数据库瓶颈：

IO 瓶颈：

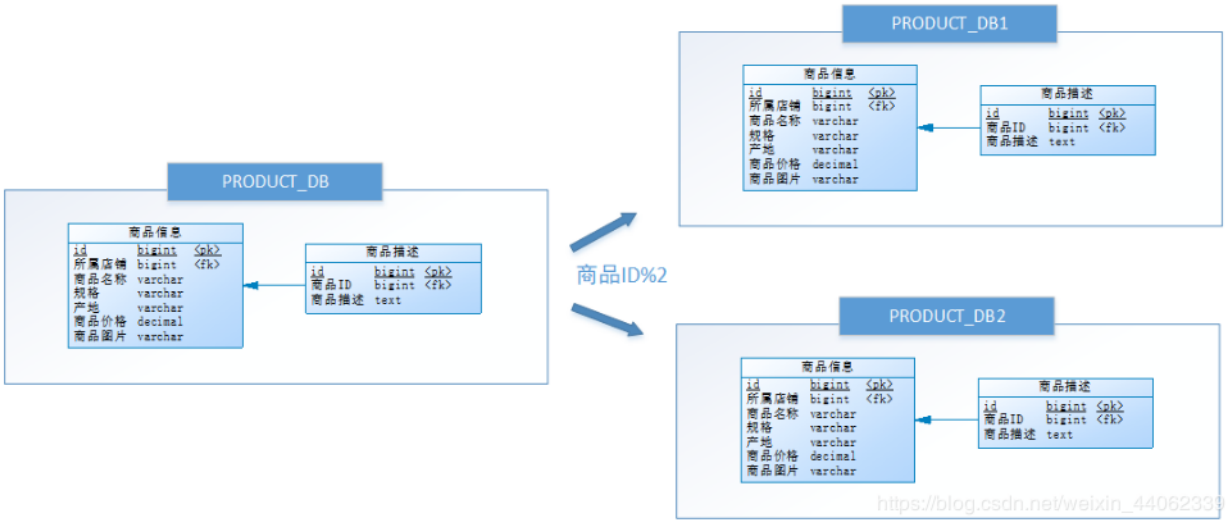
- 第一种：磁盘读 IO 瓶颈，热点数据太多，数据库缓存放不下，每次查询会产生大量的 IO，降低查询速度→分库和垂直分表。
- 第二种：网络 IO 瓶颈，请求的数据太多，网络带宽不够→分库。

CPU 瓶颈：

- 第一种：SQL 问题：如 SQL 中包含 join，group by，order by，非索引字段条件查询等，增加 CPU 运算的操作→SQL 优化，建立合适的索引，在业务 Service 层进行业务计算。
- 第二种：单表数据量太大，查询时扫描的行太多，SQL 效率低，增加 CPU 运算的操作→水平分表。

分库分表：

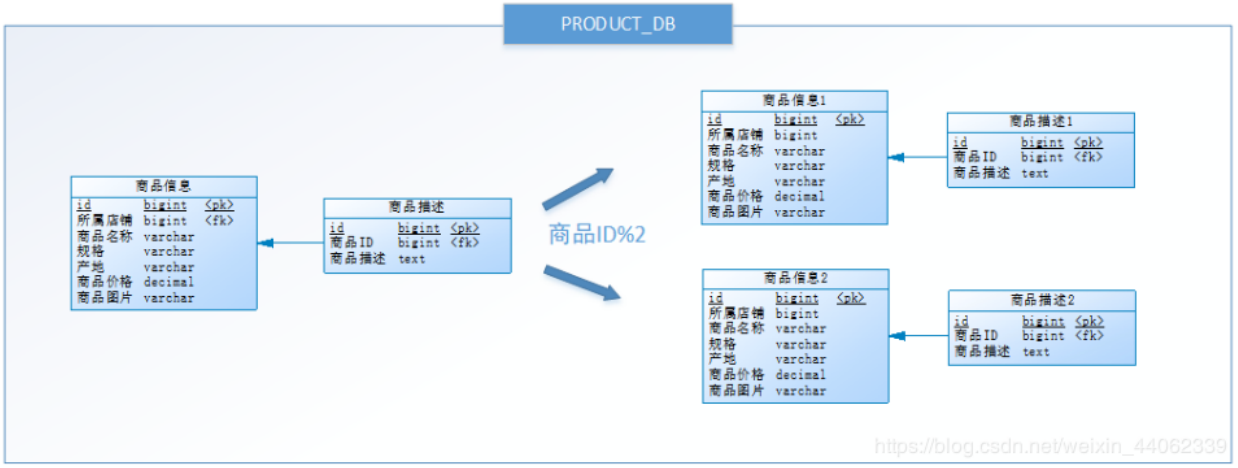
1. 水平分库：以字段为依据，按照一定策略（hash、range等），将一个库中的数据拆分到多个库中。



