

Mycat从入门到精通第2周

Mycat术语与原理

Mycat基本功能介绍

Mycat配置入门

Mycat术语与原理

分片

垂直分片 VS 水平分片

tableA

tableB

col1	col2	col3	col4	col5	col6	col7

col1 col2 col3 col4 col5 col6 col7

TableA

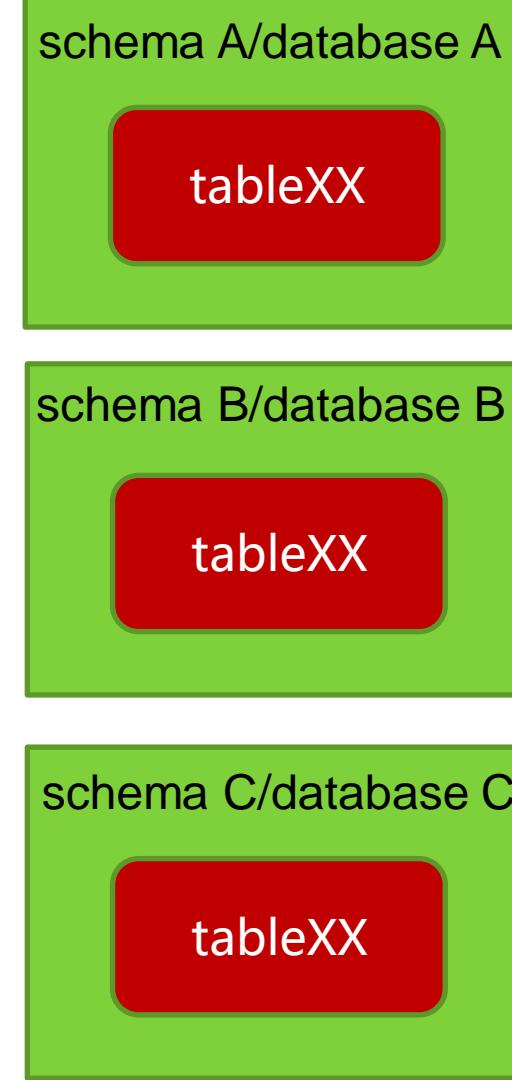
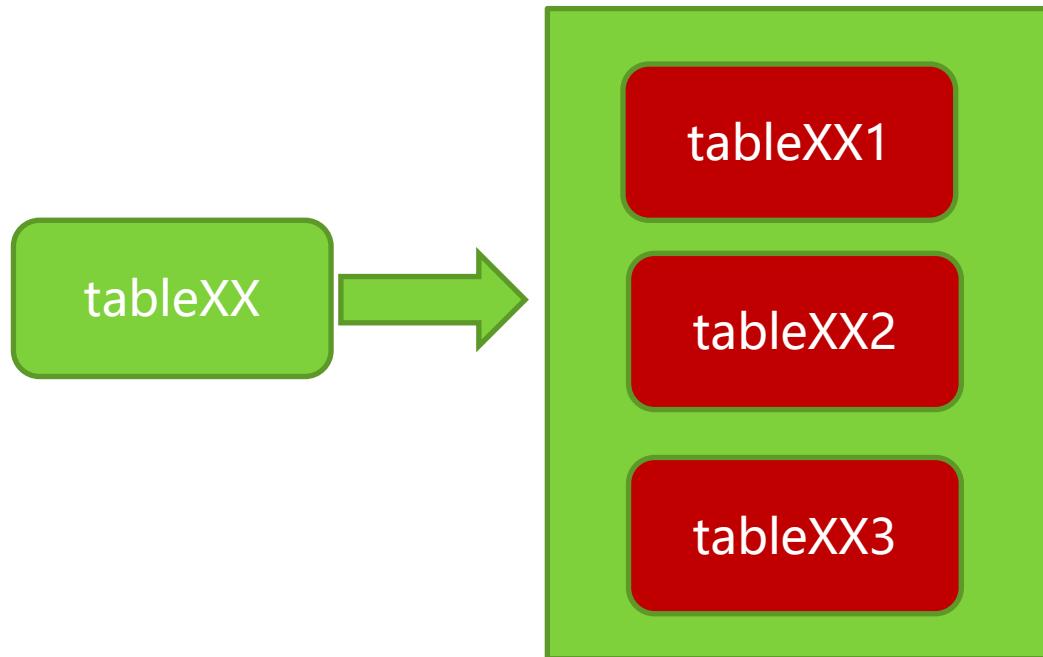
TableB

