袋鼠云 DTSTACK·COM

电商核心系统数据库拆分思路

高级数据库专家—宏翊

杭州玳数科技有限公司

数据库分布式架构设计

分布式需求

为什么要做分布式

拆分原则

先垂直，后水平
能不拆就不拆，防止过度设计

拆分的难点及解决方案

分布式设计提高了容量及并发，但会引入复杂性，如何解决

数据库规范设计

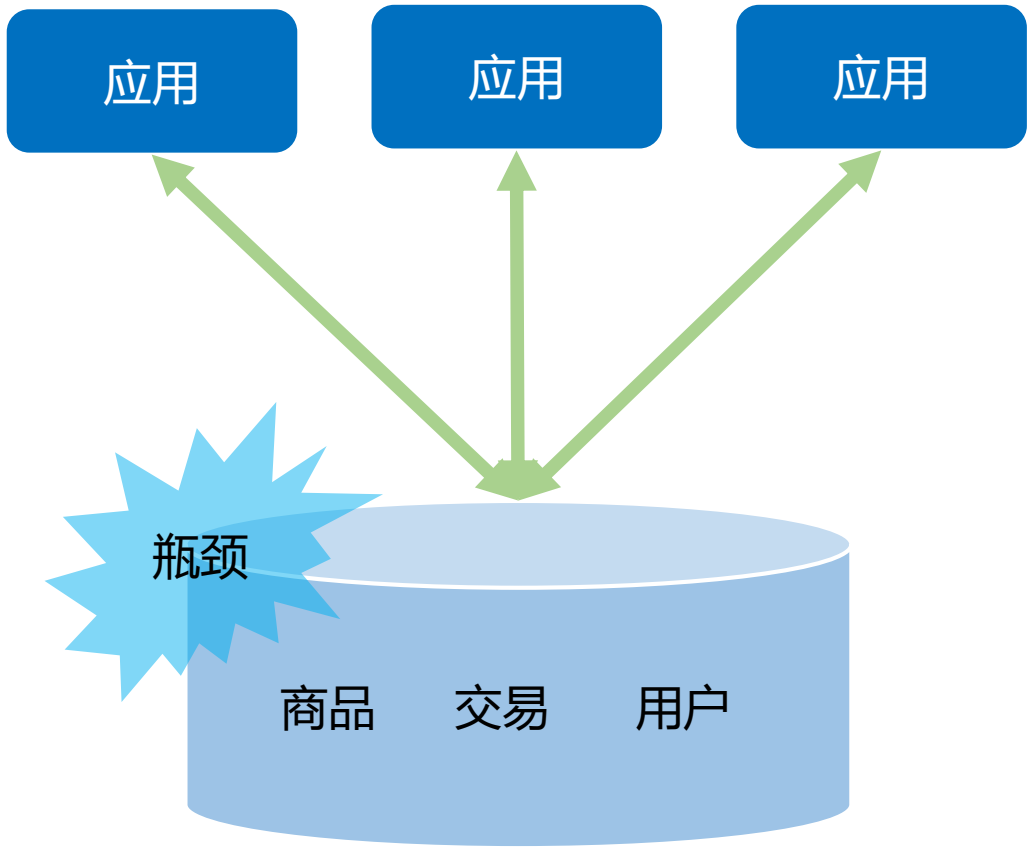
规范化使用才能发挥系统最强效能

运维相关

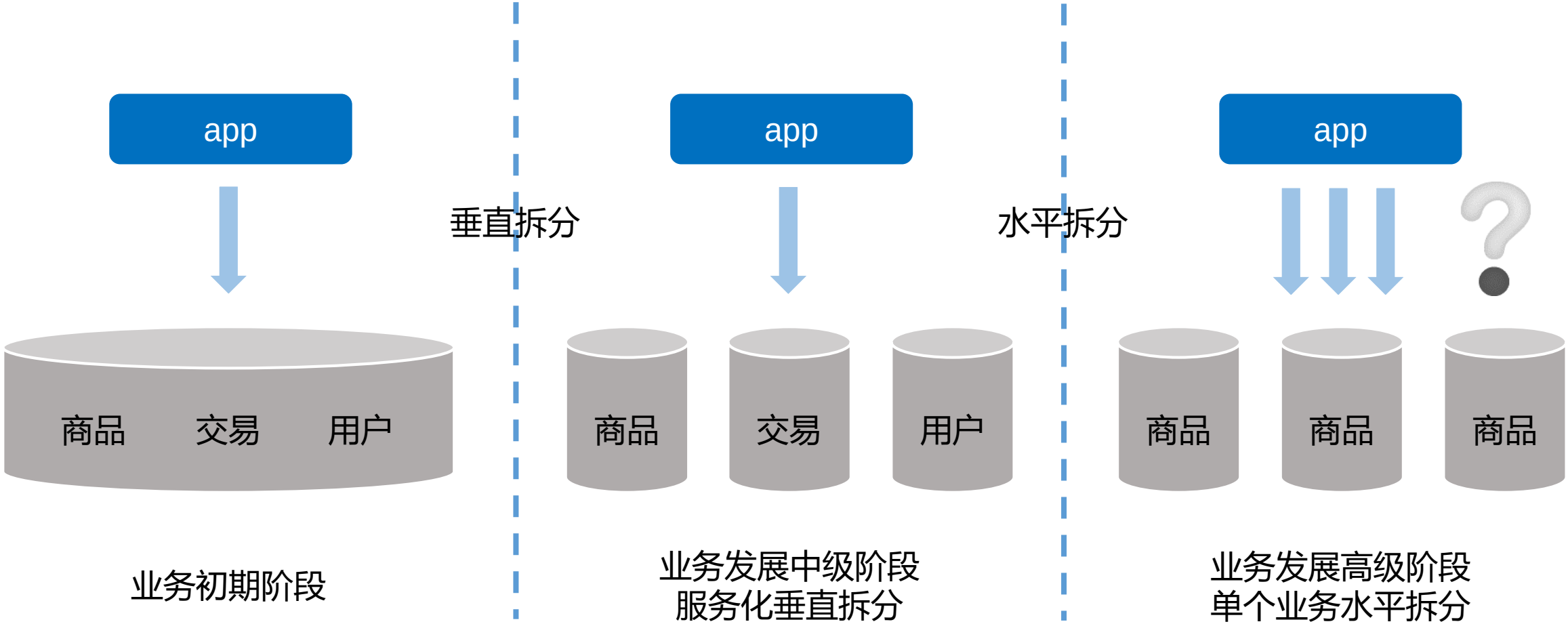
如何能够运维好分布式集群，靠人是不够的，需要借助专业的运维管理工具

为什么要做分布式

- 高并发-分布式应用带来更大量的数据库请求
- 高容量-业务增长，产生大量在线数据
- 资源向上扩展存在天花板
- 支撑业务高速发展，平滑扩容



拆分原则：循序渐进



- 先垂直，后水平
- 谨慎拆分，防止过度设计
- 紧密结合业务及应用架构设计

垂直拆分原则

- 业务逻辑要清晰，按业务领域模型进行数据拆分
- 避免跨业务领域的表关联
- 业务设计上尽量避免分布式事务
- 禁止不相关的业务直接访问数据库

水平拆分原则

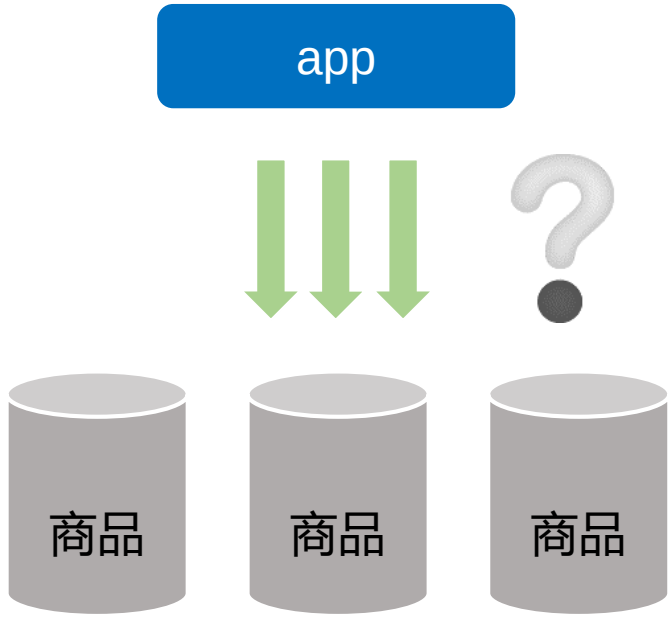
- 谨慎拆分，能不拆就不拆
- 拆分键选择

  数据/请求尽量均衡分布在底层数据库

  尽量避免跨分片的查询，大部分查询能够带上拆分条件
- 分片大小选择

  分片粒度建议细一点，方便后续扩展
- 小表拆分，和有事务绑定的大表拆分到同一个物理库
- 多维度查询，通过数据冗余的方式解决
- 避免分布式事务

水平拆分的难点



业务发展高级阶段
单个业务水平拆分

系统复杂度

系统架构设计、需要彻底的重构

技术挑战

应用需要处理复杂的分布式逻辑

稳定性挑战

分布式的局限性

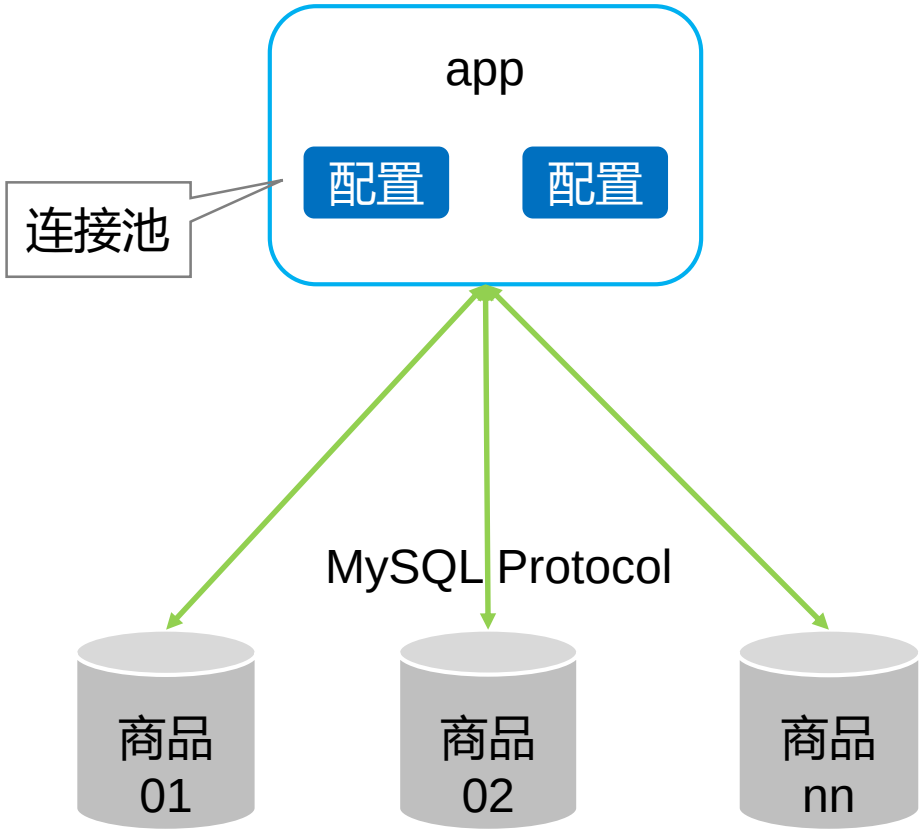
