数据库的这些性能优化，你做了吗？读写分离，分库分表

（点击上方公众号，可快速关注）

> 作者：王奎，个人公号：不止思考

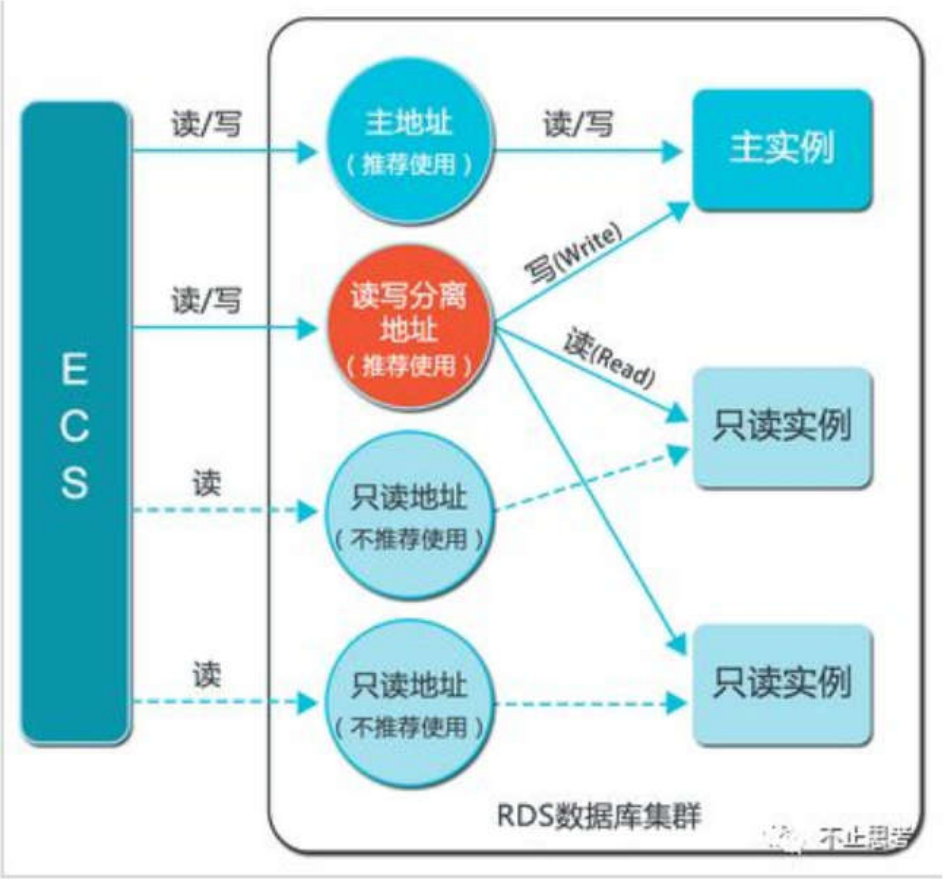
在互联网项目中，当业务规模越来越大，数据也越来越多，随之而来的就是数据库压力会越来越大。

我们可能会采取各种方式去优化，比如之前文章提到的缓存方案，SQL优化等等，除了这些方式以外，这里再分享几个针对数据库优化的常规手段：**「数据读写分离」与「数据库Sharding」**。这两点基本上是**大中型互联网项目中应用的非常普遍的方案**了。

下面我们来详细看一看，

一、从读写分离到CQRS

一、从读写分离到CQRS



由于互联网业务场景，大多数是读多写少，因此进行数据库的读写分离是一件非常简单且有效率的方案。

读写分离简单点来说就是把对数据的读操作和写操作进行分开来，让这两种操作去访问不同的数据库，这样的话，就可以减轻数据库的压力了。

例如上图中，数据库会有一个「主实例」，这个主要用来提供写操作的（偶尔也会承担一点读操作），除了「主实例」以外，还会有多个「从实例」（在图中显示的是 只读实例），「从实例」的功能只是用来承担读操作的。

那上面就出现了多个数据库了，**在多个数据库之间的数据是怎么保证一致性的呢?**

其实，我们常用的数据库就**自带这类同步功能**，比如 Mysql，它自己有一个master-slave功能，可以实现主库与从库数据的自动同步，是基于二进制日志复制来实现的。在主库进行的写操作，会形成二进制日志，然后Mysql会把这个日志异步的同步到从库上，从库再自动执行一遍这个二进制日志，那么数据就跟主库一致了。

