



# MySQL 趣味杂谈

---





分享的是经验，推广的是思路

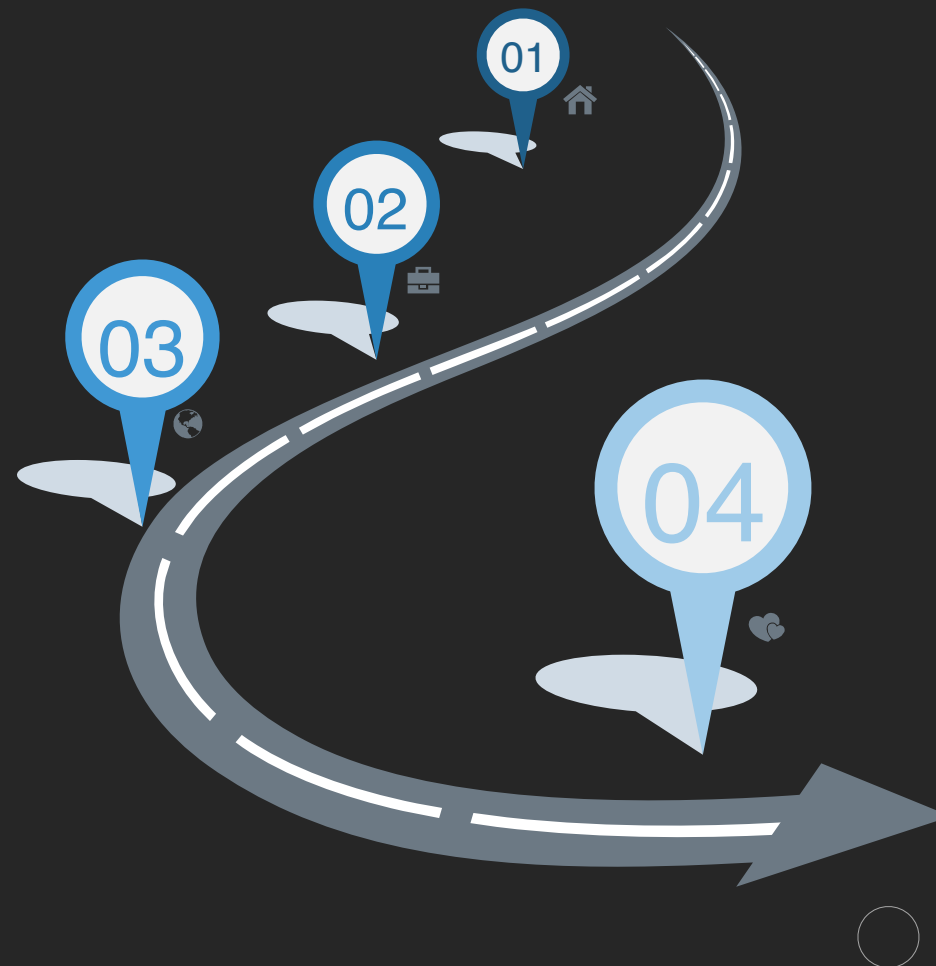
张充

[sleepduck@sina.com](mailto:sleepduck@sina.com)