不支持跨库join、分布式事务、全局sequence等

数据分布式一直以来都是个大挑战，技术门槛和改造量都很高。

解决方案：客户端实现数据路由

- 无需引入额外模块，整体架构不变
- 程序把控力强，简单场景方便使用
- 对代码侵入性强 ★
- 配置管理复杂 ★

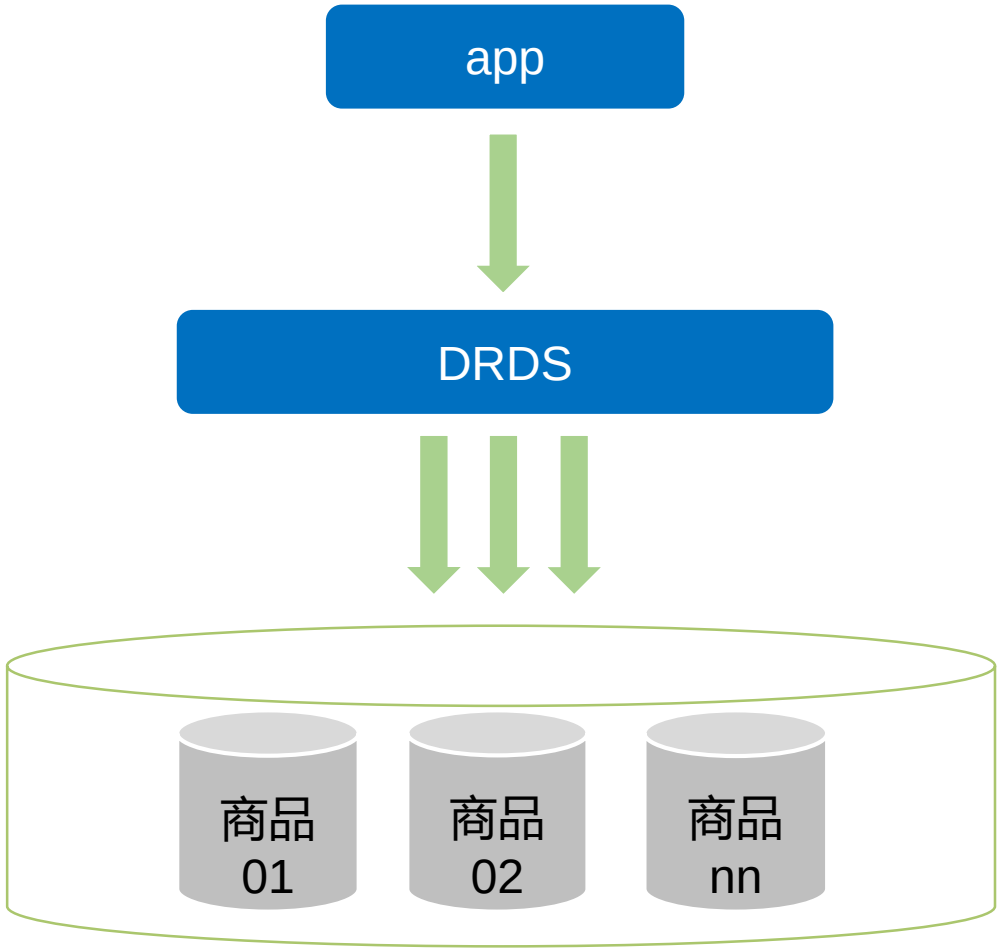
此方案不会引入额外的组建，架构上比较轻量，简单场景使用尚可，但稍复杂的场景会放大它的劣势，比如配置管理复杂等，不建议使用



解决方案：数据库中间件

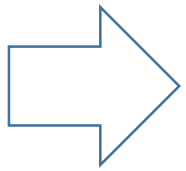
- 自动分库分表，对应用透明。
- 使用门槛极低，应用改造量小。
- 方便的动态水平扩容。
- 针对分布式的各种定制功能，如异构索引、小表广播等。★（部分中间件产品）
- 最重要的是，有了数据库中间件，应用看到还是单一的数据库。

中间件的使用最大限度的屏蔽了分布式数据库所引入的复杂性，极大降低了研发的门槛。



水平切分原理

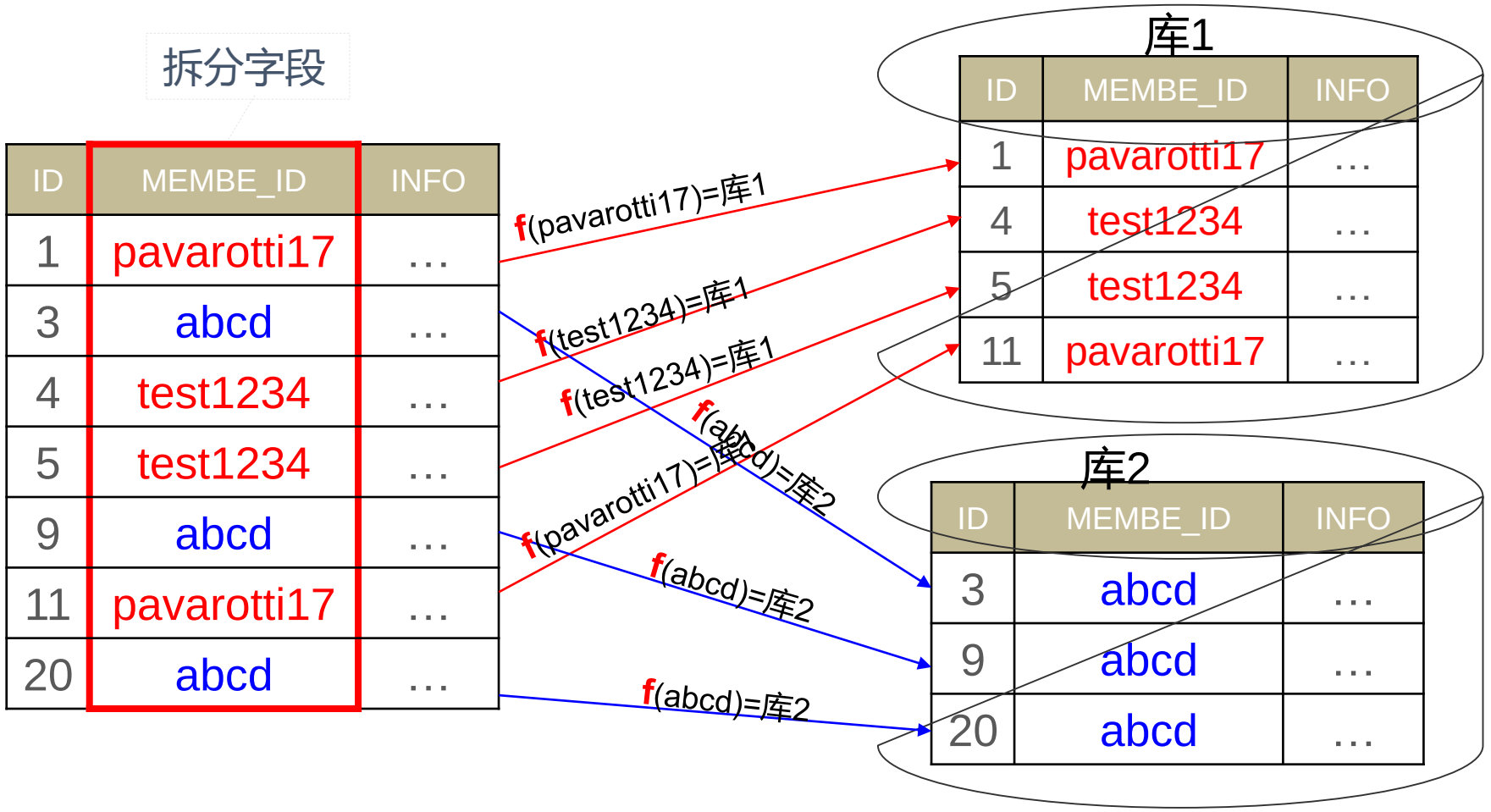
ID	MEMBE_ID	INFO
1	pavarotti17	…
3	abcd	…
4	test1234	…
5	test1234	…
9	abcd	…
11	pavarotti17	…
20	abcd	…



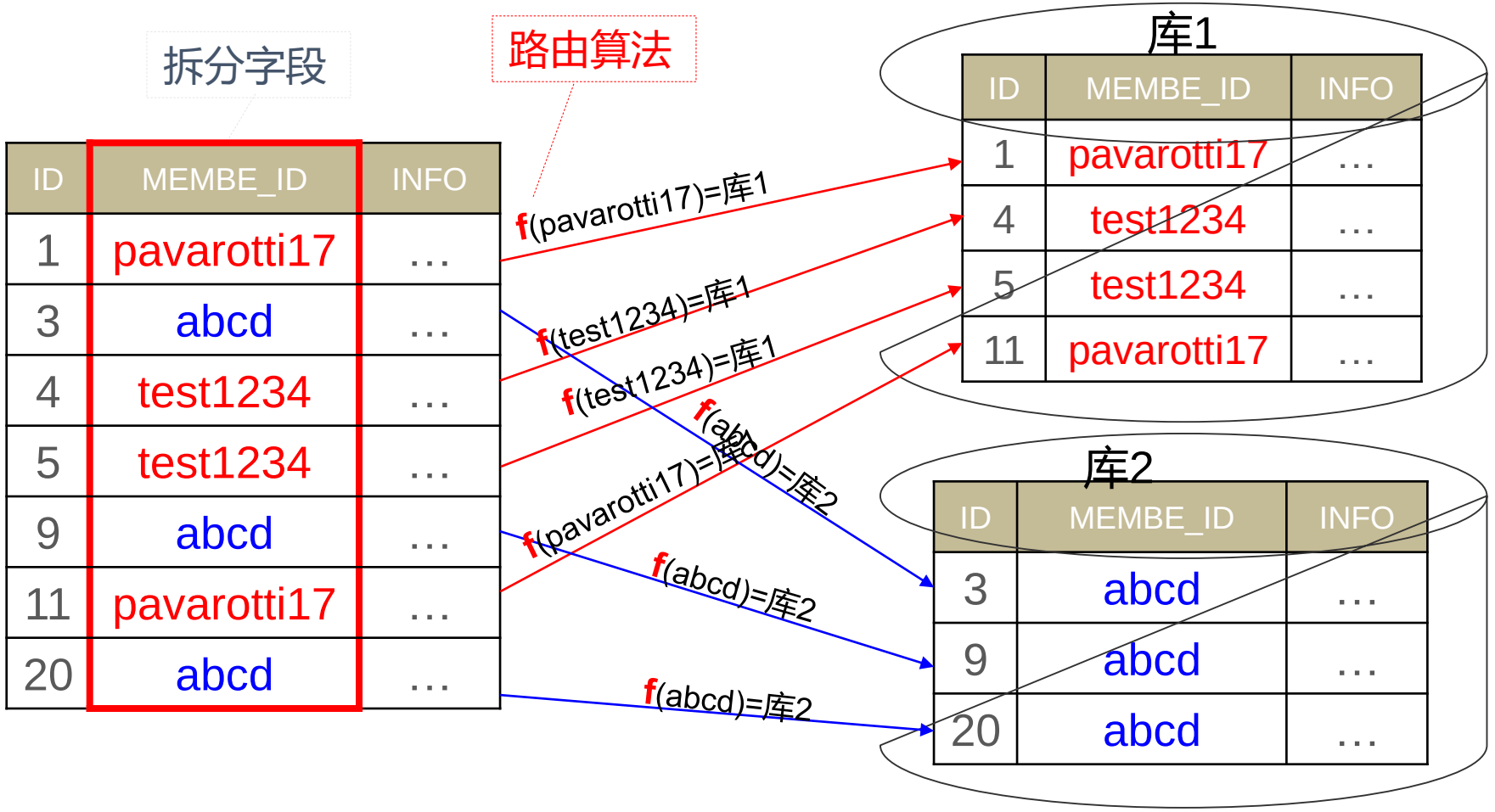
拆分字段

ID	MEMBE_ID	INFO
1	pavarotti17	…
3	abcd	…
4	test1234	…
5	test1234	…
9	abcd	…
11	pavarotti17	…
20	abcd	…

水平切分原理

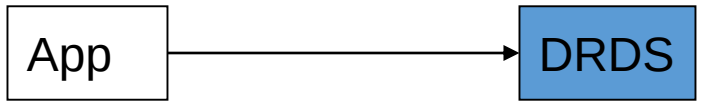


水平切分原理



数据访问-SQL转发

select * from tb1 where
member_id='test1234'



