账本

受众：架构师、应用程序开发者和智能合约开发者、管理员

账本是 Hyperledger Fabric 中的一个重要概念，它存储了有关业务对象的重要事实信息，其中既包括对象属性的当前值，也包括产生这些当前值的交易的历史。

在这个主题中，我们将谈到：

什么是账本?

账本记录着业务的当前状态，它就像一个交易日记。欧洲和中国最早的账本可以追溯到近 1000 年前，苏美尔人在 4000 年前就已经有石制账本了，不过我们还是从离我们最近的例子开始讲吧！

你可能已经习惯查看你的银行账户了。对你来说，最重要的是账户余额，它是你当时就能花的钱。如果你想看看你的余额是如何产生的，可以浏览一下相关的交易收入和支出。这是现实生活中的一个账本的示例——一个状态（您的银行余额）和一组促成该状态的有序交易（收入和支出）。Hyperledger Fabric 也致力于这两个方面，它旨在呈现一组账本状态的当前值，同时记录下促成了以上账本状态的交易的历史。

账本、事实和状态

账本储存的其实并不是业务对象本身，而是与业务对象相关的**事实**信息。当我们说“我们在账本中存储一个业务对象”时，其实是说我们正在记录与一个业务对象当前状态有关的事实，以及与促成这一当前状态的交易历史相关的事实。在一个日益数字化的世界里，我们感觉自己正在看的是一个物体本身，而不是关于这个物体的一些事实。对于数字对象来说，它可能位于一个外部数据库，但通过我们储存在账本中有关该对象的事实就能够识别出该数字对象的所在位置以及其他与之相关的关键信息。

虽然与业务对象当前状态相关的事实可能会发生改变，但是与之相关的事实历史是**不可变的**，我们可以在事实历史上增加新的事实，但无法更改历史中已经存在的事实。我们将看到，如果把区块链看作是与业务对象有关的事实历史，且该历史是不可更改的，那么我们就能够很轻松、高效地理解区块链。