# MySQL 聊点什么

-  高可用：SQL Server AlwaysOn
-  马蜂窝中间件：DB Man
-  表迁移的一些坑
-  我们还能做些什么事情





# 我对高可用的理解



# 日志对比

MySQL 日志

&

SQL Server 日志

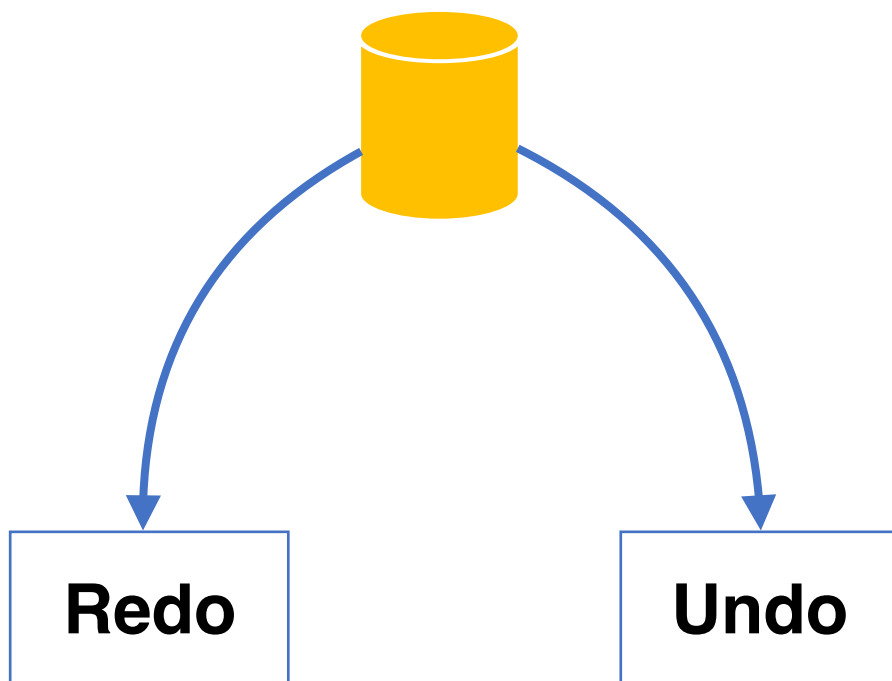
**MySQL:** Redo Log、Undo Log、Binlog

**SQL Server:** LDF日志,集成了Redo、Undo于一身

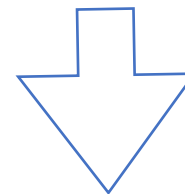


# 我对高可用的理解

Recovering



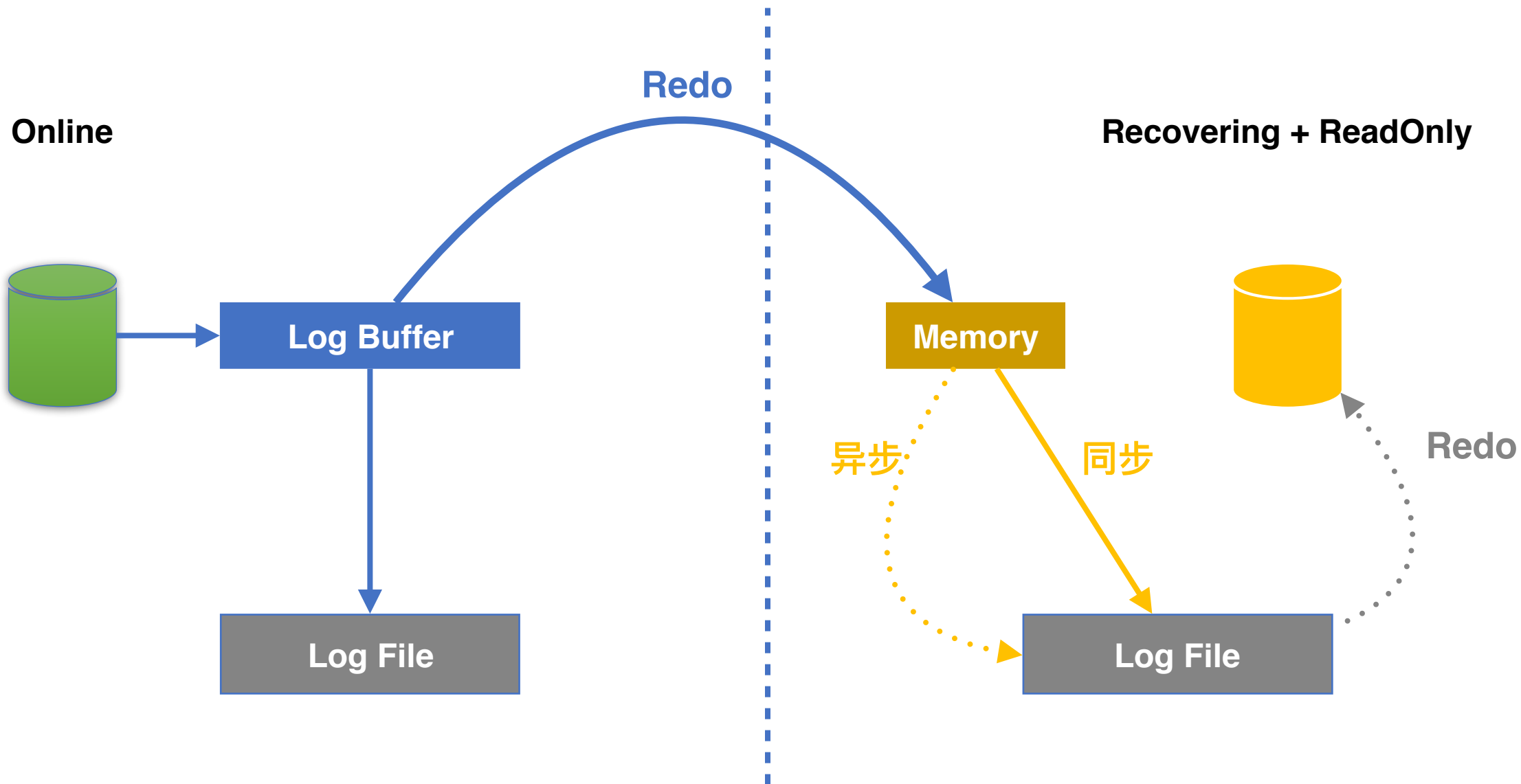
Binlog



执行Event

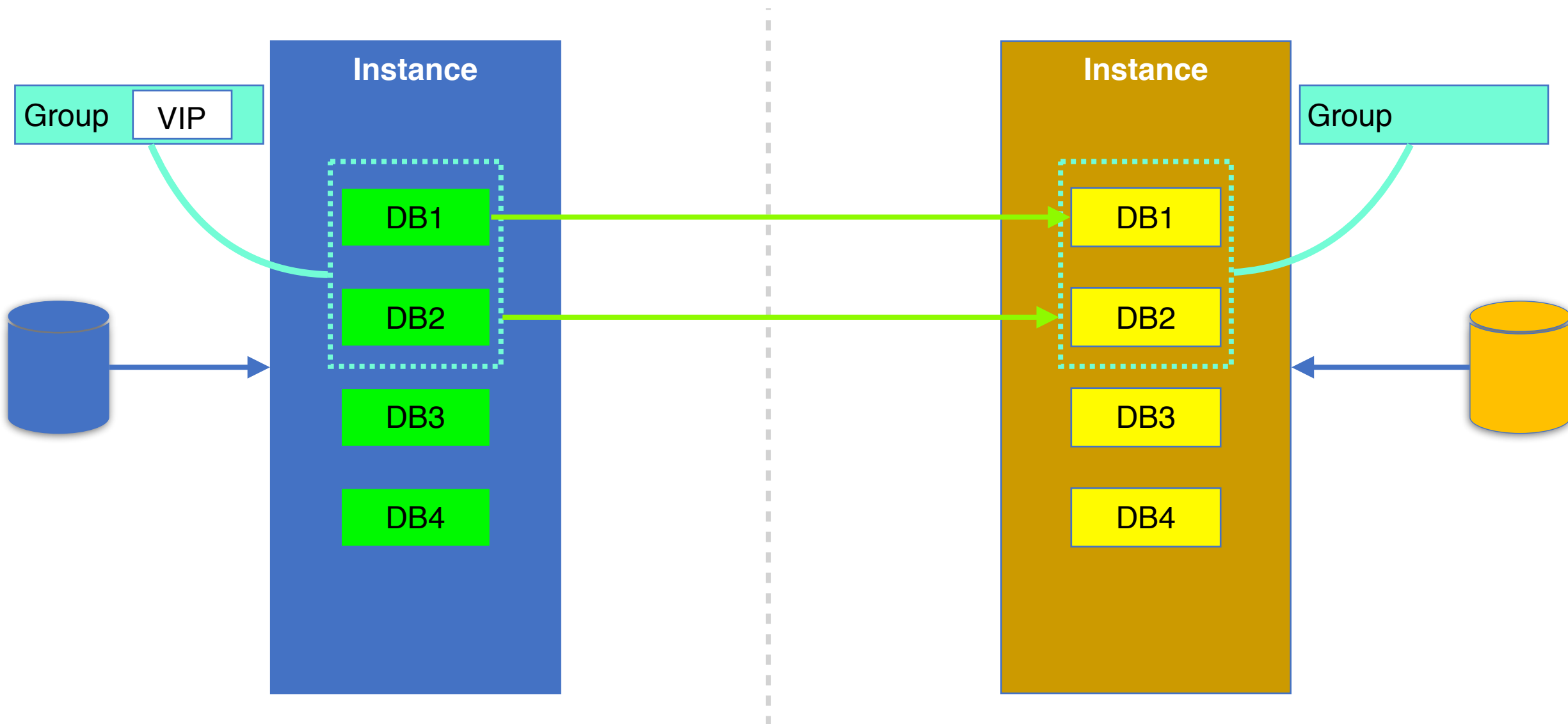


# 高可用传输-SQL Server AlwaysOn



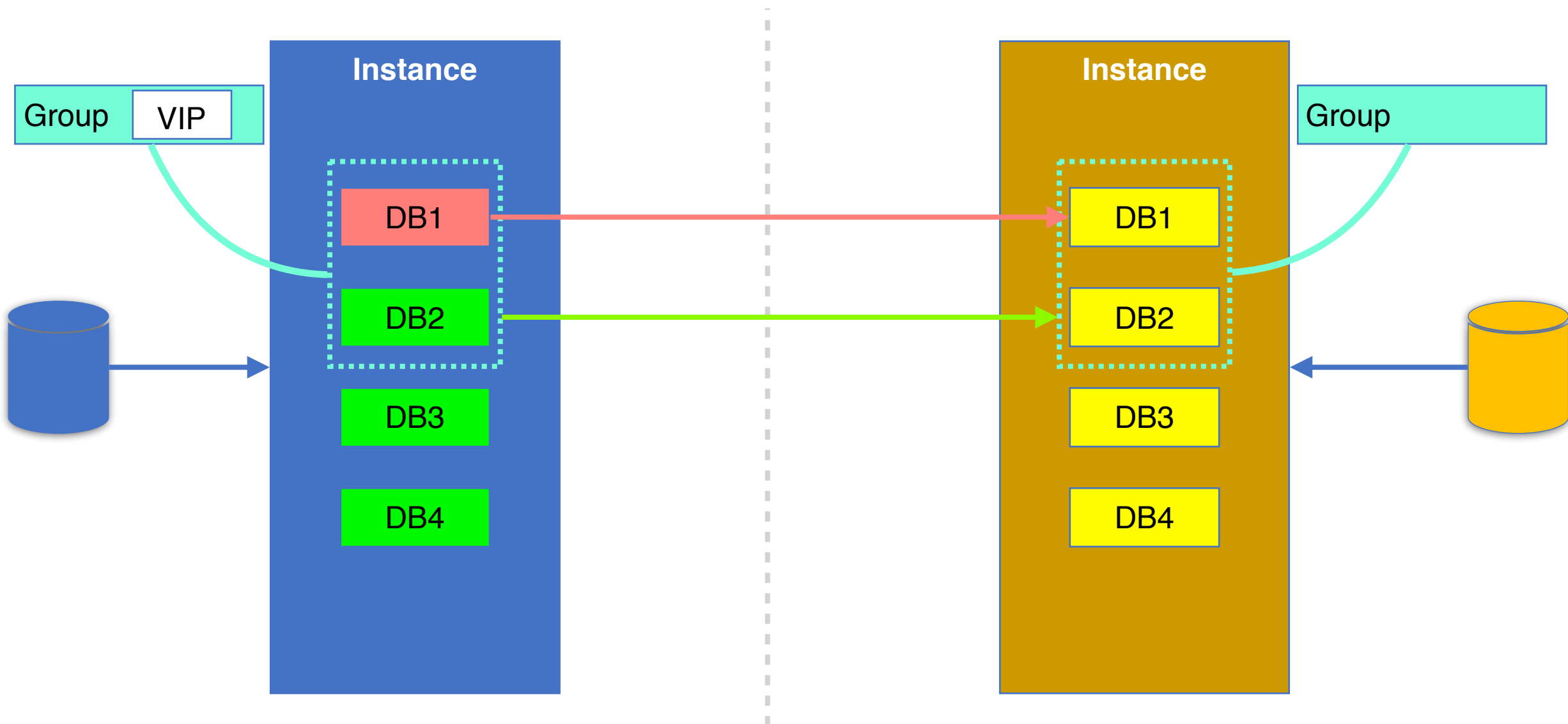


# 高可用传输-SQL Server AlwaysOn





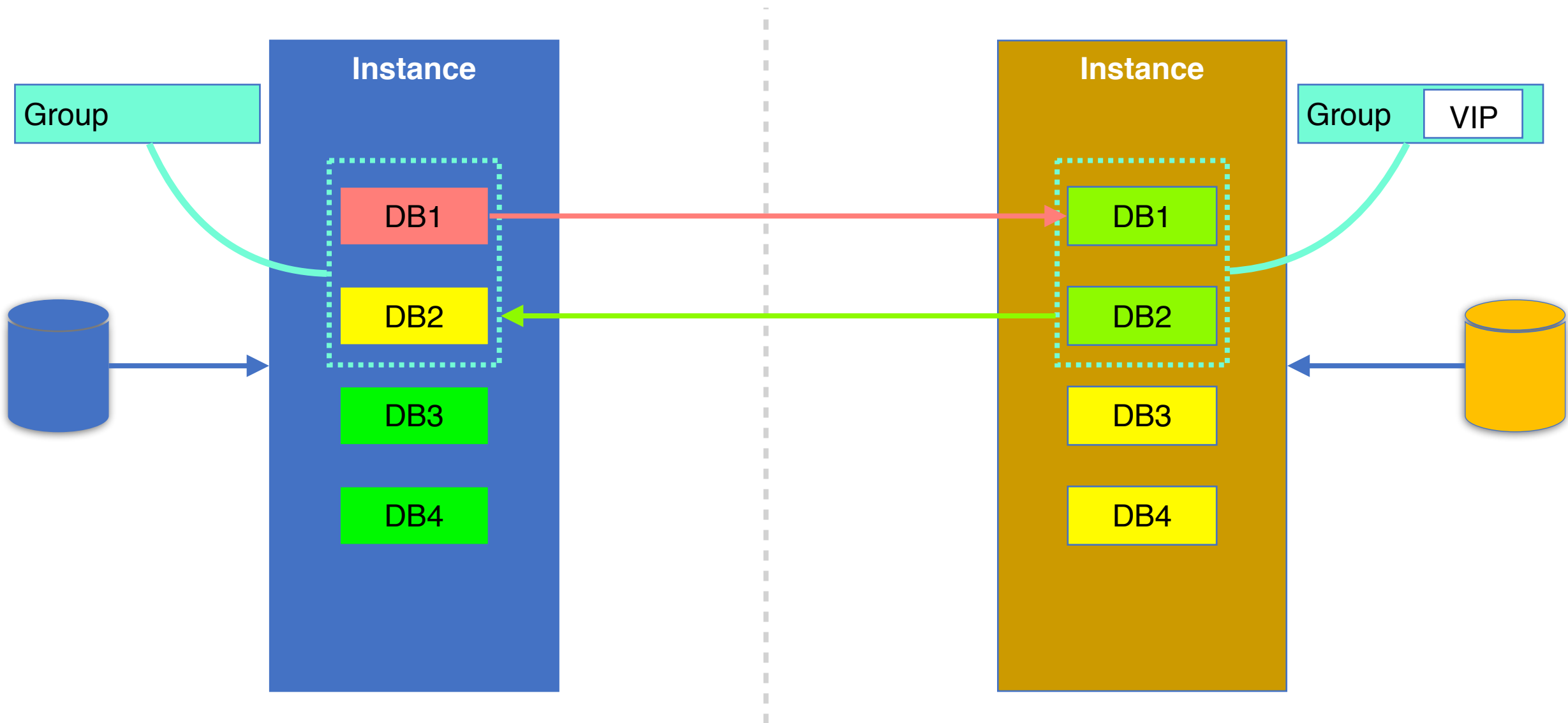
# 高可用传输-SQL Server AlwaysOn







# 高可用传输-SQL Server AlwaysOn





# 高可用的使用姿势

写库: Table 100张

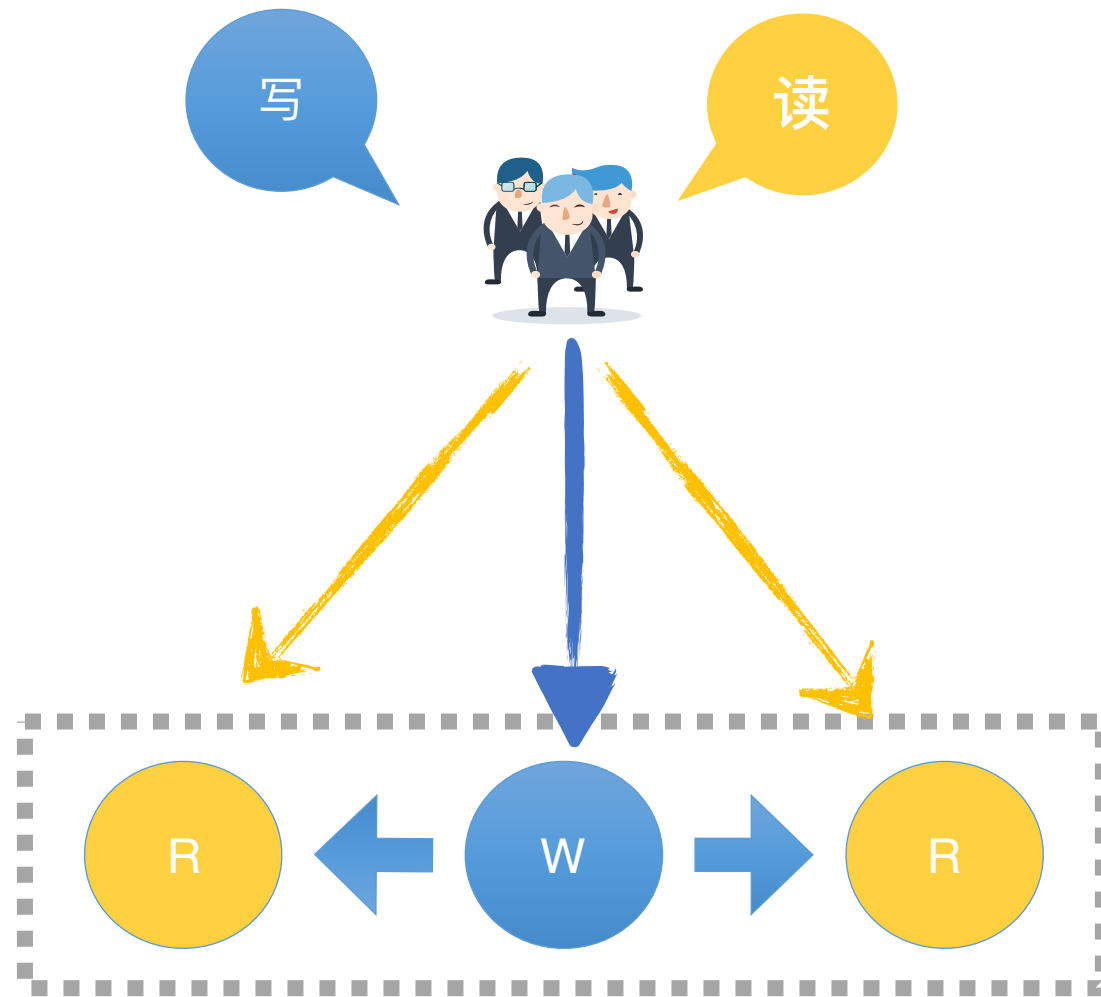
每张Table的索引: 3个

读库: 用到了几张Table?

用到了哪些列?