ID	MEMBE_ID	INFO
1	pavarotti17	…
4	test1234	…
5	test1234	…
11	pavarotti17	…

ID	MEMBE_ID	INFO
3	abcd	…
9	abcd	…
20	abcd	…

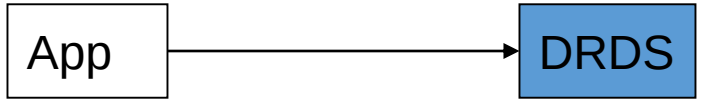
数据访问-SQL转发

ID	MEMBE_ID	INFO
1	pavarotti17	…
4	test1234	…
5	test1234	…
11	pavarotti17	…

ID	MEMBE_ID	INFO
3	abcd	…
9	abcd	…
20	abcd	…

数据访问-SQL转发

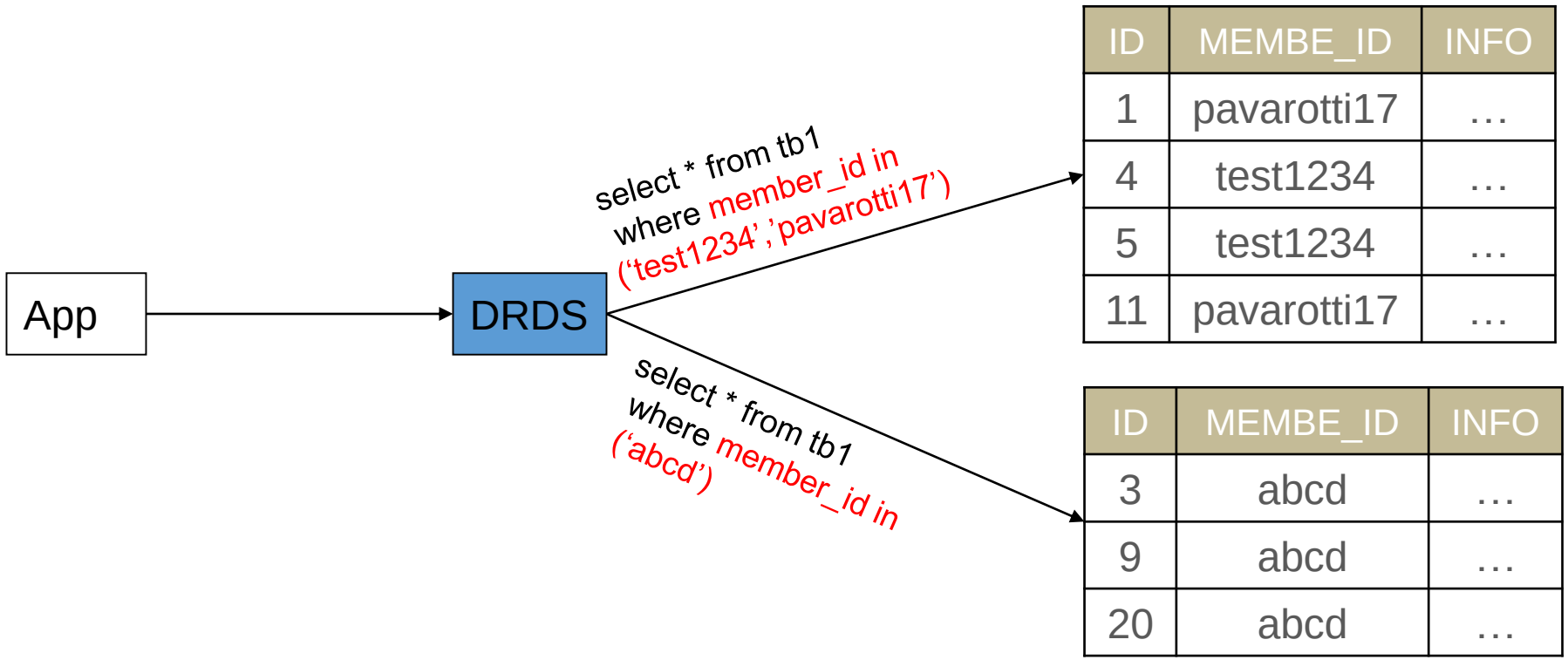
```
SELECT * FROM tb1
 WHERE member_id IN
('test1234','pavarotti17','abcd')
```



ID	MEMBE_ID	INFO
1	pavarotti17	…
4	test1234	…
5	test1234	…
11	pavarotti17	…

ID	MEMBE_ID	INFO
3	abcd	…
9	abcd	…
20	abcd	…

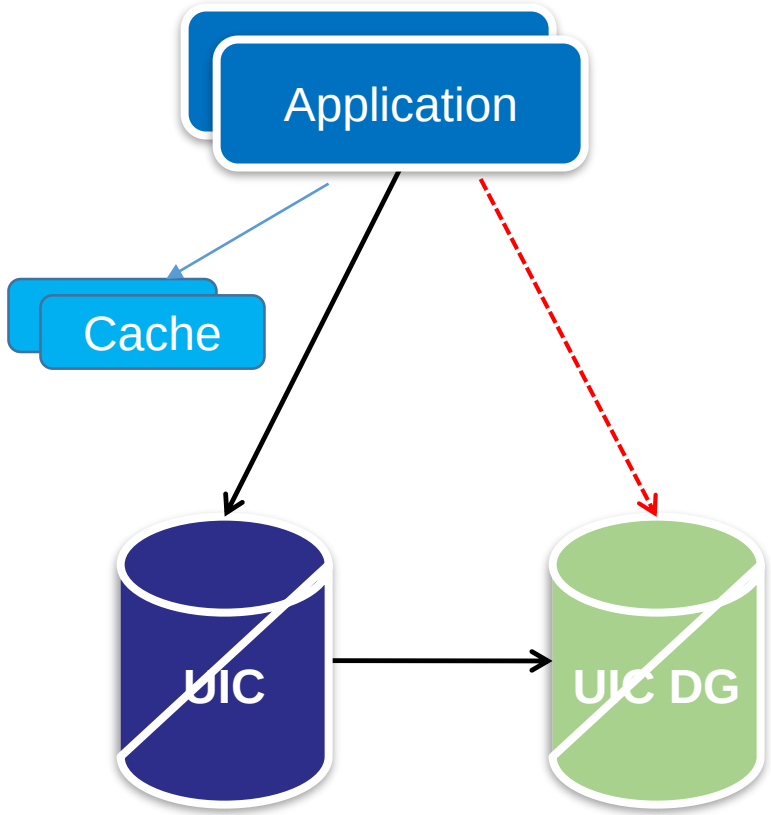
数据访问-SQL转发



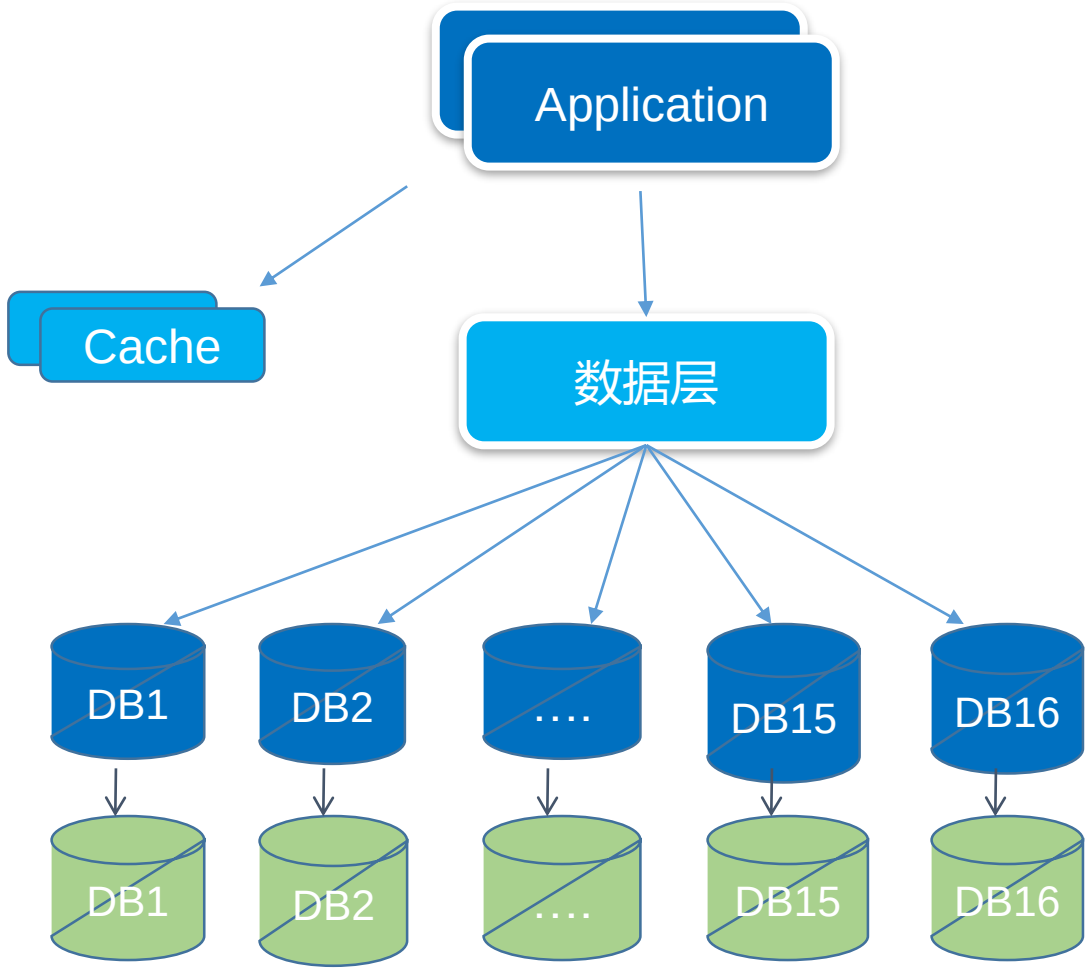
ID	MEMBE_ID	INFO
1	pavarotti17	…
4	test1234	…
5	test1234	…
11	pavarotti17	…

ID	MEMBE_ID	INFO
3	abcd	…
9	abcd	…
20	abcd	…

用户中心架构演变



读压力非常高
缓存命中率下降对数据库影响大
单写入点的风险



按USER_ID hash水平拆分
满足大部分根据user_id的查询请求

用户中心架构 - 多个维度的查询需求

多个查询维度：
会员名，邮箱，手机号，身份证，

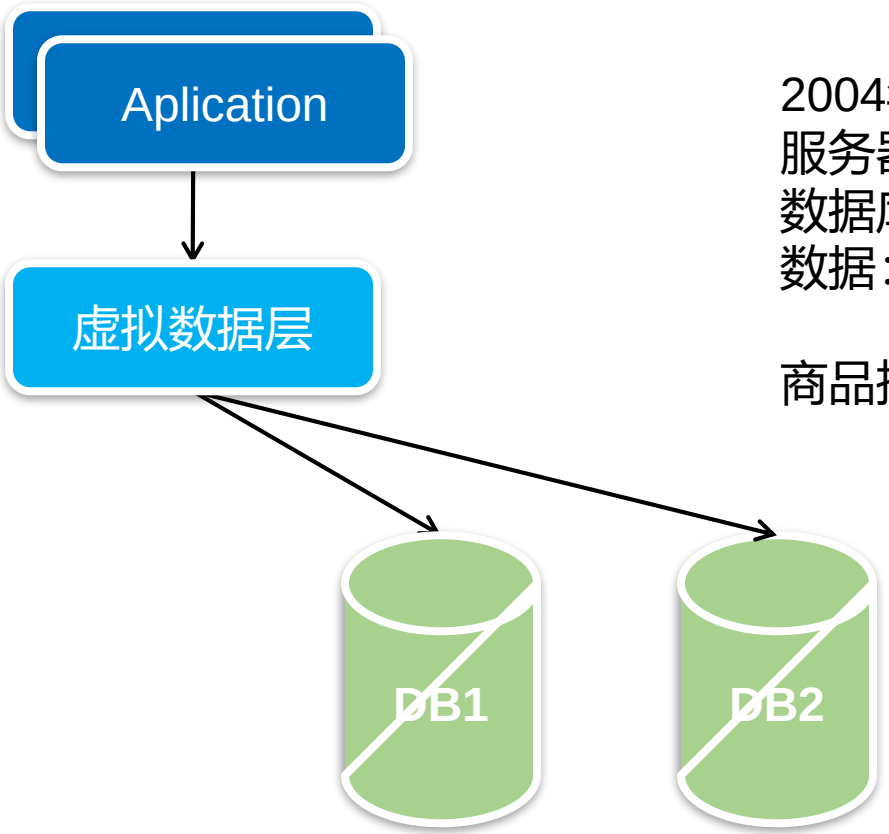
建立mapping表
user_nick_mapping(nick, user_id)
email_mapping(email, uesr_id)
phone_mapping(phone_num, user_id)
……

