# RDS

关系型数据库在大于一定数据量的情况下性能会急剧下降。在面对互联网海量数据的情况时，所有数据都存于一张表，显然很容易会达到数据表可承受的数据量阈值。

单纯分表虽然可以解决数据量过大导致检索变慢的问题，但无法解决高并发情况下访问同一个库，导致数据库响应变慢的问题。所以通常水平拆分都至少要采用分库的方式，以一并解决大数据量&高并发的问题。

但分表也有不可替代的场景。最常见的分表需求是事务问题。同一个库则不需要考虑分布式事务问题，善于使用同库不同表可有效的避免分布式事务带来的麻烦。目前，强一致性的分布式事务由于性能问题，导致使用起来性能并不一定会比不分库分表快，因此采用最终一致性的分布式事务居多。

# 分库分表

分库分表用于应对当前互联网常见的两个场景：大数据量 & 高并发。通常分为：垂直拆分 & 水平拆分。

## 垂直拆分

垂直拆分是根据**业务**将一个库（表）拆分为多个库（表）。如：将经常和不经常访问的字段拆分至不同的库（表）中，与业务关系密切。

## 水平拆分

水平拆分是根据**分片算法**将一个库（表）拆分为多个库（表）。

# Sharding-JDBC

Sharding-JDBC是当当应用框架ddframe中，从关系型数据库模块dd-rdb中分离出来的数据库水平分片框架，是继dubbox、elastic-job之后ddframe开源的第三个项目。

Sharding-JDBC直接分装jdbc协议，可理解为增强版的JDBC驱动，旧代码迁移成本几乎为零，定位为轻量级java框架，使用客户端直连数据库，以jar包形式提供服务，无proxy层。

主要包括以下特点：

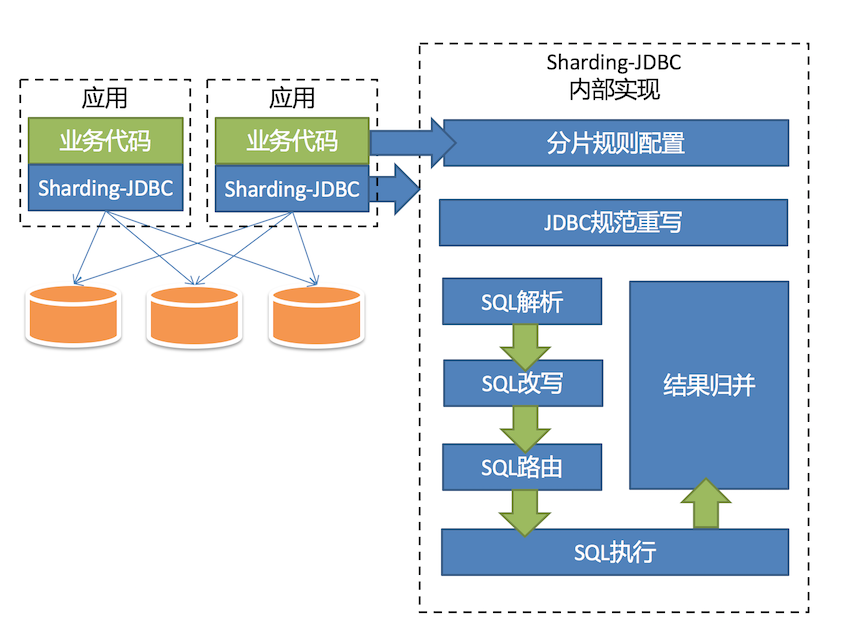
* 可适用于任何基于java的ORM框架，如：JPA、Hibernate、Mybatis、Spring JDBC Template，或直接使用JDBC
* 可基于任何第三方的数据库连接池，如：DBCP、C3P0、Durid等
* 理论上可支持任意实现JDBC规范的数据库。目前仅支持mysql
* 分片策略灵活，可支持等号、between、in等多维度分片，也可支持多分片键。
* SQL解析功能完善，支持聚合、分组、排序、limit、or等查询，并支持Binding Table以及笛卡尔积表查询。
* 性能高，单库查询QPS为原生JDBC的99.8%，双库查询QPS比单库增加94%。

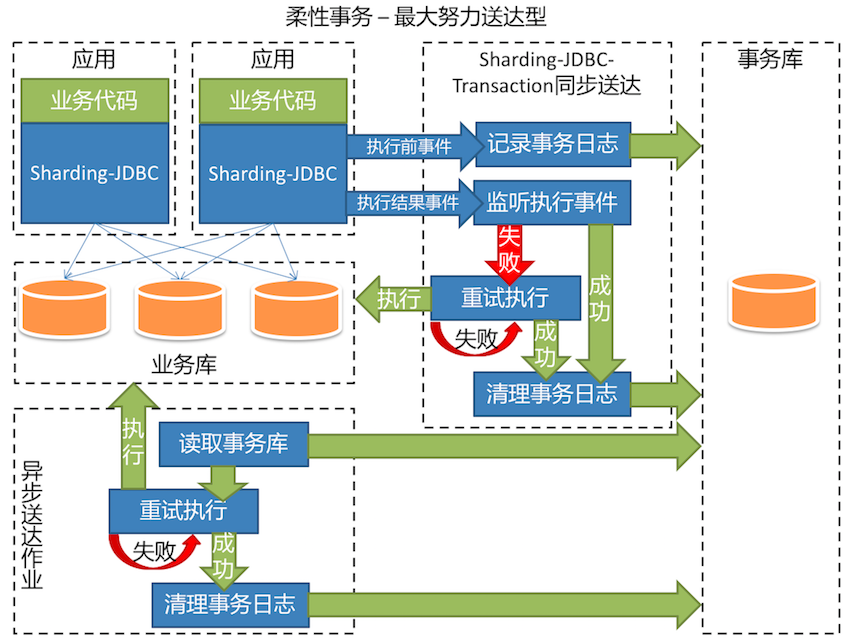
## 链接

GitHub：<https://github.com/dangdangdotcom/sharding-jdbc>

doc：<http://dangdangdotcom.github.io/sharding-jdbc/00-overview/>

## 架构





## 分片规则配置

## JDBC规范重写

## SQL解析

## SQL改写

## SQL路由

## SQL执行

## 结果归并