01 | 为什么不同类型的业务后台架构模式是通用的?

从本讲开始，咱们将一起开启本专栏的学习之旅。

国内的各大互联网公司业务模式非常丰富，所提供的业务服务也是形态各异，比如：

- 腾讯主要提供即时通信、游戏等服务
- 京东、阿里等电商提供商品购买、快递寄收件、金融投资等服务
- 滴滴提供打车服务
- 今日头条、新浪微博提供短视频、新闻资讯类阅读服务
- 美团提供商品购买、外卖订购服务
- 百度提供搜索查询服务

虽然上述列举的公司业务类型不同，但对后台开发岗位的招聘要求却很相似。在拉勾网上可以看到，除了要求你掌握某种开发语言和相关框架外，还要掌握分布式、多线程、缓存、数据库等。

（以上信息来源拉勾网）

那么，产生这个现象的根本原因是什么呢?

简单来说，是因为这些业务在技术实现上存在共性，比如技术点或者架构模式。在本讲我将和你一起探寻问题背后的原因，并向你提供一个分析业务架构共性的标准。学完本讲，**希望你在面对层出不穷的新业务、新模式时，能够洞穿出背后的架构本质，套用通用架构模式，轻松应对各项技术指标，真正做到大道归宗**。

什么是业务后台系统

在讨论一个议题之前，最重要的是确定它的定义和边界，这样才不至于在实施时出现偏差，导致事倍功半。因此，在开始本讲前，咱们先聊聊业务后台系统的定义。

乍一听“**业务后台系统**”这个称呼，你可能有点摸不着头脑，不知道它是什么。其实，业务后台系统和你在拉勾网上看到的“**后台开发**”大体上是一个意思。我将以外卖订购服务为例，帮助你理解和明确它的定义和边界。