Mycat术语与原理

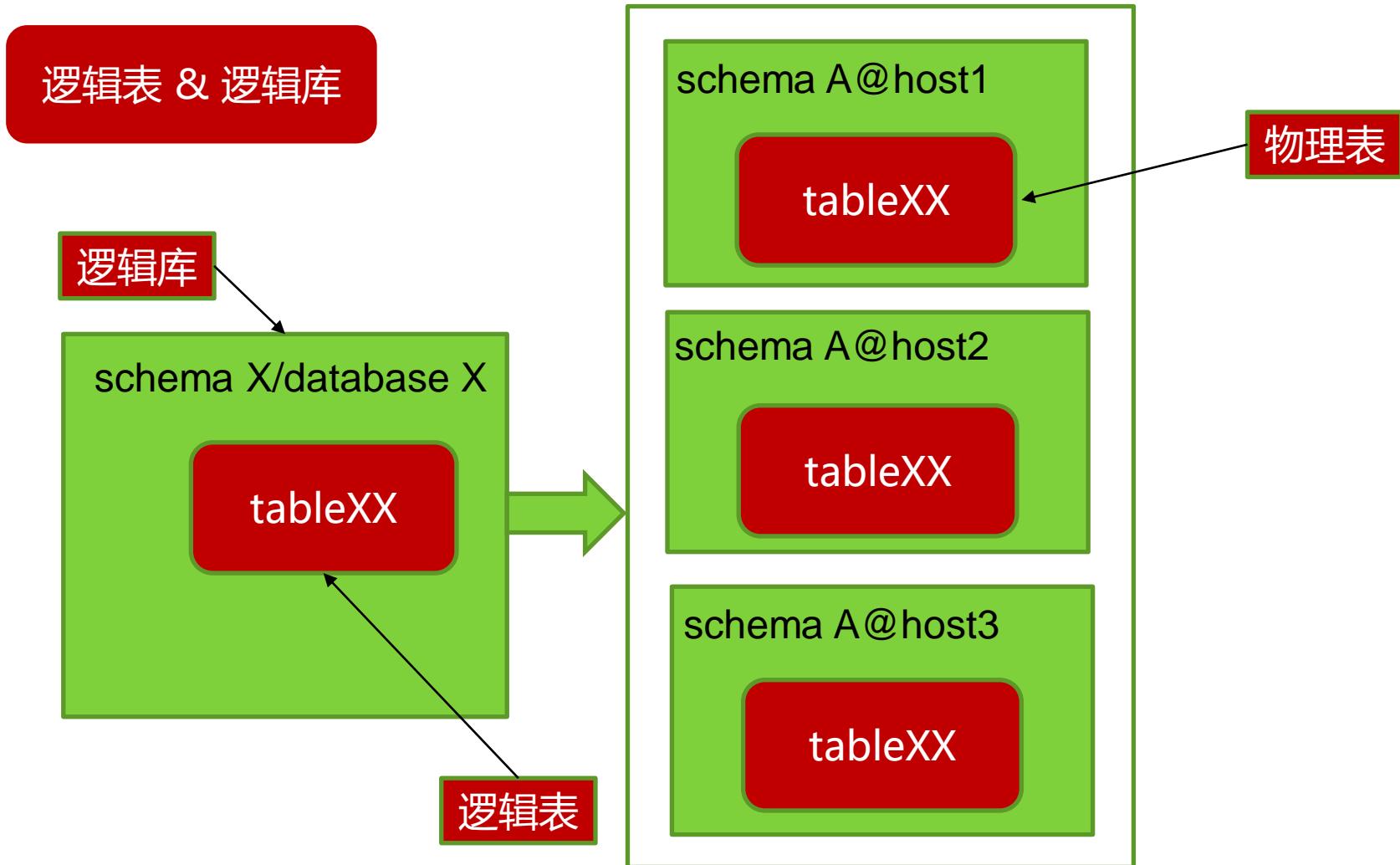
分片

分表 VS 分库

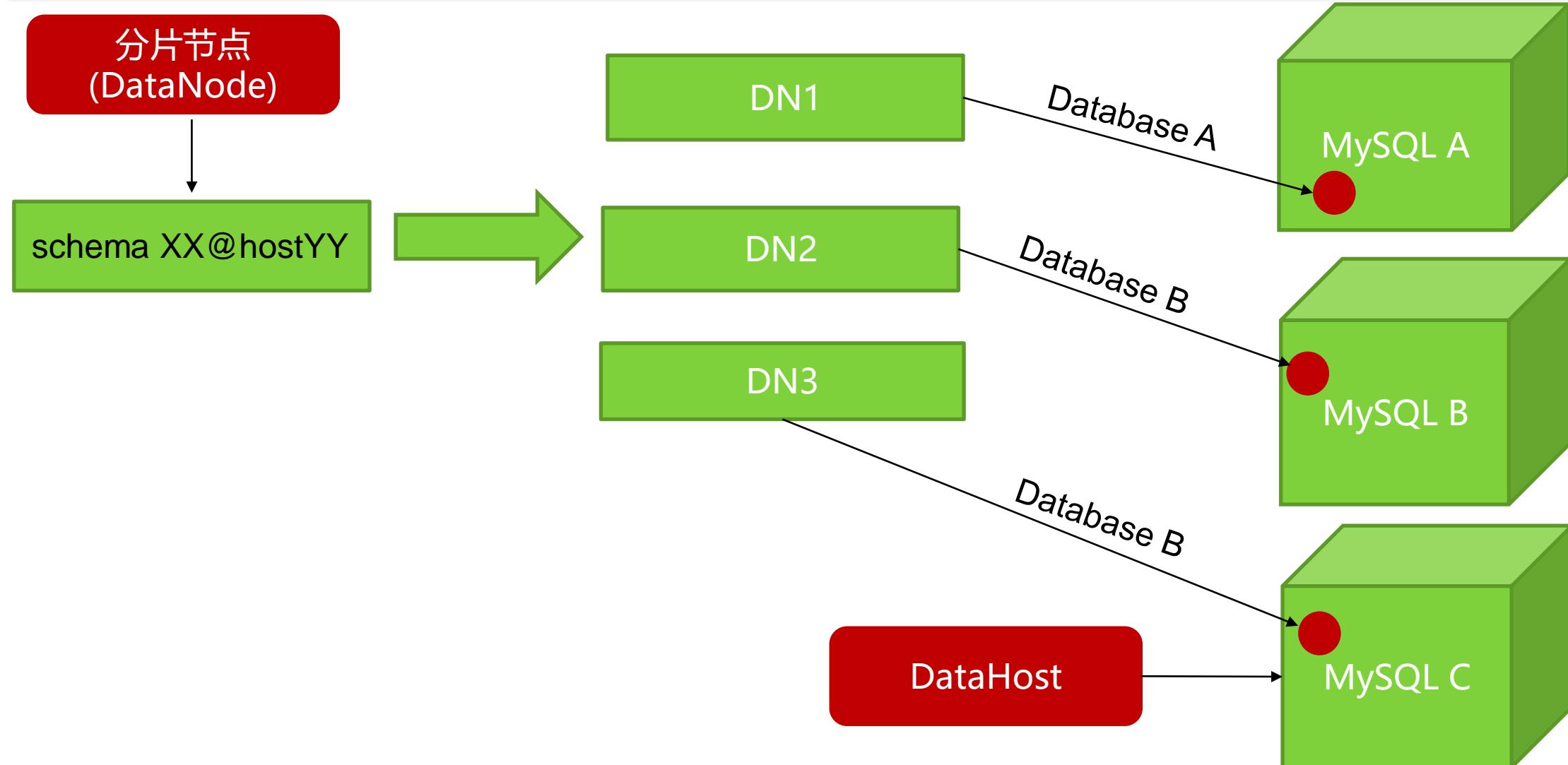
schema A/database A



Mycat术语与原理

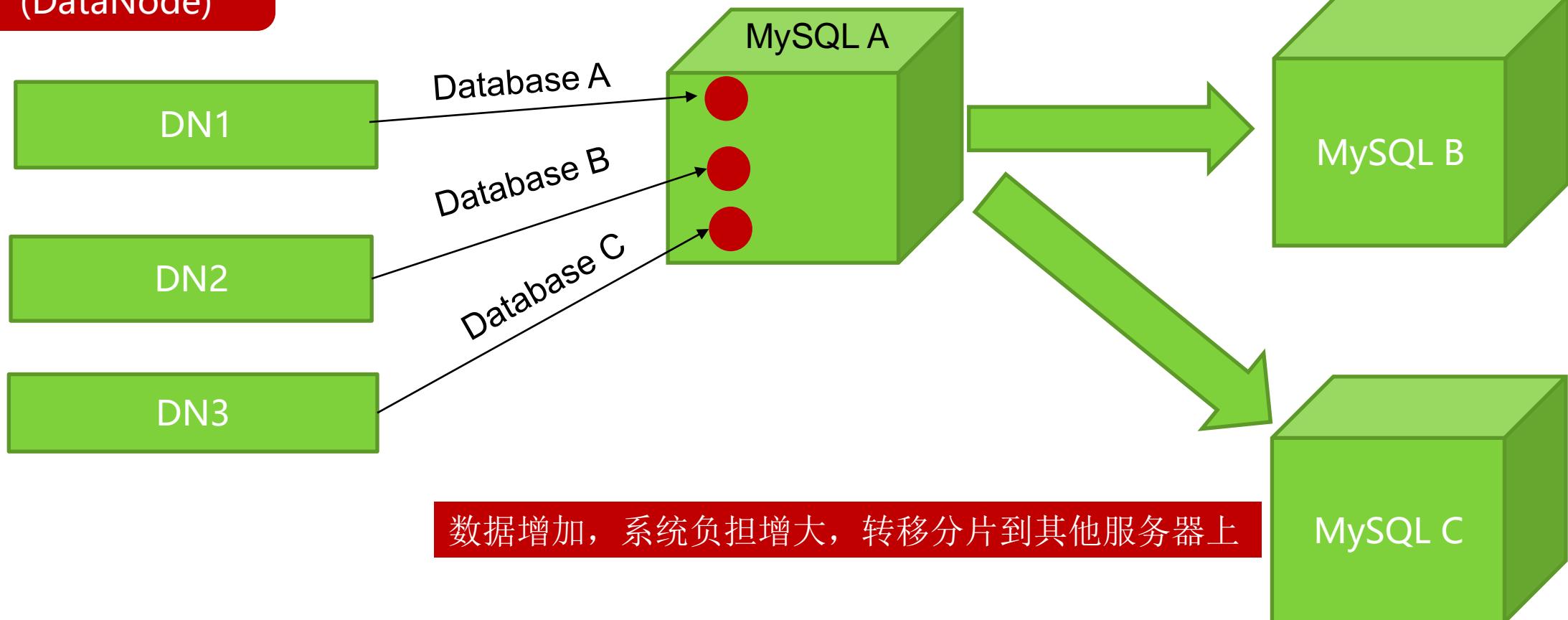


Mycat术语与原理



Mycat术语与原理

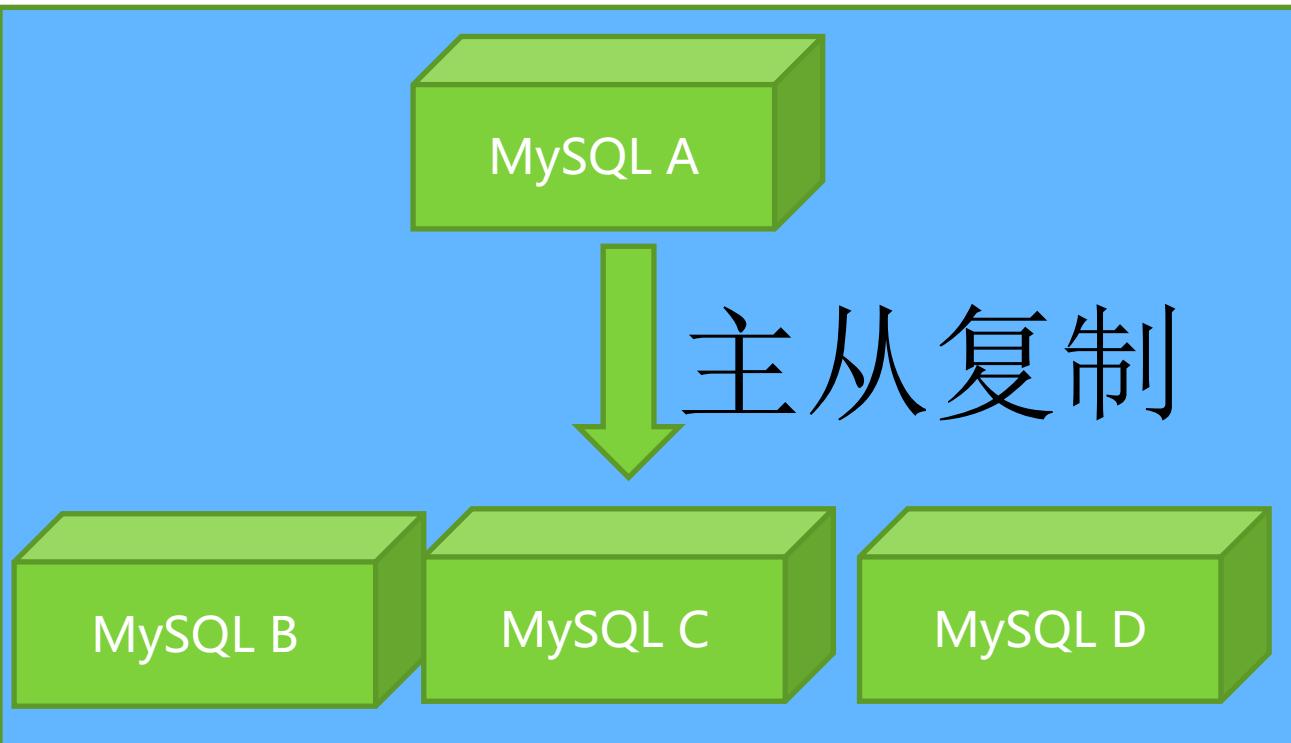
分片节点
(DataNode)



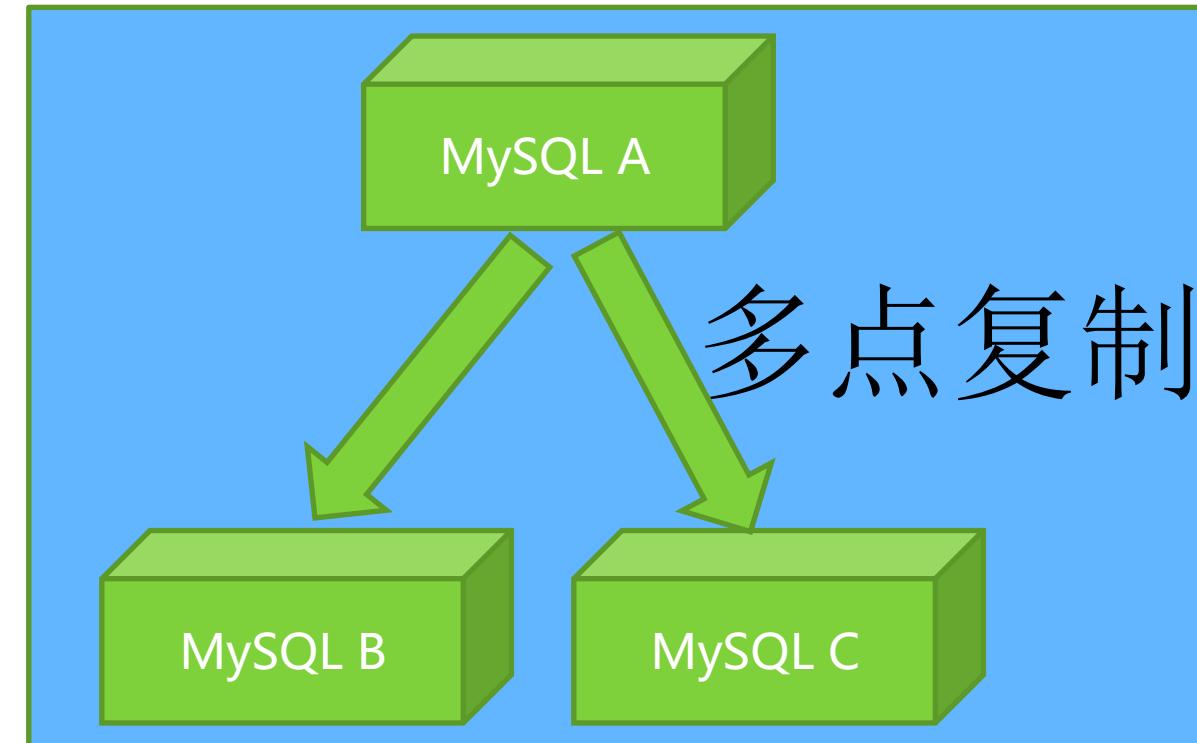
Mycat术语与原理

DataHost(MySQL Rep Group)