除了Mysql以外，像Oracle等商业数据库都有类似的功能，甚至是网络上还有很多开源的第三方数据同步工具，也有很多成熟好用的。

好了，「主实例」与「从实例」之间的数据同步问题解决了，那现在还有一个问题就是，**项目中是怎样让 写请求 去访问「主实例」，让 读请求 去访问「从实例」的，这个路由规则是怎么实现的呢?**

常规的有2种方式：

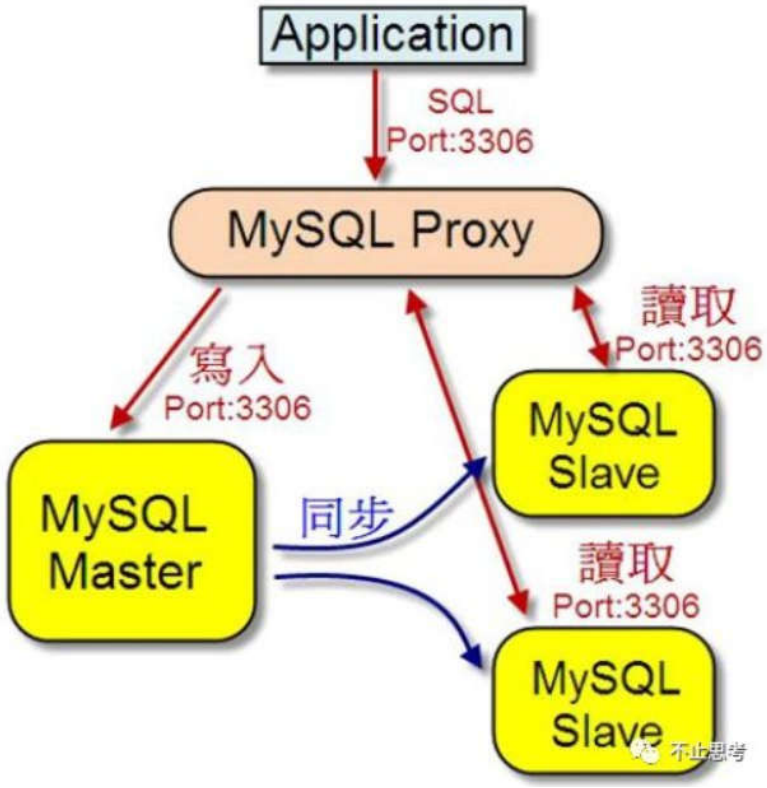
使用编码方式

这个方式主要是靠开发同学在编码的时候，**根据读写不同的操作需求，去调用不同的数据源。**例如在数据操作层（DAO层）将读数据与写数据分开为两个方法（函数），然后为这两个方法分别指定不同的数据库即可。

但是这种方式有点硬编码的味道了，而且对开发同学而言还得额外关注这个事情，多了一个编码成本且容易不小心忽略掉。

使用中间件

这种方式就是在后端数据库的前面，前置一个 数据库代理服务，如下图的：MySQL-Proxy 是Mysql提供的一个中间件，用于实现读写分离请求，但这个组件实际用的人不多，我们可以选择其它的一些开源的组件替代，例如：MyCat、ProxySQL 等等，但大致的原理比较类似，通过这个图很容易理解这个模式。

