycat.org.cn/ 。

重要特性一目了然，独立，是他最大的标签。然后在他的Git仓库中，对产品有一个简单的介绍：

```
MySQL Proxy using Java NIO based on Sharding SQL,Calcite ,simple and fast
```

立足于Sharding分库分表。简单、快速是MyCat2对自己最直白的声明。他的定位其实是代表了一整套基于MySQL的分布式数据处理系统，可以让MySQL拥有堪比大数据的数据处理能力。而在实际开发过程中，MyCat2用得最多的功能就是他的读写分离和分库分表，后面也是重点分享这两个功能。

2、MyCat2的故事：英雄迟暮，重装出行。

MyCat最早的版本完成与2013年年底。最早是叫做OpencloudDB，后来改成MyCat。这其中有一个重要的原因，就是打算入驻Apache，与Tomcat异曲同工。MyCat的前身阿里内部的Cobar框架。Cobar是阿里研发的关系型数据库分布式处理系统，最重要的特性就是分库分表。而MyCat则是对Cobar的一些问题进行改良，并开源出来的。

MyCat从诞生之初就带上了程序员桀骜不驯的标签，不依托于任何商业公司，以社区的形式进行维护。曾经的MyCat社区，聚集了国内大部分的IT经营，一度是中国最大的IT社区。MyCat产品也是在迅猛发展。从GitHub上使用者公布的案例中看到，在某些数量很大的系统中，MyCat能支持处理单表单月30亿级别的数据量。

但是，社区化运营毕竟给产品开发带来了不稳定性。MyCat社区一度想要开发一个叫做MycloudOA的云平台产品。但是，目标过大的结果是团队凝聚力逐渐下降。MyCat的产品版本发展到1.6就沉寂了很长一段时间。

而现在MyCat2产品相当于是重新收拾之前MyCat留下的烂摊子，重整山河，再次出发。以下是官方公布的MyCat1.6和MyCat2的功能比较。

功能	1.6	2
多语句	不支持	支持
blob值	支持一部分	支持
全局二级索引	不支持	支持
任意跨库join(包含复杂查询)	catlet支持	支持
分片表与分片表JOIN查询	ER表支持	支持
关联子查询	不支持	支持一部分
分库同时分表	不支持	支持
存储过程	支持固定形式的	支持更多
支持逻辑视图	不支持	支持
支持物理视图	支持	支持
批量插入	不支持	支持
执行计划管理	不支持	支持
路由注释	支持	支持
集群功能	支持	支持更多集群类型
自动hash分片算法	不支持	支持
支持第三方监控	支持mycat-web	支持普罗米斯,kafka日志等监控
流式合拼结果集	支持	支持
范围查询	支持	支持
单表映射物理表	不支持	支持
XA事务	弱XA	支持,事务自动恢复
支持MySQL8	需要更改mysql8的服务器配置支持	支持
虚拟表	不支持	支持
joinClustering	不支持	支持

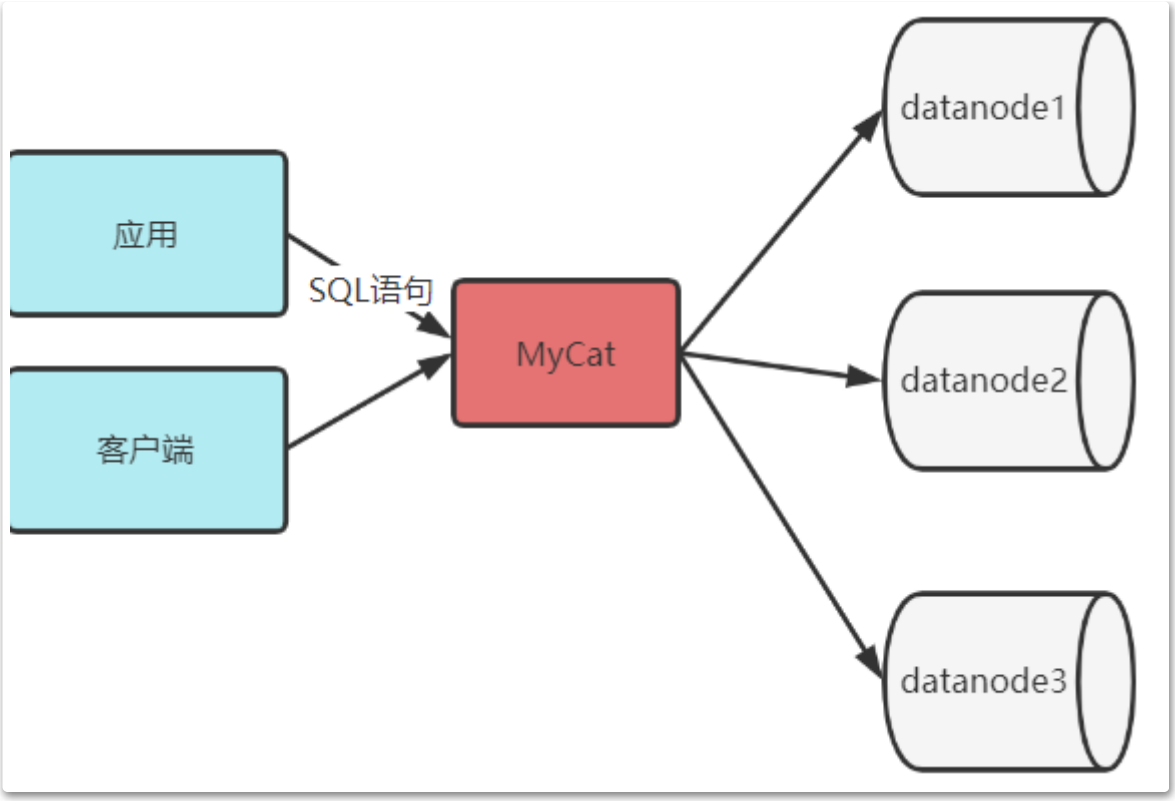
功能	1.6	2
union all语法	不支持	支持
BKAJoin	不支持	支持
优化器注释	不支持	支持
ER表	支持	支持
全局序列号	支持	支持
保存点	不支持	支持
离线迁移	支持	支持（实验）
增量迁移	CRC32算法支持	BINLOG追平（实验）
安全停机	不支持	支持（实验）
HAProxy协议	不支持	支持
会话粘滞	update后select会粘滞	update后select会粘滞且支持设置时间
全局表插入支持全局序列号	不支持	支持
全局表插入支持主表插入自增结果作为序列号	不支持	支持
外部调用的分片算法	不支持但可定制	支持