Mysql Master/Slave



Mysql Galera Cluster



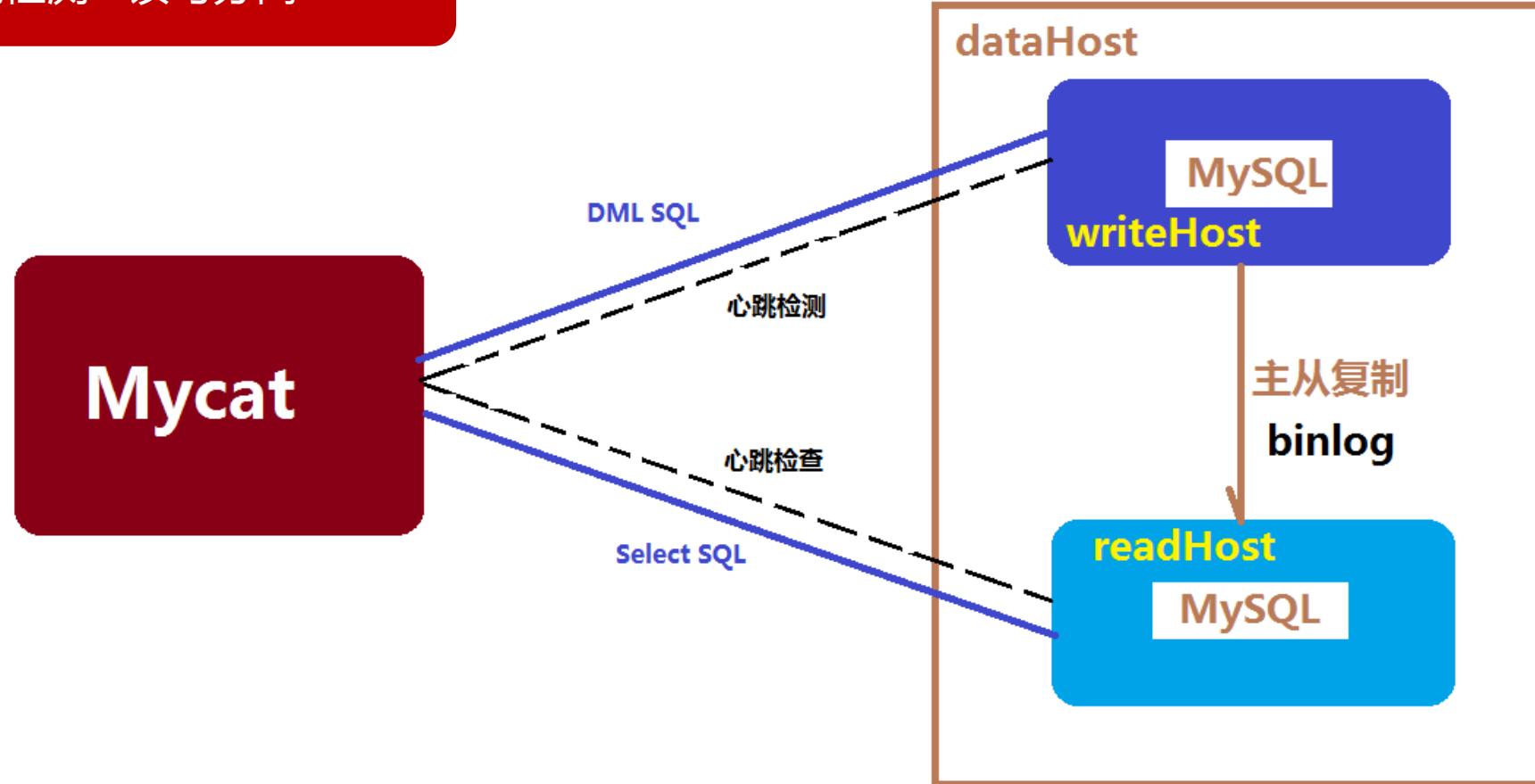
Mycat术语与原理

```
<dataNode name="dn1" dataHost="localhost1" database="db1" />
<dataNode name="dn2" dataHost="localhost1" database="db2" />
<dataNode name="dn3" dataHost="localhost1" database="db3" />

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0" writeType="0" dbType="mysql"
dbDriver="native" switchType="1" slaveThreshold="100">
<heartbeat>select user()</heartbeat>
<!-- can have multi write hosts -->
<writeHost host="hostM1" url="localhost:3306" user="root" password="123456">
<!-- can have multi read hosts -->
</writeHost>
<writeHost host="hostS1" url="localhost:3316" user="root" password="123456" />
```

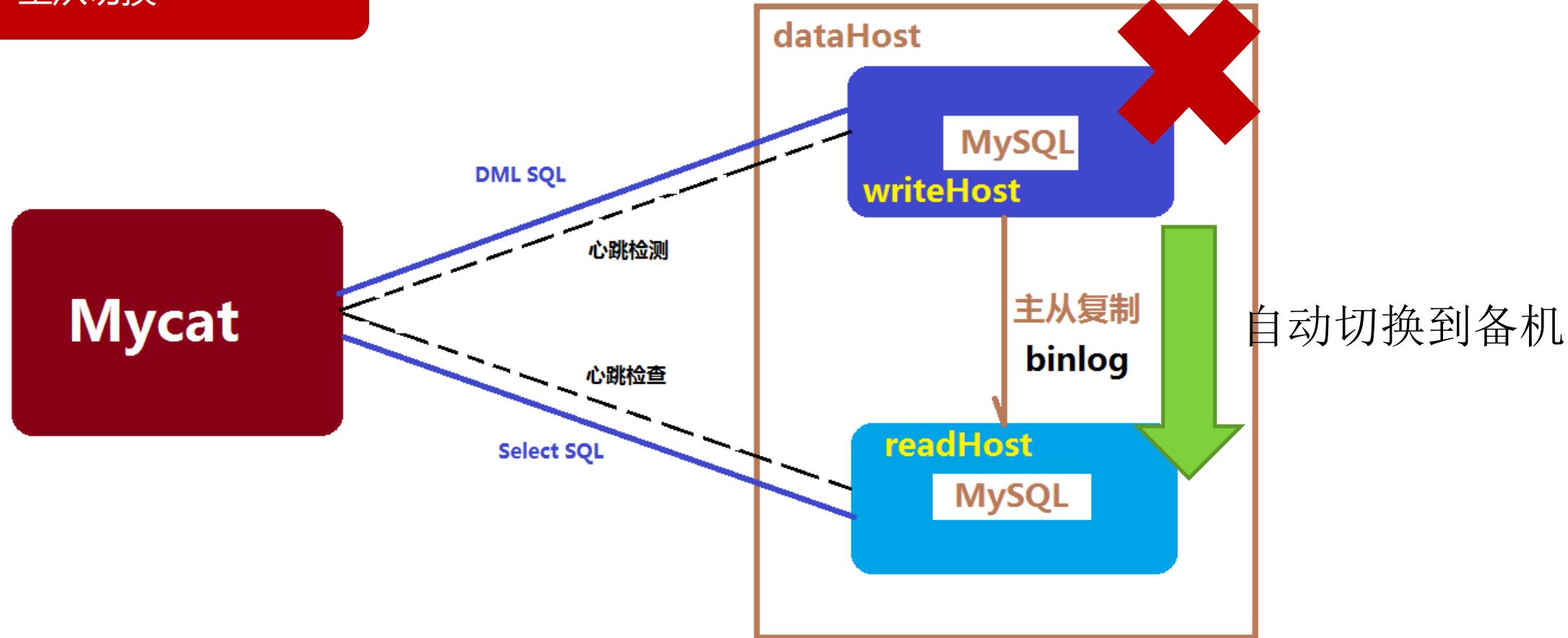
Mycat术语与原理

心跳检测&读写分离

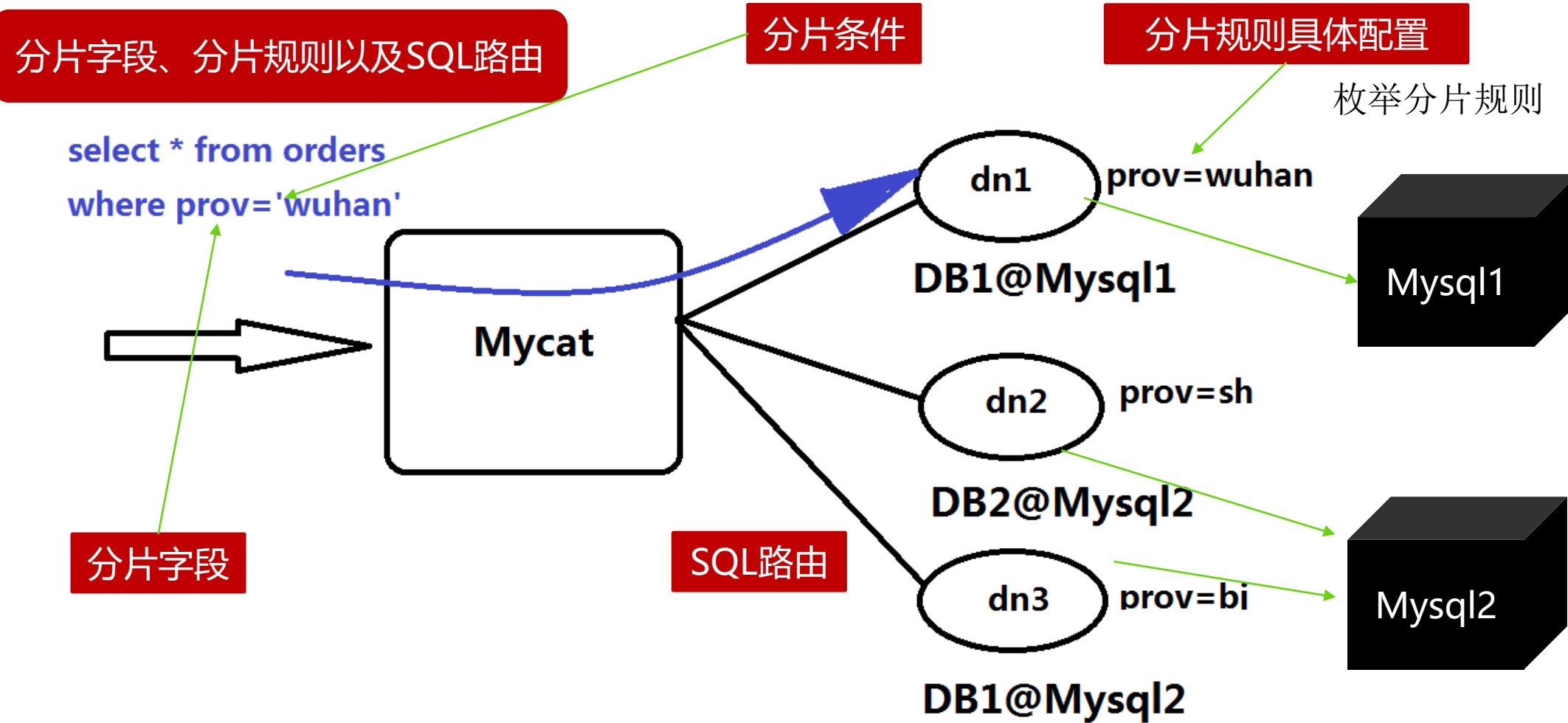


Mycat术语与原理

主从切换



Mycat术语与原理



Mycat术语与原理

分片字段、分片规则

```
<table name="employee" primaryKey="ID" dataNode="dn1,dn2" rule="sharding-by-intfile" />
```