用户主表拆分键: user_id
Mapping表拆分键: mapping字段hash

数据查询：先从mapping表查user_id，再查主表

新增数据：先写mapping表，再写主表
更新数据：删掉老的mapping，新增新的mapping，更新主表（数据拆分后，不能更新拆分键）

商品中心架构演变



2004年：
服务器：pc服务器
数据库：oracle
数据：　存放到独立存储中

商品拆分成db1、db2两个数据库

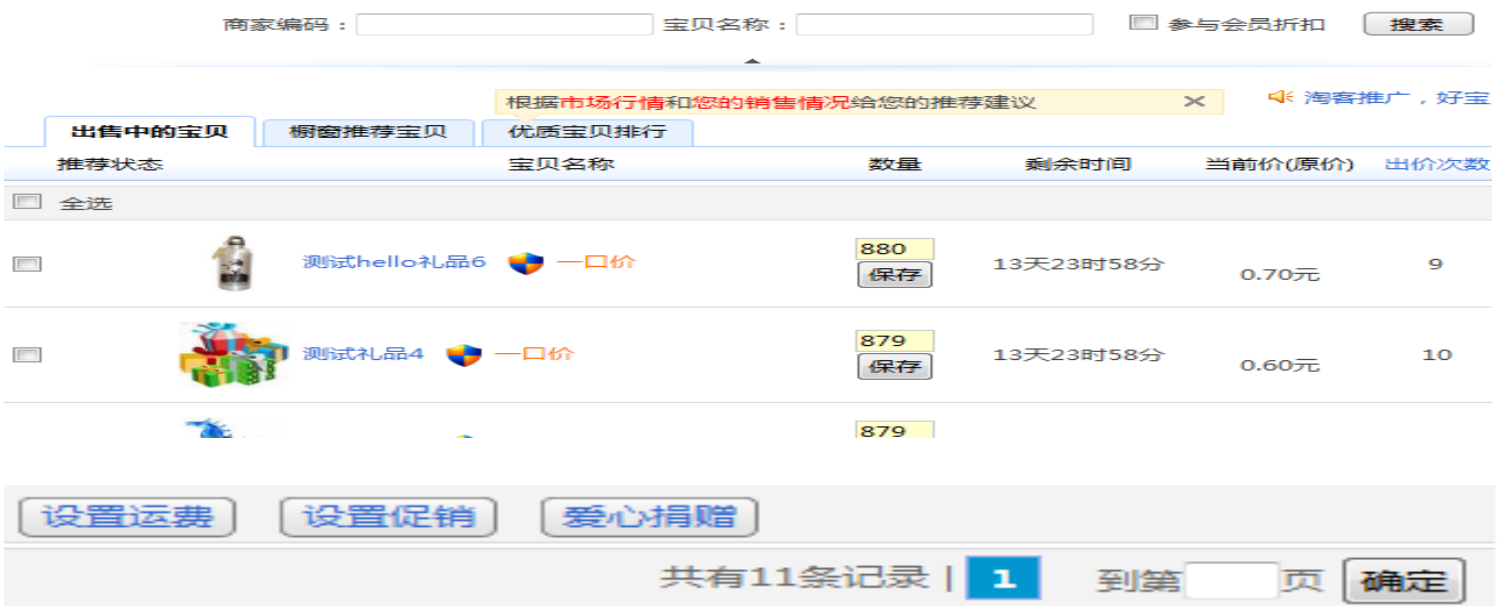
商品中心架构演变

2005年：
数据库整体架构不变，引入高端硬件，减少硬件的故障概率，同时高端的硬件的引入，为未来的高增长奠定了基础。

商品中心架构演变

问题定位：
1.大卖家商品后台管理，count操作，list查询
2.商品标题auction_title like模糊查询，大卖家通常对几十万的商品标题模糊查询，消耗了大量的资源。

我们可以认为是卖家的后台管理操作导致了数据库瓶颈

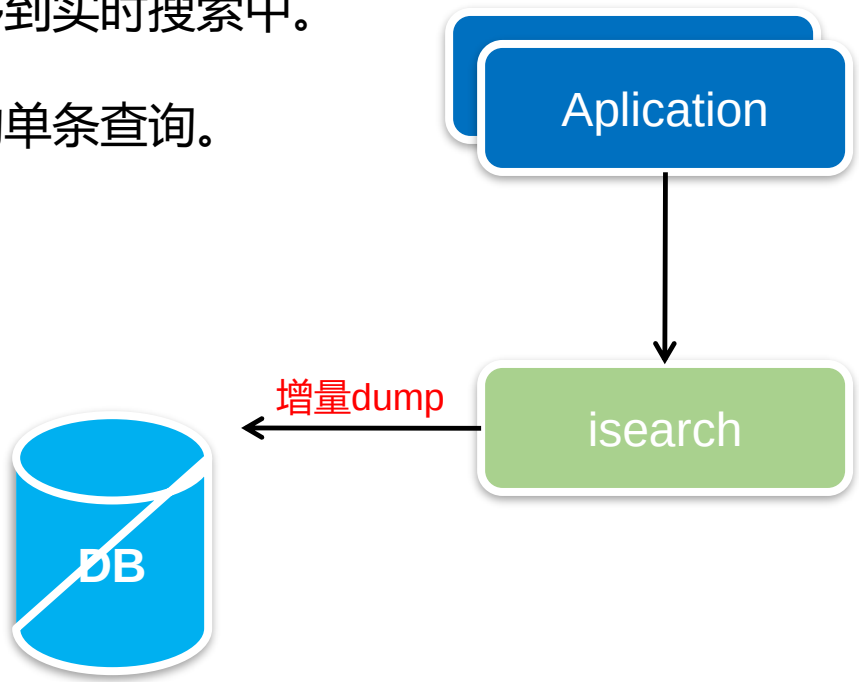


商品中心架构-读写分离

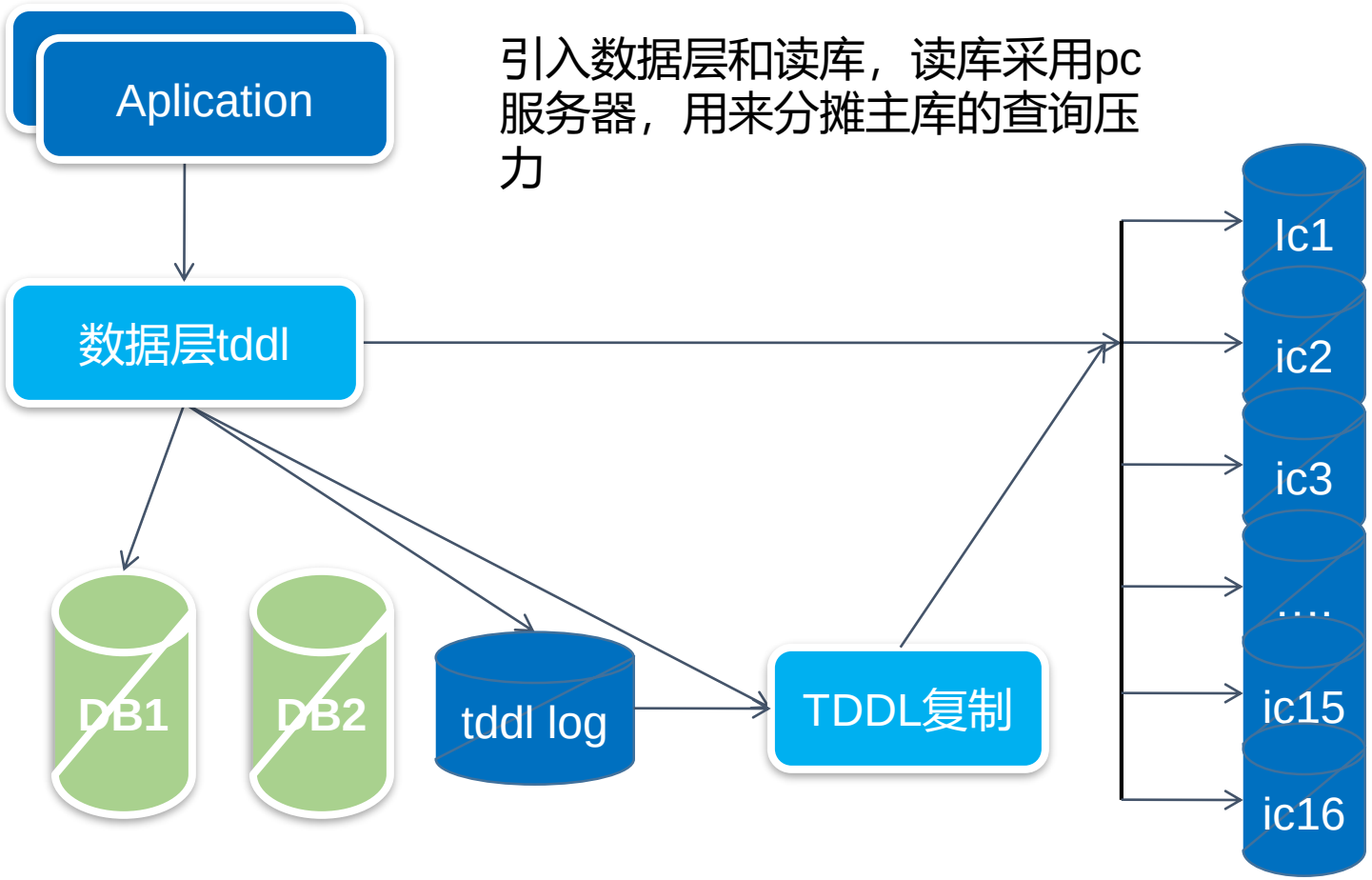
2008年:
开始引入了实时搜索，相比其他的方案，搜索的好处在于很好解决了 title like 的查询。