其中有个比较明显的区别在于MyCat1.6中，分库和分表规则是分开配置的，而MyCat2中，直接一起配置分库分表规则。

不过到目前为止，MyCat2社区的人数和规模相比当年MyCat1.6，还是差距很大，不过英雄归来，未来可期。

3、MyCat2的原理

MyCat的工作原理其实并不复杂。就是对请求进行“拦截”。他会拦截客户端发送过来的SQL语句，对SQL语句做一些特定的分析，如分片分析、路由分析、读写分离分析、缓存分析等。然后将此SQL发往后端的真实数据库。并将返回的结果做适当的处理，比如结果聚合、排序等，最终返回给用户。



而在工作方式上，MyCat会部署成一个MySQL服务，程序员只需要像操作普通的单机MySQL服务一样去操作MyCat。MyCat再去完成分库分表功能。而后端的数据库产品，建议还是使用MySQL。当然，MyCat2目前也在开始提交与更多数据库产品的兼容。目前提交了Oracle的兼容版本，但是还没有到发布版本。

二、MyCat2安装启动

1、准备测试环境

MyCat2是基于Java开发的，所以他的运行环境是比较简单的，只需要安装JDK即可。接下来准备一台Linux机器，搭建JDK8版本。初始搭建时，建议在这台服务器上也搭建一个MySQL服务。

> 在MyCat2的git仓库中明确说明，MyCat2目前只支持JDK8版本，其他版本的适配会在后续推出。

然后，为了上手使用MyCat2，需要搭建后端测试的MySQL服务。接下来会准备两个MySQL8服务实例，并搭建完成MySQL的主从同步集群。

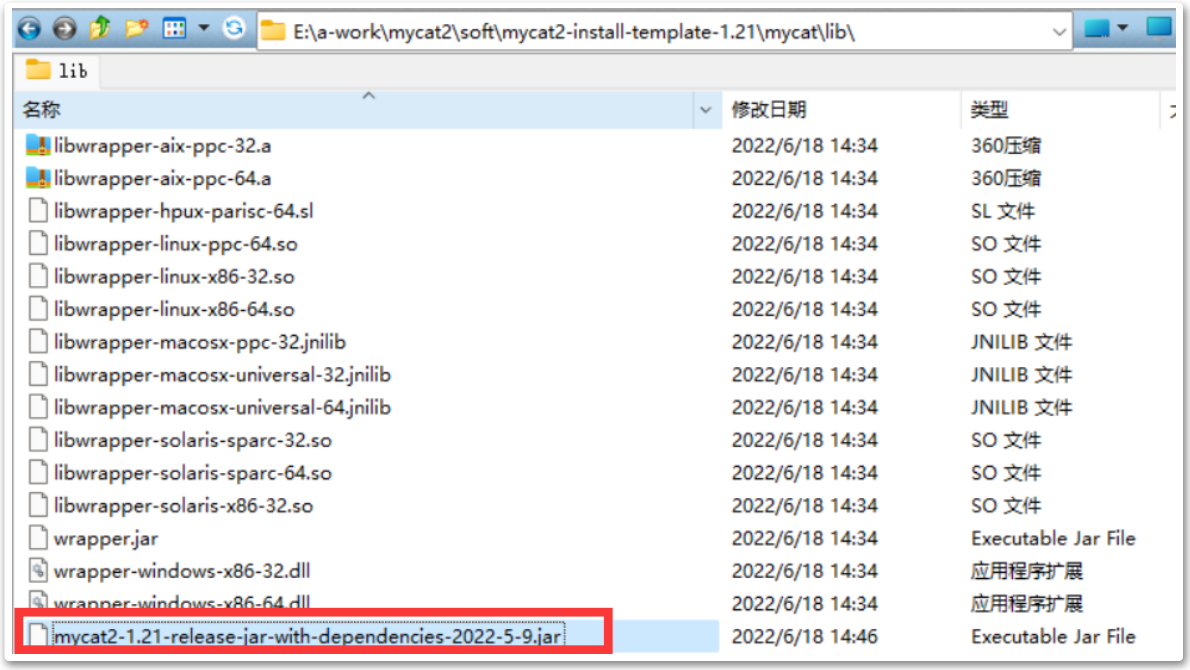
> MySQL服务搭建步骤参见：《MySQL生产环境快速搭建.pdf》

2、下载安装包

先下载一个运行环境框架包。下载地址： http://dl.mycat.org.cn/2.0/install-template/mycat2-install-template-1.20.zip。