店铺ID为单数的和店铺ID为双数的商品信息分别放在两个库中。

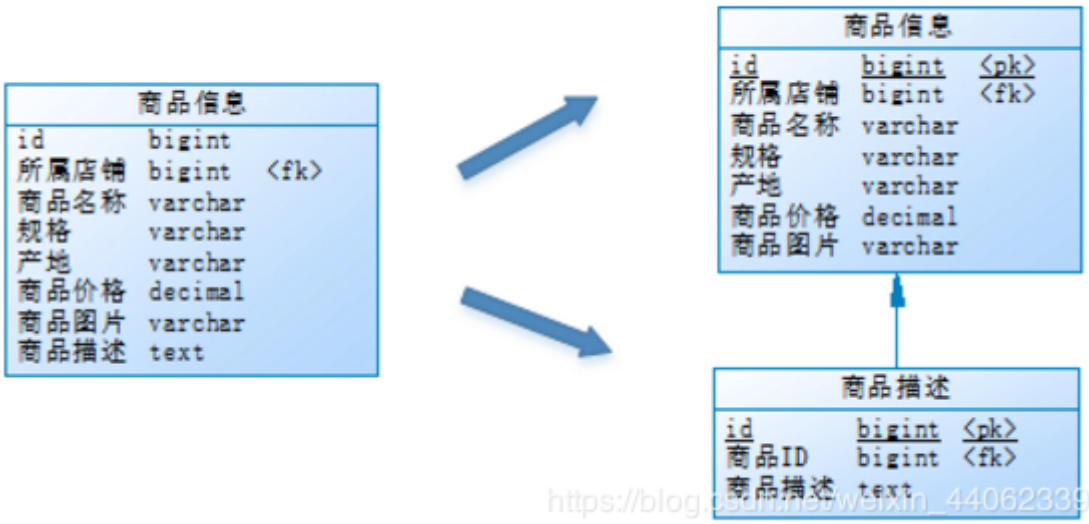
水平分库是把同一个表的数据按一定规则拆到不同的数据库中，每个库可以放在不同的服务器上。

2. 水平分表：以字段为依据，按照一定策略（hash、range 等），讲一个表中的数据拆分到多个表中。



水平分表是在同一个数据库内，把同一个表的数据按一定规则拆到多个表中。

3. 垂直分表：以字段为依据，按照字段的活跃性，将表中字段拆到不同的表中（主表和扩展表）。



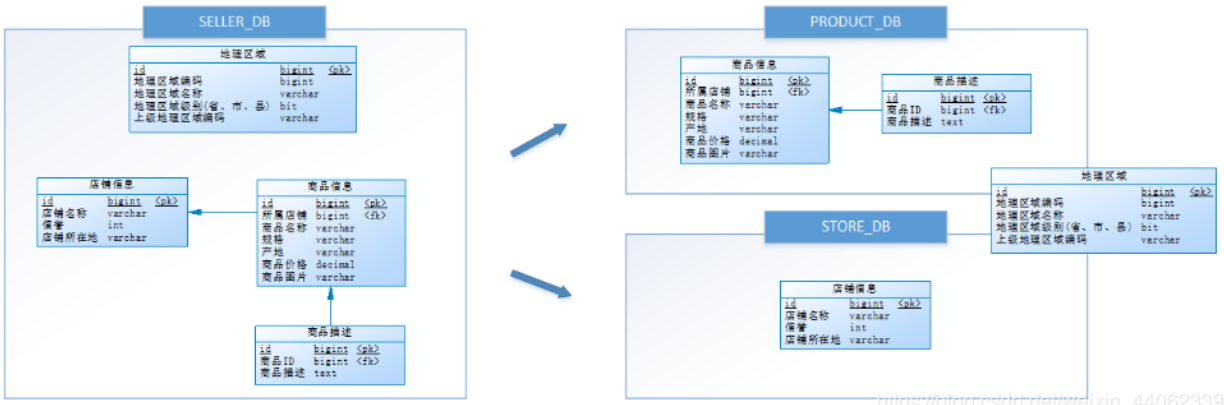
用户在浏览商品列表时，只有对某商品感兴趣时才会查看该商品的详细描述。因此，商品信息中商品描述字段访问频次较低，且该字段存储占用空间较大，访问单个数据IO时间较长；商品信息中商品名称、商品图片、商品价格等其他字段数据访问频次较高。