假如读库, 我们确定了用到了哪些对象

没用到的对象, 就成为了节点之间数据同步的累赘。



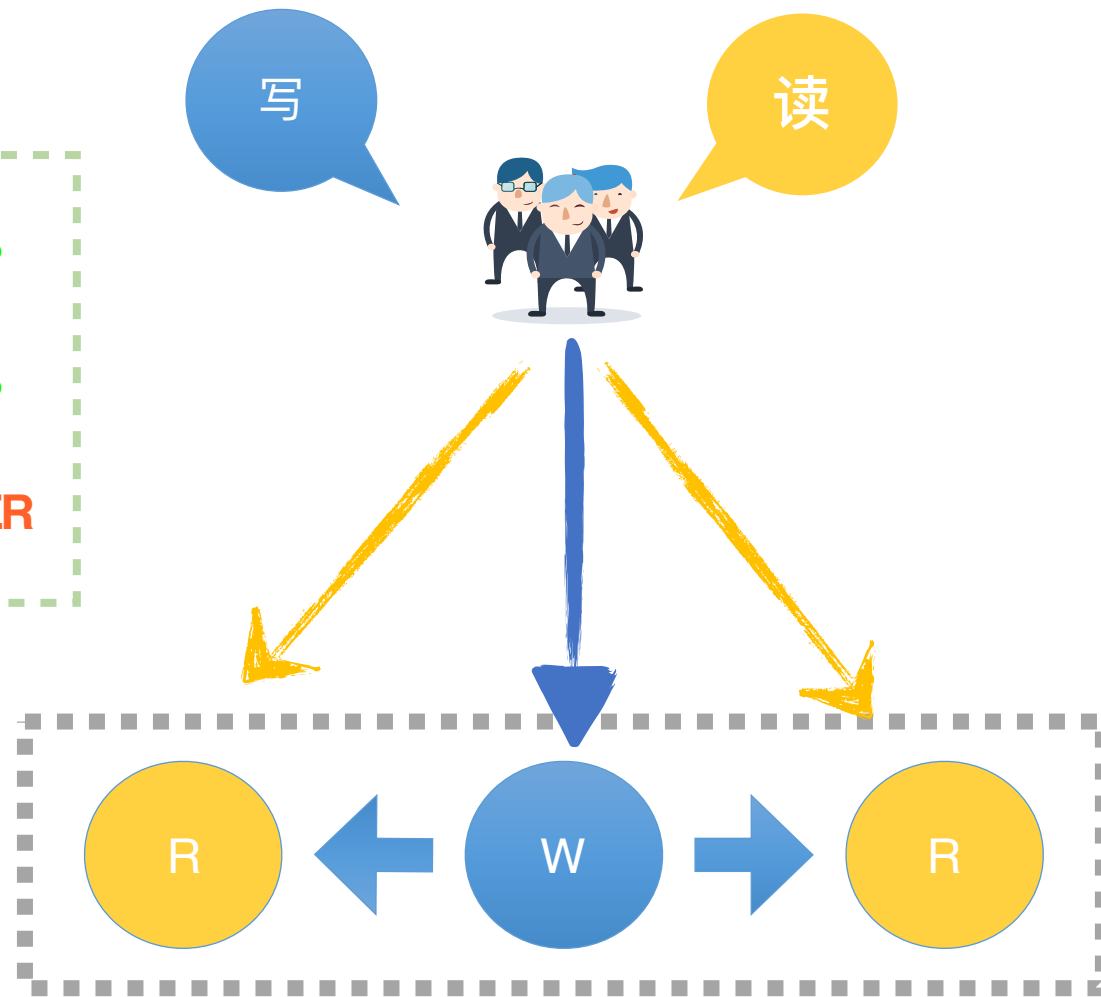


# 高可用的使用姿势

写库负载: 60%	从库负载: 0%	切换后负载: = 60%
写库负载: 60%	从库负载: 10%	切换后负载: > 70%
写库负载: 60%	从库负载: 40%	切换后负载: = OVER

高可用方案:

为了让数据库可以在故障时, 提供原有的服务能力, 为业务正常访问。





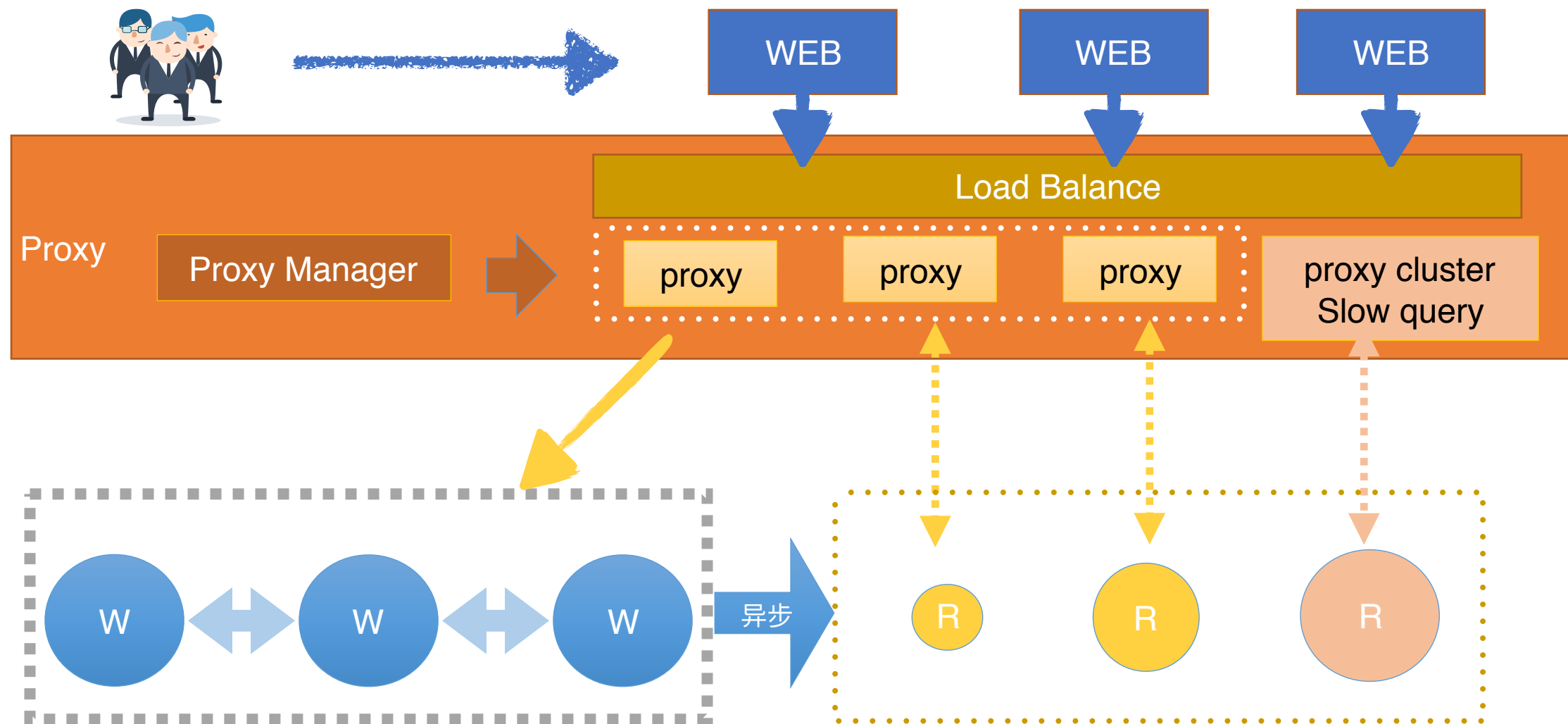
# 高可用的使用姿势

高可用就是高可用，读写分离就是读写

两者结合，妥协一些，承担一些

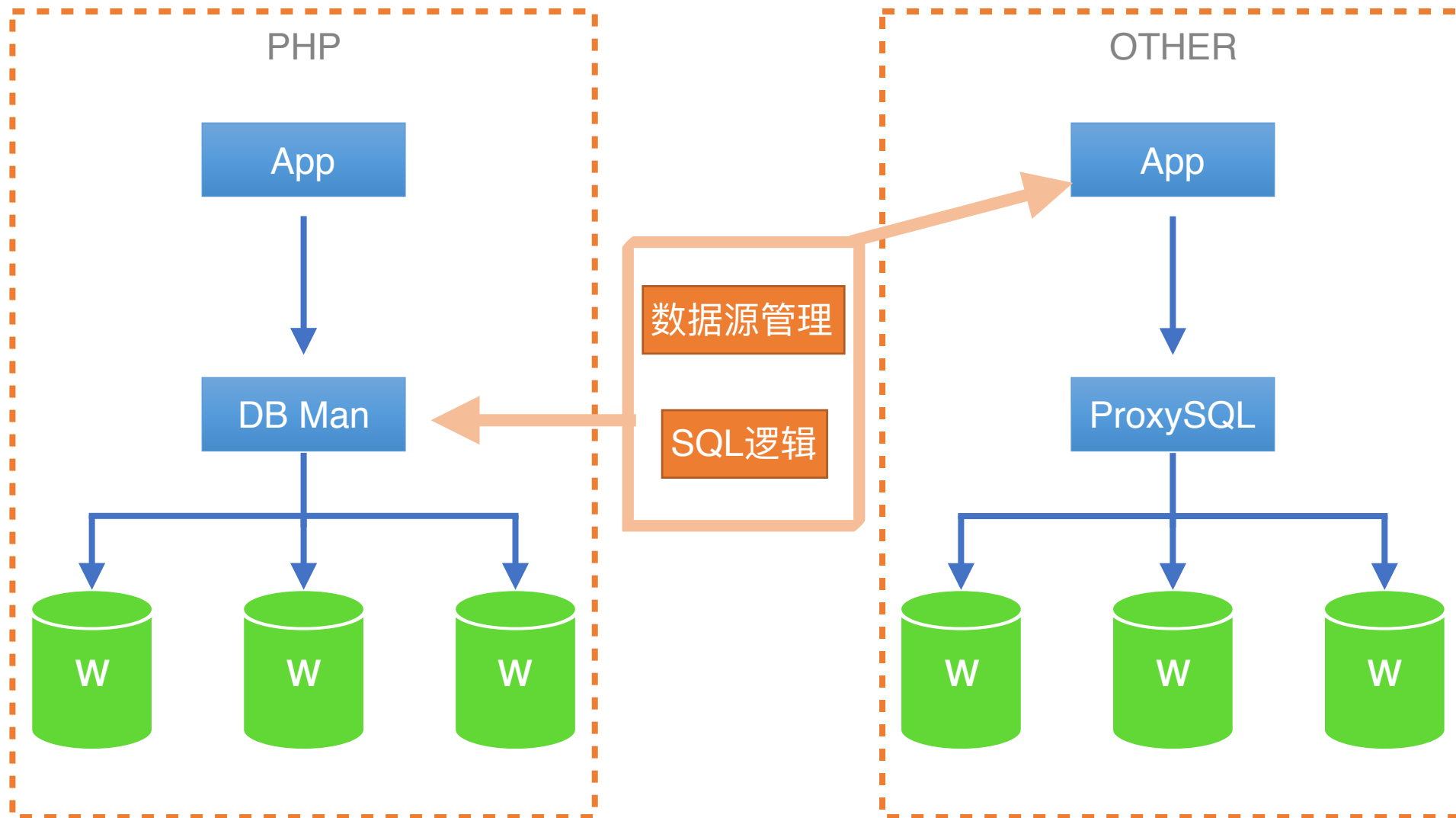


# 高可用的使用方式





# MySQL 马蜂窝的访问方式

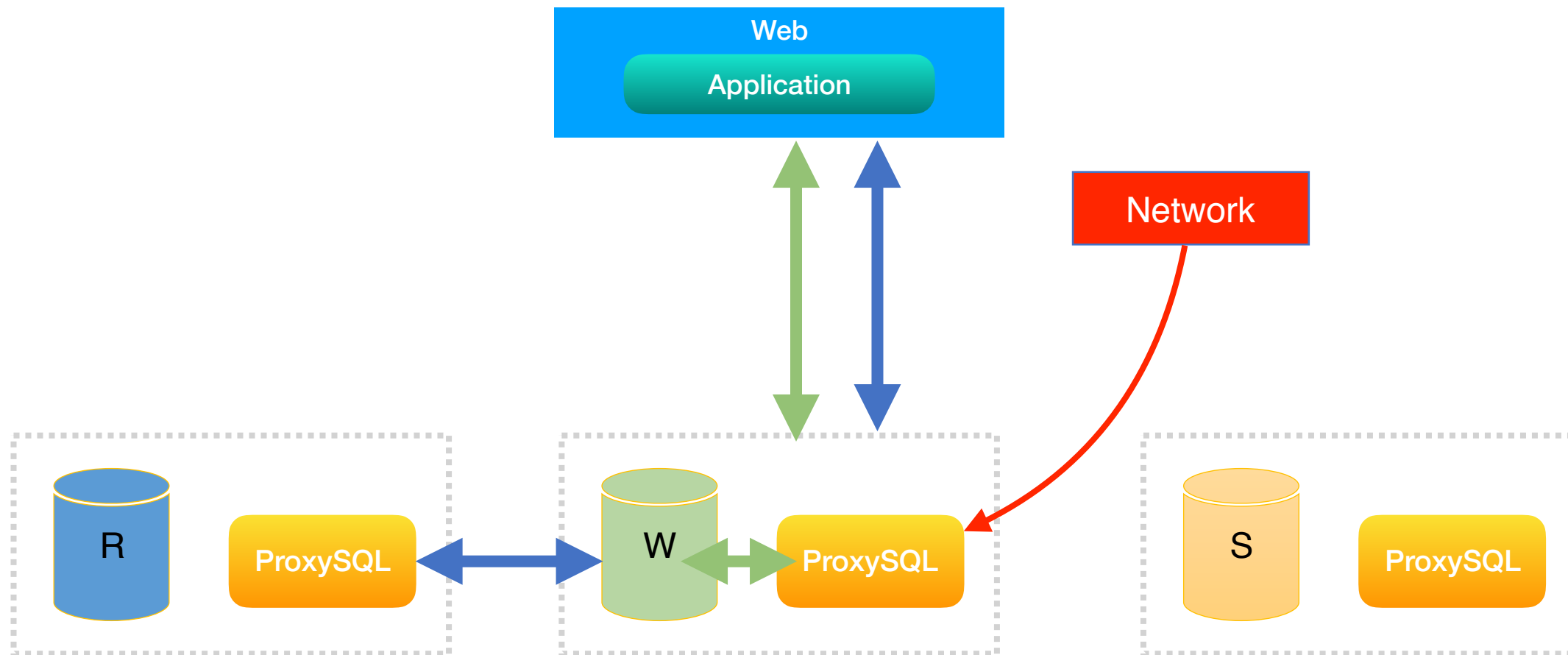




## 直连DB的中间件 - ProxySQL



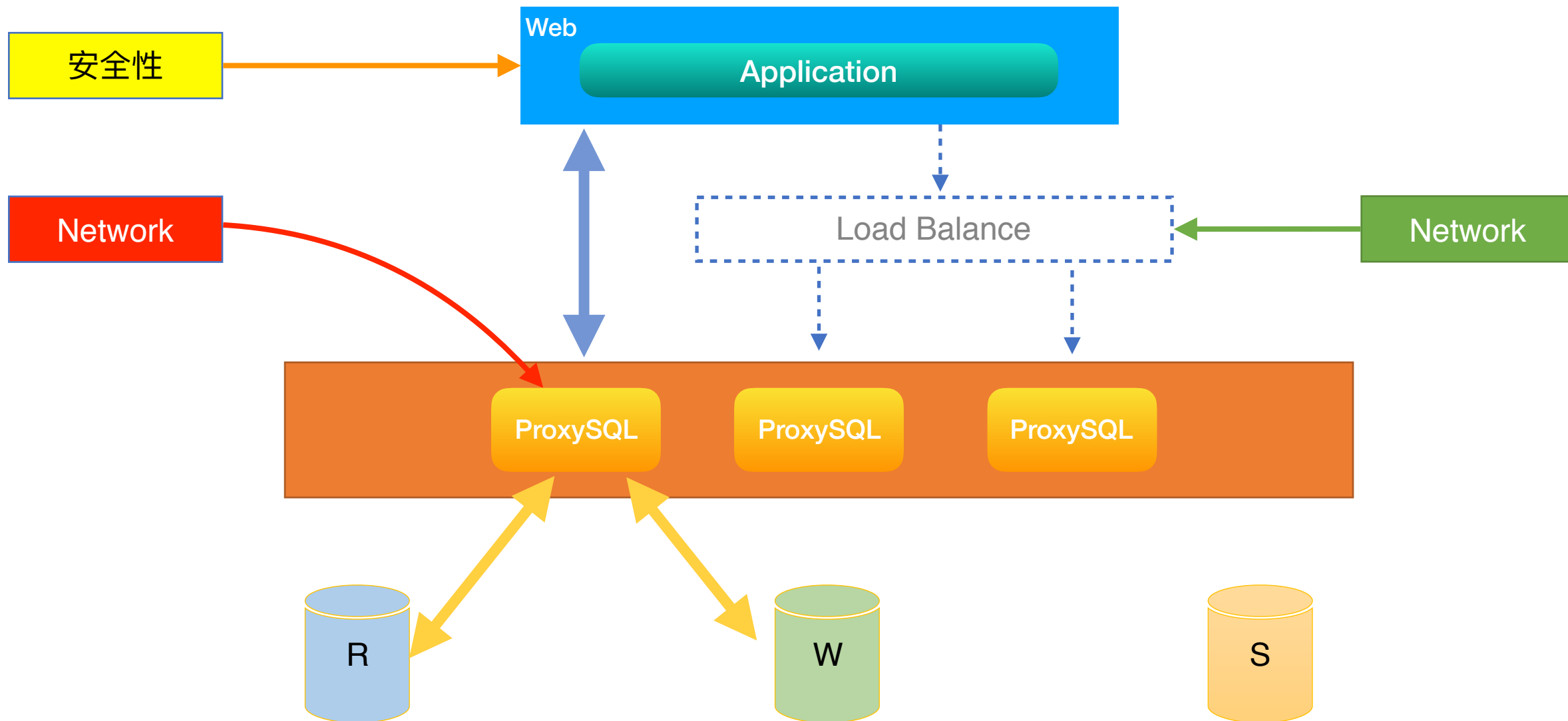
# ProxySQL 与 MGR & PXC





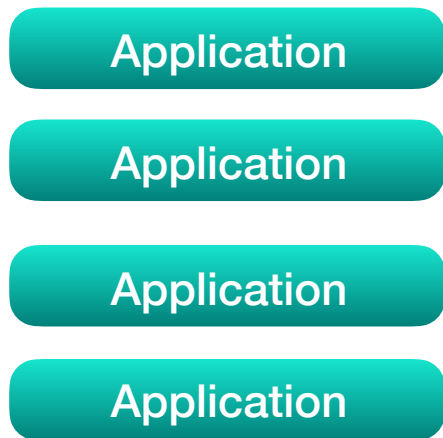


# ProxySQL 与 MGR & PXC





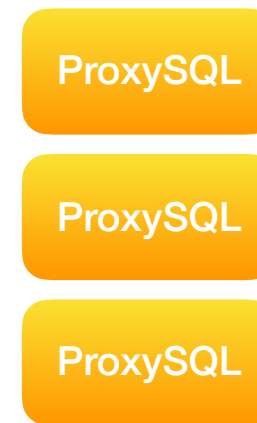
# ProxySQL



?

=

?



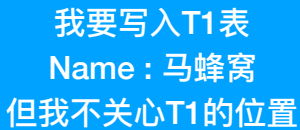
## 如何管理?