然后可以下载MyCat2的运行包。地址：http://dl.mycat.org.cn/2.0/1.21-release/ (选择较新的jar包版本)

下载下来后，运行框架包解压，将MyCat2的运行包上传到mycat/lib目录下



然后上传到Linux服务器上。这其中，libwrapper-linux和mycat/bin目录下的mycat指令，是在linux环境上运行的文件，要调整下文件的权限，保证有运行权限。

```
chmod 777 libwrapper-linux-*
chmod 777 ../bin/mycat
```

这样MyCat2的环境就基本安装完成了。

3、启动服务

在启动MyCat服务之前，需要先修改下mycat的prototype元数据信息。这是MyCat运行所需要的基本信息。配置文件在mycat/config/datasource/prototypeDs.datasource.json.

```
{
        "dbType":"mysql",
        "idleTimeout":60000,
        "initSqls":[],
```

```
        "initSqlsGetConnection":true,
        "instanceType":"READ_WRITE",
        "maxCon":1000,
        "maxConnectTimeout":3000,
        "maxRetryCount":5,
        "minCon":1,
        "name":"prototypeDs",
        "password":"root",
        "type":"JDBC",
        "url":"jdbc:mysql://localhost:3306/mysql?
useUnicode=true&serverTimezone=Asia/Shanghai&characterEncoding=UTF-8",
        "user":"root",
        "weight":0
}
```

主要是通过url，user，password三个属性指向一个MySQL服务。如果指向的MySQL服务无法连接，那么MyCat2在启动阶段就会报错。

接下来，使用bin目录下的mycat指令，就可以启动mycat服务。

```
cd mycat/bin
./mycat start #启动 jps进程名WrapperSimpleApp
./mycat stop #停止
./mycat console #前台运行
./mycat restart #重启服务
./mycat pause #暂停
./mycat status #查看启动状态…
```

这样，就可以使用mysql的客户端工具去连接mycat了。注意端口需要改成8066。登录时的用户名和密码通过MyCat2下的conf/[用户名].user.json文件配置。默认配置了一个root用户，密码是123456。

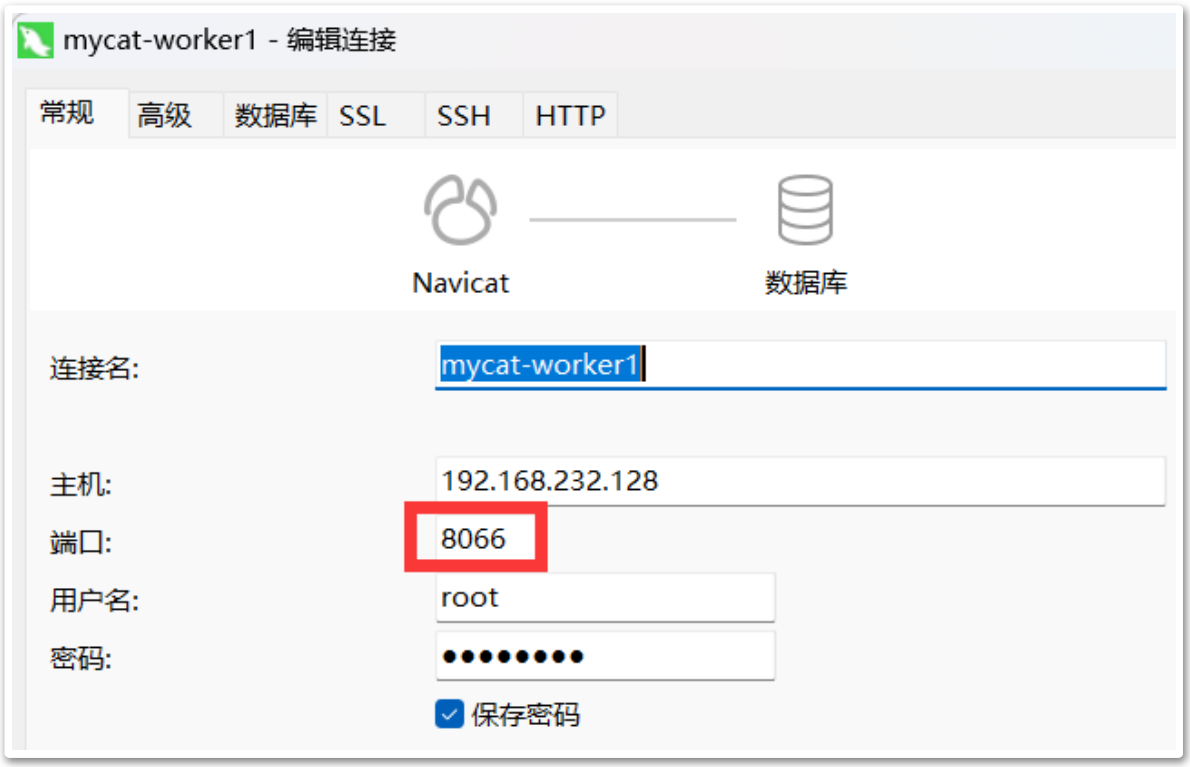
```
[oper@worker1 bin]$ /usr/local/mysql/bin/mysql -uroot -p -P 8066 -h
127.0.0.1
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.20 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