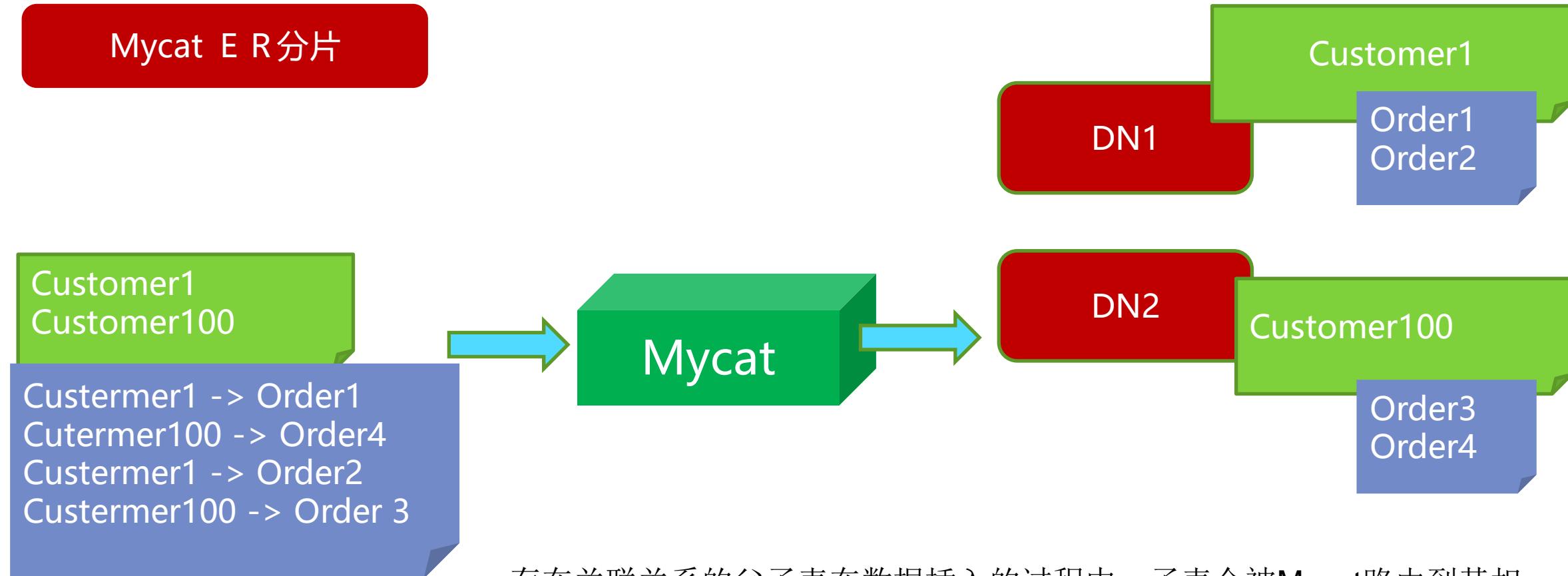
分片规则=分片字段+分片函数

```
<tableRule name="sharding-by-intfile">
  <rule>
    <columns>sharding_id</columns>
    <algorithm>hash-int</algorithm>
  </rule>
</tableRule>
```

简单映射函数，根据分片字段的值，返回分片Id

```
<function name="hash-int" class="org.opencloudb.route.function.PartitionByFileMap">
  <property name="mapFile">partition-hash-int.txt</property>
</function>
```

Mycat术语与原理

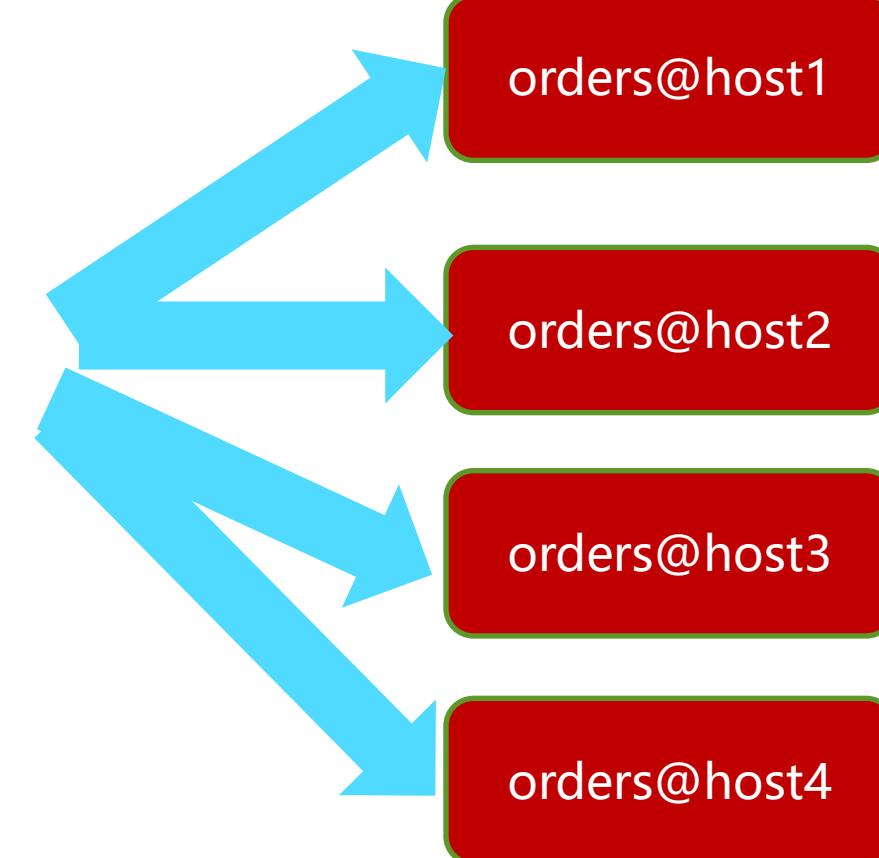


存在关联关系的父子表在数据插入的过程中，子表会被Mycat路由到其相关父表记录的节点上，从而父子表的Join查询可以下推到各个数据库节点上完成，这是最高效的跨节点Join处理技术，也是Mycat首创

Mycat术语与原理

Mycat全局表

insert into orders (xxx)



每个节点同时并发插入和更新数据，每个节点都可以读取数据，提升读性能的同时解决跨节点Join的效率

Mycat术语与原理

全局序列号

分片情况下，MySQL自身的自增长主键无法保证唯一

在数据库中建立一张表，存放sequence名称(name)，sequence当前值(current_value)，步长(increment) int类型每次读取多少个sequence，假设为K)等信息