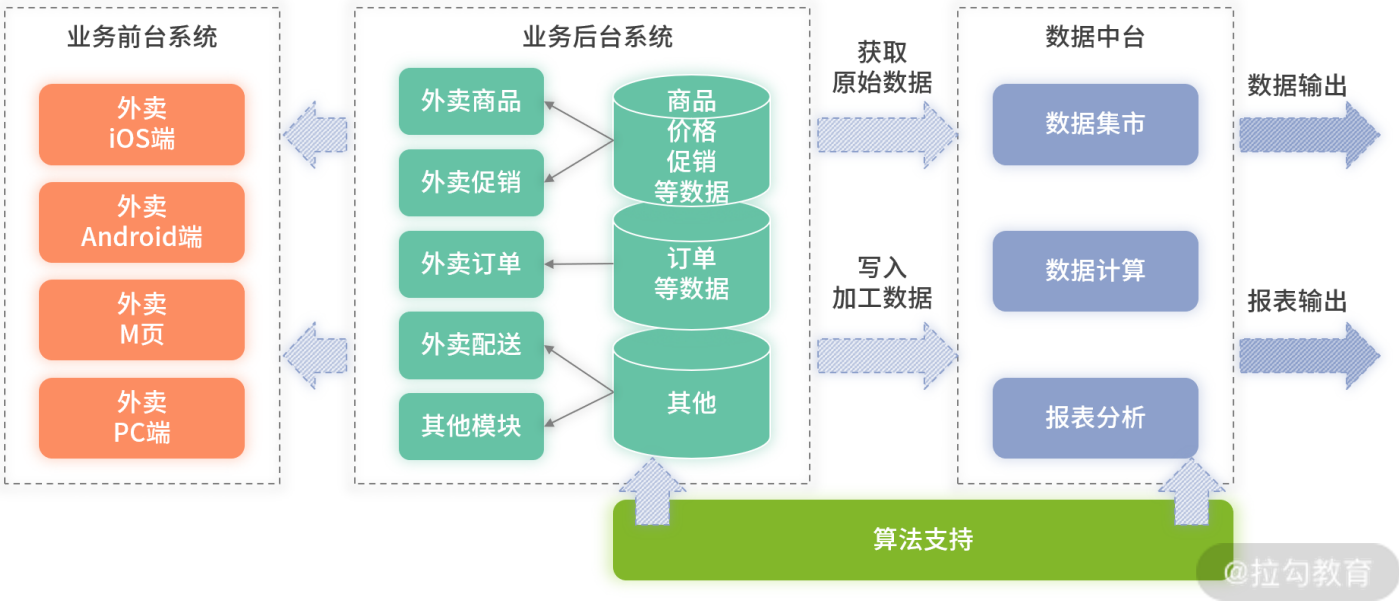

图 1：外卖系统全局架构

图 1 展示的是一个外卖系统全局架构，如果某公司要进军外卖业务，就需要开发一套外卖系统，其中包含用户可直接使用的各个终端，如 iOS 端、Android 端、M 页及 PC 站点。提供内容展示和系统交互，称为**业务前台系统**。

业务前台系统中展示的内容，比如外卖商品种类、商品图片等信息，其实是由业务后台系统提供的，业务前台并没有这些数据，它只负责展示。当用户在前台选择商品点外卖后，实际接受并存储订单及调度配送的系统称为**业务后台系统**。

总的来看，业务后台系统是指直接接受前台的请求，同时给前台返回数据或者保存前台数据的系统。“业务”这个词只是个代称，代指各类业务系统。

很多公司有一些大数据、BI、数据挖掘等相关的系统，它们并不属于业务后台系统，可以称为业务大数据系统，现在时髦的叫法是：**数据中台**。数据中台有一个特点，不直接接受业务前台系统的请求也不直接生产数据，而是直接对业务后台系统产生的数据进行分析、再加工等。

还有一些算法研究的岗位，他们主要对算法进行研究和调优，并将这些成熟的、可用于生产环境的算法提供给业务后台开发工程师。再由他们集成进业务后台系统里，提升业务的体验，比如推荐算法、语音识别算法等。

至此，你应该对业务后台系统的边界有一个比较清晰的了解。**通过上述案例，我们还可以举一反三，比如**：

- 在短视频和资讯类（微博、新闻）业务里，提供查询视频基本信息和资讯的系统、能够保存短视频的系统，都属于后台系统。
- 在电商业务里，提供查询商品信息的系统、保存订单的系统，都属于后台系统。

其他互联网公司提供的系统，你可以按此思路来判断它们的业务后台系统类别。

后台系统的共性探究与归类

在探讨不同业务后台系统的共性前，需要明确一个非常重要的概念，即目的性。如果不明确此概念，你会发现不同的业务后台系统的所有技术实现都是 CRUD（增、查、改、删），按此归类后，很难看出各类业务后台系统之间的区别，更别说从中提炼共性技术和通用架构模式了。

因此，**我们首先要确定归类的维度。具体来说，就是需要寻找一个新的维度来对业务后台系统进行归类，并基于此归类提炼技术共性**。在我看来，这个维度就是目的性。在确定归类的维度后，就可以对各类业务后台进行分类了。

接下来，我会以资讯、发布及购买、库存及支付这三类业务为案例，讲解如何基于目的性进行归类。

资讯类业务

下面请你思考一个问题：对于微博、知乎等类资讯业务系统，它们的目的性是什么？

你可以从日常使用的角度来思考，显而易见，这类 App 主要为用户提供阅读和浏览信息的业务，这就是资讯业务系统的目的性。

比如你平时上微博和知乎的目的是娱乐或者学习，我们总结为阅读“新鲜事”。当然，你偶尔也会发布几条动态。想象一下，如果系统出现 Bug，导致你曾经发布的某条消息丢失，当下你可能感知不到，除非某一天你去查看历史动态。但如果系统异常导致不能阅读“新鲜事”，用户当下即可感知到系统出现故障。而互联网时代舆论传播非常快，这会给公司带来很大的负面影响。因此这类事故产生的影响更大。