```

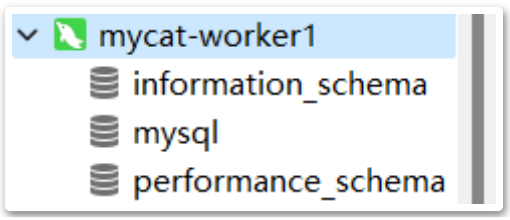
```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| `Database`         |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

到这里，MyCat2的服务就搭建起来了，接下来可以配置MyCat2的服务。

> 这里注意下，show databases；指令看到的是MyCat2服务当中的逻辑库，跟之前在prototypeDs.datasource.json配置文件中指定的数据库中的databases会有不同。

然后，既然可以用mysql指令连接MyCat2服务，那自然也可以用MySQL的客户端工具来连。例如使用navicate配置连接。

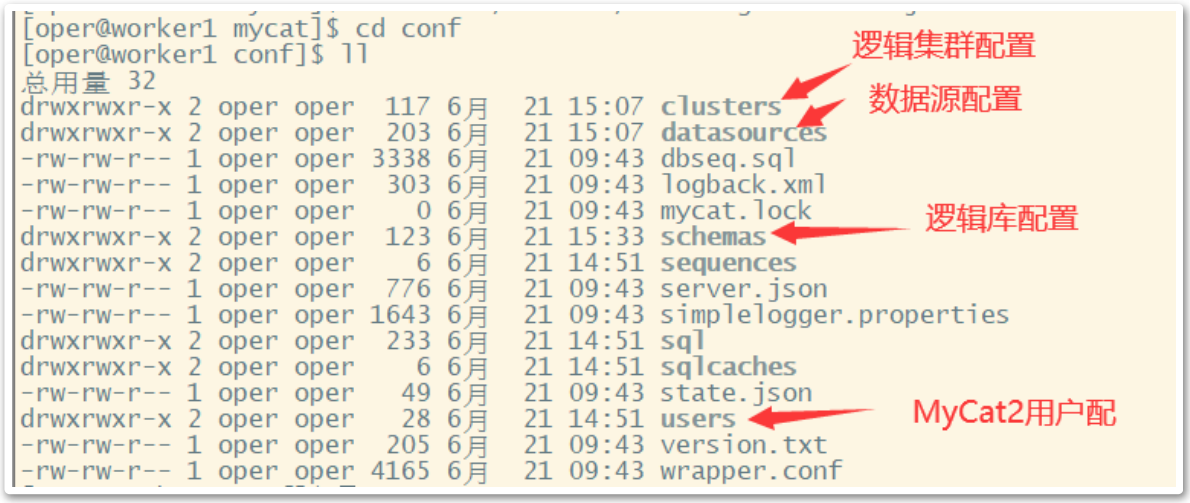




三、MyCat2的主要配置文件

MyCat2定位是一个数据库中间件，他并不存储数据。MyCat2所有的功能都可以理解为通过一系列配置文件定制一系列业务规则，通过与其他数据库(目前主要是MySQL)协作，提供具体的业务功能。所以，目前MyCat2的所有功能都体现在他的配置文件当中。

我们先来了解一下MyCat2的主要配置文件。



1、用户 users

目录： mycat/conf/users

命名规则：{用户名}.user.json

主要配置内容：

```
mycat/conf/users/root.user.json
{
"ip":null,
"password":"123456",
"transactionType":"xa",
"username":"root",
"isolation":3
}

```

主要字段含义：

- #ip：客户端访问ip，建议为空,填写后会对客户端的ip进行限制

- username：MyCat2登录的用户名
- password：MyCat2登录的密码
- dialect：数据库类型。目前就MySQL
- transactionType：事务类型

  可选值:
  - proxy 本地事务，兼容性最好，但是分布式场景事务控制不严格。
  - xa XA分布式事务,需要后端数据库能够支持 XA
  - 可以通过语句实现切换 set transaction_policy = 'xa'set transaction_policy = 'proxy'
  - 可以通过语句查询 SELECT @@transaction_policy

2、数据源 datasource

目录： mycat/conf/datasources

命名规则：{数据源名字}.datasource.json

主要配置内容：

```
mycat/conf/datasources/prototype.datasources.json
{
        "dbType":"mysql",
        "idleTimeout":60000,
        "initSqls":[],
        "initSqlsGetConnection":true,
        "instanceType":"READ_WRITE",
        "maxCon":1000,
        "maxConnectTimeout":3000,
        "maxRetryCount":5,
        "minCon":1,
        "name":"prototypeDs",
        "password":"123456",
        "type":"JDBC",
        "url":"jdbc:mysql://localhost:3306/mysql?useUnicode=true&serverTimezone=Asia/Shanghai&characterEncoding=UTF-8",
        "user":"root",
        "weight":0
}
```

主要字段含义：

- dbType：数据库类型 mysql
- name：在MyCat2中定义的数据源名字。

- user, password,url： 实际数据库的JDBC属性
- instanceType：配置实例只读还是读写。可选值有READ_WRITE,READ,WRITE
- weight：负载均衡权重
- idleTimeout：空闲连接超时时间。
- 其他连接相关参数。

3、集群 clusters

目录： mycat/conf/clusters

命名规则：{集群名字}.cluster.json

主要配置内容：