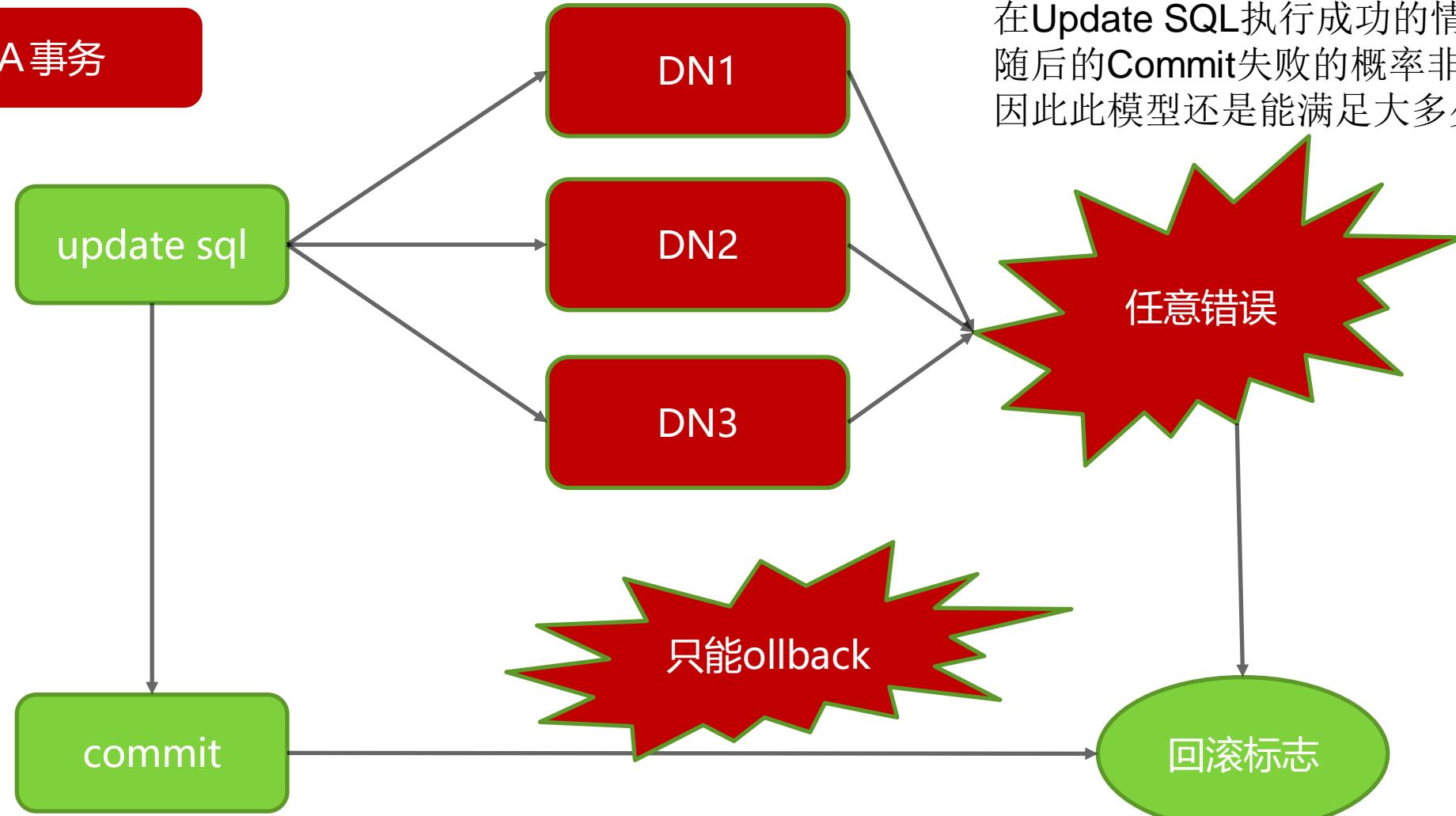
next value for MYCATSEQ_seqXXX

Mycat自增长主键

基于全局序列号，每个需要定义自增长主键的表创建一个全局序列号

Mycat术语与原理

弱XA事务

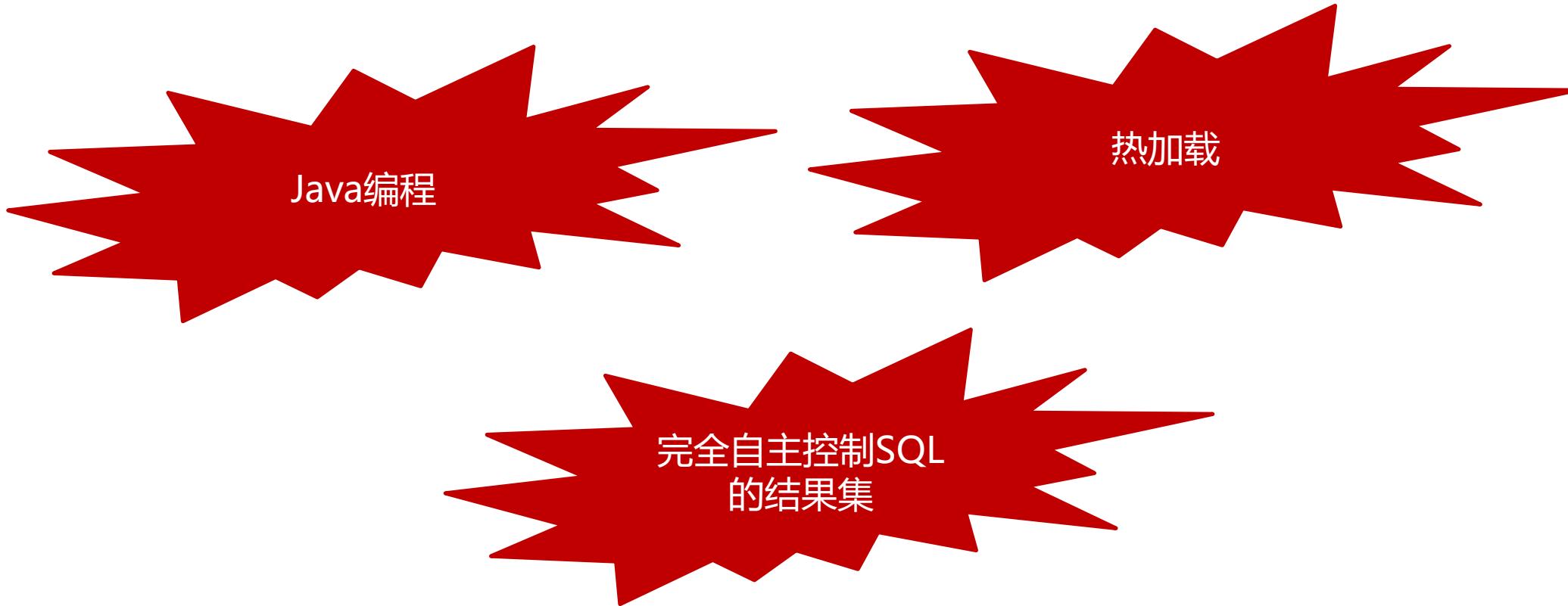


在Update SQL执行成功的情况下，
随后的Commit失败的概率非常小
因此此模型还是能满足大多少应用的要求

Mycat术语与原理

Mycat Catlet (HBT技术)

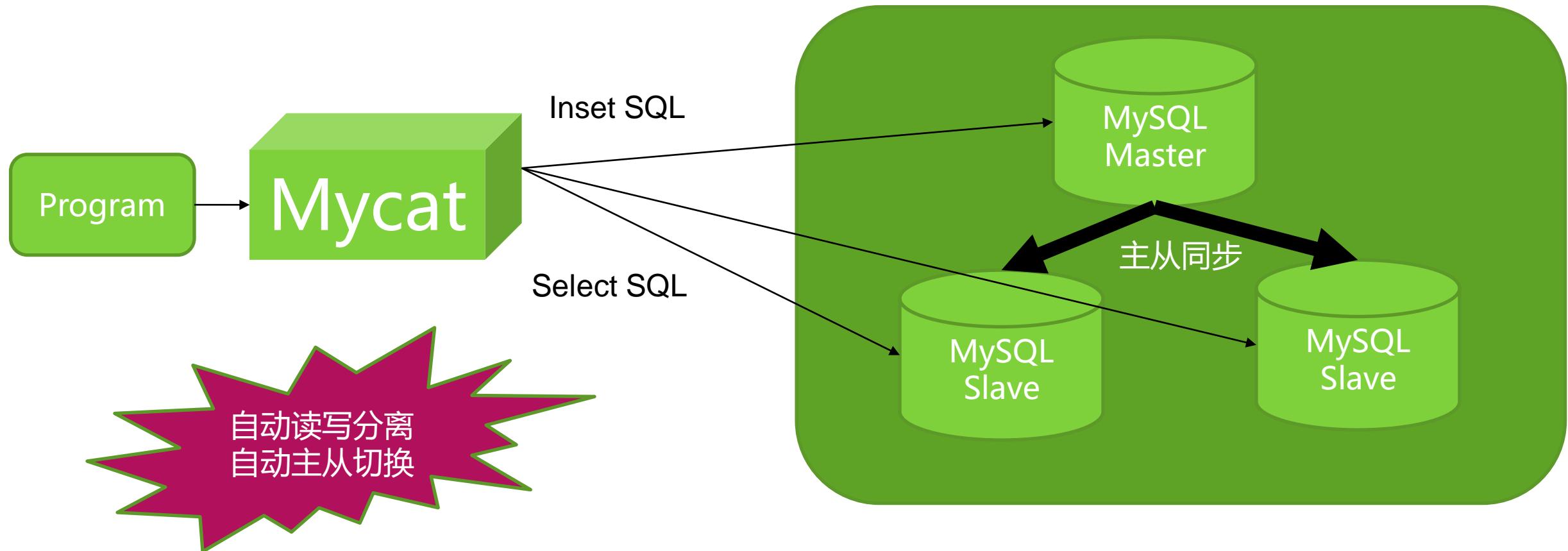
Catlet是Java编写的一段程序，类似数据库中的存储过程，可以实现任意复杂SQL的Join、Group、Order等功能



Mycat基本功能介绍

一：高可用性与MySQL读写分离

基于主从时延的主从同步能力



Mycat基本功能介绍

事务内的SQL，默认走写节点，以注释/*balance*/开头，则会根据balance="1"或"2"去获取 b. 非事务内的SQL，开启读写分离默认根据balance="1"或"2"去获取，以注释/*balance*/开头则会走写解决部分已经开启读写分离，但是需要强一致性数据实时获取的场景走写

负载均衡类型，目前的取值有3种：

1. balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的writeHost上。
2. balance="1", 全部的readHost与stand by writeHost参与select语句的负载均衡，简单的说，当双主双从模式(M1->S1, M2->S2, 并且M1与 M2互为主备)，正常情况下，M2,S1,S2都参与select语句的负载均衡。
3. balance="2", 所有读操作都随机的在writeHost、readhost上分发。
4. balance="3", 所有读请求随机的分发到writerHost对应的readhost执行，writerHost不负担读压力

Mycat基本功能介绍

```
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"          writeType="0"
dbType="mysql" dbDriver="native" switchType="2" slaveThreshold="100">
<heartbeat>show slave status </heartbeat>
    <writeHost host="hostM1" url="localhost:3306" user="root"                  password="123456"
/>
    <writeHost host="hostS1" url="localhost:3316" user="root"                  password="123456"
/>
</dataHost>
```

Mycat心跳机制通过检测 show slave status 中的 "Seconds_Behind_Master", "Slave_IO_Running", "Slave_SQL_Running" 三个字段来确定当前主从同步的状态以及Seconds_Behind_Master主从复制时延，当Seconds_Behind_Master>slaveThreshold时，读写分离筛选器会过滤掉此Slave机器

Mycat基本功能介绍

switchType="3" ,MyCAT心跳检查语句配置为 show status like 'wsrep%'，开启MySQL集群复制状态状态绑定的读写分离与切换机制

```
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0" writeType="0" dbType="mysql"  
dbDriver="native" switchType="3" > <heartbeat> show status like 'wsrep%'</heartbeat> <writeHost  
host="hostM1" url="localhost:3306" user="root" password="123456"> </writeHost> <writeHost  
host="hostS1" url="localhost:3316" user="root" password="123456" ></writeHost> </dataHost>
```

Mycat基本功能介绍

conf/log4j.xml中配置日志输出级别为debug时，当选择节点的时候，会输出如下日志：

16:37:21.660 DEBUG [Processor0-E3] (PhysicalDBPool.java:333) -select read source hostM1 for dataHost:localhost1

16:37:21.662 DEBUG [Processor0-E3] (PhysicalDBPool.java:333) -select read source hostM1 for dataHost:localhost1

根据这个信息，可以确定某个SQL发往了哪个读（写）节点，据此可以分析判断是否发生了读写分离。

用MySQL客户端连接到Mycat的9066管理端口，执行show @@datanode，也能看出负载均衡的情况，其中execute字段表明该分片上执行过的SQL累计数：

NAME	DATHOST	INDEX	TYPE	ACTIVE	IDLE	SIZE	EXECUTE	TOTAL_TIME	MAX_TIME	MAX_SQL	RECOVERY_TIME	LAST_SQL
dn1	localhost1/db1	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[0]	localhost1/db\$0	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[1]	localhost1/db\$1	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[2]	localhost1/db\$2	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[3]	localhost1/db\$3	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[4]	localhost1/db\$4	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[5]	localhost1/db\$5	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[6]	localhost1/db\$6	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[7]	localhost1/db\$7	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[8]	localhost1/db\$8	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[9]	localhost1/db\$9	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[10]	localhost1/db\$10	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[11]	localhost1/db\$11	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[12]	localhost1/db\$12	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[13]	localhost1/db\$13	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[14]	localhost1/db\$14	0	mysql	0	1	1000	1	0	0	0	-1	
dn1[15]	localhost1/db\$15	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[16]	localhost1/db\$16	0	mysql	0	0	1000	0	0	0	0	-1	
dn1[17]	localhost1/db\$17	0	mysql	0	0	1000	0	0	0	0	-1	

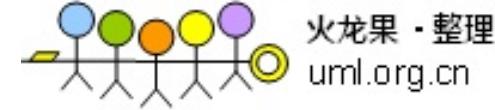
Mycat基本功能介绍

Mycat主从切换

需要配置多个WriteHost节点

switchType属性

- -1 表示不自动切换 - 1 默认值，自动切换 - 2 基于MySQL主从同步的状态决定是否切换
心跳语句为 `show slave status`
- 3 基于MySQL galaxy cluster的切换机制（适合集群）（
1.4.1）
心跳语句为 `show status like 'wsrep%`