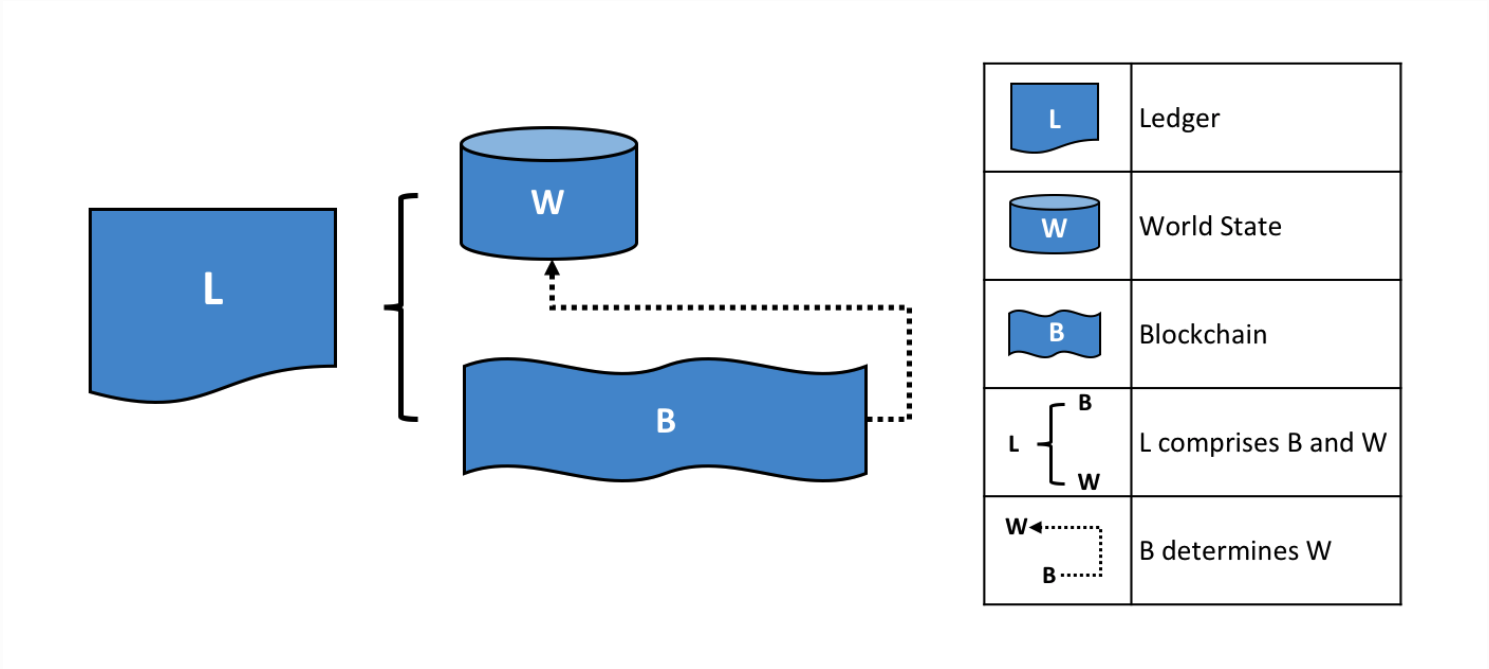
现在我们来深入探讨一下 Hyperledger Fabric 的账本结构！

账本

Hyperledger Fabric 中的账本由“世界状态“和”区块链“这两部分组成，它们彼此不同但却相互关联。二者都代表了与业务对象有关的一些事实。

首先，**世界状态**是一个数据库，它存储了一组账本状态的**当前值**。通过世界状态，程序可以直接访问一个账本状态的当前值，不需要遍历整个交易日志来计算当前值。默认情况下，账本状态是以**键值**对的方式来表示的，稍后我们将看到 Hyperledger Fabric 如何提供这一方面的灵活性。因为我们可以创建、更新和删除状态，所以世界状态能够频繁更改。

其次，**区块链**是交易日志，它记录了促成当前世界状态的所有改变。交易被收集在附加到区块链的区块中，能帮助我们理解所有促成当前世界状态的改变的历史。区块链数据结构与世界状态相差甚远，因为一旦把数据写入区块链，就无法修改，它是**不可篡改的**。



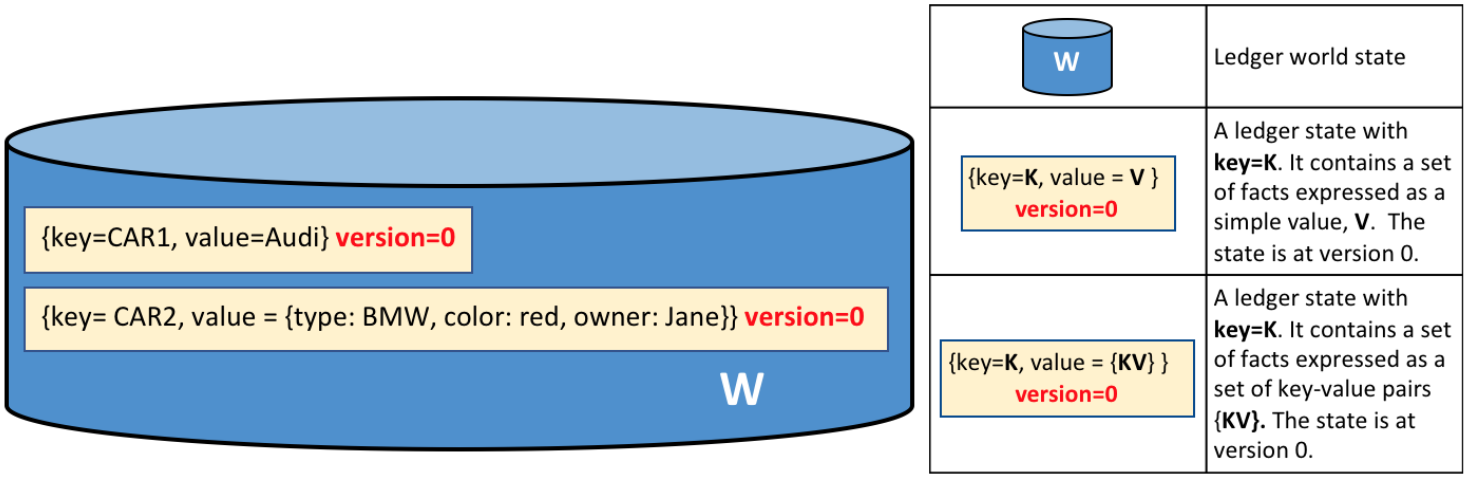
账本 L 由区块链 B 和世界状态 W 组成，其中世界状态 W 由区块链 B 决定。我们也可以说世界状态 W 是源自区块链 B。