```
mycat/conf/clusters/prototype.cluster.json
{
        "clusterType":"MASTER_SLAVE",
        "heartbeat":{
                "heartbeatTimeout":1000,
                "maxRetry":3,
                "minSwitchTimeInterval":300,
                "slaveThreshold":0
        },
        "masters":[
                "prototypeDs"
        ],
        "maxCon":200,
        "name":"prototype",
        "readBalanceType":"BALANCE_ALL",
        "switchType":"SWITCH"
}
```

主要字段含义：

- clusterType：集群类型。主要用得多的是 SINGLE_NODE:单一节点；MASTER_SLAVE:普通主从；另外针对高可用集群还支持MHA和MGR
- readBalanceType：查询负载均衡策略 可选值：

  BALANCE_ALL(默认值)：获取集群中所有数据源；
  BALANCE_ALL_READ：获取集群中允许读的数据源；
  BALANCE_READ_WRITE：获取集群中允许读写的数据源,但允许读的数据源优先；
  BALANCE_NONE：获取集群中允许写数据源,即主节点中选择

- switchType：切换类型 NOT_SWITCH:不进行主从切换；SWITCH:进行主从切换
- masters：主从集群中的主库名字。 从库用replicas配置。

4、逻辑库 schema

目录： mycat/conf/schemas

命名规则：{库名}.schema.json

主要配置内容：

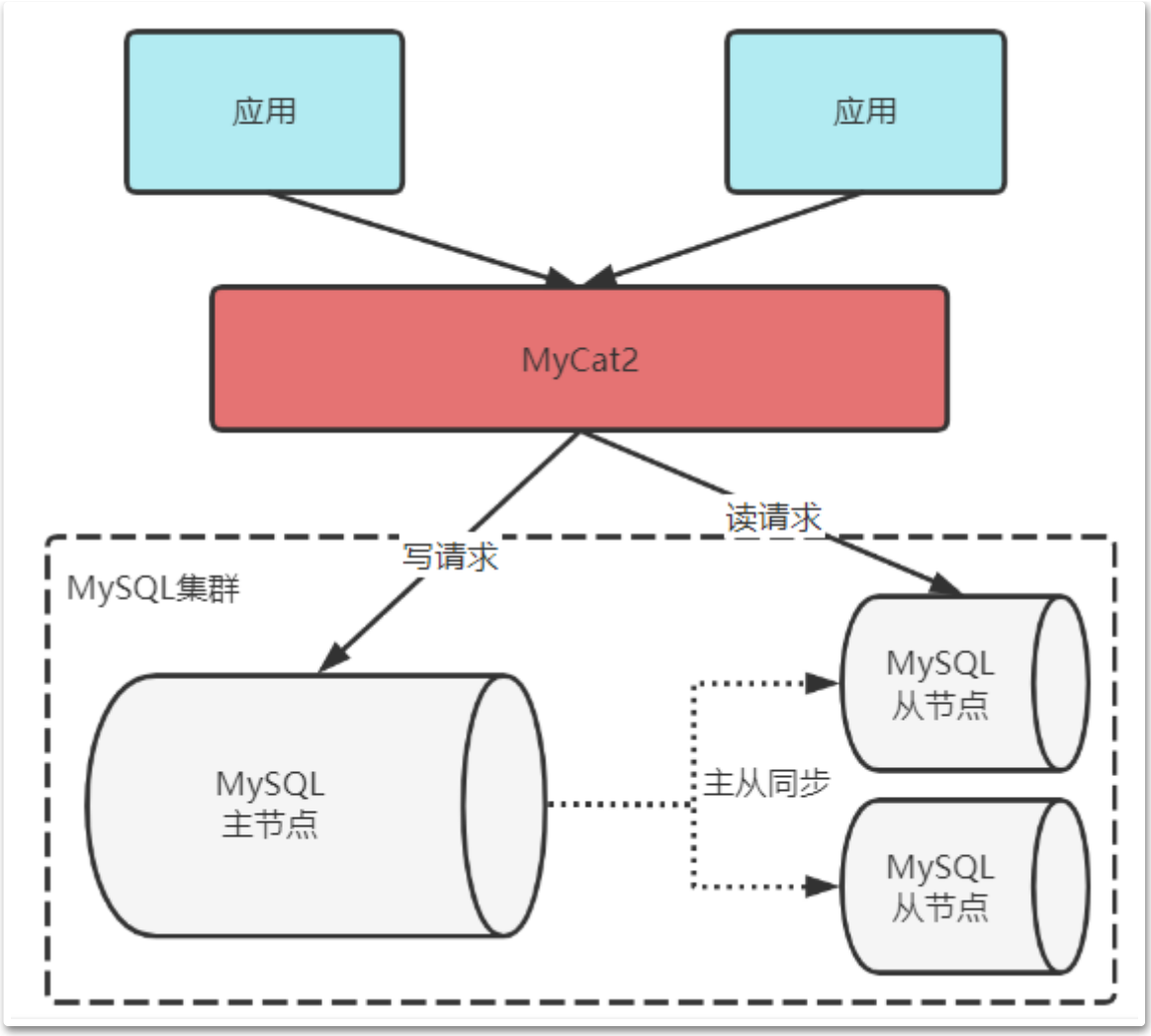
```
{
        "customTables":{},
        "globalTables":{},
        "normalProcedures":{},
        "normalTables":{},
        "schemaName":"wrdb",
        "shardingTables":{},
        "targetName":"",
        "views":{}
}
```

主要字段含义：

- schema：逻辑库名
- targetName： 目的真实数据源或集群
- normalTables：常规表。如果物理表已经存在或者在每次启动服务时需要加载表定义，就可以写在这里。

四、MyCat2读写分离实战

1、读写分离实验方案说明：



读写分离的基本原理是MyCat2将Update、Delete、Insert这一类写数据的请求转发到MySQL集群中的主节点，然后将Select类的读数据的请求转发到MySQL集群中的相关节点，主要是从节点。而MySQL的主节点与从节点之间，通常会使用MySQL的主从同步机制来进行数据同步，从而保证读请求能够读取到最新的数据结果。

在这个过程中，MySQL的主从同步其实是与MyCat2无关的，接下来的实验，我们会把重点放在MyCat2的读写分离功能上，而不要太注重主从之间的数据一致性。

2、MyCat2读写分离规则配置

在新版本MyCat2中配置读写分离规则，相比MyCat会简单很多。按照以下几个步骤配置即可：

step1、配置真实数据源

登录MyCat2后，可以使用MyCat2提供的注解方式配置真实数据源。