卖家后台管理开始全部迁移到实时搜索中。

数据库中尽量只支持主键的单条查询。



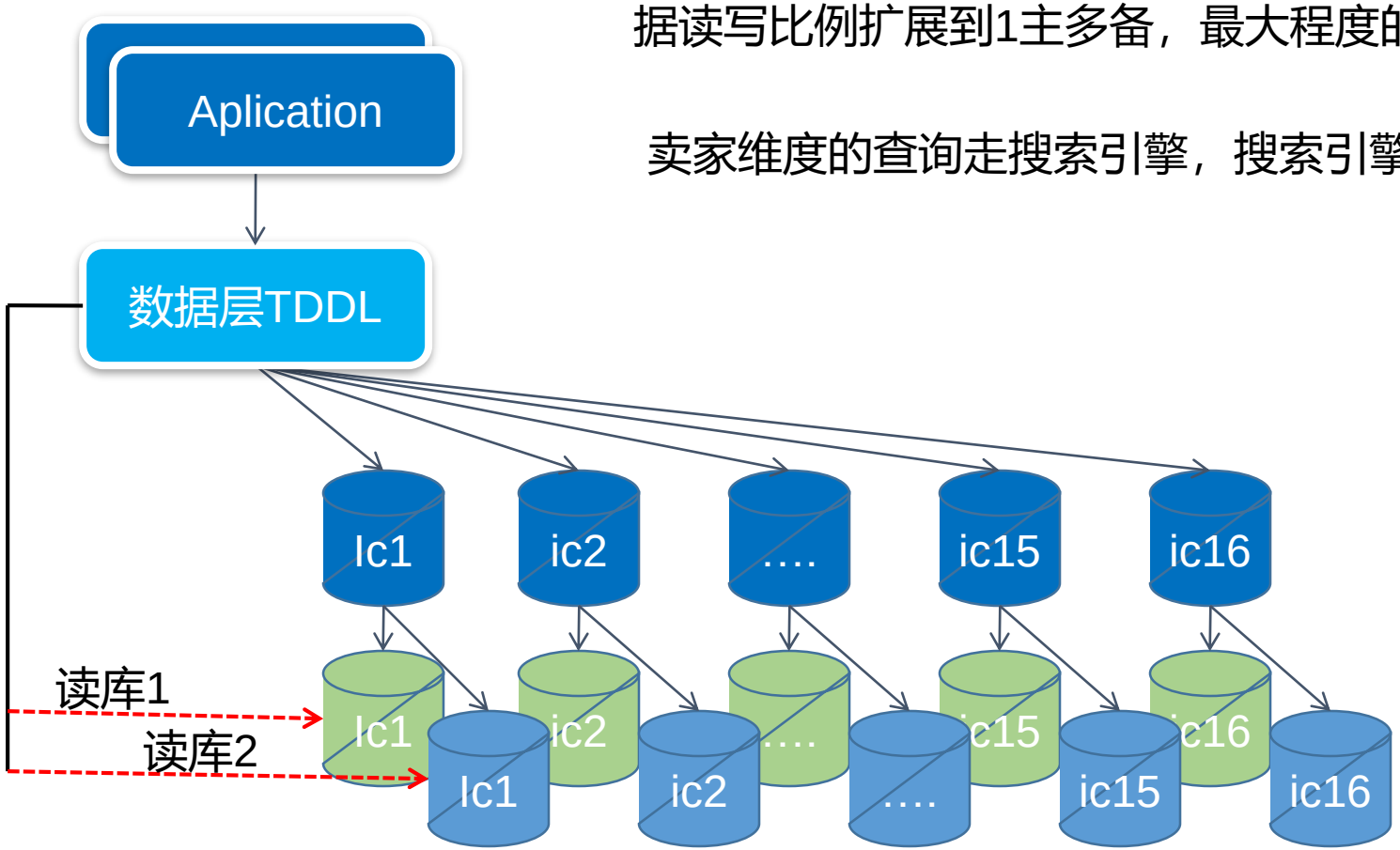
商品中心架构-读写分离



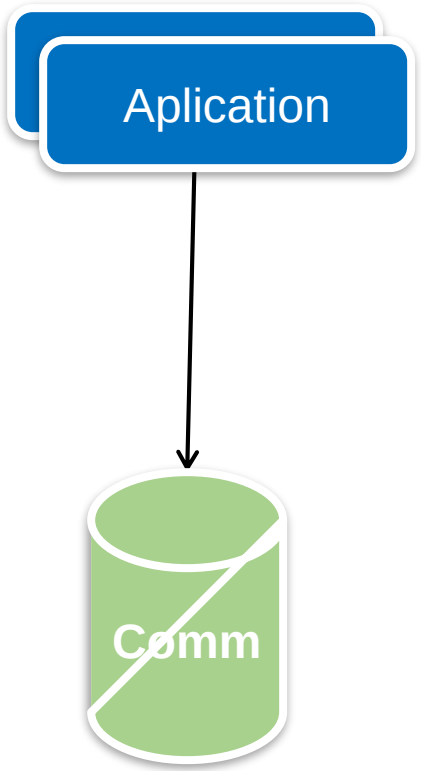
商品中心架构-主库拆分

IC拆分，通过商品ID来分库，备库可以提供读服务，我们可以根据读写比例扩展到1主多备，最大程度的解决了读瓶颈。

卖家维度的查询走搜索引擎，搜索引擎也是读写分离的一个选择



交易中心架构演进



服务器：IBM小型机
数据库：oracle
数据：　存放到独立存储中

相对于商品，交易只负责交易流程，逻辑上相对简单。

交易中心架构演进

相对于商品，交易只负责交易流程，逻辑上相对简单

主要查询场景：买家：1.已买到 2.交易明细列表；
卖家：3.已卖出 4.大卖家模糊查询

交易中心架构演进

一口价：400.00 元
至浙江：快递: 5.0元　EMS: 25.0元
累积售出：261 件
特色服务：

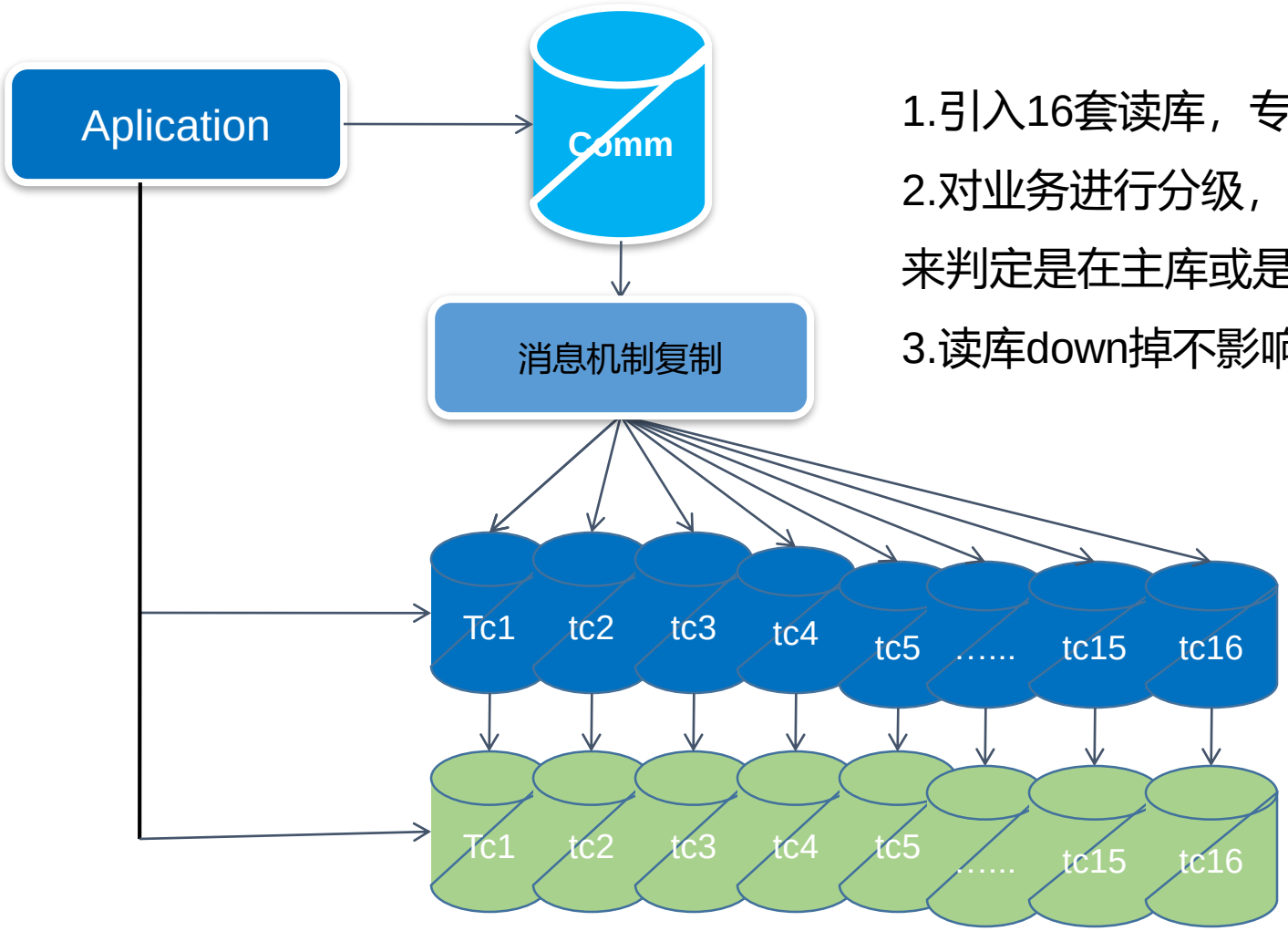
成交记录(261件)

价格：400.00元　和我联系　收藏该宝贝

最近一个月成交记录：

买家	宝贝名称	出价	购买数量
lalae18	新款-专柜正品耐克/NIKE360-男子blazer low休闲鞋-317552-007 颜色:主图颜色;运动鞋尺码:41	400	1
蓝魔水	新款-专柜正品耐克/NIKE360-男子blazer low休闲鞋-317552-007 颜色:主图颜色;运动鞋尺码:41	400	1
sunyaoyu722	新款-专柜正品耐克/NIKE360-男子blazer low休闲鞋-317552-007 颜色:主图颜色;运动鞋尺码:42	400	1
bolipppp	新款-专柜正品耐克/NIKE360-男子blazer low休闲鞋-317552-007 颜色:主图颜色;运动鞋尺码:41	400	1
q64418698	新款-专柜正品耐克/NIKE360-男子blazer low休闲鞋-317552-007 颜色:主图颜色;运动鞋尺码:42	400	1