为帮助理解，可以这样认为：Hyperledger Fabric 网络中存在一个**逻辑**账本。实际上，Fabric 网络维护着一个账本的多个副本，这些副本通过名为**共识**的过程来与其他副本保持一致。**分布式账本技术**（DLT）这个术语经常与这种账本联系在一起，这种账本在逻辑上是单个的，但是在整个网络中却分布着许多彼此一致的副本。

现在让我们更细致地研究一下世界状态和区块链数据结构。

世界状态

世界状态将业务对象属性的当前值保存为唯一的账本状态。这很有用，因为程序通常需要对象的当前值，如果遍历整个区块链来计算对象的当前值会很麻烦——从世界状态中可以直接获取当前值。



一个账本世界状态包含两个状态。第一个状态是： key=CAR1 和 value=Audi。第二个状态中有一个更复杂的值：key=CAR2 和 value={model:BMW, color=red, owner=Jane} 。两个状态的版本都是0。

账本状态记录了一组与特定业务对象有关的事实。我们的示例展示的是 CAR1 和 CAR2 这两辆车的账本状态，二者都各有一个值和一个键。应用程序可以调用智能合约，该合约使用简单的账本 API 来**获取**、**写入**和**删除**状态。注意状态值可以是简单值（Audi...），也可以是复合值（type:BMW...）。经常会通过查询世界状态来检索具有某些特定属性的对象，例如查找所有红色宝马汽车。

世界状态被作为数据库来实现。这一点很有意义，因为数据库为有效存储和状态检索提供了充分的算子。稍后我们将看到，我们可以将 Hyperledger Fabric 配置为使用不同的世界状态数据库来满足以下需求：不同类型的状态值，应用程序所需的访问模式，例如，当遇到复杂查询的情况时。

应用程序提交那些会更改世界状态的交易，这些交易最终被提交到账本区块链上。应用程序无法看到 Hyperledger Fabric SDK（软件开发工具包）设定的共识机制的细节内容，它们能做的只是调用智能合约以及在交易被收进区块链时收到通知（所有被提交的交易，无论有效与否，都会被收进区块链）。Hyperledger Fabric 的关键设计在于，只有那些受到相关**背书组织签名**的交易才会更新世界状态。如果一个交易没有得到足够背书节点的签名，那么它不会更新世界状态。您可以阅读更多关于应用程序如何使用智能合约以及如何开发应用程序的信息。

您还会注意到，每个状态都有一个版本号，在上面的图表中，状态 CAR1 和 CAR2 都处于它们的初始版本 0。版本号是供 Hyperledger Fabric 内部使用的，并且每次状态更改时版本号会发生递增。每当更新状态时，都会检查该状态的版本，以确保当前状态与背书时的版本相匹配。这就确保了世界状态是按照预期进行更新的，没有发生并发更新。

最后，首次创建账本时，世界状态是空的。因为区块链上记录了所有代表有效世界状态更新的交易，所以任何时候都可以从区块链中重新生成世界状态。这样一来就变得非常方便，例如，创建节点时会自动生成世界状态。此外，如果某个节点发生异常，重启该节点时能够在接受交易之前重新生成世界状态。

区块链

现在让我们把注意力从世界状态转移到区块链上。世界状态存储了与业务对象当前状态相关的事实信息，而区块链是一种历史记录，它记录了这些业务对象是如何到达各自当前状态的相关事实。区块链记