DB Man 采用了折中方案



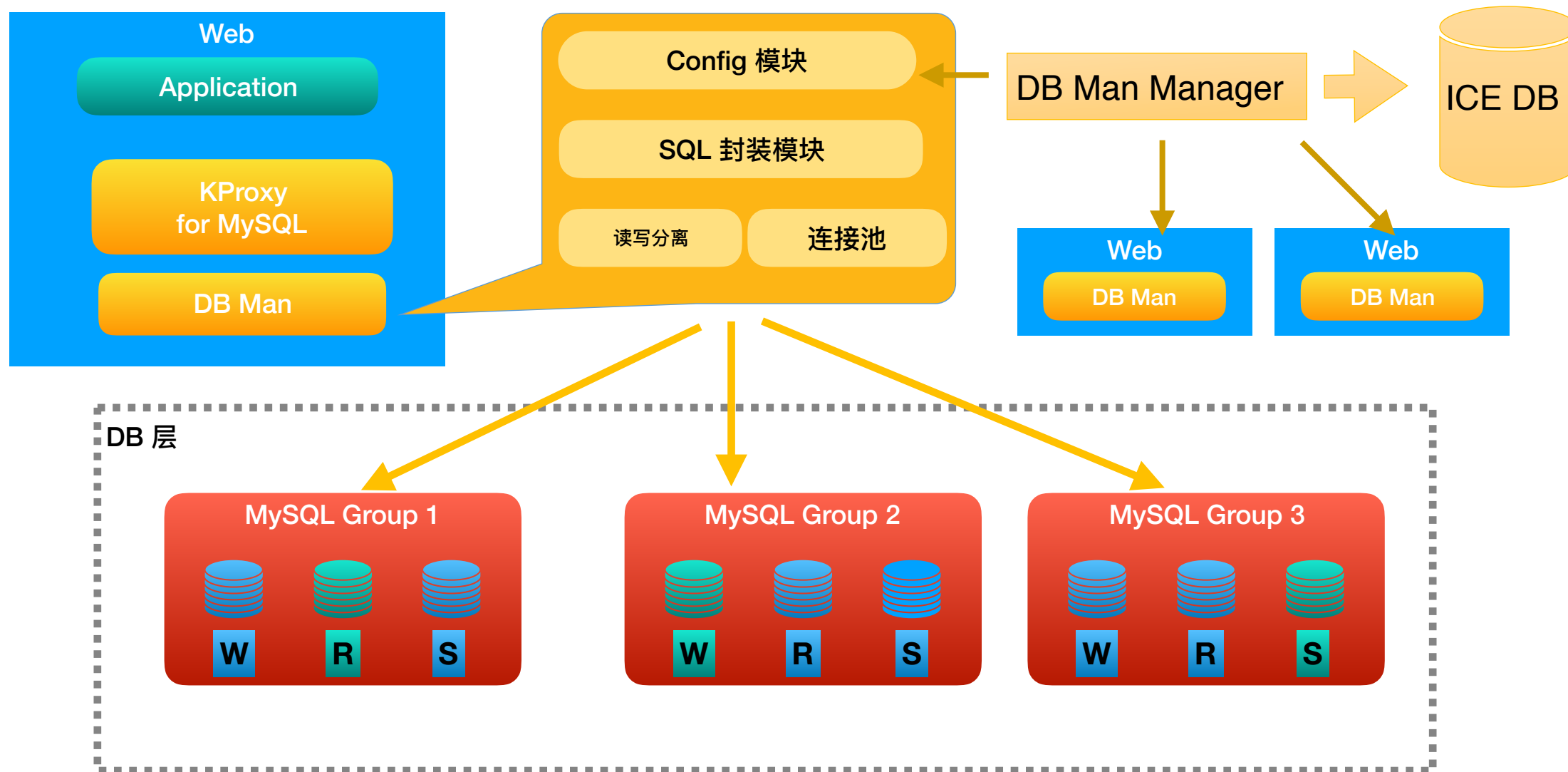
马蜂窝中间件 - DB Man



我来封装语句：  
insert into T1 (name) values('马蜂窝')  
我知道T1的位置



# DB Man

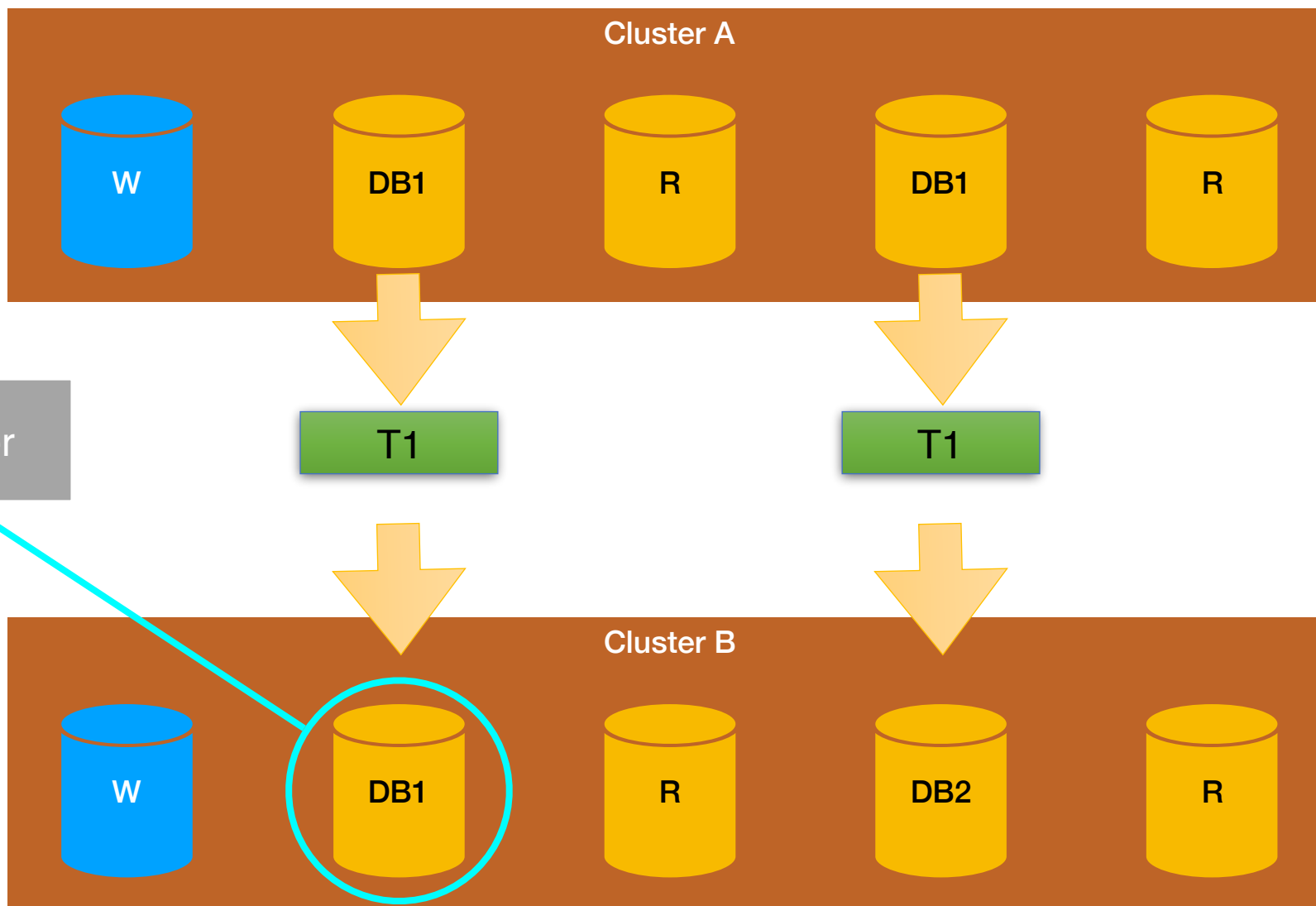




表迁移



# 表迁移



**CHANGE REPLICATION FILTER filter[, filter][, ...]**

**Filter:**

- REPLICATE\_DO\_DB = (db\_list)**
- I REPLICATE\_IGNORE\_DB = (db\_list)**
- I REPLICATE\_DO\_TABLE = (tbl\_list)**
- I REPLICATE\_IGNORE\_TABLE = (tbl\_list)**
- I REPLICATE\_WILD\_DO\_TABLE = (wild\_tbl\_list)**
- I REPLICATE\_WILD\_IGNORE\_TABLE = (wild\_tbl\_list)**
- I REPLICATE\_REWRITE\_DB = (db\_pair\_list)**





# 表迁移

**1.Replication Filter 是作用在 SQL Thread上吗?**

**2.MGR 会和Replication Filter产生效果吗?**

**3.PXC 会和Replication Filter产生效果吗?**



# MGR - Channel

**1.Group\_Rplication\_Reovery**

**2.Group\_Rplication\_Applier**

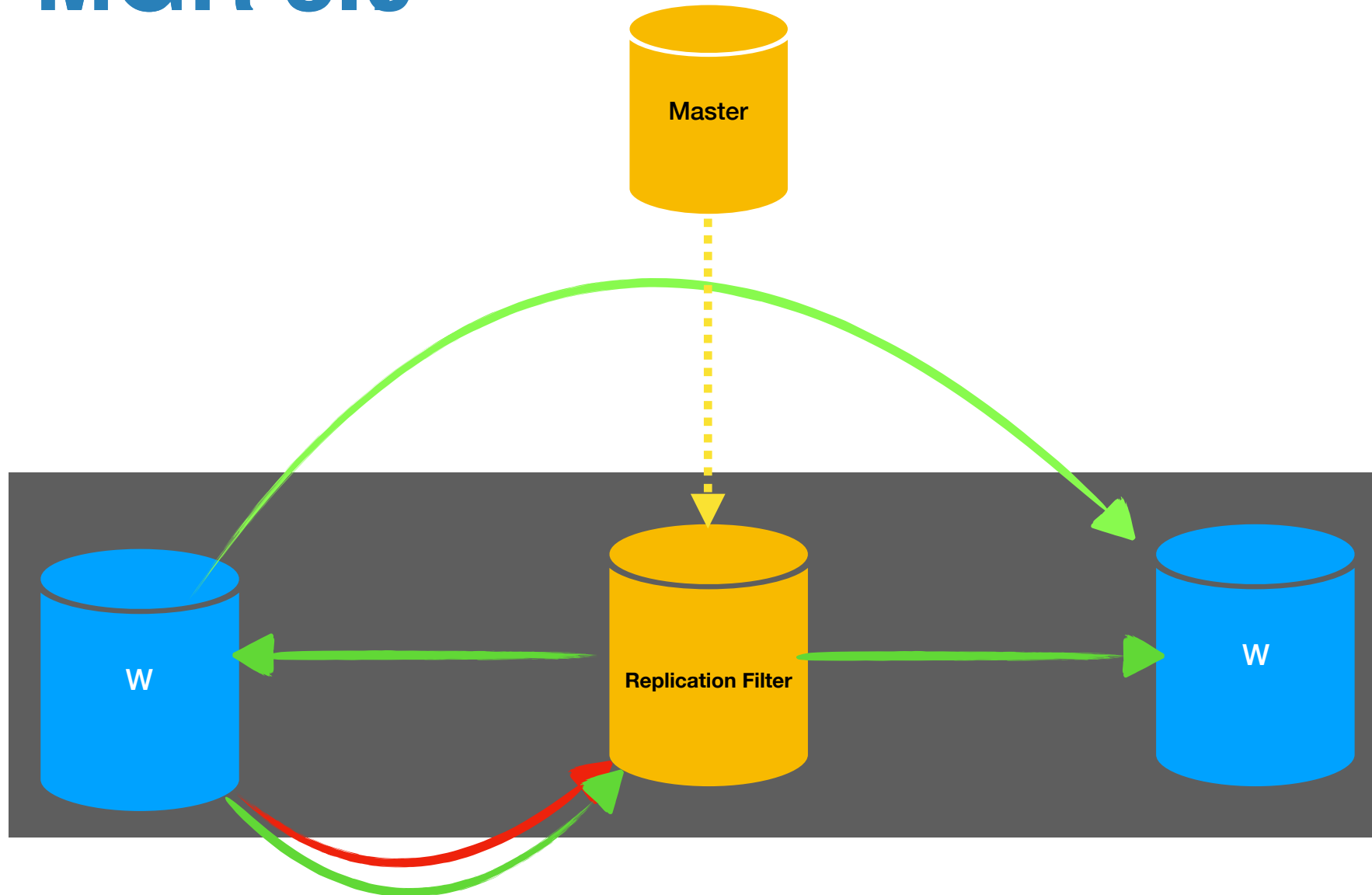
# MGR

## change replication filter replicate\_do\_table=(test.zc1);

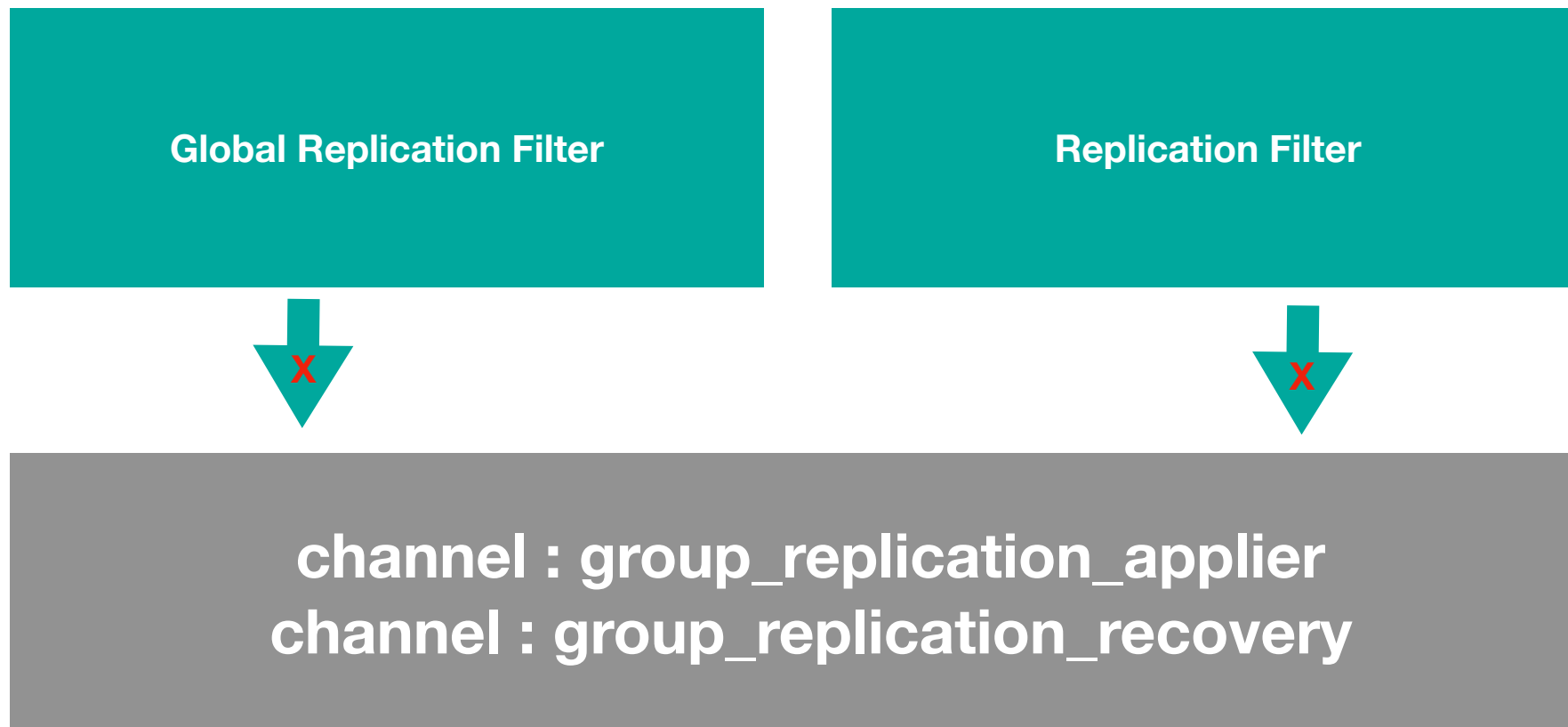
```
zhangchong@172.18.12.52 18:26:26 [(none)]> show slave status for channel 'group_replication_recovery'\G
```

```
***** 1. row *****
Slave_IO_State:
  Master_Host: <NULL>
  Master_User: rpl_user
  Master_Port: 0
  Connect_Retry: 60
  Master_Log_File:
  Read_Master_Log_Pos: 4
  Relay_Log_File: 024523309-relay-bin-group_replication_recovery.000001
  Relay_Log_Pos: 4
  Relay_Master_Log_File:
  Slave_IO_Running: No
  Slave_SQL_Running: No
  Replicate_Do_DB:
  Replicate_Ignore_DB:
  Replicate_Do_Table: test.zc1
  Replicate_Ignore_Table:
  Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
  Last_Errno: 0
  Last_Error:
  Skip_Counter: 0
  Exec_Master_Log_Pos: 0
  Relay_Log_Space: 499
  Until_Condition: None
  Until_Log_File:
  Until_Log_Pos: 0
  Master_SSL_Allowed: No
  Master_SSL_CA_File:
  Master_SSL_CA_Path:
  Master_SSL_Cert:
  Master_SSL_Cipher:
  Master_SSL_Key:
  Seconds_Behind_Master: NULL
```