交易中心架构演进 - 读写分离

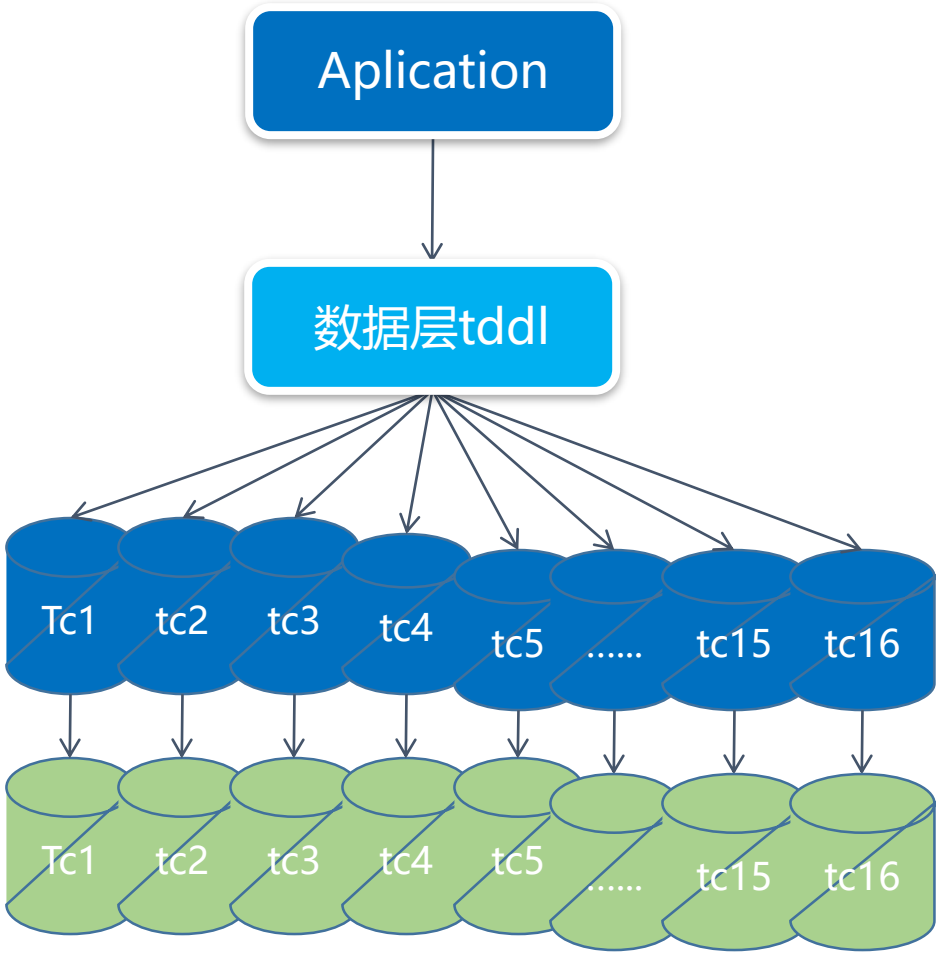


1.引入16套读库，专门用来提供查询

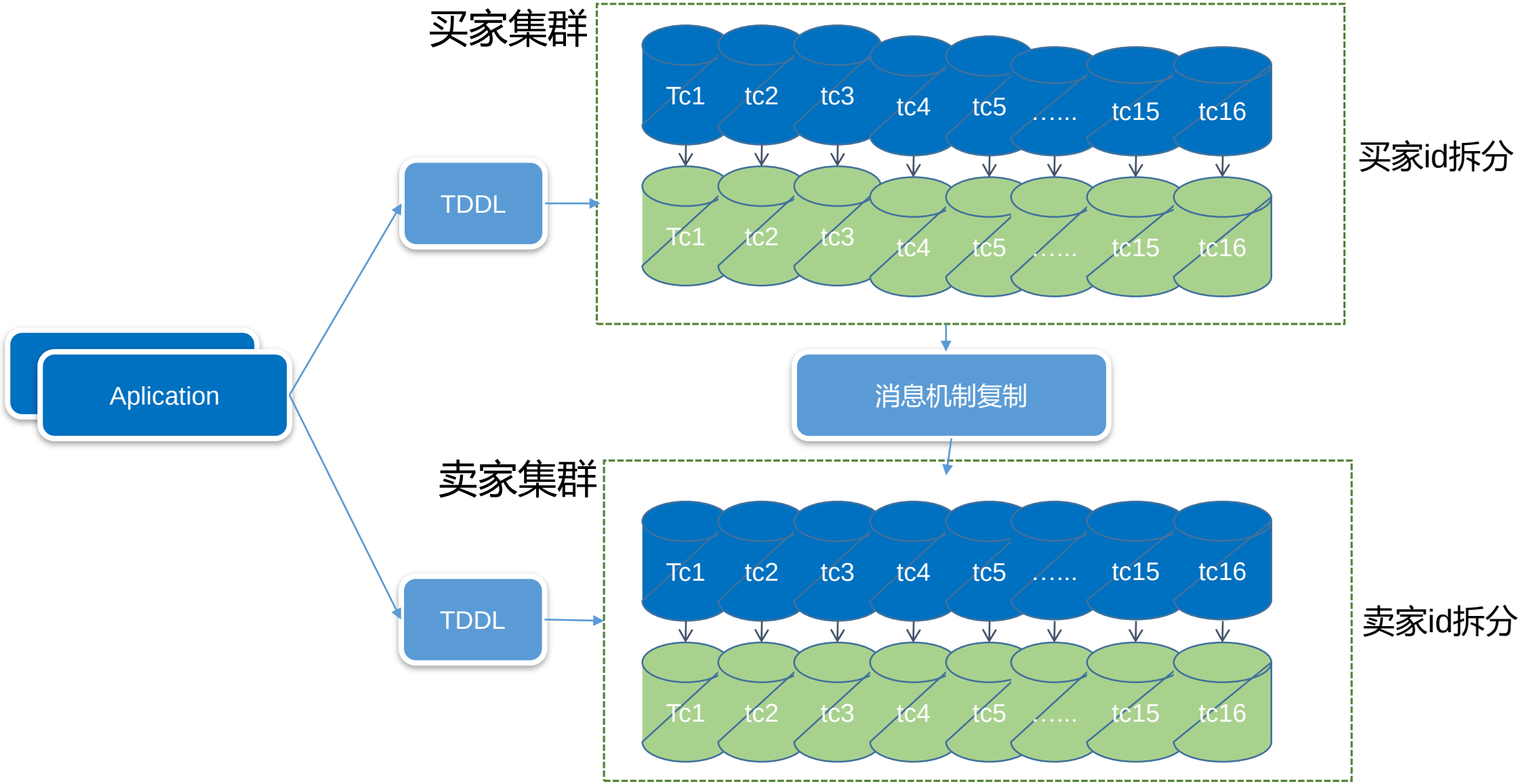
2.对业务进行分级，根据业务优先级来判定是在主库或是读库上查询

3.读库down掉不影响交易主流程

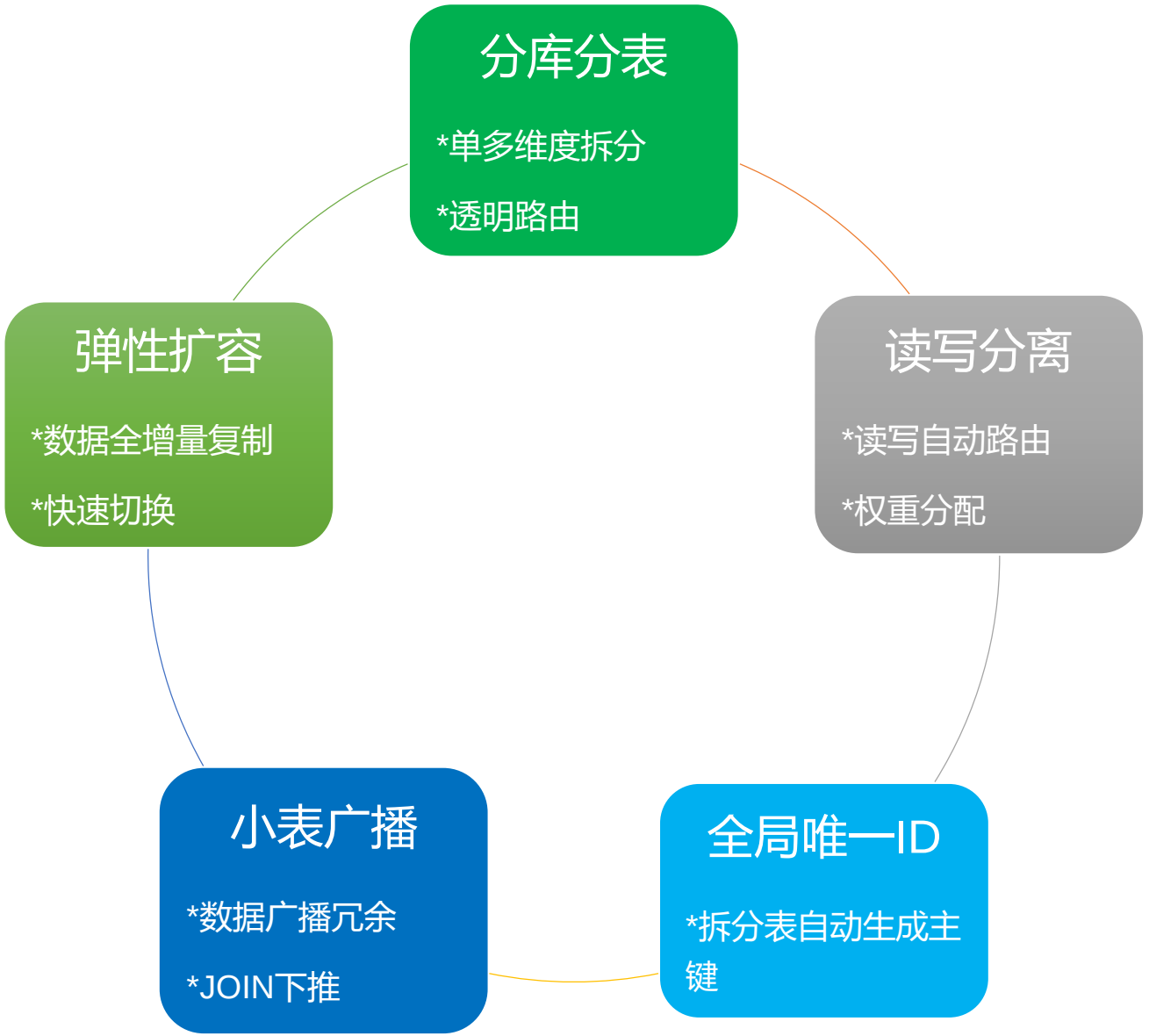
交易中心架构演进-主库拆分



交易中心架构-主库拆分

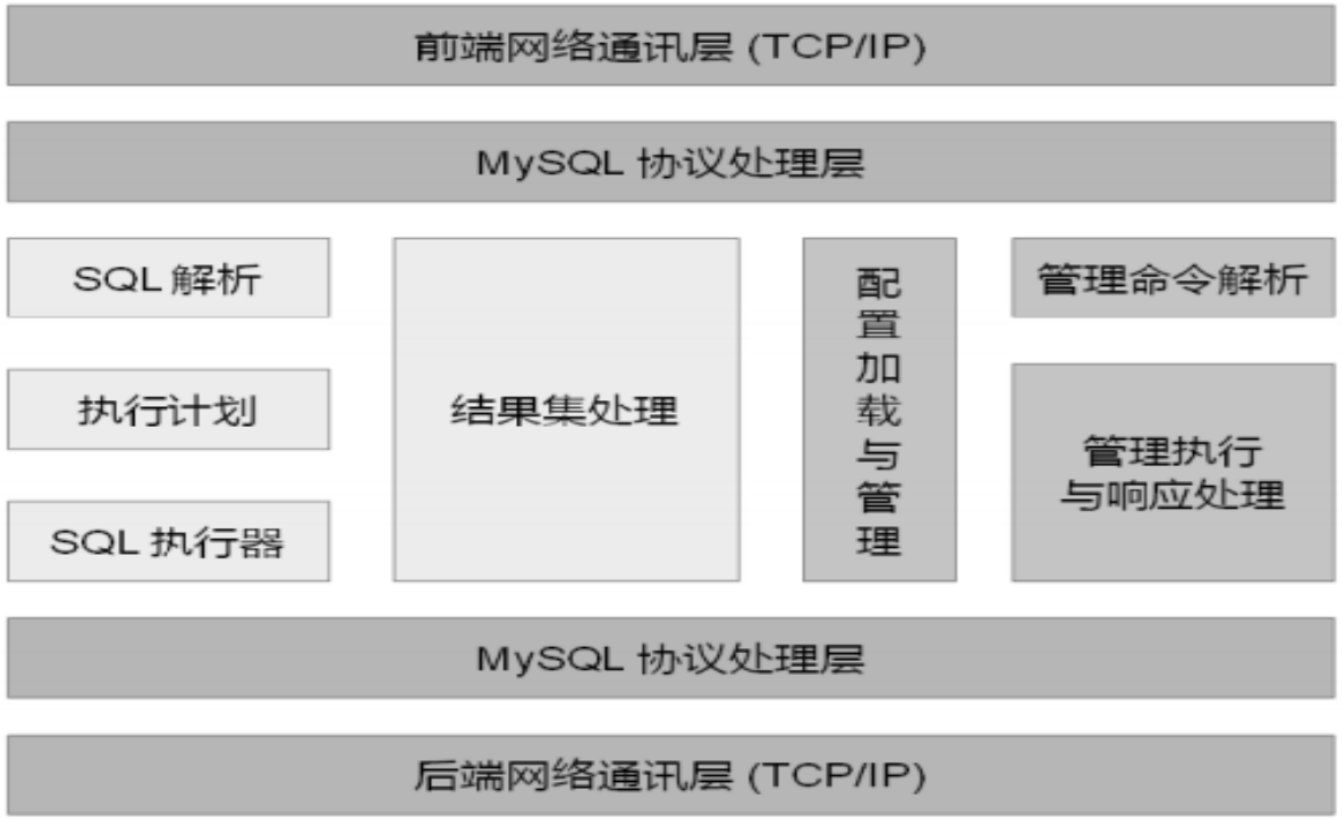


DRDS功能简介



- 分库分表是 DRDS 的核心功能，支持数据的多维度切分和路由访问；
- 内建读写分离功能，可以灵活配置访问权重；
- 自带全局唯一 ID 组件；
- 查询引擎识别和下推复杂查询，兼容 98% MySQL 语法；
- 弹性扩容组件实现自动化在线水平扩容。