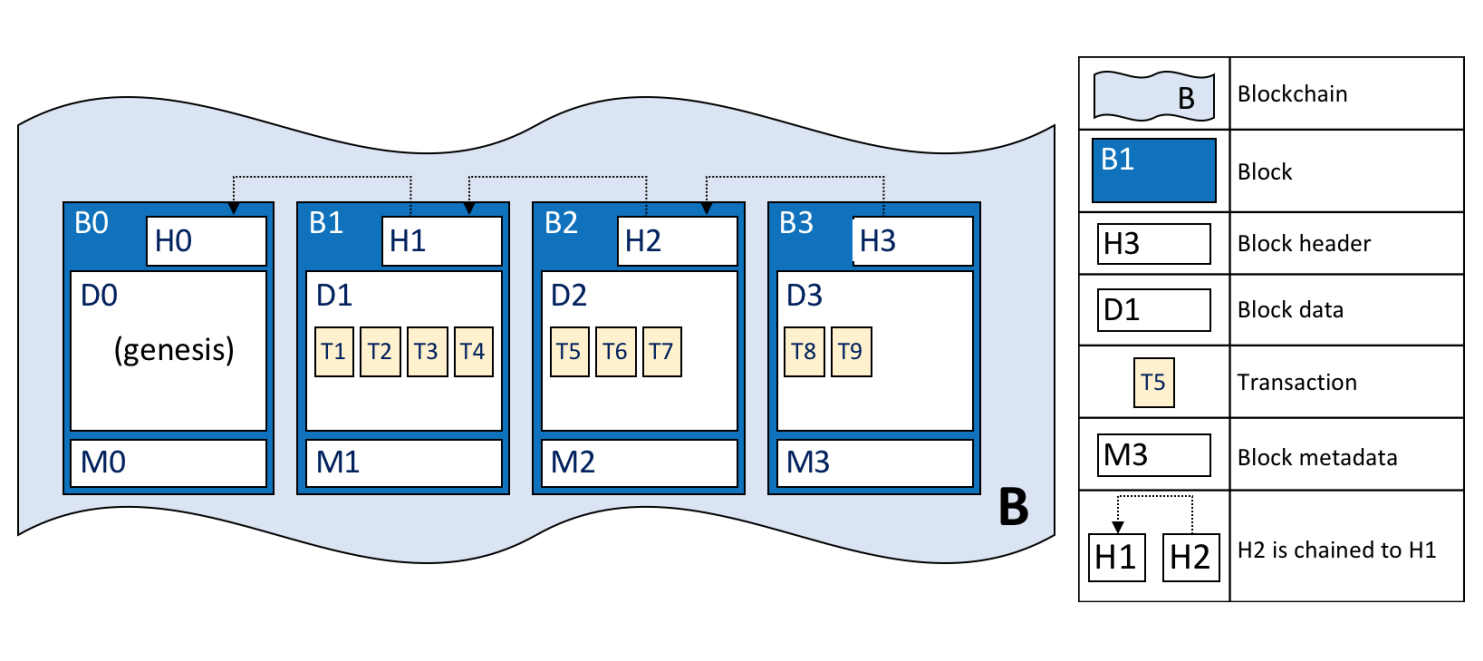
录了每个账本状态之前的所有版本以及状态是如何被更改的。

区块链的结构是一群相互链接的区块的序列化日志，其中每个区块都包含一系列交易，各项交易代表了一个对世界状态进行的查询或更新操作。我们在其他地方讨论了排序交易的确切机制；其中重要的是区块排序以及区块内的交易排序，这一机制是在 Hyperledger Fabric 的**排序服务**组件首次创建区块时被建立起来的。

每个区块的头部都包含区块交易的一个哈希，以及前一个区块头的哈希。这样一来，账本上的所有交易都被按序排列，并以密码方式连接在一起。这种哈希和链接使账本数据变得非常安全。即使某个保存账本的节点被篡改了，该节点也无法让其他节点相信自己拥有“正确的”区块链，这是因为账本被分布在一个由独立节点组成的网络中。

区块链总是以文件实现，而与之相反的是，世界状态以数据库实现。这是一个明智的设计，因为区块链数据结构高度偏向于非常小的一组简单操作。第一项操作被放在区块链的末尾，就目前来说，查询操纵相对少见。

让我们更详细地看看区块链的结构。



区块链 B 包含了 B0、B1、B2、B3这四个区块。B0 是该区块链的第一个区块，也叫创世区块。

在上面的图中我们可以看到，**区块** B2 有一个**区块数据** D2，该数据包含了 B2 的所有交易：T5、T6、T7。

最重要的是，B2 有一个**区块头** H2，H2 包含了 D2 中所有交易的加密**哈希**以及前一个区块中 H1 的一个哈希。这样一来，所有区块彼此紧密相连，不可篡改，术语**区块链**很好地描述了这一点！