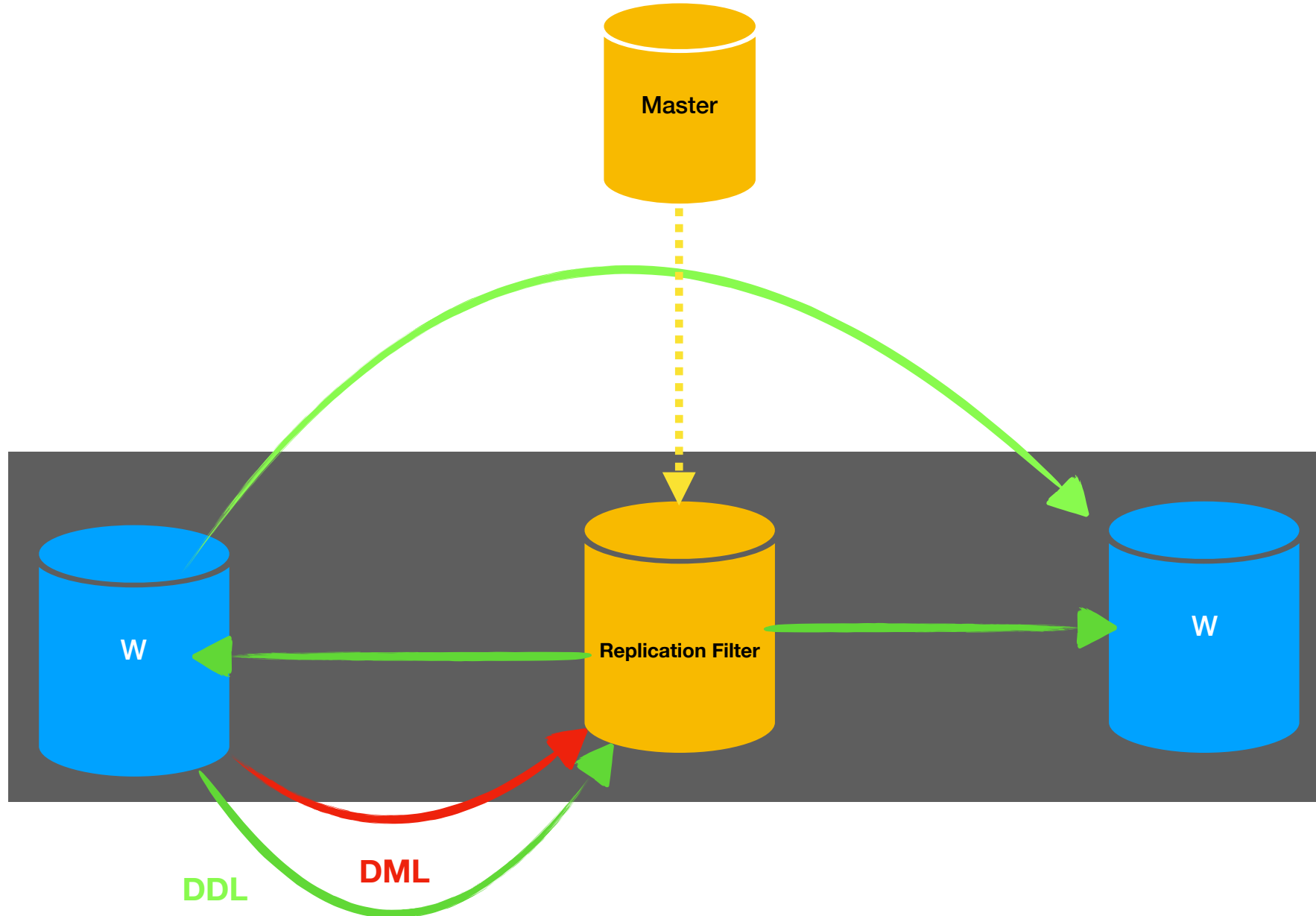
# MGR 8.0



# MGR 8.0



# PXC 5.7

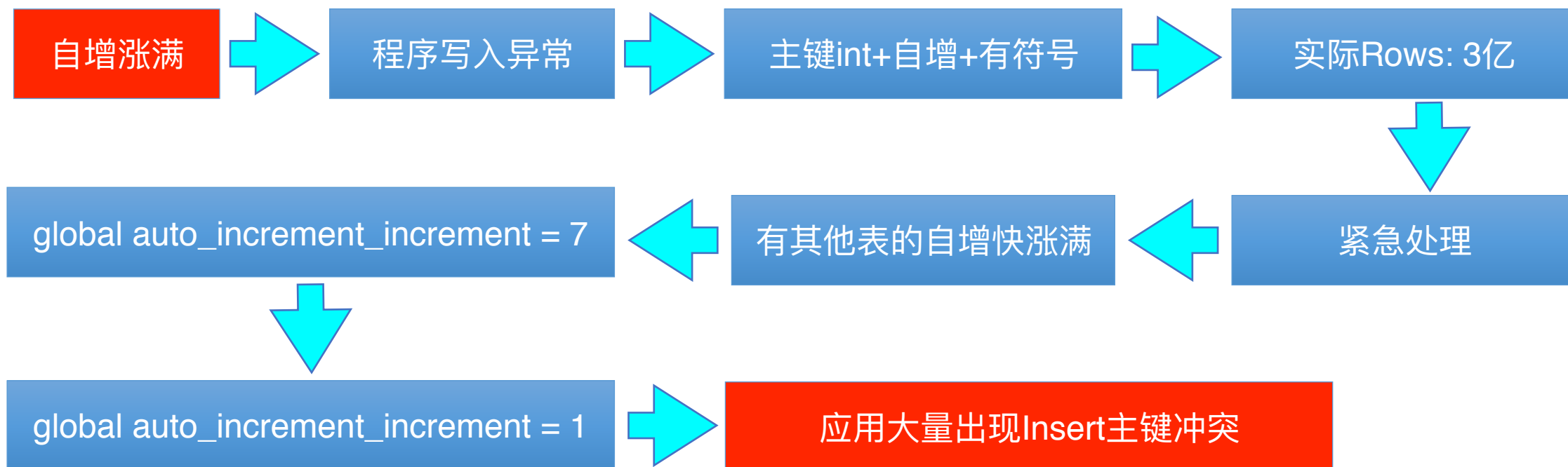




一次简单而又悲催的事故



# MGR 一次简单的故障



长连接中，获取最新auto\_increment\_increment会被影响。

新连接正常。



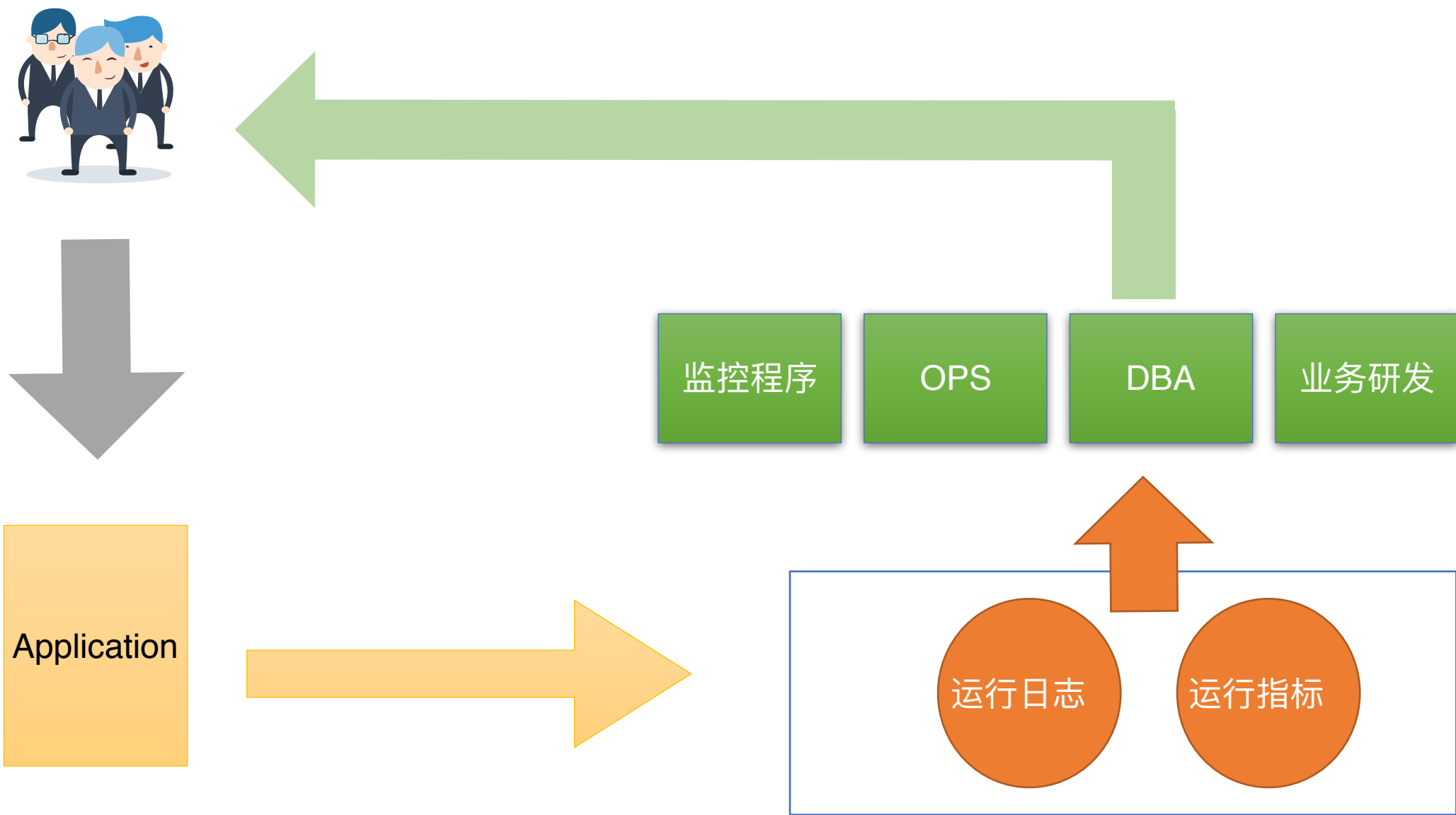


# 表迁移 - 集群建议

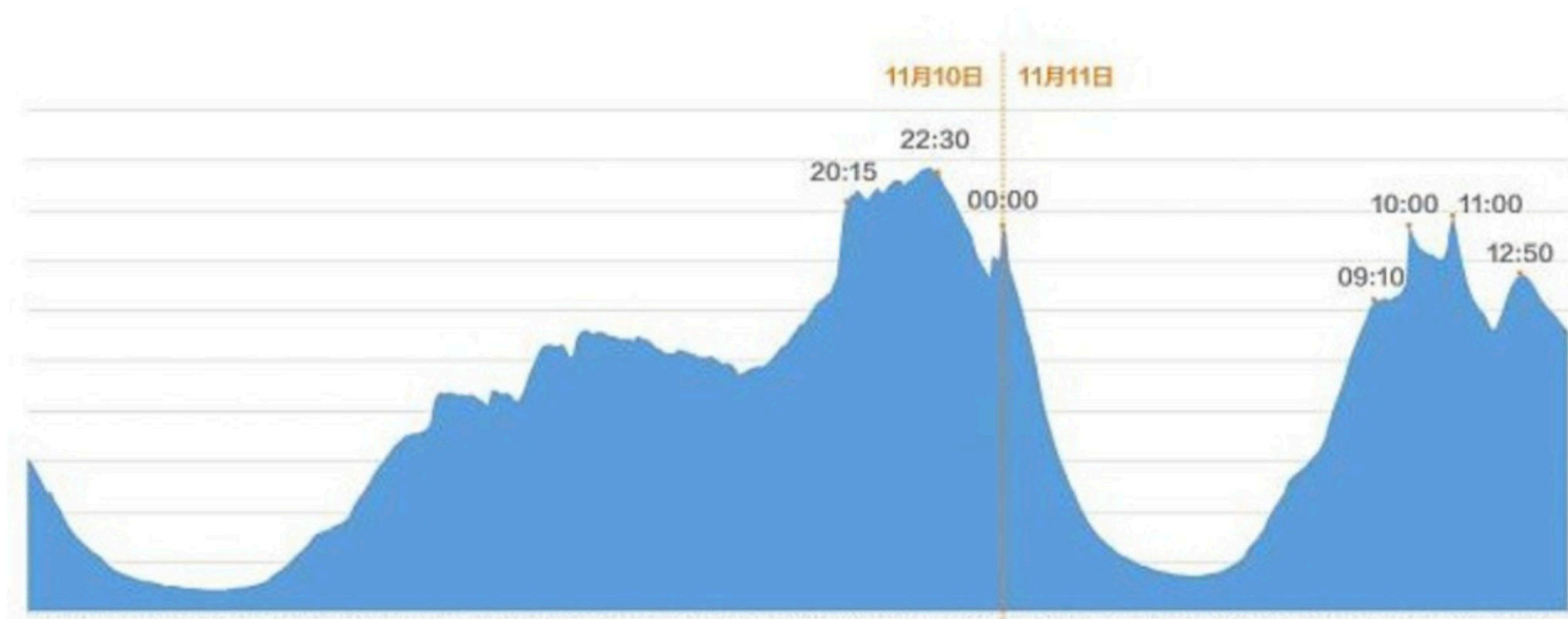
1. 调整合适的集群自增步长
2. 提前确定是否开启Single-Primary模式
3. 集群的写库充当Slave节点，来接收迁移数据  
集群是多节点写入架构，慎重
4. MySQL 8.0 MGR即可多节点作为slave节点



监控继续完善，扩展监控思路



我们在用“用户行为产生的指标”来衡量“对用户提供服务的可用性”



被动式-监控



主动式-监控



APPLICATION

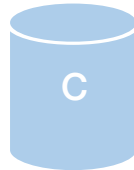
核心功能请求

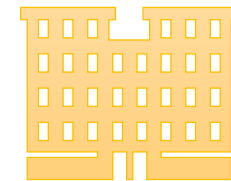


Script Manager: a.py \ b.py

Temp Monitor Result: last 5Min

Service





告警通知  
告警标题: 192.168.4.98 MySQL 3306 is down  
告警状态: 故障中  
下发时间: 2019-03-15 08:17:01  
告警时间: 2019-03-15 08:16:35  
告警来源: zabbix  
告警级别: 严重告警  
告警内容:  
192.168.4.98 MySQL status on 3306:Down (0)



192.168.4.98 是啥机子?