我们总结一下，资讯类业务系统的主要目的性是尽最大的可能性保证读的可用性和优化用户体验。根据最终目标，我们称它为读类型的业务后台系统，或者叫读业务。

到这里，你可能会有疑问，读业务和 CRUD 里的 R 不是一个意思吗？答案是：完全不同。它们主要有 2 点区别：

1. 这里的“读”是从目的性推导而来，是有限定要求的，它要求的“读”是能够满足用户体验的高性能及高可用；
2. 定义的面不同，读业务最重要的是保证系统可读，但此系统仍然会提供写删修的功能，但对这些功能的关注度和要求指标均较低。

和上述读业务类似的场景有：

- **短视频业务**，在系统出现故障的极端情况下，可以不能发送新的短视频，但需要能够浏览历史视频；
- **电商或者外卖业务里的商品系统**，在系统出现故障的极端情况下，商家可以不能创建新的商品，但历史商品需要能够被客户浏览并下单。

可以看到，上述第二个场景是某个业务中的一个系统，而第一个场景短视频则是一个大的业务。从这两个差异的场景案例可以看出，**目的性这个衡量指标可以是某一个大的业务，也可以是某一个业务中的一个具体细小场景**。

除了读业务之外，还有另外两个场景，分别是**数据写场景和扣减场景**。

发布及购买类业务

这里请你思考一个问题：对于电商、外卖和打车等交易类的业务场景，它们最重要的目的性是什么？

不管这些业务里有多少形形色色的系统，当出现一些难以恢复的故障时，比如钻石会员不能使用优先打车通道、不能显示此次打的是出租车还是快车等，但只要用户能够提交订单打车即可。因为如果不能提交订单，将直接减少企业真金白银的收入，在商业上是不允许的。**因此提单的写入是此类业务场景中的重中之重，也是提供电商、外卖和打车服务的企业的最终目的。所以提供一个高可用的写入服务十分重要**。

库存及支付类业务

最后，我们再来聊聊库存及支付类业务的目的性。

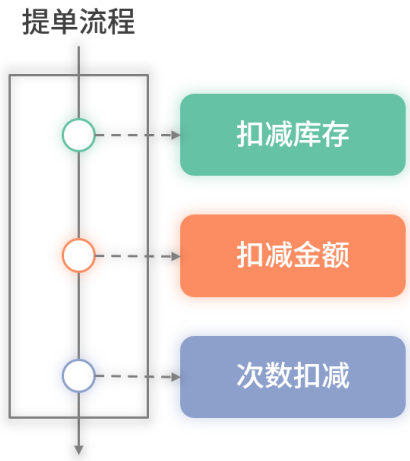

图 2：提单扣减流程

图 2 展示的是扣减场景中的提单扣减流程，属于一个大型业务之下的某个系统的技术诉求，比如库存的扣减、次数的限制、支付金额的扣减等。虽然这几个系统都会对外提供诸如查询库存、次数等能力，**但它最重要的是保障扣减的高可用，因为一般扣减都是和提单共同发生。如果扣减失败，那么提单也无法成功，所以，扣减业务也是一个需要重点保障的场景**。

以上，我们使用目的性这个维度对不同类型的公司业务（短视频、微博、新闻资讯、电商、打车等）梳理分类，得出结论 1：

业务后台系统在系统实现上均可分为读业务、写业务、扣减业务。

因为业务类型是可归类的、通用的，所以得出结论 2：

这三大类业务后台系统在技术实现上也是类似的，甚至可以说是统一的。

各类型的技术实现关注点

通过将形态各异的业务分成三大类，你应该能够解答本讲开篇提出的问题：为什么不同公司的业务后台开发岗的招聘要求有很多重叠？可以把你的思考和答案写在留言区，再来看我接下来的分析。

因为很多公司的业务或者其中的某一个系统，都归属于同一大类，而这些类别的技术实现基本上大同小异，因此所有招聘或者任职要求都是类似的。

上一小节，我从业务场景的角度介绍了读业务、写业务、扣减业务，接下来我们再来看看这三大类场景在技术实现上各自有哪些要求。

读业务是越快越好

首先介绍的是读业务场景。任何业务最基本的要求是高可用，随时保障服务可用。那么读业务除了此要求之外还有其他什么要求吗？

从上述的几个案例中，你会发现资讯类业务（微博、知乎、短视频），它们的“写”即发微博、发短视频，和用户的“读”即浏览新鲜事的次数相差非差大。一个正常的用户，可能阅读了 100 条微博，才会发 1 条微博。这里的读写比例在十倍、百倍的量级，因此读的并发量级非常大。