最后，如图所示，区块链中的第一个区块被称为**创世区块**。虽然它并不包含任何用户交易，但却是账本的起始点。相反的，创世区块包含了一个配置交易，该交易含有网络配置（未显示）的初始状态。我们将会在讨论区块链网络和通道 时更详细地探讨初始区块。

区块

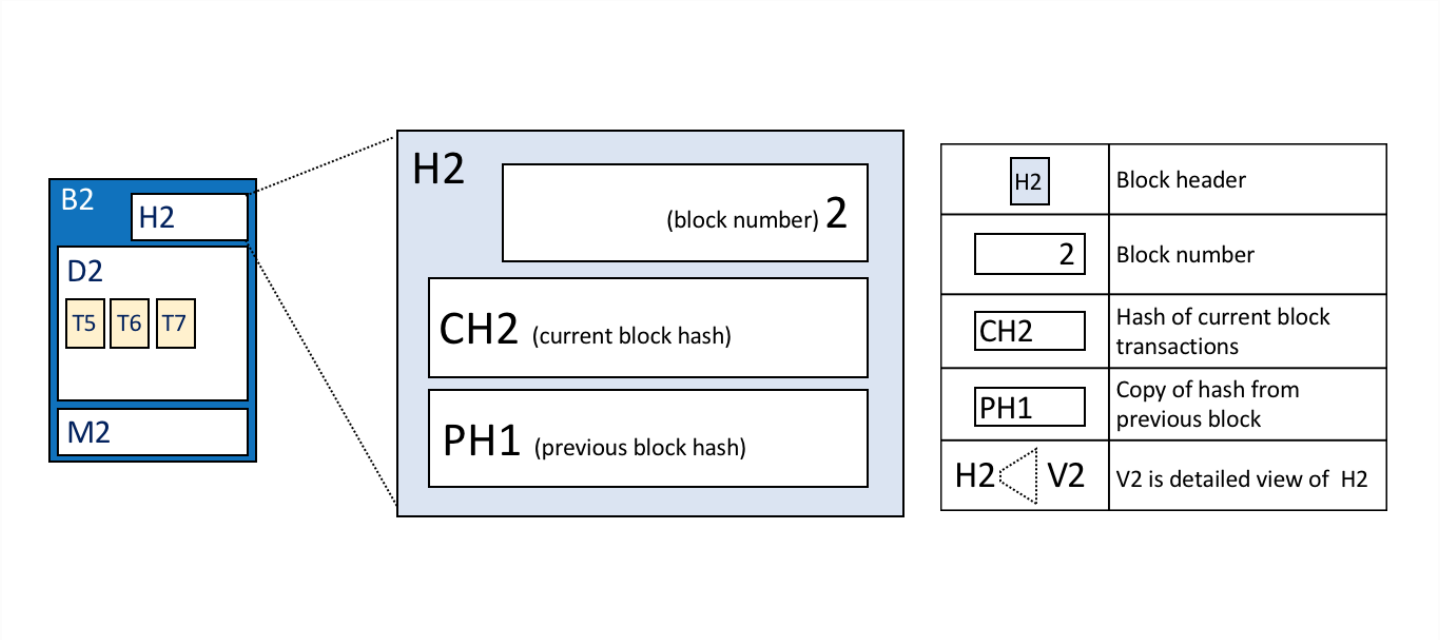
让我们仔细看看区块的结构。它由三个部分组成

- **区块头**

  这个部分包含三个字段，这些字段是在创建一个区块时候被写入的。

  - **区块编号**：编号从0（初始区块）开始，每在区块链上增加一个新区块，编号的数字都会加1。
  - **当前区块的哈希值**：当前区块中包含的所有交易的哈希值。
  - **前一个区块头的哈希值**：区块链中前一个区块头的哈希值。