```
# 创建数据源
# 创建dbw写库，指向集群中的master服务
/*+ mycat:createDataSource{"name":"dbW",
"url":"jdbc:mysql://192.168.232.128:3306/mysql?
useSSL=false&characterEncoding=UTF-8&useJDBCCompliantTimezoneShift=true",
"user":"root", "password":"root" } */;
# 创建dbR读库，指向集群中的slave服务。如果有多个从库，依次继续创建
/*+ mycat:createDataSource{ "name":"dbR",
"url":"jdbc:mysql://192.168.232.129:3306/mysql?
useSSL=false&characterEncoding=UTF-8&useJDBCCompliantTimezoneShift=true",
"user":"root", "password":"root" } */;

#查询配置数据源结果
/*+ mycat:showDataSources{} */;
```

> 1、真实数据源也就是实际存储数据的数据源，例如之前在prototypeDs.datasource.json文件中配置的prototypeDs也是一个真实数据源
>
> 2、创建了两个数据库后，进入MyCat的部署目录，在conf/datasource目录下，可以看到对应的数据库配置文件dbR.datasource.json和dbW.datasource.json。这是MyCat2自动创建的配置文件，里面有关于这个数据源的详细配置信息。
>
> 配置文件中的几个属性：
>
> # idleTimeout：空闲连接超时时间
>
> # initSqls：初始化sql
>
> # initSqlsGetConnection：对于 jdbc 每次获取连接是否都执行initSqls
>
> # instanceType：配置实例只读还是读写 。 可选值:READ_WRITE,READ,WRITE

step2、配置MySQL集群

接下来需要在MyCat2中配置一个包含真实数据源的MySQL集群，通过集群来实现读写分离。

```
#更新集群信息,添加dr0从节点.
/*! mycat:createCluster{"name":"WRSplitCluster","masters":
["dbW"],"replicas":["dbR"]} */;
#查看配置集群信息
/*+ mycat:showClusters{} */;
```

1、这样就创建一个名为WRSplitCluster的集群，masters指定主库，replicas指定从库。其中，replicas从库可以配置多个，表示有多个从库。masters也可以配置多个，第一个表示主库，后面的表示备用库。

2、配置集群后，MyCat2会在conf/cluster目录下创建对应的集群配置文件，WRSplitCluster.cluster.json。里面包含了关于这个集群的详细配置信息。另外，也可以看到，MyCat2默认情况下也提供了一个名为prototype的集群。

配置文件中几个重要的配置：

clusterType：集群类型

可选值:　SINGLE_NODE:单一节点

　MASTER_SLAVE:普通主从

　MHA：MHA 集群

　MGR：MGR 集群

readBalanceType：查询负载均衡策略

可选值:　BALANCE_ALL(默认值)：获取集群中所有数据源

　BALANCE_ALL_READ：获取集群中允许读的数据源

　BALANCE_READ_WRITE：获取集群中允许读写的数据源,但允许读的数据源优先

　BALANCE_NONE：获取集群中允许写数据源,即主节点中选择

switchType：切换类型

可选值:　NOT_SWITCH:不进行主从切换

> SWITCH:进行主从切换

step3、配置逻辑库

有了集群后，还需要在MyCat2中配置一个逻辑库，指向这个集群。这样，客户端就只需要操作这个逻辑库，由MyCat2去完成真实的读写分离逻辑。

```
-- 1、在mycat2的服务中声明一个逻辑库。
create database wrdb;
-- 2、在mycat2的部署目录下，找到对应的配置文件,conf/schema/wrdb.schema.json。在其中增加targetName:WRSplitCluster属性。指向真实的集群。