Mycat基本功能介绍

Mycat主从切换

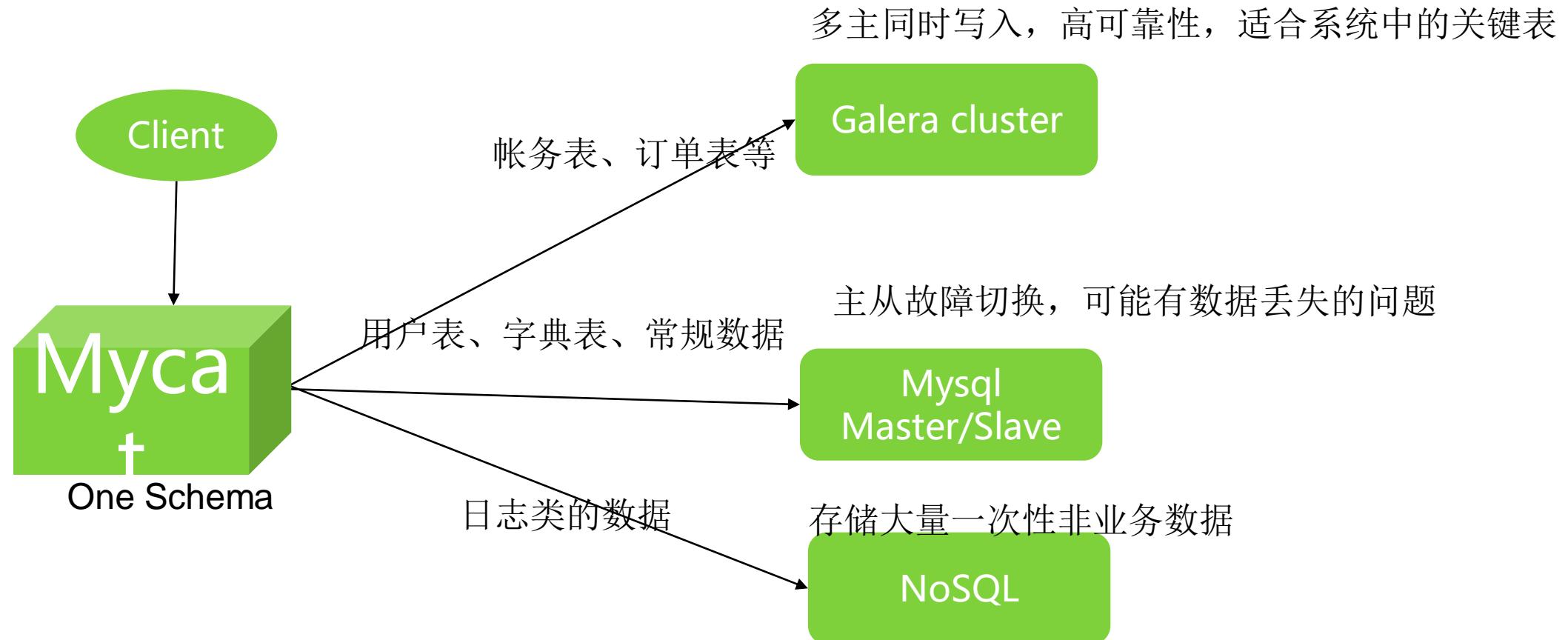
`conf/dnindex.properties`文件记录了当前使用的是哪个写节点

```
#update  
#Mon Oct 05 13:49:48 CST 2015  
localhost1=0
```