  这些字段是通过在内部对区块数据进行加密哈希而生成的。它们确保了每一个区块和与之相邻的其他区块紧密相连，从而组成一个不可更改的账本。



  区块头详情：区块 B2 的区块头 H2 包含了区块编号 2，当前区块数据 D2 的哈希值 CH2，以及前一个区块头 H1 的哈希值。

- **区块数据**

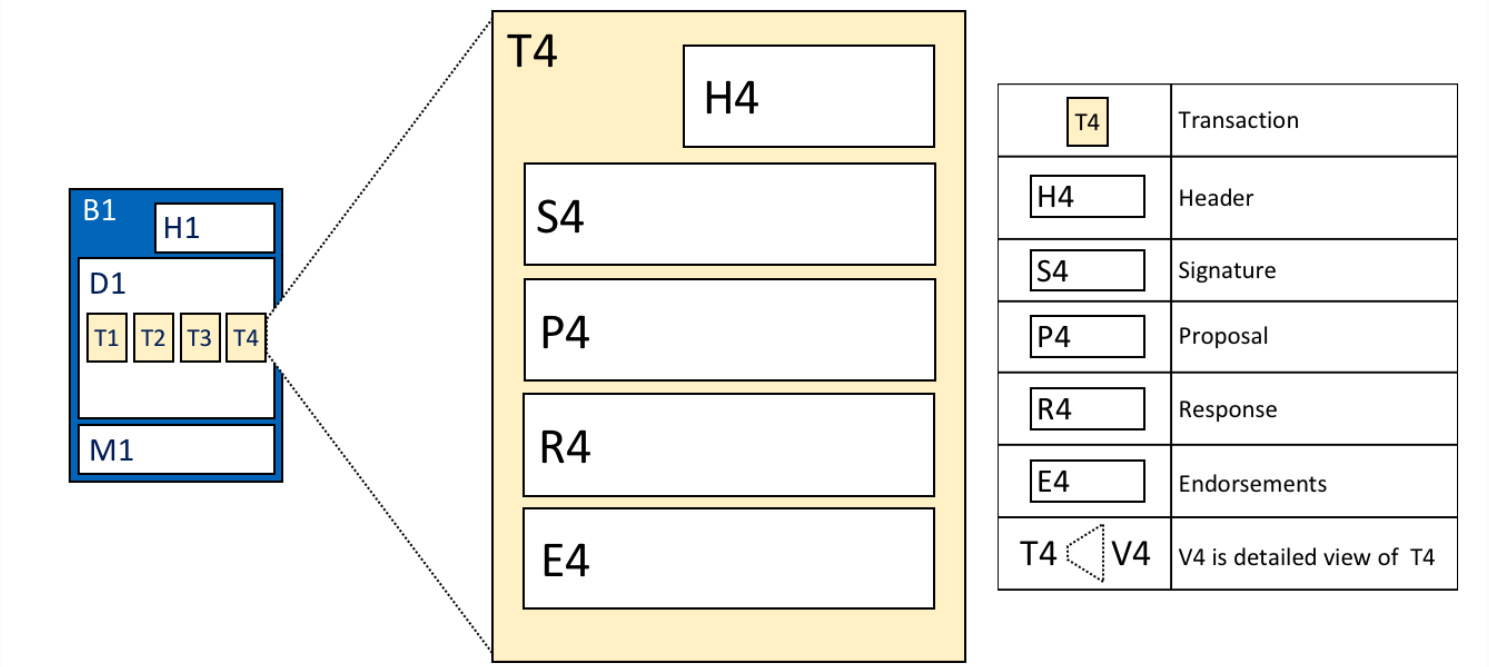
  这部分包含了一个有序的交易列表。区块数据是在排序服务创建区块时被写入的。这些交易的结构很复杂但也很直接，我们会在后边进行讲解。

- **区块元数据**

  这个部分包含了区块被写入的时间，还有区块写入者的证书、公钥以及签名。随后，区块的提交者也会为每一笔交易添加一个有效或无效的标记，但由于这一信息与区块同时产生，所以它不会被包含在哈希中。

交易

正如我们所看到的，交易记录了世界状态发生的更新。让我们来详细了解一下这种把交易包含在区块中的**区块数据**结构。



交易详情：交易 T4 位于区块 B1 的区块数据 D1 中，T4 包括的内容如下：交易头 H4，一个交易签名 S4，一个交易提案 P4，一个交易响应 R4 和一系列背书 E4。

在上面的例子中，我们可以看到以下字段：

- **头**

  这部分用 H4 表示，它记录了关于交易的一些重要元数据，比如，相关链码的名字以及版本。
- **签名**

  这部分用 S4 表示，它包含了一个由客户端应用程序创建的加密签名。该字段是用来检查交易细节是否未经篡改，因为交易签名的生成需要用到应用程序的私钥。
- **提案**