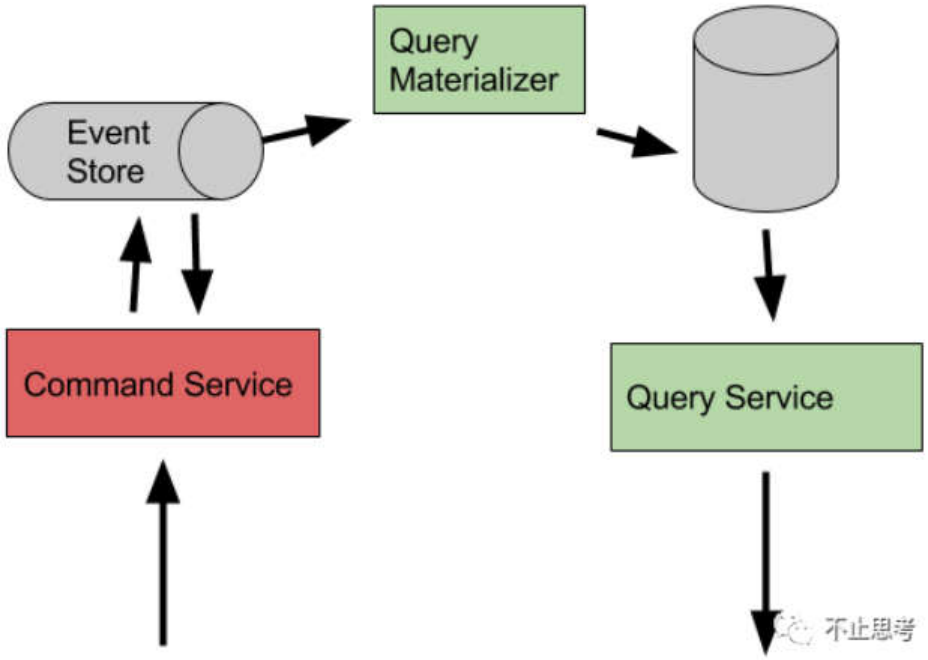


好了，基础的读写分离就讲完了，但感觉这个方式虽然实用是实用，就是不怎么有逼格。

OK，想要有逼格是吧，满足你，那我们就来聊一聊**另一个有逼格的读写分离概念：「 CQRS 」**

CQRS：Command Query Responsibility Segregation

命令（增删改）和查询的责任分离



我们还是先看图，通过上图可以简单的理解一下CQRS

CQRS 重点强调的就是 Query（读） 和 Command（写）的分离，在业务上将职责分离清晰，**Command 主要做业务逻辑的执行，Query来负责数据查询和展示。**同时 这两种操作是**基于不同的数据源，**甚至是一个是数据库，另外一个是NoSQL都可以，Query去查询的数据源可以直接按照领域模型进行存储，而并不是按照数据模型去存储，这样查询出来就立即可以展示，而不用转换，且查询效率高。

其实CQRS是由鼎鼎大名的 Martin Fowler 提出，搞计算机的应该都认识。想要更深入的去学习CQRS，可以翻看Martin Fowler公开的资料。

二、Sharding（分库分表）

上面讲完了**数据库的读写分离**，现在我们来聊一下数据库的Sharding。

随着数据库里的数据越来越大，单表查询的性能已经不能满足业务要求了，这个时候就需要进行分表处理了，将大表拆分为若干个小表，不同的分表中数据也不一样，这样可以分散查询压力，提高处理效率。

然而，当表越来越多，所有的数据都在一个数据库上时，网络IO以及文件IO也都会集中在一个数据库上，可能会超过单台服务器的容量， CPU、内存、文件IO、网络IO 都会成为系统的瓶颈，QPS/TPS也会超过单数据库实例的处理极限。那么这个时候就需要对数据库进行分片处理。

因为分表和分库的思路类似，因此下面统一来聊技术方案。

其实**分库分表**只是我们通俗的便于理解的说话，**正确的描述应该是：数据分片**

数据的分片主要有2种模式：

- 垂直拆分
- 水平拆分

两种拆分应用的场景是不同的：

垂直拆分，是指按照业务模块进行拆分。简单来讲，就是把业务紧密的模块的字段/表放在一起，放在同一个数据库或者服务器上。将不同业务的字段/表进行独立，拆到不同的数据库或者服务器上。比如一个游戏系统中，可以将玩家基本信息与道具公会等信息进行拆分。

如图示例：

（图片来源网络）

水平拆分，是指纯粹的按照某种数据规则/格式进行拆分。例如 按照数**据唯一ID的哈希散列拆分**、**按照数据的日期拆分**、**按照某种范围拆分**等等。水平拆分需要注意的是，随着数据动态的变化，分片数量可能需要随之动态调整，另外就是水平分片是没有考虑业务特征的，因此在进行业务汇总查询或者分片中事物处理的时候就比较麻烦一些。

如图示例：

另外，在实际应用中，两种拆分模式一般会结合在一起使用，效果更佳。

以上就是数据库性能优化之「数据读写分离」与「数据库Sharding」方法，欢迎大家一起交流。