由于这两种数据的特性不一样，因此他考虑将商品信息表拆分如下：

将访问频次低的商品描述信息单独存放在一张表中，访问频次较高的商品基本信息单独放在一张表中。

缺点：数据还是始终限制在一台服务器，库内垂直分表只解决了单一表数据量过大的问题，但没有将表分布到不同的服务器上，因此每个表还是竞争同一个物理机的CPU、内存、网络IO、磁盘。

4. 垂直分库：以表为依据，按照业务归属不同，将不同的表拆分到不同的库中。



商品信息与商品描述业务耦合度较高，因此一起被存放在PRODUCT_DB(商品库)；而店铺信息相对独立，因此单独被存放在STORE_DB(店铺库)。

总结

垂直分表：可以把一个宽表的字段按访问频次、是否是大字段的原则拆分为多个表，这样既能使业务清晰，还能提升部分性能。拆分后，尽量从业务角度避免联查，否则性能方面将得不偿失。

垂直分库：可以把多个表按业务耦合松紧归类，分别存放在不同的库，这些库可以分布在不同服务器，从而使访问压力被多服务器负载，大大提升性能，同时能提高整体架构的业务清晰度，不同的业务库可根据自身情况定制优化方案。但是它需要解决跨库带来的所有复杂问题。

水平分库：可以把一个表的数据(按数据行)分到多个不同的库，每个库只有这个表的部分数据，这些库可以分布在不同服务器，从而使访问压力被多服务器负载，大大提升性能。它不仅需要解决跨库带来的所有复杂问题，还要解决数据路由的问题(数据路由问题后边介绍)。

水平分表：可以把一个表的数据(按数据行)分到多个同一个数据库的多张表中，每个表只有这个表的部分数据，这样做能小幅提升性能，它仅仅作为水平分库的一个补充优化。

一般来说，在系统设计阶段就应该根据业务耦合松紧来确定垂直分库，垂直分表方案，在数据量及访问压力不是特别大的情况，首先考虑缓存、读写分离、索引技术等方案。若数据量极大，且持续增长，再考虑水平分库水平分表方案。

分库分表中间件：

- **Sharding-JDBC（当当）**
- **TSharding（蘑菇街）**
- **Atlas（奇虎 360）**
- **Cobar（阿里巴巴）**
- **MyCAT（基于 Cobar）**
- **Oceanus（58 同城）**
- **Vitess（谷歌） 各种工具的利弊自查**

sharding-jdbc和mycat使用不同的理念，sharding-jdbc目前是基于jdbc驱动，无需额外的proxy，因此也无需关注proxy本身的高可用。Mycat 是基于 Proxy，它复写了 MySQL 协议，将 Mycat Server 伪装成一个 MySQL 数据库，而 Sharding-JDBC 是基于 JDBC 接口的扩展，是以 jar 包的形式提供轻量级服务的。

分库分表带来的问题

[文档详情]

Mycat

MyCat是目前最流行的基于Java语言编写的数据库中间件，是一个实现了MySQL协议的服务器，前端用户可以把它看作是一个数据库代理，用MySQL客户端工具和命令行访问，而其后端可以用MySQL原生协议与多个MySQL服务器通信，也可以用JDBC协议与大多数主流数据库服务器通信，其核心功能是分库分表。配合数据库的主从模式还可实现读写分离。

MyCat是基于阿里开源的Cobar产品而研发，Cobar的稳定性、可靠性、优秀的架构和性能以及众多成熟的使用案例使得MyCat变得非常的强大。

原理 ：原理中最重要的一个动词是“拦截”，它拦截了用户发送过来的SQL 语句，首先对SQL 语句做了一些特定的分析：如分片分析、路由分析、读写分离分析、缓存分析等，然后将此SQL 发往后端的真实数据库，并将返回的结果做适当的处理，最终再返回给用户。

安装：官网下载解压即可

常用命令：

可以使用如下命令启动mycat服务：mycat.bat start

启动后可以通过如下命令查看mycat的运行状态：mycat.bat status

可以使用如下命令停止mycat服务：mycat.bat stop

重启服务命令：mycat.bat restart

配置：schema.xml配置文件是mycat中重要的配置文件之一，它涵盖了mycat的逻辑库、表、分片规则、分批按节点及数据源。点击查看

Mycat 配置文件server.xml

[Mycat 配置文件schema.xml]

[Mycat 配置文件rule.xml]

表分类：

逻辑表：逻辑表，可以是数据切分后，分布在一个或多个分片库中，也可以不做数据切分，不分片，只有一个表构成

分片表：是指那些原有的很大数据的表，需要切分到多个数据库的表，这样，每个分片都有一部分数据，所有分片构成了完整的数据。

ER表：所有子表和主表在一个库，避免跨库查询

全局表：数据量不多的公共表所有库的保存，通过数据冗余解决跨库查询

读写分离：配置文件schema.xml里的writeHost内添加一个readHost