{
        "customTables":{},
        "globalTables":{},
        "normalProcedures":{},
        "normalTables":{},
        "schemaName":"wrdb",
        "targetName":"WRSplitCluster",
        "shardingTables":{},
        "views":{}
}
--3、手动修改配置文件之后，需要重启MyCat2服务，让配置文件生效。
```

> 逻辑库即客户端直接操作的库。后续在库中建表，建视图、存储过程等，最终都会体现在对应的schema.json中。

step4、读写分离测试

这样就完成了MyCat2的读写分离配置。接下来在MySQL主从集群的基础上测试MyCat2的读写分离。

首先，在主库创建一个wrdb1的数据库，在其中简单创建一张表demotable。这样就会在从库上也创建相同的表。

```
CREATE TABLE `wrdb`.`demotable`  (
  `id` int(0) NULL
);
```

然后，在MyCat2中也同样创建这张表，schema同样指向wrdb。

接下来，在MyCat2中往demotable中插入一条数据。测试MyCat2是否将写请求转发到主库。

```
insert into wrdb.demotable values (1);
```

这时，可以检查下主库和从库对应的demotable表中是否有数据插入。如果也正常插入了测试数据，说明MyCat2已经正常将数据插入到了主库(因为只有插入了主库，从库才会有测试数据。)

接下来，在MyCat2中查询demotable的数据，测试MyCat2是否将读请求在集群之间转发。

```
select * from wrdb.demotable;
```

这时，能正常查到数据，但是并不能验证MyCat2对SQL语句做了转发。简单的验证方式可以将从库中的demotable表中的数据手动删除掉。然后在MyCat2中多次尝试查询demotable数据，会发现数据时而能查到，时而查不到。(查主库时能查到，查从库时就查不到)这就说明MyCat2在集群中不断转发读请求。

> 注意，这样会破坏主从之间的数据一致性。

接下来，可以自行测试下在WRSplitCluster.cluster.json中配置不同的负载均衡策略。例如BALANCE_ALL_READ只查从库。

> 从这个配置过程可以理解到， MyCat2只是个数据库中间件，不存储数据。MyCat2所有的功能都是通过conf目录下的配置文件进行定制。而客户端的所有相关配置语句都会最终写入到conf目录下的配置文件当中。

五、MyCat2分库分表配置

1、分库分表规则配置

对于分库分表功能，MyCat2提供了非常简单的配置方式。可以在MyCat2客户端直接完成配置，不需要手动调整配置文件。基础的配置方式依然是数据源 -》 集群 。 然后在建表时指定分库分表规则。

Step1： 配置数据源

```
#第一个写库
/*+
mycat:createDataSource{"name":"dw0","url":"jdbc:mysql://192.168.232.128:3306
","user":"root","password":"root"} */;
#第一个读库
/*+
mycat:createDataSource{"name":"dr0","url":"jdbc:mysql://192.168.232.128:3306
","user":"root","password":"root"} */;
#第二个写库
/*+
mycat:createDataSource{"name":"dw1","url":"jdbc:mysql://192.168.232.129:3306
","user":"root","password":"root"} */;
#第二个读库
/*+
mycat:createDataSource{"name":"dr1","url":"jdbc:mysql://192.168.232.129:3306
","user":"root","password":"root"} */;
```

Step2： 配置集群

```
#在mycat2终端输入
#配置第一组集群
/*!mycat:createCluster{"name":"c0","masters":["dw0"],"replicas":["dr0"]}*/;
#配置第二组集群
/*!mycat:createCluster{"name":"c1","masters":["dw1"],"replicas":["dr1"]}*/;
```

> 这里集群名c0,c1是MyCat2默认支持的分片集群名字。不建议修改集群名字。

Step3：配置全局表

全局表表示在所有数据分片上都相同的表。比如字典表。

```
#添加数据库db1
CREATE DATABASE shardingdb;
#在建表语句中加上关键字 BROADCAST（广播，即为全局表）
CREATE TABLE shardingdb.`t_dict` (
`id` INT NOT NULL,
`dict_id` INT,
`item_id` INT,
`item_value` VARCHAR(32) NULL,
PRIMARY KEY(id),
KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 BROADCAST;
```

配置完成后的全局表会写入到mycat/conf/schemas/db1.schema.json文件中。这样下次启动服务时就能够初始化表结构。

然后可以插入几条数据进行测试：

```
insert into shardingdb.t_dict values(1,1,0,'正常');
insert into shardingdb.t_dict values(2,1,1,'正常');
insert into shardingdb.t_dict values(3,2,0,'男');
insert into shardingdb.t_dict values(4,2,1,'女');
insert into shardingdb.t_dict values(5,3,1,'会员');
insert into shardingdb.t_dict values(6,3,1,'非会员');
```

数据会插入到两个库中，并且两个库中都有全部的数据。

Step4：配置分片表

分片表表示逻辑表中的数据会分散保存到多个数据分片中。