›mysql -utest -ptest -P9066

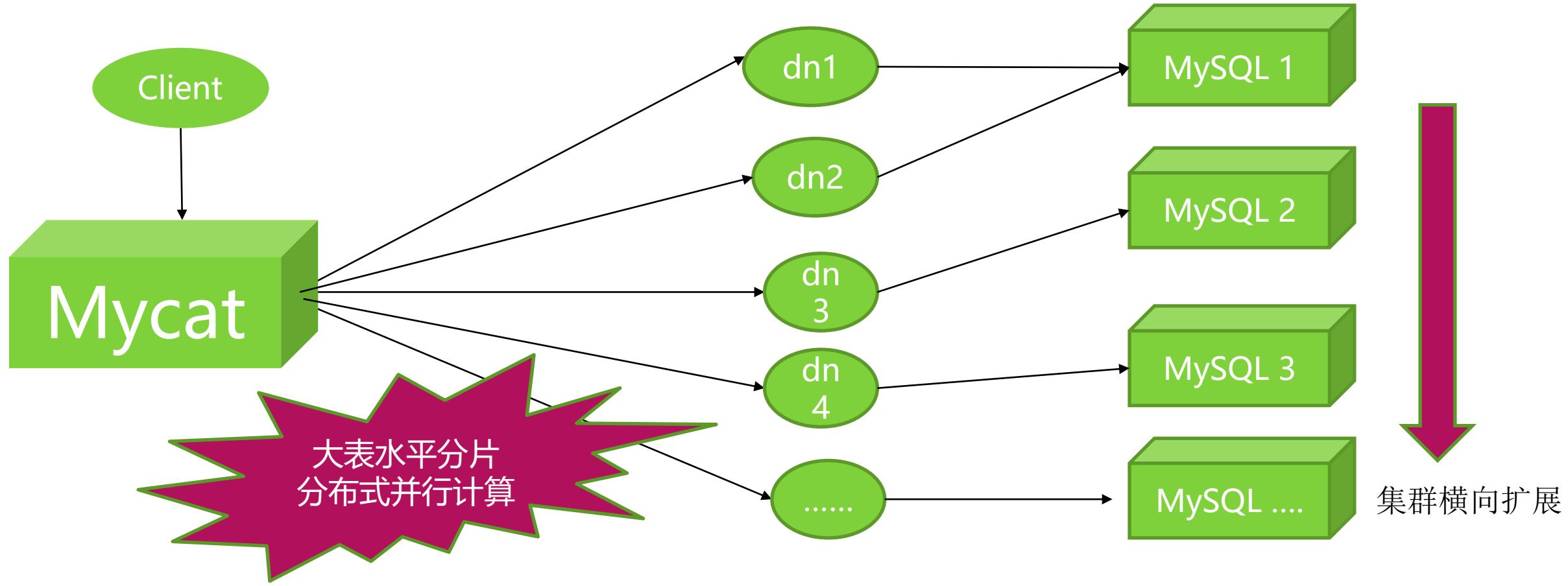
Mycat基本功能介绍

二：业务数据分级存储保障



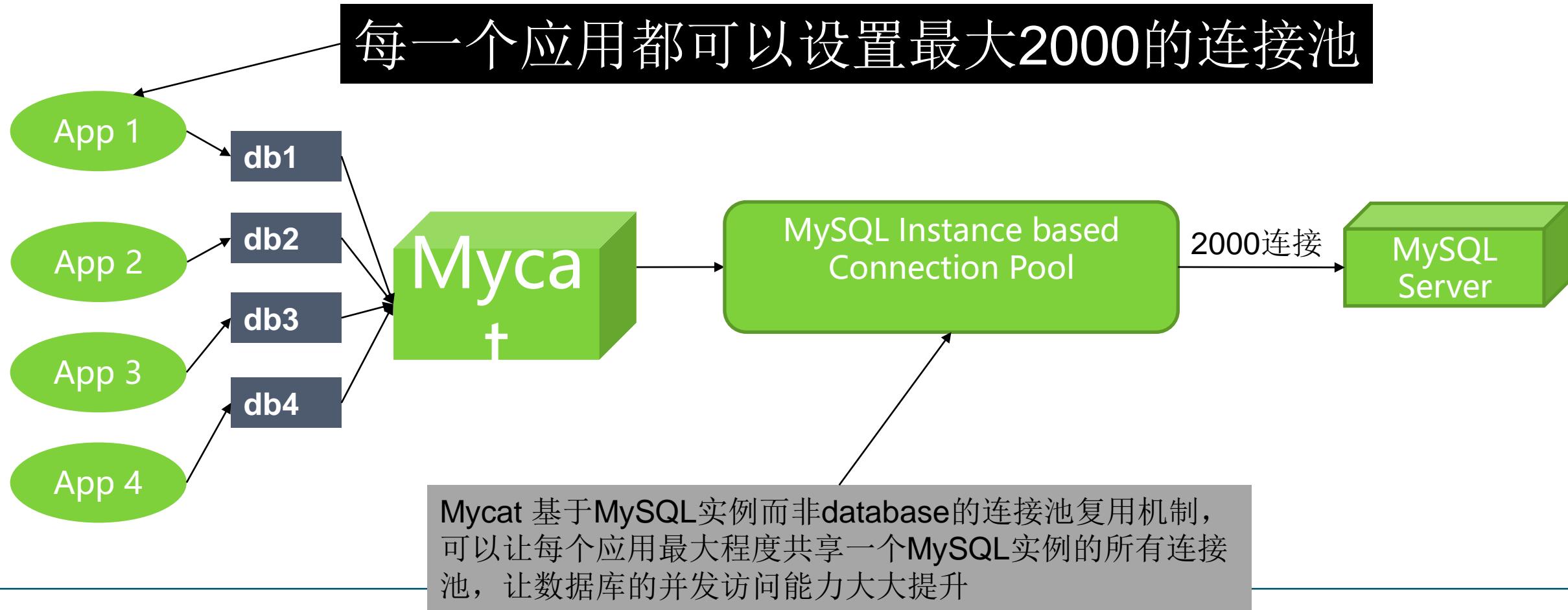
Mycat基本功能介绍

三：100亿大表水平分片，集群并行计算



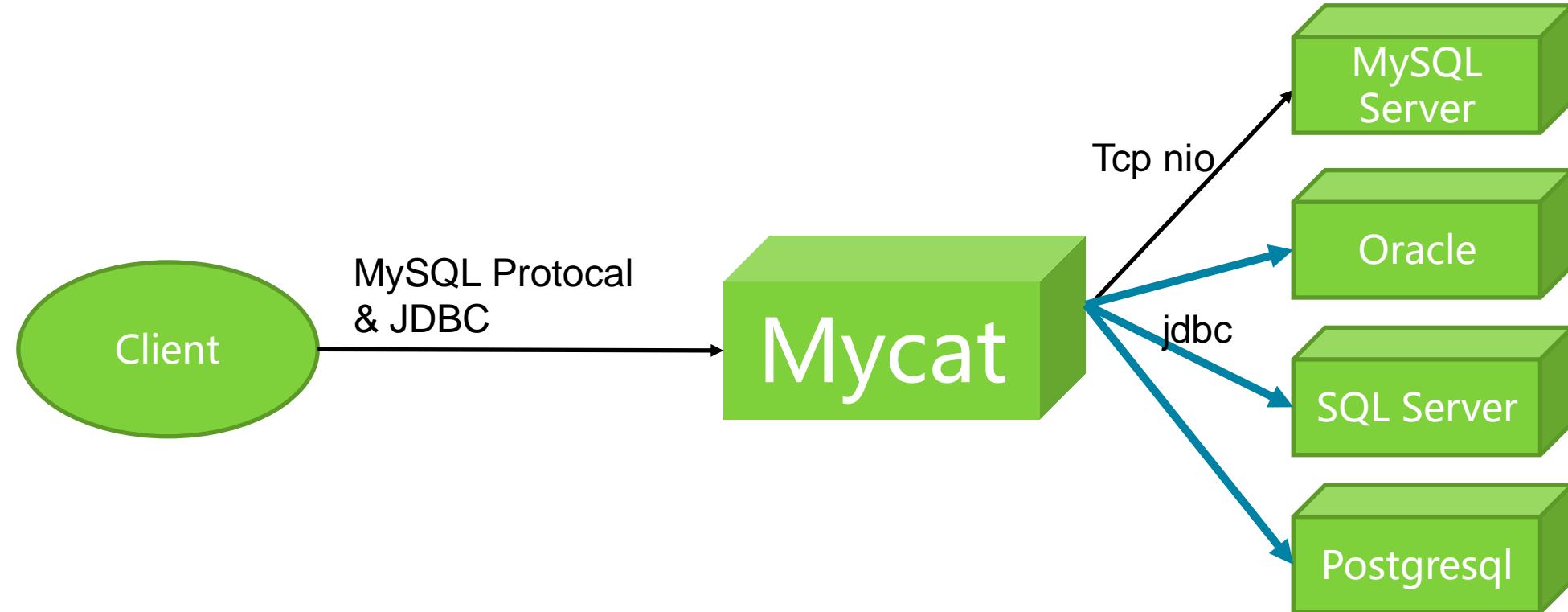
Mycat基本功能介绍

四：数据库路由器：大大提升数据库服务能力



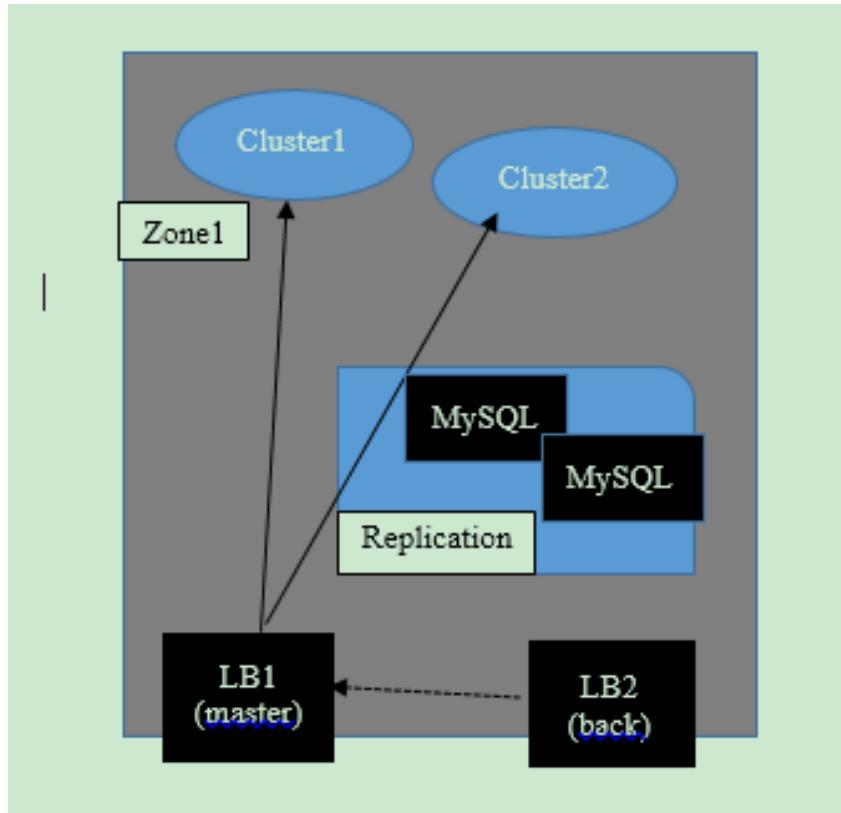
Mycat基本功能介绍

五：数据库路由器：整合多种数据源



Mycat配置入门

1.5的过度阶段，本地XML配置与基于ZK的yaml方式共存



config/zk-create.yaml用来初始化基于ZK的配置信息

config/myid.properties用来定义本Mycat节点的Id以及zk路径配置

loadZk=false

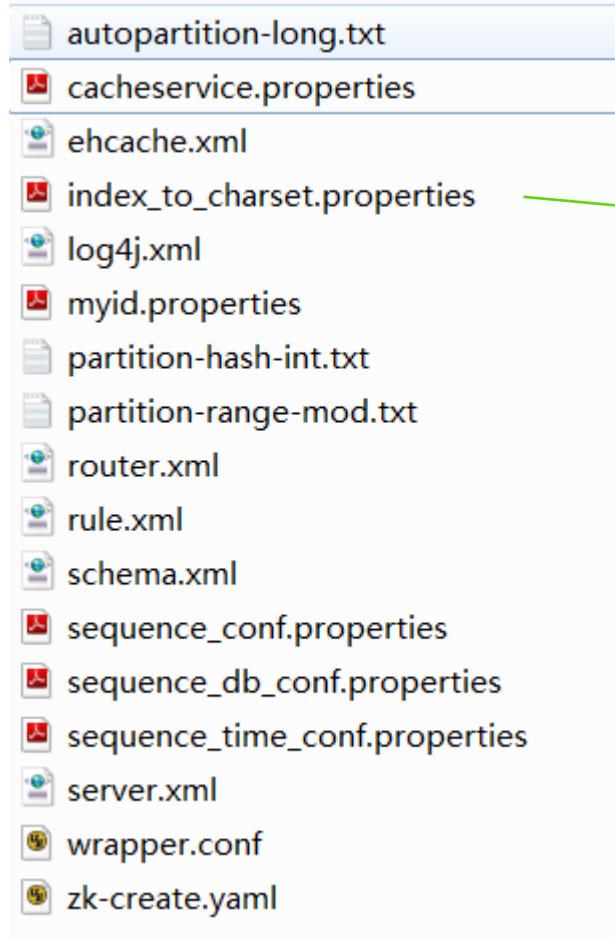
zkURL=127.0.0.1:2181

myid=mycat_fz_01

bin/init_zk_data.bat用来将上述配置信息导入到ZK中

当修改好zk-create.yaml的文件内容，并且用init_zk_data工具导入到ZK中以后，就可以修改myid.properties里的loadZK属性为true，让MyCat从ZK加载

Mycat配置入门



字符集映射关系

Mycat配置入门

```
<mycat:server xmlns:mycat="http://org.opencloudb/">
  <system>
    <property name="defaultSqlParser">druidparser</property>
    <property name="processors">32</property> <property name="processorExecutor">32</property>
    <property name="serverPort">8066</property> <property name="managerPort">9066</property>
    <property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</property>
  </system>
  <user name="test">
    <property name="password">test</property>
    <property name="schemas">TESTDB</property>
  </user>

  <user name="user">
    <property name="password">user</property>
    <property name="schemas">TESTDB</property>
    <property name="readOnly">true</property>
  </user>
</mycat:server>
```

show @@sysparam;

系统参数

```
12/29 22:32:48.263 INFO [main] (MyCatServer.java:195) =====
12/29 22:32:48.263 INFO [main] (MyCatServer.java:196) -MyCat is ready to startup ...
12/29 22:32:48.263 INFO [main] (MyCatServer.java:206) -Startup processors ...,total processors:8,aio thread pool size:16
each process allocated socket buffer pool bytes ,buffer chunk size:4096 buffer pool's capacity(buferPool/bufferChunk) is:8000
12/29 22:32:48.264 INFO [main] (MyCatServer.java:207) -sysconfig params:SystemConfig [processorBufferLocalPercent=100, frontSocketSoRcvbuf=1048576, backSocketSoRcvbuf=4194304, backSocketSoSndbuf=
12/29 22:32:48.289 INFO [main] (MycatServer.java:266) -using nio network handler
12/29 22:32:48.426 INFO [main] (MycatServer.java:284) -$_MyCatManager is started and listening on 9066
12/29 22:32:48.427 INFO [main] (MycatServer.java:288) -$_MyCatServer is started and listening on 8066
12/29 22:32:48.427 TNFO [main] (MycatServer.java:290) =====
```

Mycat配置入门

```

<schema name="TESTDB" checkSQLSchema="false" sqlMaxLimit="100">
  <!-- auto sharding by id (long) -->
  <table name="travelrecord" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" />
  <!-- global table is auto cloned to all defined data nodes ,so can join
      with any table whose sharding node is in the same data node -->
  <table name="company" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />
  <table name="goods" primaryKey="ID" type="global" dataNode="dn1,dn2" />
  <!-- random sharding using mod sharind rule -->
  <table name="hotnews" primaryKey="ID" dataNode="dn1,dn2,dn3"
    rule="mod-long" />
  <!-- <table name="dual" primaryKey="ID" dataNode="dnx,dnoracle2" type="global"
  needAddLimit="false"/> <table name="worker" primaryKey="ID" dataNode="jdbc_dn1,jdbc_dn2,jdbc_dn3"
  rule="mod-long" /> -->
  <table name="employee" primaryKey="ID" dataNode="dn1,dn2"
    rule="sharding-by-intfile" />
  <table name="customer" primaryKey="ID" dataNode="dn1,dn2"
    rule="sharding-by-intfile" />
  <childTable name="orders" primaryKey="ID" joinKey="customer_id"
    parentKey="id">
    <childTable name="order_items" joinKey="order_id"
      parentKey="id" />
  </childTable>
  <childTable name="customer_addr" primaryKey="ID" joinKey="customer_id"
    parentKey="id" />
  </table>
  <!-- <table name="oc_call" primaryKey="ID" dataNode="dn1$0-743" rule="latest-month-calldate"
  /> -->
</schema>
<!-- <dataNode name="dn1$0-743" dataHost="localhost1" database="db$0-743"
  /> -->
<dataNode name="dn1" dataHost="localhost1" database="db1" />
<dataNode name="dn2" dataHost="localhost1" database="db2" />
<dataNode name="dn3" dataHost="localhost1" database="db3" />
<!--<dataNode name="dn4" dataHost="sequoiadb1" database="SAMPLE" />
<dataNode name="jdbc_dn1" dataHost="jdbchost" database="db1" />
<dataNode name="jdbc_dn2" dataHost="jdbchost" database="db2" />
<dataNode name="jdbc_dn3" dataHost="jdbchost" database="db3" /> -->
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">
  <heartbeat>select user()</heartbeat>
  <!-- can have multi write hosts -->
  <writeHost host="hostM1" url="localhost:3306" user="root"
    password="123456">
    <!-- can have multi read hosts -->
  </writeHost>
  <writeHost host="hostS1" url="localhost:3316" user="root"
    password="123456" />
  <!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->
</dataHost>

```

逻辑库

逻辑表

分片

物理数据库

```
<table name="goods" primaryKey="ID" type="global"  
dataNode="dn1,dn2" />  
<!-- random sharding using mod sharind rule -->  
<table name="hotnews" primaryKey="ID" dataNode="dn1,dn2,dn3"  
rule="mod-long" />  
  
<function name="mod-long" class="org.opencloudb.route.function.PartitionByMod">  
  <!-- how many data nodes -->          <property name="count">3</property>  
  </function>
```