另外，阅读作为一切业务发起的起点，对于速度的要求至关重要。不管是电商还是现在短视频、微博里的直播带货，首先要保障用户能够快速浏览和切换商品，然后才是进入下一步的购买页面。你可以想象一下，如果一个商品图片加载很慢，或一个资讯类应用新闻半天不展示，你还会耐心等待吗？

因此，作为大多数业务的起点，除了完成高可用外，读业务的实现还要求能够在海量读请求下保障高性能。

写业务需要 101% 高可用

在上一小节，我们在读业务的技术实现分析里提到，保障高可用是基本要求。那么，写入业务如提交订单等场景，还需要再提及高可用吗？

答案当然是要，此小节的标题我用了一个夸张的写法“101% 高可用”。我在本讲写入的场景介绍里提到，写入基本上是提交订单，它和实实在在的企业收入相关。因此，我们需要尽“101%” 的努力去保障可用性。

如果读服务出现存储或应用故障，可以在前端或者终端进行前置缓存抗一段时间。缓存给研发或者运维提供了分钟级别的故障处理、数据修复的可能。

但写入服务是无法使用缓存的。此外，对于各大电商、打车、外卖平台来说，故障恢复的时间需要在一分钟内或者秒级别。故我们在架构设计时需要做到“101%”的高可用，这样在实际生产环境才能高效应对故障的发生。

扣减业务要抗住并发和保障数据一致性

对于扣减业务，从目的性上讲，最重要的就是抗住并发的扣减量。除开高可用外，你会觉得扣减和写入有很多类似的地方，甚至可以归为一类。

对于写入业务，以提交订单为例，在写入的时候，所有的数据都是用户从表单里提交过来的，比如购买商品的名称和数量、收货地址等，这些数据是这个用户私有的。在技术实现上，我们只要能够尽“101%”的可能性把它保存下来即可。

而对于扣减则不是，以库存为例，扣减的请求只会包含购买的商品和对应的数量，而具体能不能买，则依赖后台系统存储的当前剩余库存数量。另外，**不同用户在同一时刻可能购买同一个商品，此处就存在并发更新，这种在并发情况下的扣减一致性需要格外注意**。

虽然，扣减类业务也会对外提供一些诸如剩余数量和金额的查询接口，但扣减类系统提供的最重要的能力是被各类订单所依赖的扣减接口。它的稳定性决定了提交订单的稳定性。因此，**扣减类业务的重点就是在并发情况下保障扣减的准确性和抗击高并发的能力**。

总结

在本讲，我向你提供了一个对形态各异的业务模式归类的方法。不管各大公司提供的具体业务是什么，从技术的目的性上看，它们都是提供了三大类技术角度的业务。这也就解答了，为什么各大公司的后台开发岗位的招聘要求是类似的，穿过业务的外皮，它们底层逻辑相同。

另外，有一个点需要你注意，本讲对各公司大的业务层面进行了划分，相信你已经对谁属于读业务、谁又属于写业务有了通透的理解。此时不妨尝试问自己一个问题：是不是经过上述定性后，就可以认为某个公司的系统都是某一特性的业务了（不是读业务就是写业务）？

其实不然，如订单系统，从大的目的性来看它属于写业务。但进一步剖析，会发现它对外提供的订单详情和列表是读业务。**本讲的案例只是在一个比较高的维度对各大公司进行了划分，你可以继续利用目的性对它们做进一步的划分**。

通过不断地划分，你可以将你负责的系统与本讲介绍的三大类业务进行对应，并将你现在负责的业务架构和本专栏后续讲解的技术架构进行对照，有针对性地学习。

最后，给你留一道思考题。**你所负责或者了解的属于这三大类型的业务形态有哪些？可以试着举一些例子写在留言区，咱们一起讨论吧**。