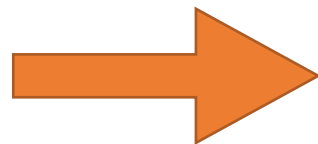
3306 是什么业务?

日志最后输出的是什么内容?

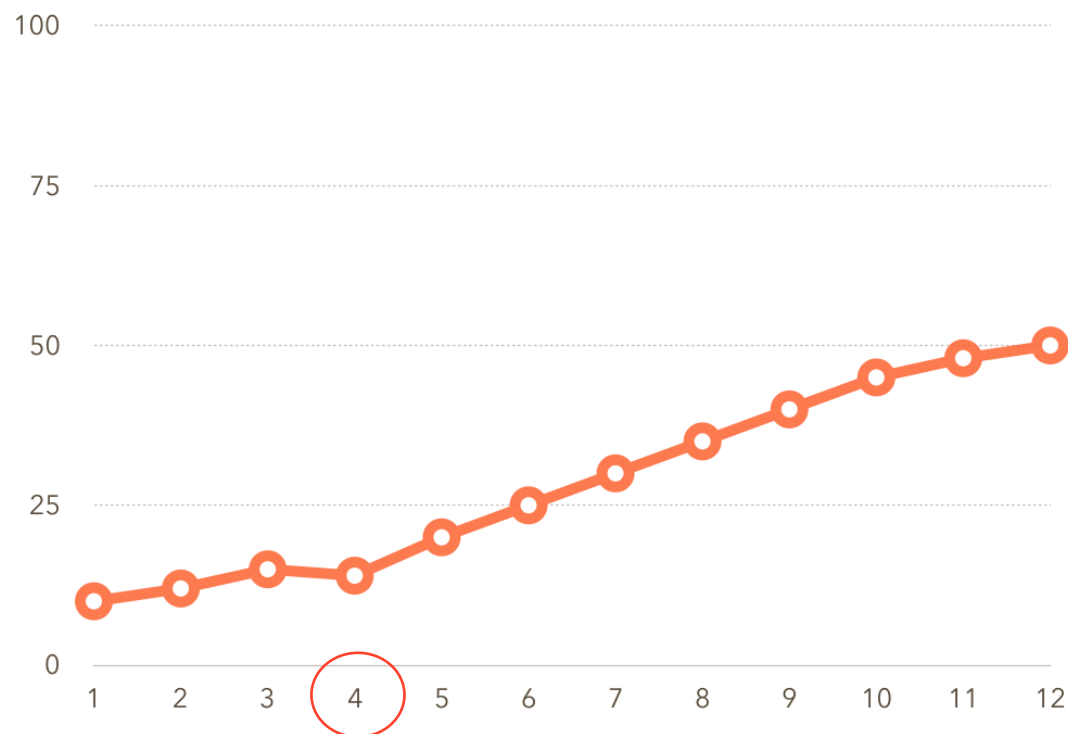
数据库运行情况如何?

但我能做的只是到公司或者靠边停车

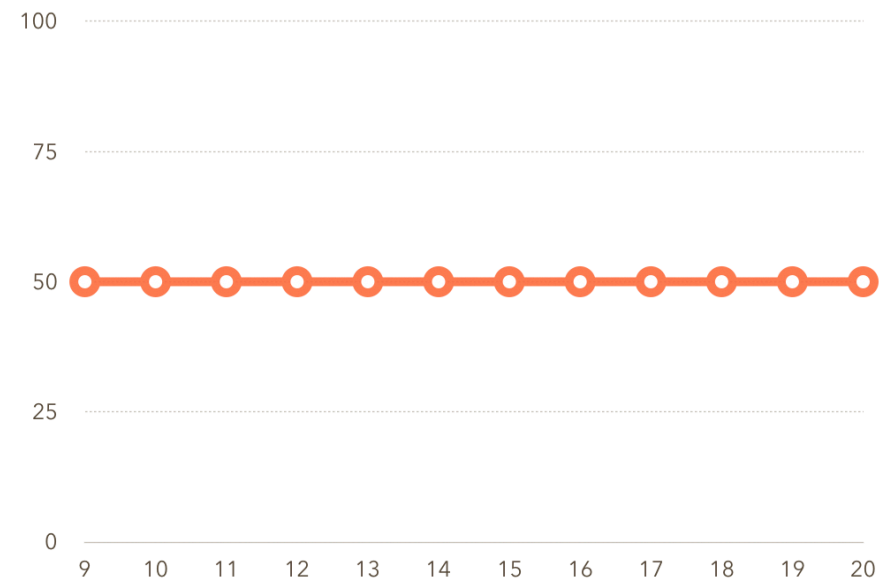
CPU报警阈值: 50%



CPU



发布了一版程序



趋势监控

Day

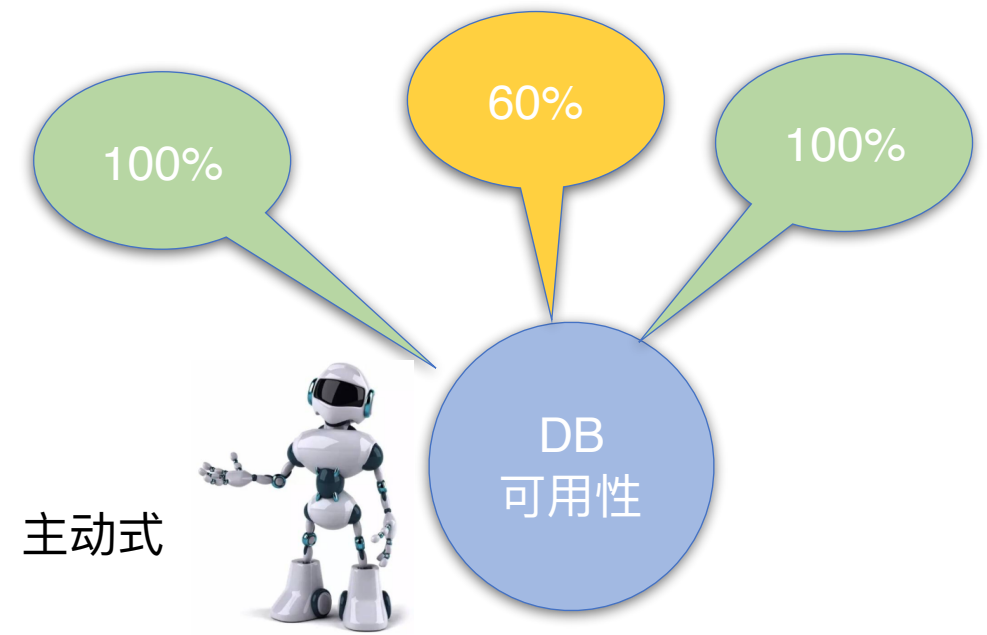
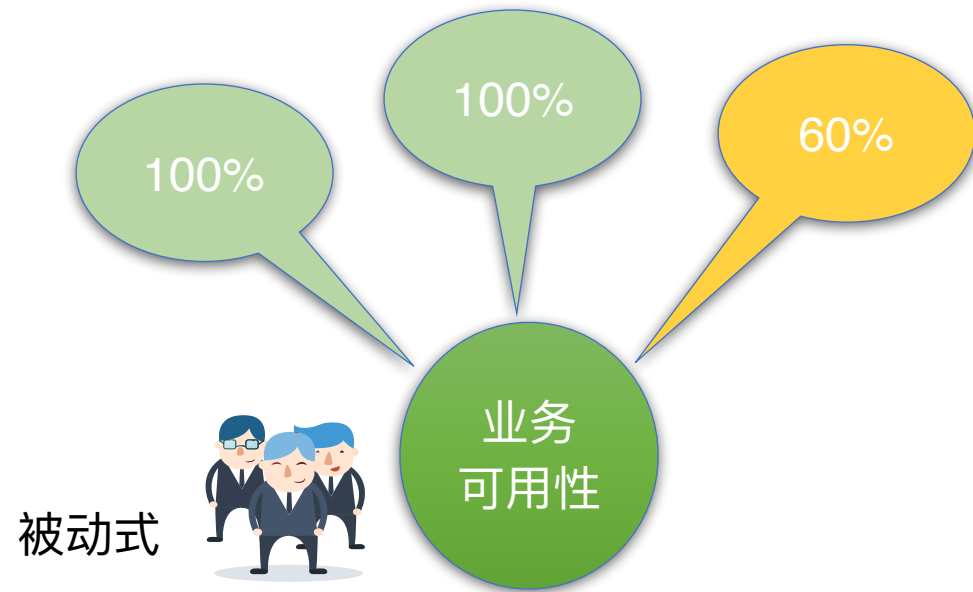
Week

