通常让大家去判断一个区块链适用的应用场景，往往都会感到无从下手。本篇章就以此出发，为大家提供思路和方法，帮助你建立分析区块链应用场景的能力。

众所周知，区块链本质上就是一个去中心化的数据库，使用数据库的各方都可以存储交易数据。需要注意的是，这里的交易指的是广义的交易，并不局限于货币和金融的交易，一切会产生数据状态变化的事务都称之为交易，例如，创建账户，修改密码等等。

前面提到了区块链可以简单理解为一个分布式的账本，那么相比于传统的中心化数据库，两者有什么区别呢？

【插图】

<https://www.quora.com/Whats-the-difference-between-blockchain-and-a-database>

<https://www.coindesk.com/information/what-is-the-difference-blockchain-and-database>

区块链会永久保留历史数据

任意一笔交易不会覆盖区块链账本上已经存在的记录，只会将新的交易信息链接到最尾端。

而数据库既可以添加新的内容，同样也可以删除、更新原来的信息。这里补充解释下超级账本中的世界状态（world state）。因为世界状态只存储最新的账本信息，有些人会觉得这是不是和我们刚提到的概念冲突。其实不然，世界状态独立于区块链自身存储空间，引入世界状态主要方便快速获取账本的当前状态值，我们不再需要从创世区块开始遍历整个区块链，你可以直接从世界状态获取。比如说，原本你的钱包有100元，世界状态里存储的钱包余额同样也是100元。当某天你消费了10元，世界状态里的余额就变成了90元，不过区块链账本中100元的记录依旧存在，这时会添加一条新的交易信息记录消费10元。为了提供简单有效的存储，可回溯的账本状态，世界状态当前采用数据库的实现方式。

区块链保证数据的完整性

虽然传统的数据库安全性也是有保障的，但是不排除黑客攻击或者使用者的误操作，会导致数据内容的丢失，篡改。然而区块链是有完美的机制保证数据的完整性，每个网络节点都存储区块链账本的信息，纵使某些节点不甚被黑客攻击，整个区块链网络依旧可以运转正常，

因为黑客不可能攻破区块链网络中遍布全世界超过半数的节点，唯一的账本信息还是会遵循多数节点所存储的内容，而这是传统数据库无法保证的。

接下来我们讨论区块链的应用场景，下图是一个较为严格的区块链应用判断流程，目的是给大家提供一个快速识别区块链应用的方法，但很多场景下这些约束条件可以进一步放宽。下面将对流程中的信息进行逐一简单阐述，方便大家理解。

【插图】

1. 是否储存状态

虽然之前提到了区块链和传统数据库的不同，不少人还是觉得区块链本质上还是一个数据库。不过区块链使用场景只是数据库使用场景中很小的一部分。这里有一个需要讨论的问题是，是否所有的交易数据都需要存储在区块链上？答案是不一定，那么什么样的数据不适合上链以及什么样的数据适合上链？

首先来看看什么样的数据不适合上链。从业务角度看，不需要共享的数据不适合上链。 例如，用户账户密码，这本身就是高度隐私化的数据，相信没有人希望自己的密码有被其他人知晓的机会。尽管区块链加密技术可以保证数据的隐私性，但是也有泄露的可能性，因此不需要共享的数据就没有必要上链。从性能角度讲，过于庞大的数据和更新过于频繁的数据也不适合上链。例如，时钟信息，如果把每秒的时钟信息都记录在区块链上，相信没过多久，区块链的存储空间会变得非常大，这会严重的影响到同步性能。过于庞大的数据也是相同的道理，区块链作为分布式账本意味着，更新的交易数据会被同步到所有的节点，这意味着庞大的数据不仅会占据本地的存储空间，更会影响网络上的所有节点。因此，我们需要谨慎使用区块链的存储空间。

那么什么样的数据适合上链呢？简单来说就是需要共享的、需要具备可信度、不能被篡改并且需要可追溯的数据。例如保险行业的保单信息，用户签署了什么样的保险协议，需要被妥善保存。因为将来一旦出险，存储在区块链上的保单信息将会作为索赔的依据。因为区块链带来的数据不可篡改性，保险公司无法抵赖，用户的权益得以保障。也因为可以共享和追溯，一旦产生纠纷，监管部门、法院也可以追溯取证。