  这部分用 P4 表示，它负责对应用程序供给智能合约的输入参数进行编码，随后该智能合约生成提案账本更新。在智能合约运行时，这个提案提供了一套输入参数，这些参数同当前的世界状态一起决定了新的账本世界状态。
- **响应**

  这部分用 R4 表示，它是以**读写集** （RW-set）的形式记录下世界状态之前和之后的值。交易响应是智能合约的输出，如果交易验证成功，那么该交易会被应用到账本上，从而更新世界状态。
- **背书**

  就像 E4 显示的那样，它指的是一组签名交易响应，这些签名都来自背书策略规定的相关组织，并且这些组织的数量必须满足背书策略的要求。你会注意到，虽然交易中包含了多个背书，但它却只有一个交易响应。这是因为每个背书都对组织特定的交易响应进行了有效编码，那些不完全满足背书的交易响应肯定会遭到拒绝、被视为无效，而且它们也不会更新世界状态，所以没必要放进交易中。

  在交易中只包含一个交易响应，但是会有多个背书。这是因为每个背书包含了它的组织特定的交易响应，这意味着不需要包含任何没有有效的背书的交易响应，因为它会被作为无效的交易被拒绝，并且不会更新世界状态。

以上总结了交易的一些主要字段，其实还有其他字段，但是上述几种是您需要了解的基本字段，便于您对账本数据结构有一个很好的了解。

世界状态数据库选项

世界状态是以数据库的形式实现的，旨在提供简单有效的账本状态存储和检索。正如我们所看到的，账本状态可包含简单值或复合值，为了适应这一点，世界状态数据库可以多种形式实现，从而对这些值进行有效实现。目前，世界状态数据库的选项包括 LevelDB 和 CouchDB 。

LevelDB 是世界状态数据库的默认选项，当账本状态是简单的键值对时，使用 LevelDB 非常合适。LevelDB 数据库与 peer 节点位于相同位置，它被嵌入与 peer 节点相同的操作系统进程中。

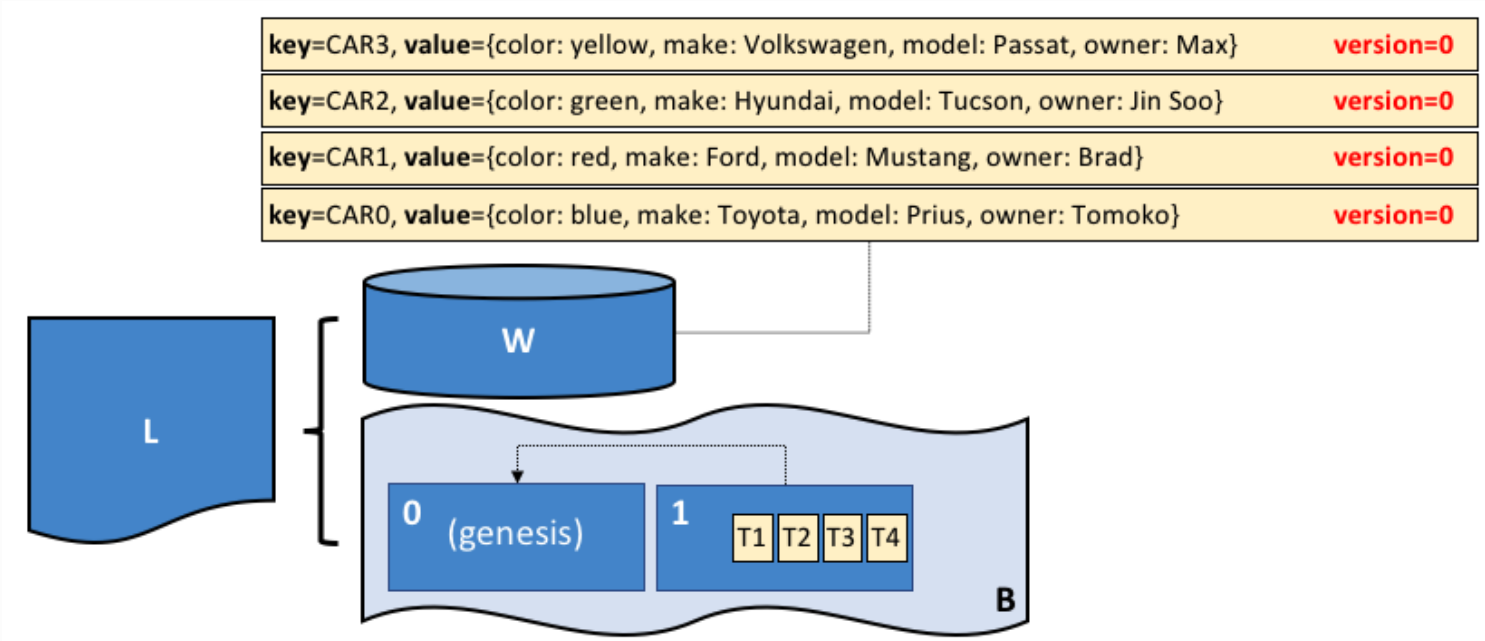
当账本状态结构为 JSON 文档时，以 CouchDB 来实现世界状态非常合适，这是因为业务交易涉及的数据类型通常十分丰富，而 CouchDB 可支持对这些数据类型进行各种形式的查询和更新。在实现方面，CouchDB 是在单独的操作系统进程中运行的，但是节点和 CouchDB 实例之间仍然存在1:1的关系。智能合约无法看到上述任何内容。有关 CouchDB 的更多信息，请参见 CouchDB 作为状态数据库。