Month



可用性





历史

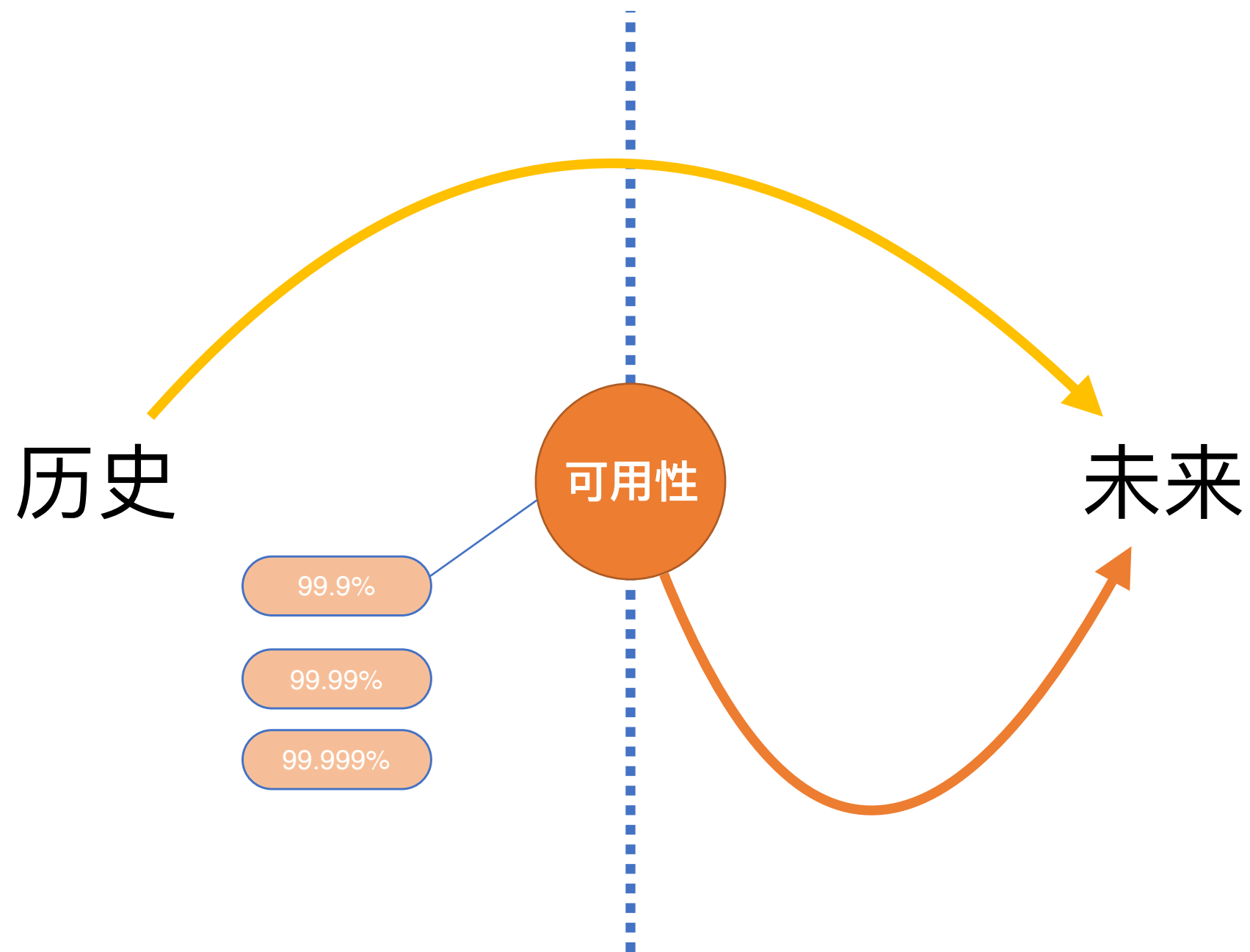
可用性

未来

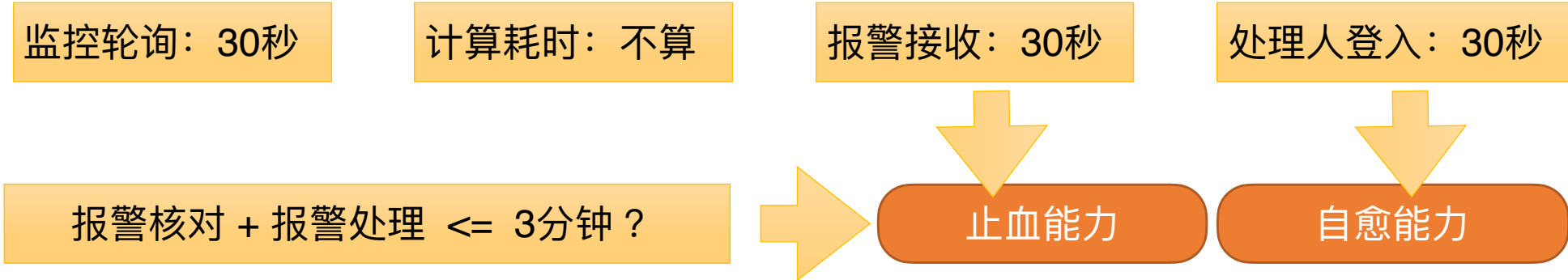
99.9%

99.99%

99.999%



可用性指标	计算方式	不可用时间（分钟）	每月不可用时间（分钟）
99.9%	$0.1\% * 365 * 24 * 60$	525.6	43.8
99.99%	$0.01\% * 365 * 24 * 60$	52.56	4.38
99.999%	$0.001\% * 365 * 24 * 60$	5.256	0.438



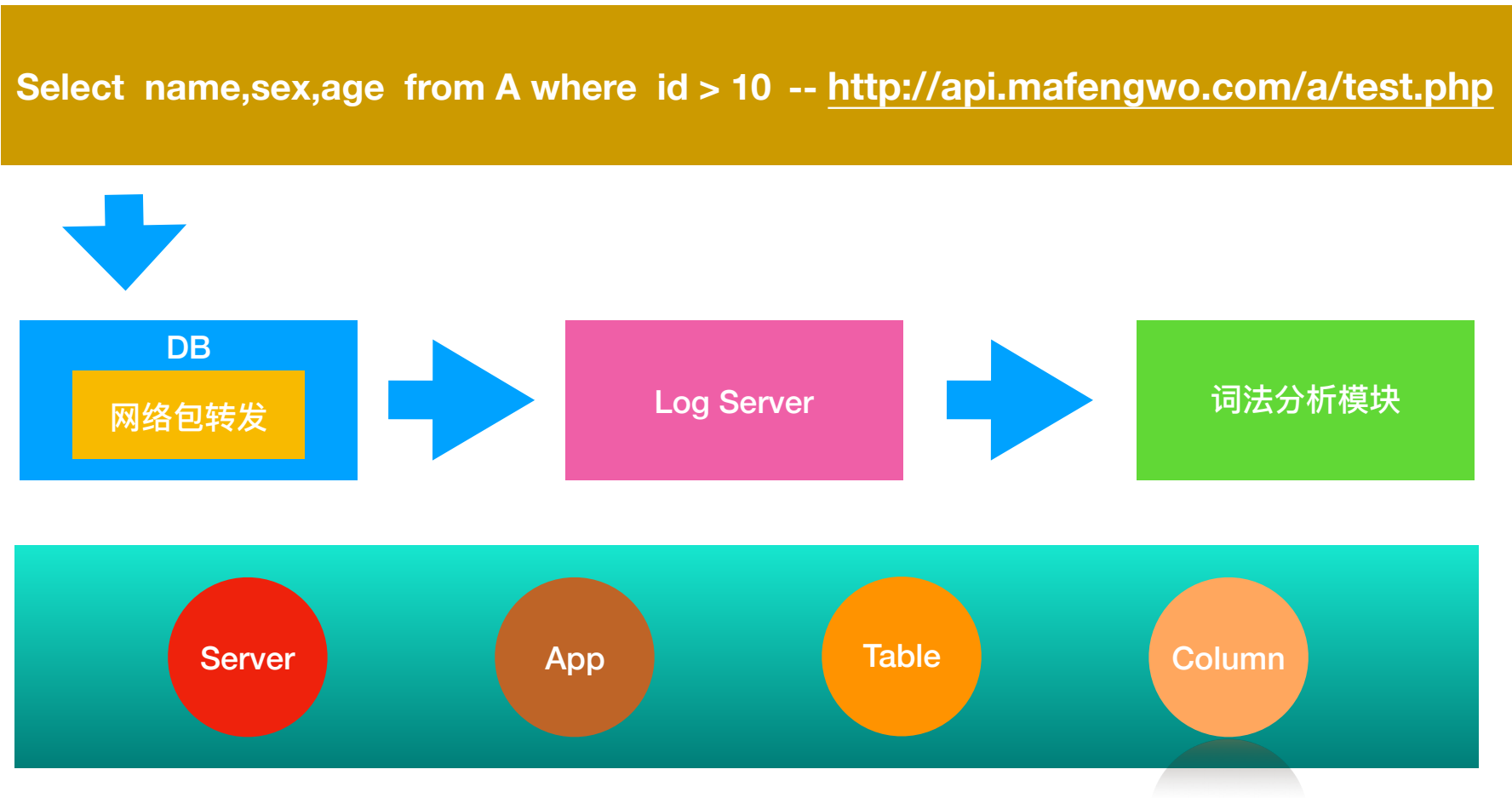


# 继续思考

1. 是否对写库和读库的使用情况了解?
2. 是否对应用程序和对象的关系了解?
3. DBA 如何了解以上问题?



$(\text{QPS:1000}) * (\text{时长:24小时}) * (\text{实例:100个}) = 86\text{亿条SQL}$



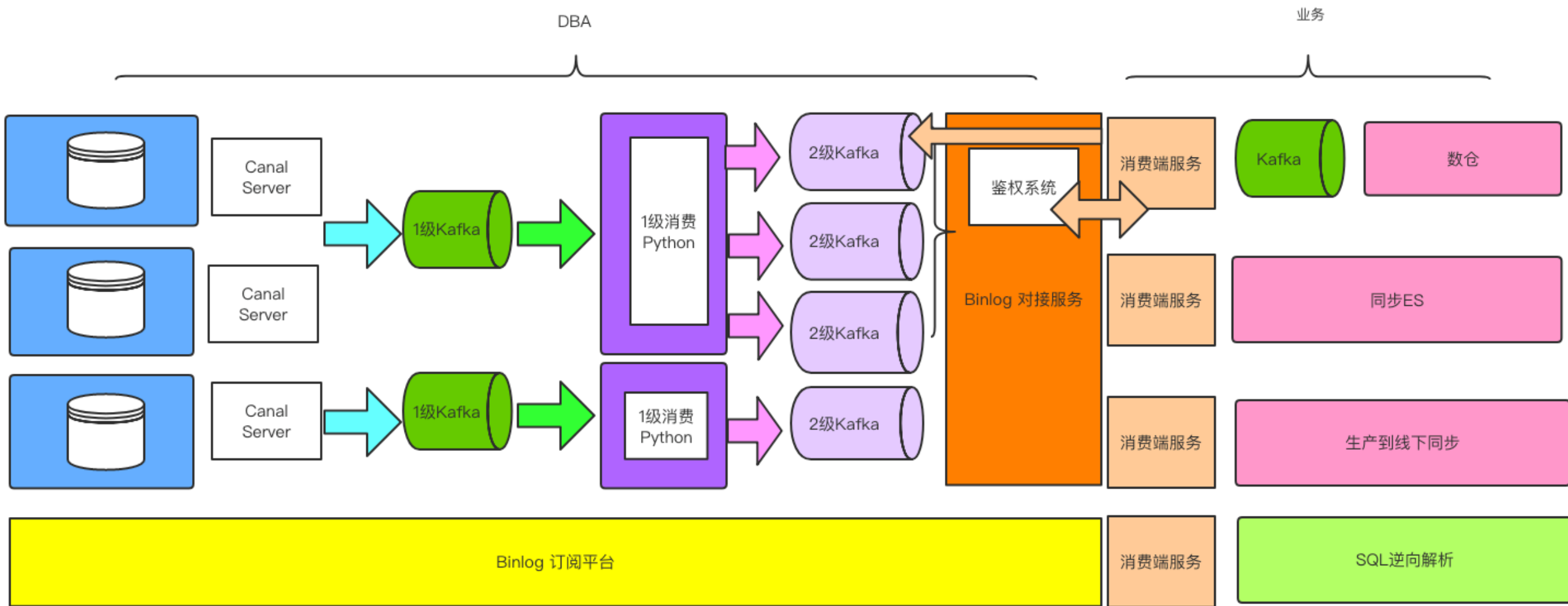
**Data Map**



我们的一些“接地气”架构



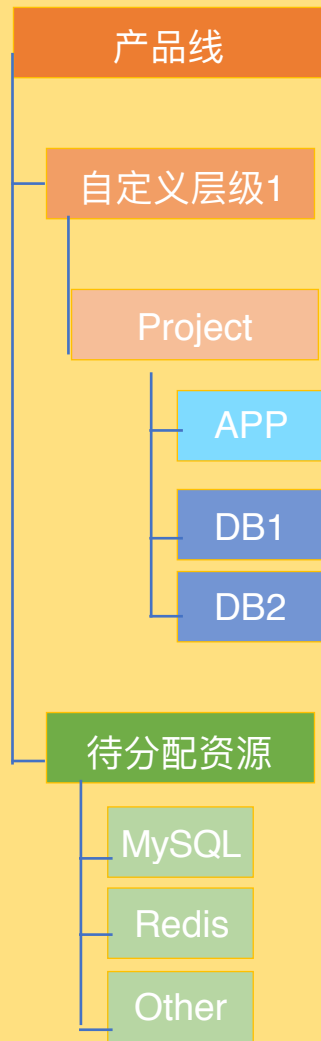
# Binlog 订阅平台





# SQL自动发布平台

## 服务树



CMDB

配置中心

APP:A1

## SQL发布

1.工单模块

2.SQL解析

3.SQL自动审核

4.人工审核

5.SQL发布

6.Other

## 统一工单

4.1工单模块

4.2审批模块

## OA

4.2.1 组织架构API





# THANK YOU

张充



## 关于「3306π」社区

围绕 **MySQL** 核心技术，将互联网行业中最重要数据化解决方案带到传统行业中；囊括其他开源技术Redis、MongoDB、Hbase、Hadoop、ElasticSearch、Storm、Spark等；分享干货知识，即便是赞助商，也要求如此，拒绝放水。



社区公众号



社区QQ群