此外，我们需要确保交易数据是可以准确度量，并且数字化存储到数据库中。假使世界环保组织想基于区块链开展一个项目，将区块链节点遍布全球主要国家，每个节点测量自己国家的C02排放量，然后基于区块链的同步机制，大家最终共享全球C02排放量。听起来的确利用到了区块链本身的同步以及信息共享机制，不过该方案可行性有待商榷。因为目前为止，没有一种有效的方式能够精确测量每个国家的C02释放量，也就是说无法找到一种方式关联感兴趣的实物和存储在区块链上的交易信息。如果不能将交易信息准确数字化存储到数据库中，那么使用区块链就甚至传统数据库都没有必要了，这是讨论区块链应用场景的前提条件。

如果满足了该条件，就一定要使用区块链吗？不一定，如果可以用传统的数据库解决，比如学校想要备份期末试卷信息，传统数据库本身就可以很好的解决，那就没必要使用区块链。需要明确的是，区块链不是万能的，不是所有问题的最优解。如果现有的技术已经可以完美解决问题，就完全没有必要刻意使用区块链。区块链网络的搭建维护，本身也消耗很多的资源。接下来，我们会一起探讨在哪些场景下，我们有可能需要使用区块链而传统的数据库是没法实现的。

b. 是否多方协同写入

有人曾经说，区块链的核心就在于去中心话，换种方式说，也就是多方协同写入才能激发区块链的潜力。比如我们常提到的比特币，每一个参与者都有自己的钱包，他们可以在任意时间发起交易参与到区块链网络中。

接着我们探讨一个线上自行车销售商户的例子，店家打算升级系统将客户采购订单信息记录到区块链上以保证数据的完整性。目前，当客户线上订购一款自行车，店家先确认店内是否有这款自行车，并验证客户付款方式的有效性。如果信息确认无误，店家会将新的订单添加到记录中。通常情况下，只有店家才有权访问、添加、修改订单的信息，而客户是无法访问的。如果所有的交易信息都是由店家一方完成的，那么想要通过区块链存储订单信息就没必要了，多方协同写入准则在这里是没有满足的。设想个稍微不同的场景，淘宝电商平台。淘宝上的所有店家在准入之后，都可以自行销售商品，这种场景下多方写入的条件就很好的满足了。那么淘宝网需要搭建个区块链网络记录所有的交易信息嘛？这又牵扯到前面提到的更新过于频繁或者庞大的数据不适合上链。

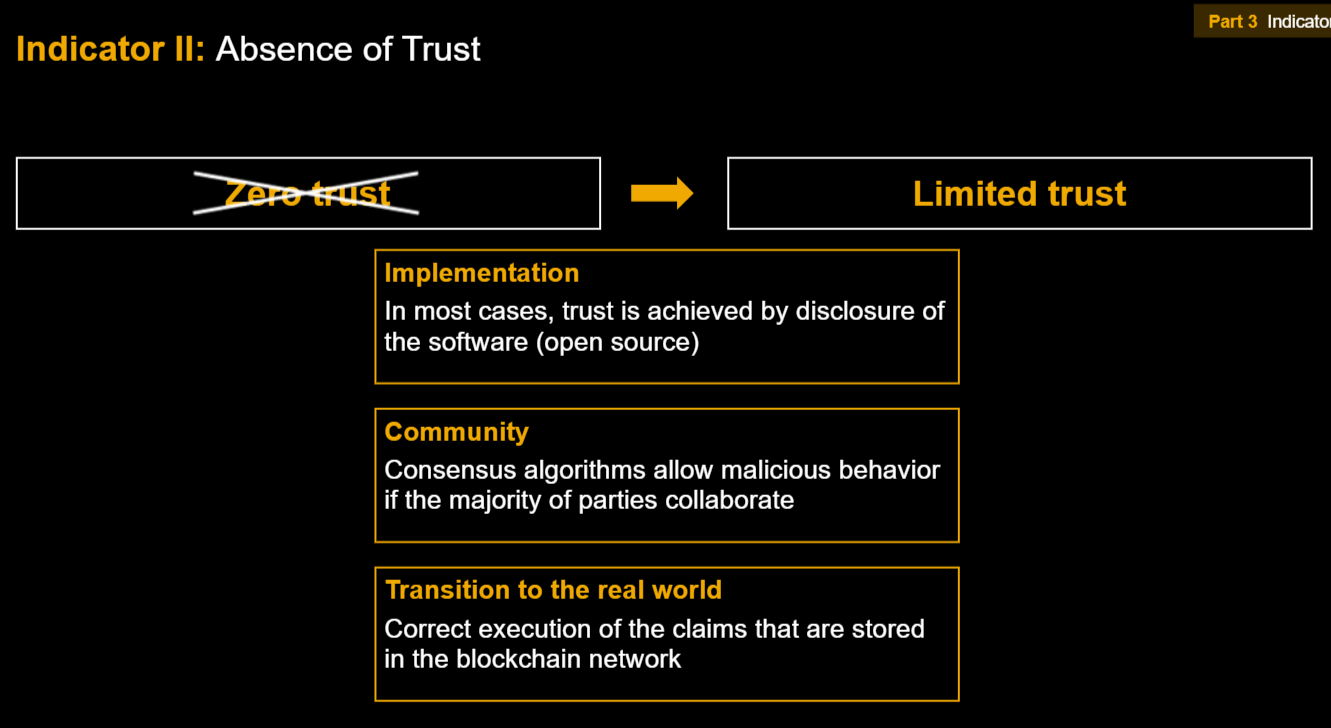
简单来说，如果一个区块链网络只有一个写入者，那么无论拥有多少共识节点都是没有意义的，因为写入者可以随意写入、变更数据，本质上又变成了一个集中式的系统。一个合理的区块链应用应该是参与的各方都可以具备预先设定好的写入权限，从而真正意义上达到去中心话的目的。