在 LevelDB 和 CouchDB 中，我们看到了 Hyperledger Fabric 的一个重要方面——它是*可插拔*的。世界状态数据库可以是关系数据存储、图形存储或时态数据库。这极大提升了可被有效访问的账本状态类型的灵活性，使得 Hyperledger Fabric 能够处理多种不同类型的问题。

示例账本fabcar

关于账本的讨论即将结束，让我们来看一个示例账本。如果您已经运行了 fabcar 示例应用程序，那么您就已经创建了这个账本。

fabcar 示例应用程序创建了 10 辆车，每辆车都有独一无二的身份；它们有不同的颜色，制造商，型号和拥有者。以下是前四辆车创建后的账本。



账本 L 包含了一个世界状态 W 和一个区块链 B。其中 W 包含了四个状态，各状态的键分别是：CAR0，CAR1，CAR2 和 CAR3 。而 B 包含了两个区块 0和 1。区块1 包含了四笔交易：T1，T2，T3，T4。

我们可以看到世界状态包含了对应于 CAR0、CAR1、CAR2 和 CAR3 的状态。CAR0 中包含的值表明了这是一辆蓝色的丰田普锐斯（Toyota Prius），目前车主是 Tomoko，其他车辆的的状态和值也与此类

似。此外，我们还可以看到所有车辆状态的版本号都是0，这是它们的初始版本号，也就是说这些车辆状态自创建以来一直没有被更新过。

我们还可以看到区块链包含两个区块。其中区块0是创世区块，但它并不包含任何与汽车相关的交易。而区块1包含交易 T1、T2、T3、T4，这些交易与生成世界状态中 CAR0 到 CAR3 这四辆车初始状态的交易相符。同时区块1与区块0是相连的。

我们没有介绍区块或交易中的其他字段，特别是（区块/交易）头和哈希，如果你对这部分内容感兴趣的话，可以阅读文件中的其他部分。读完后你会对整个区块和交易有更加透彻的认识，但现在，你对 Hyperledger Fabric 账本的概念已经足够了解了。很好！

命名空间

上文中我们讨论账本时，似乎它只包括一个世界状态和一条区块链，但这显然过于简单化了。实际上，每个链码都有自己的世界状态，并且与所有其他链码的世界状态分离。世界状态位于一个命名空间中，因此只有位于同一链码中的智能合约才能访问一个给定的命名空间。

区块链没有命名空间。它包含来自许多不同智能合约命名空间的交易。您可以在此主题中阅读更多关于链码命名空间的信息。

现在让我们看看命名空间的概念是如何被应用到 Hyperledger Fabric 通道中的。

通道

在 Hyperledger Fabric 中，每个通道都有一个完全独立的账本。这意味着完全独立的区块链和完全独立的世界状态，包括命名空间。应用程序和智能合约可以在通道之间通信，以便在通道间访问账本信息。

在本主题中，您可以阅读更多关于账本如何与通道一起工作的信息。

更多信息

要深入了解交易流程、并发控制和世界状态数据库，请查阅交易流程、读写集语义和 CouchDB 作为状态数据库主题。