分片函数返回的是dataNode的顺序号： 0,1,2

primaryKey的特殊意义

```
<tableRule name="age-mod-long"><rule><columns>age</columns><algorithm>mod-long</algorithm></rule></tableRule>
```

定义一个表，采用上述分片

```
<table name="TESTONLY" primaryKey="ID" dataNode="dn1,dn2,dn3" rule="age-mod-long" />
```

创建表：

```
create table testonly (id bigint not null primary key,age int);
```

```
mysql> explain select * from testonly;
+----+-----+
| DATA_NODE | SQL           |
+----+-----+
| dn1      | SELECT * FROM testonly LIMIT 100 |
| dn2      | SELECT * FROM testonly LIMIT 100 |
| dn3      | SELECT * FROM testonly LIMIT 100 |
+----+-----+
3 rows in set (0.08 sec)
```

```
insert into testonly (id ,age) values (1,18);
insert into testonly (id ,age) values (2,28);
insert into testonly (id ,age) values (3,38);
```

Mycat配置入门

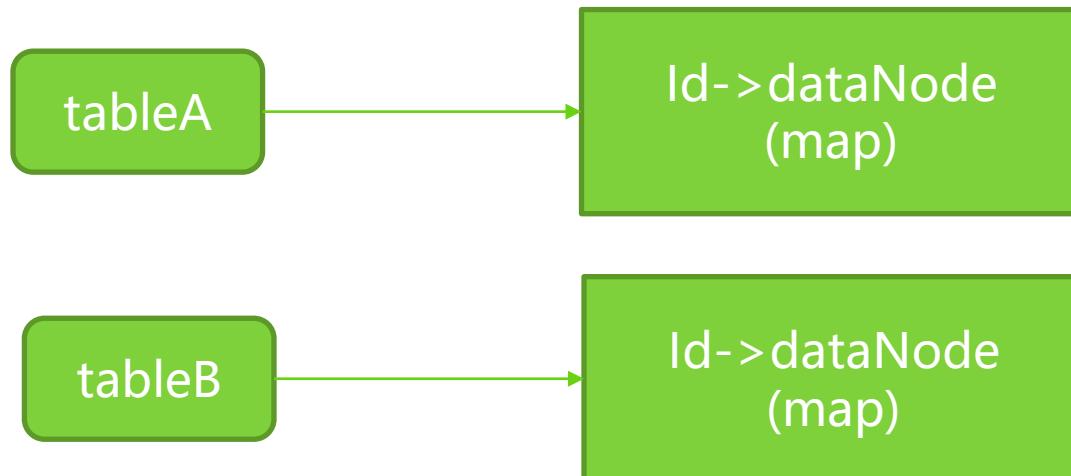
Table以及分片规则入门

primaryKey的特殊意义

```
mysql> explain select * from testonly where id=1;
+-----+-----+
| DATA_NODE | SQL
+-----+-----+
| dn1      | select * from testonly where id=1 |
| dn2      | select * from testonly where id=1 |
| dn3      | select * from testonly where id=1 |
+-----+-----+
3 rows in set (0.05 sec)
```

```
mysql> explain select * from testonly where age=18;
+-----+-----+
| DATA_NODE | SQL
+-----+-----+
| dn1      | SELECT * FROM testonly WHERE age = 18 LIMIT 100 |
+-----+-----+
1 row in set (0.00 sec)

mysql> explain select * from testonly where age=28;
+-----+-----+
| DATA_NODE | SQL
+-----+-----+
| dn2      | SELECT * FROM testonly WHERE age = 28 LIMIT 100 |
+-----+-----+
1 row in set (0.00 sec)
```



primaryKey的特殊意义

```

)R-3-RW] (EnchachePool.java:76) -SQLRouteCache miss cache ,key:TESTDBselect * from testonly where id=2
)R-3-RW] (RouterUtil.java:951) -try to find cache by primary key
)R-3-RW] (DefaultLayedCachePool.java:80) -create child Cache: TESTDB_TESTONLY for layered cache TableID2DataNodeCache, size 10000, expire seconds 18000
)R-3-RW] (CacheManager.java:794) -Attempting to create an existing singleton. Existing singleton returned.
)R-3-RW] (Cache.java:955) -No BootstrapCacheLoaderFactory class specified. Skipping...
)R-3-RW] (Cache.java:929) -CacheWriter factory not configured. Skipping...
)R-3-RW] (MemoryStore.java:153) -Initialized net.sf.ehcache.store.NotifyingMemoryStore for TableID2DataNodeCache.TESTDB_TESTONLY
)R-3-RW] (Cache.java:1165) -Initialised cache: TableID2DataNodeCache.TESTDB_TESTONLY
)R-3-RW] (EnchachePool.java:76) -TableID2DataNodeCache.TESTDB_TESTONLY miss cache ,key:2
)R-3-RW] (NonBlockingSession.java:113) -ServerConnection [id=1, schema=TESTDB, host=0:0:0:0:0:0:1, user=test, txIsolation=3, autocommit=true, schema=TES
)R-3-RW] (MultiNodeQueryHandler.java:82) -execute mutinode query select * from testonly where id=2
)R-3-RW] (PhysicalDBPool.java:452) -select read source hostM1 for dataHost:localhost1
)R-3-RW] (MySQLConnection.java:445) -con need syn ,total syn cmd 2 commands SET names utf8;SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;schema
)R-3-RW] (PhysicalDBPool.java:452) -select read source hostM1 for dataHost:localhost1
)R-3-RW] (MySQLConnection.java:445) -con need syn ,total syn cmd 2 commands SET names utf8;SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;schema
)R-4-RW] (MultiNodeQueryHandler.java:171) -received ok response ,executeResponse:false from MySQLConnection [id=4, lastTime=1451908067248, user=root, sch
)R-6-RW] (MultiNodeQueryHandler.java:171) -received ok response ,executeResponse:false from MySQLConnection [id=6, lastTime=1451908067248, user=root, sch
)R-3-RW] (PhysicalDBPool.java:452) -select read source hostM1 for dataHost:localhost1
)R-6-RW] (MultiNodeQueryHandler.java:171) -received ok response ,executeResponse:false from MySQLConnection [id=6, lastTime=1451908067248, user=root, sch
)R-4-RW] (MultiNodeQueryHandler.java:171) -received ok response ,executeResponse:false from MySQLConnection [id=4, lastTime=1451908067248, user=root, sch
)R-3-RW] (MySQLConnection.java:445) -con need syn ,total syn cmd 2 commands SET names utf8;SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;schema
)R-4-RW] (MultiNodeQueryHandler.java:241) -on row end reseponse MySQLConnection [id=4, lastTime=1451908067248, user=root, schema=db1, old shema=db1, borr
)R-4-RW] (NonBlockingSession.java:229) -release connection MySQLConnection [id=4, lastTime=1451908067248, user=root, schema=db1, old shema=db1, borrowed=
)R-2-RW] (MultiNodeQueryHandler.java:171) -received ok response ,executeResponse:false from MySQLConnection [id=10, lastTime=1451908067248, user=root, sc
)R-4-RW] (PhysicalDatasource.java:403) -release channel MySQLConnection [id=4, lastTime=1451908067248, user=root, schema=db1, old shema=db1, borrowed=true
)R-2-RW] (MultiNodeQueryHandler.java:171) -received ok response ,executeResponse:false from MySQLConnection [id=10, lastTime=1451908067248, user=root, sc
)R-6-RW] (EnchachePool.java:59) -TableID2DataNodeCache.TESTDB_TESTONLY add cache ,key:2 value:dn2

```

Mycat配置入门

Table以及分片规则入门

primaryKey的特殊意义

```

2 01/04 19:53:12.712 DEBUG [$_NIOREACTOR-3-RW] (EnchachePool.java:76) -SQLRouteCache miss cache ,key:TESTDBselect * from testonly where id=2
3 01/04 19:53:12.713 DEBUG [$_NIOREACTOR-3-RW] (RouterUtil.java:951) -try to find cache by primary key
4 01/04 19:53:12.714 DEBUG [$_NIOREACTOR-3-RW] (EnchachePool.java:70) -TableID2DataNodeCache.TESTDB_TESTONLY hit cache ,key:2
5 01/04 19:53:12.715 DEBUG [$_NIOREACTOR-3-RW] (NonBlockingSession.java:113) -ServerConnection [id=1, schema=TESTDB, host=0:0:0:0:0:1, user=test,txIsola
6   1 -> dn2{select * from testonly where id=2}
7 } rns

```

```

Current database: TESTDB
+----+-----+
| id | age |
+----+-----+
| 2 | 28 |
+----+-----+
1 row in set (0.12 sec)

mysql> select * from testonly where id=2;
+----+-----+
| id | age |
+----+-----+
| 2 | 28 |
+----+-----+
1 row in set (0.01 sec)

mysql> select * from testonly where id=2;
+----+-----+
| id | age |
+----+-----+
| 2 | 28 |
+----+-----+
1 row in set (0.01 sec)

mysql> select * from testonly where id=2;
+----+-----+
| id | age |
+----+-----+
| 2 | 28 |
+----+-----+
1 row in set (0.01 sec)

```

CACHE	MAX	CUR	ACCESS	HIT	PUT	LAST_ACCESS	LAST_PUT
ER_SQL2PARENTID	1000	0	0	0	0	0	0
SQLRouteCache	10000	0	5	0	0	1451908567857	0
TableID2DataNodeCache.TESTDB_ORDERS	50000	0	0	0	0	0	0
TableID2DataNodeCache.TESTDB_TESTONLY	10000	1	5	4	1	1451908567859	1451908067254

4 rows in set (0.01 sec)

cacheservice.properties

```

1 #used for mycat cache service conf
2 factory.encache=org.opencloudb.cache.impl.EnchachePooFactory
3 #key is pool name ,value is type,max size, expire seconds
4 pool.SQLRouteCache=encache,10000,1800
5 pool.ER_SQL2PARENTID=encache,1000,1800
6 layedpool.TableID2DataNodeCache=encache,10000,18000
7 layedpool.TableID2DataNodeCache.TESTDB_ORDERS=50000,18000

```

Mycat配置入门

index_to_charset.properties 做了MySQL字符集的映射关系

sequence_db_conf.properties，存放了全局序列号的配置信息

wrapper.conf是JVM的参数，包括堆大小问题

-XX:MaxDirectMemorySize=2G

wrapper.java.additional.6=-Dcom.sun.management.jmxremote

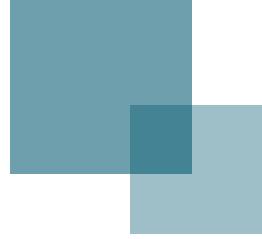
wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=1984

wrapper.java.additional.8=-Dcom.sun.management.jmxremote.authenticate=false

wrapper.java.additional.9=-Dcom.sun.management.jmxremote.ssl=false

wrapper.java.additional.10=-Xmx4G

wrapper.java.additional.11=-Xms1G



Thanks

FAQ时间