c. 是否存在信任缺失

区块链的意义在于使得互不信任的各方可以通过区块链传递和获取信任，并且这种信任建立的成本是很低的而且性价比极高。如果参与方本身不存在信任的问题，这种场景下区块链的信任传递特性反而有些多此一举。举例来说，一个研发团队通过git管理产品代码，每个人作为项目一员都是参与者，信任缺失的问题一定程度上是不存在的，而且git本身也有权限管理的功能，区块链就不太适合该场景。有些人或许会反驳说，团队成员之间的信任并不是百分之百的信任，也有可能存在蓄意销毁代码库的行为。的确有可能发生，不过一定程度上的信任已经建立了，区块链的引入主要还是想扭转信任缺失的情况。

需要指出的是区块链技术很重要的一点：区块链的参与方可以不完全信任彼此，但是至少需要建立最低程度的信任。这种最低程度的信任可以从以下三方面理解：

1. 参与方需要信任区块链网络的实现方式。多数情况下，该信任是通过开源区块链网络代码来建立的。例如超级账本，多链以及其他公开的区块链网络实现方式。
2. 参与方需要信任由所有参与方组成的整个社区。这就牵扯到区块链网络的共识算法， 公链比特币的POW算法依赖超过50%的网络节点是可信的，这是非常重要的假设条件。如果该条件不能满足，区块链的不可篡改性是完全无法保证的。虽然超级账本中共识算法有所不同，不过整个系统的可靠性还是依赖某些参与方的真实性。
3. 参与方需要信任数据库数据到真实场景的过渡。区块链可以很好的确保数据的真实性，这些真实的数据接下来的使用、实现参与方也需要建立一定的信任。



前面提到区块链颠覆的核心在于去中心化，那么区块链与生俱来具备的互信特性就是去中心化的基础。没有最基本的信任，去中心化等所有特性都会变得毫无意义。

d. 是否需要取代第三方

比特币区块链网络的引入，使得数字交易不需要第三方金融机构的参与，点对点可以高效安全的完成。回想传统的跨境交易需要经由各个国家金融机构的审批，整个过程不仅会产生手续费，也会产生时间成本。可以说去除中间商，一直也是区块链相比传统行业很大的优势。

可以看到一定情况下取代中间商会带来时间，成本上的节约。不过不是所有的场景，中间商都需要被取代或者能被取代。如果可信任的第三方已经存在而且能够解决问题， 那么可能就没有必要上区块链。切记，区块链不是万能钥匙，本身也捆绑很多资源成本。

Airbnb, 是一家联系旅游人士和家有空房出租的房主的服务型网站，它可以为用户提供多样的住宿信息。可以看到，Airbnb并不仅仅只是一个房屋信息提供平台，同时需要做很多的沟通工作，协调房主以及旅客。而这些内容是不能被区块链的实施就能取代的。因此，我们在判断应用是否适合使用区块链的时候，不能简单认为有中间商的应用场景都适用区块链。

需要注意的是，本书中提到的判断标准是作为判断区块链应用的充分不必要条件，也就是说，如果满足四大判断准则就基本可以肯定为区块链应用，但没有全部满足的应用也很可能是区块链应用。