```
#在 Mycat 终端直接运行建表语句进行数据分片
CREATE TABLE shardingdb.orders(
id BIGINT NOT NULL,
order_type INT,
customer_id INT,
amount DECIMAL(10,2),
PRIMARY KEY(id),
KEY `id` (`id`)
)ENGINE=INNODB DEFAULT CHARSET=utf8
dbpartition BY mod_hash(id) tbpartition BY mod_hash(id)
tbpartitions 1 dbpartitions 2;
#数据库分片规则，表分片规则，以及各分多少片
```

核心就是后面的分片规则。 dbpartition表示分库规则，tbpartition表示分表规则。而mod_hash表示按照customer_id字段取模进行分片。

接下来可以往测试表中插入部分测试数据进行验证。

```
#在MyCat2终端可以插入数据
INSERT INTO shardingdb.orders(id,order_type,customer_id,amount)
VALUES(1,101,100,100100);
INSERT INTO shardingdb.orders(id,order_type,customer_id,amount)
VALUES(2,101,100,100300);
INSERT INTO shardingdb.orders(id,order_type,customer_id,amount)
VALUES(3,101,101,120000);
INSERT INTO shardingdb.orders(id,order_type,customer_id,amount)
VALUES(4,101,101,103000);
INSERT INTO shardingdb.orders(id,order_type,customer_id,amount)
VALUES(5,102,101,100400);
INSERT INTO shardingdb.orders(id,order_type,customer_id,amount)
VALUES(6,102,100,100020);

SELECT * FROM shardingdb.orders;
```

而数据都分片保存在两个真实数据库中。

Step5：配置关联表

关联表也成为绑定表或者ER表。表示数据逻辑上有关联性的两个或多个表，例如订单和订单详情表。对于关联表，通常希望他们能够有相同的分片规则，这样在进行关联查询时，能够快速定位到同一个数据分片中。

```
#在 Mycat 终端创建一个与order表关联的订单详情表
CREATE TABLE shardingdb.orders_detail(
`id` BIGINT NOT NULL,
detail VARCHAR(2000),
order_id INT,
PRIMARY KEY(id)
)ENGINE=INNODB DEFAULT CHARSET=utf8
dbpartition BY mod_hash(order_id) tbpartition BY mod_hash(order_id)
tbpartitions 1
```

这个订单详情表通过order_id字段关联到订单表，所以这两个表希望能够有相同的分片规则。而MyCat2中对于关联表，不需要有过多的声明，他可以根据分年规则自行判断。

```
#在MyCat2终端查看关联表关系。
/*+ mycat:showErGroup{}*/
#groupId相同的表示相同的组，该组中的表具有相同的存储分布。
```

然后可以插入一些测试数据来验证一下关联表关系。

```
INSERT INTO orders_detail(id,detail,order_id) VALUES(1,'detail1',1);
INSERT INTO orders_detail(id,detail,order_id) VALUES(2,'detail1',2);
INSERT INTO orders_detail(id,detail,order_id) VALUES(3,'detail1',3);
INSERT INTO orders_detail(id,detail,order_id) VALUES(4,'detail1',4);
INSERT INTO orders_detail(id,detail,order_id) VALUES(5,'detail1',5);
INSERT INTO orders_detail(id,detail,order_id) VALUES(6,'detail1',6);

SELECT * FROM orders o INNER JOIN orders_detail od ON od.order_id=o.id;
```

在MyCat2中能查询到全部的数据，但是真实数据也是分布在不同的数据库中。

2、其他常用的分片规则

(1) MOD_HASH

如果分片值是字符串则先对字符串进行hash转换为数值类型

分库键和分表键是同键

分表下标=分片值%(分库数量*分表数量)

分库下标=分表下标/分表数量

分库键和分表键是不同键

分表下标= 分片值%分表数量

分库下标= 分片值%分库数量

```
create table travelrecord (
 ....
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by MOD_HASH (id) dbpartitions 6
tbpartition by MOD_HASH (id) tbpartitions 6;
```

(2) RANGE_HASH

RANGE_HASH(字段1, 字段2, 截取开始下标)

仅支持数值类型,字符串类型

当时字符串类型时候,第三个参数生效

计算时候优先选择第一个字段,找不到选择第二个字段

如果是字符串则根据下标截取其后部分字符串,然后该字符串hash成数值

根据数值按分片数取余

要求截取下标不能少于实际值的长度

两个字段的数值类型要求一致

```
create table travelrecord(
...
)ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by RANGE_HASH(id,user_id,3) dbpartitions 3
tbpartition by RANGE_HASH(id,user_id,3) tbpartitions 3;
```

（3）RIGHT_SHIFT

RIGHT_SHIFT(字段名,位移数)
仅支持数值类型
分片值右移二进制位数,然后按分片数量取余

```
create table travelrecord(
 ...
)ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by RIGHT_SHIFT(id,4) dbpartitions 3
tbpartition by RIGHT_SHIFT(user_id,4) tbpartitions 3;
```

（4）YYYYMM

仅用于分库
(YYYY*12+MM)%分库数.MM 是 1-12

```
create table travelrecord (
 ....
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by YYYYMM(xxx) dbpartitions 8
tbpartition by xxx(xx) tbpartitions 12;
```

（4） YYYYWEEK 支持分库分表

(YYYY*54+WEEK)%分片数
WEEK的范围是1-53

```
create table travelrecord (
 ....
) ENGINE=InnoDB DEFAULT CHARSET=utf8
dbpartition by YYYYWEEK(xx) dbpartitions 8
tbpartition by xxx(xx) tbpartitions 12;
```