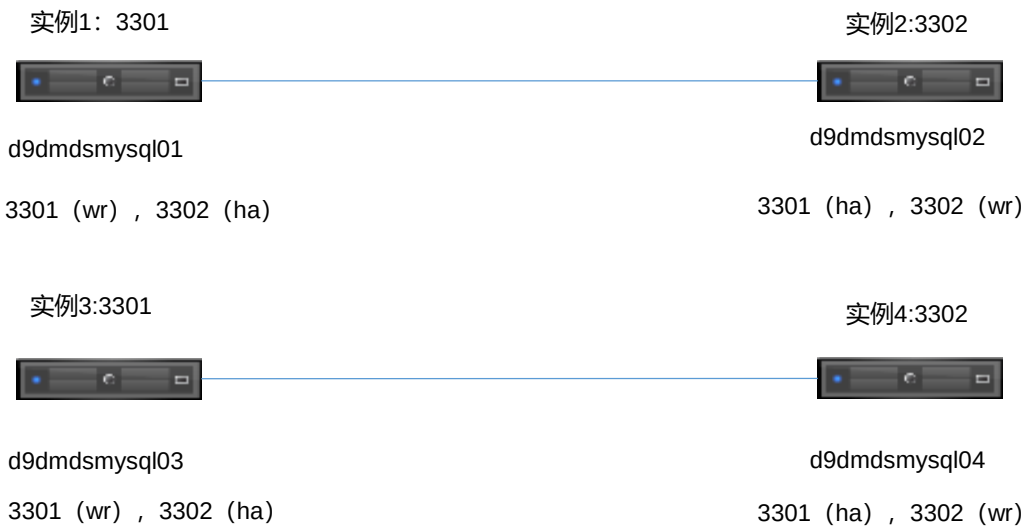
DRDS框架

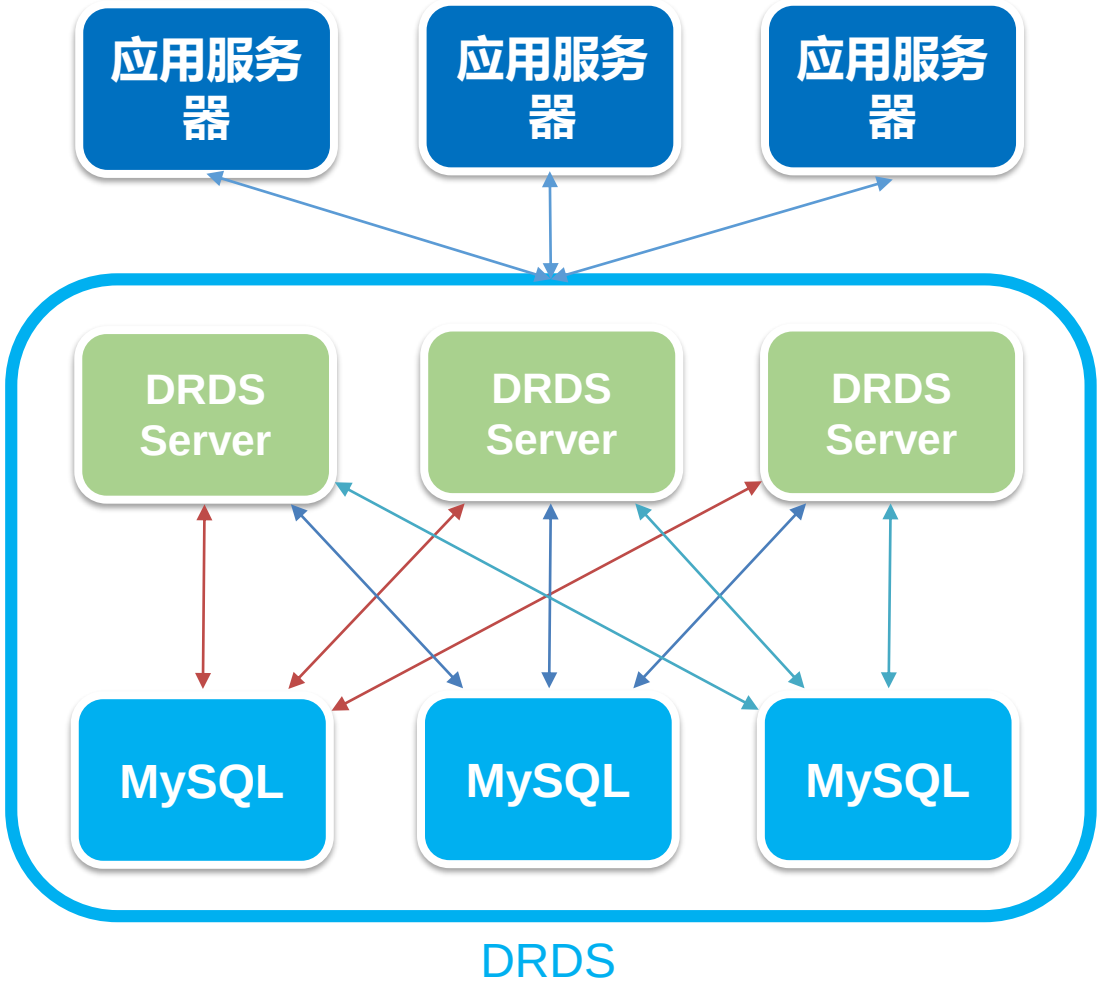


模块架构图

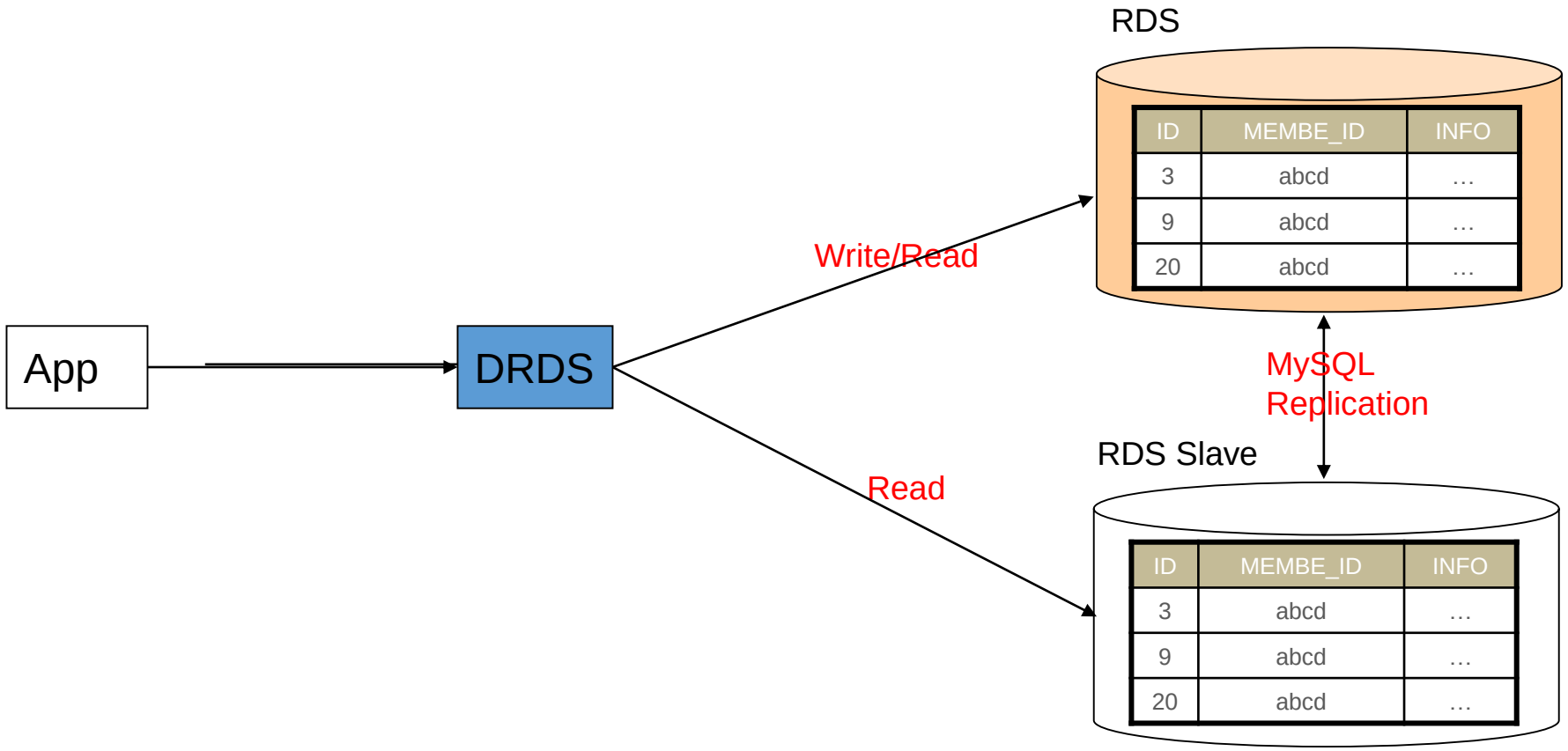
DRDS物理部署框架

- 数据拆分， 能够组合1K个MySQL
- 分布式SQL查询引擎与高度的SQL兼容性
- 数据存储的平滑扩容
- 处理性能的弹性伸缩
- 读写分离（应用透明）
- 小表广播、跨库join、全局sequence





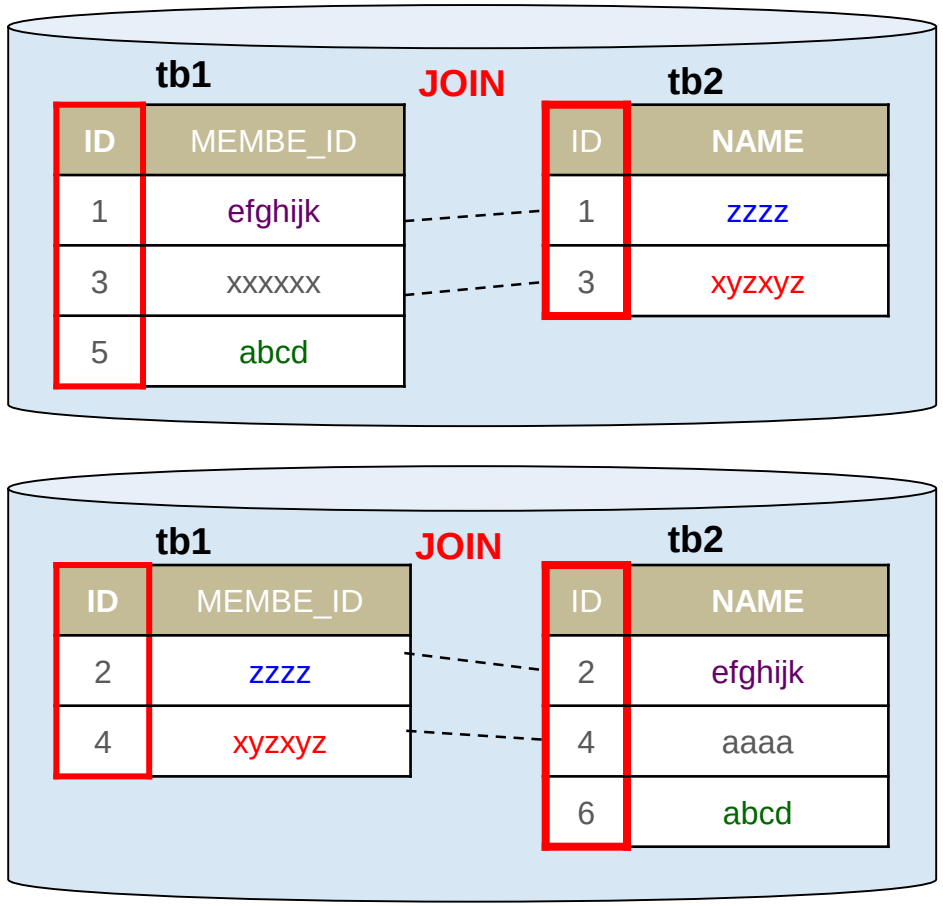
DRDS特性-读写分离



DRDS特性-下推join

```
SELECT *
  FROM tb1 INNER JOIN tb2
    ON t1.ID=t2.ID
```

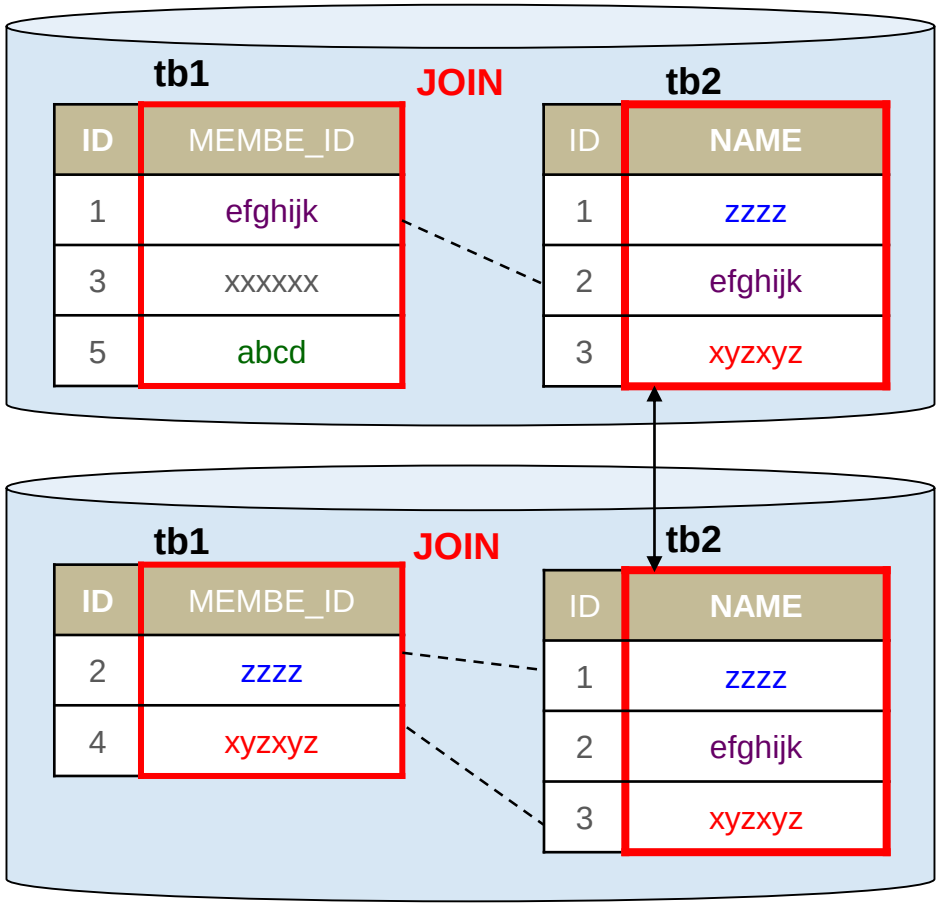
ID	MEMBER_ID	ID	NAME
1	efghijk	1	zzzz
3	xxxxxx	3	xyzxyz
2	zzzz	2	efghijk
4	xyzxyz	4	aaaa



DRDS特性-广播表join

```
SELECT *
  FROM tb1 INNER JOIN tb2
    ON t1.MEMBER_ID=t2.NAME
```

ID	MEMBER_ID	ID	NAME
1	efghijk	2	efghijk
2	zzzz	1	zzzz
4	xyzxyz	3	xyzxyz



分布式SQL规范设计：最佳实践

- ✧ 查询尽可能带上分库条件
  - ➢ 单库查询
    - ➢ select * from t1 where id=1
  - ➢ 单库Join
- ✧ Join
  - ➢ 尽可能参与Join的每张表都带上相同的分库条件
    - ➢ select * from t1 join t2 on t1.name=t2.name where t1.id=1 and t2.id=1
  - ➢ 分库键=分库键的Join
    - ➢ select * from t1 join t2 on t1.id=t2.id
  - ➢ 广播表Join
    - ➢ select * from t1 join broadcast on t1.id=broadcast.id
- ✧单库事务
  - ➢ 事务尽量限制在单库范围内，避免引入分布式事务

如何运维

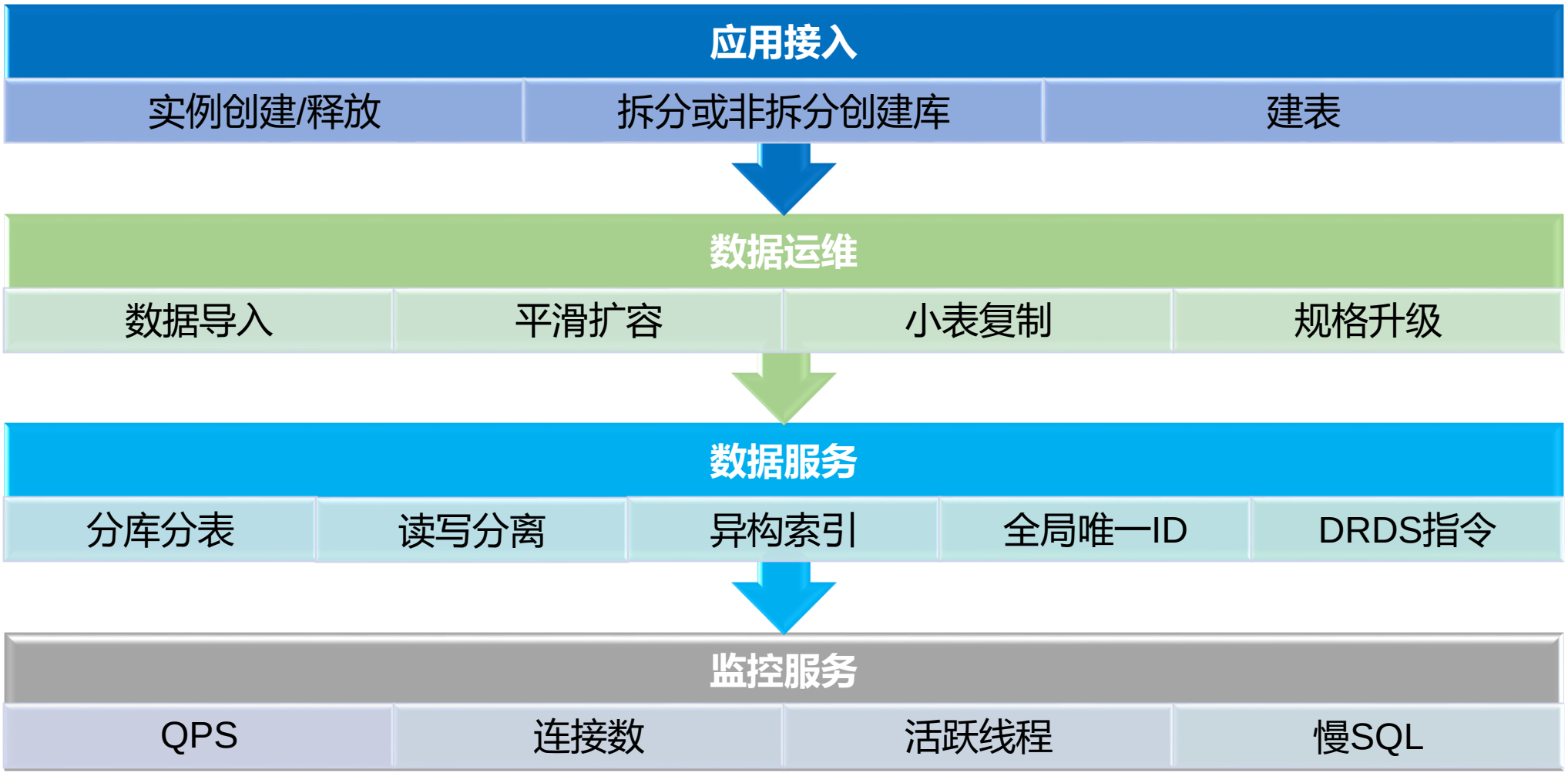
DRDS MANAGER

EasyDB

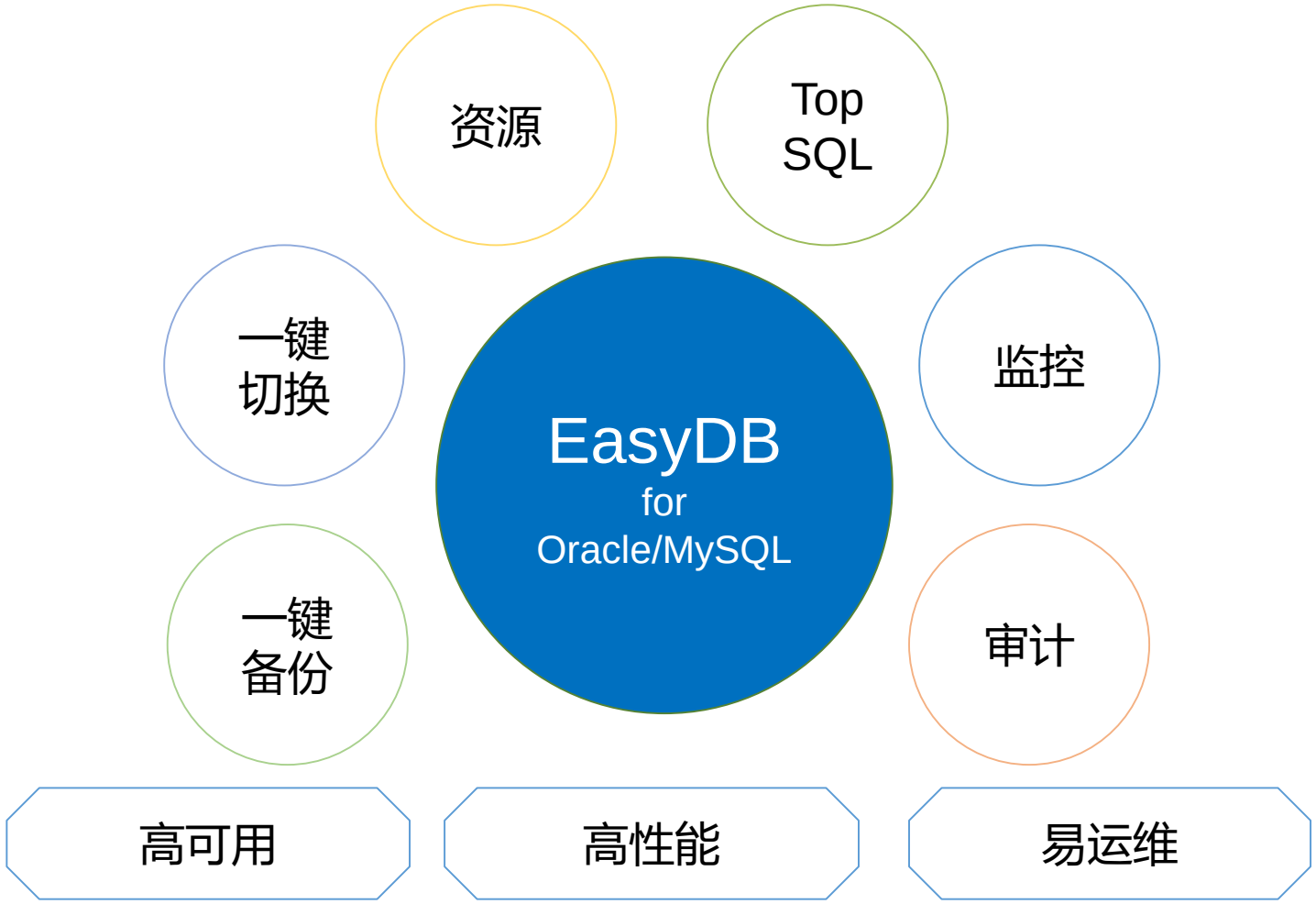
DRDS MANAGER提供DRDS逻辑数据库相关的运维

EasyDB提供MySQL物理数据库相关的运维

DRDS manager



EasyDB：数据库自动化管理平台



典型客户：特步、贵州交警、GQY、比亚迪、陆金所等

核心功能：

资源：实时监控数据库和服务器空间的使用状态。

高可用：云上高可用架构设计，故障自动切换。

备份：定期的数据库全量，增量备份，可灵活配置。

监控：异常情况下自动捕获和告警，支持短信，邮件和微信通知。

性能：超过50个指标性能趋势和SQL采集，实时监控数据库的运行状态。

日志：数据库错误日志采集。

安全：数据库账号和操作的审计，基于服务器的安全设计。

大盘：数据库运维大盘，全方面掌握数据库的运行情况。

袋鼠云 DTSTACK·COM

袋鼠云 DTSTACK·COM | IT大咖说 知识共享平台

数据智能
让未来变成现在