流程和数据设计

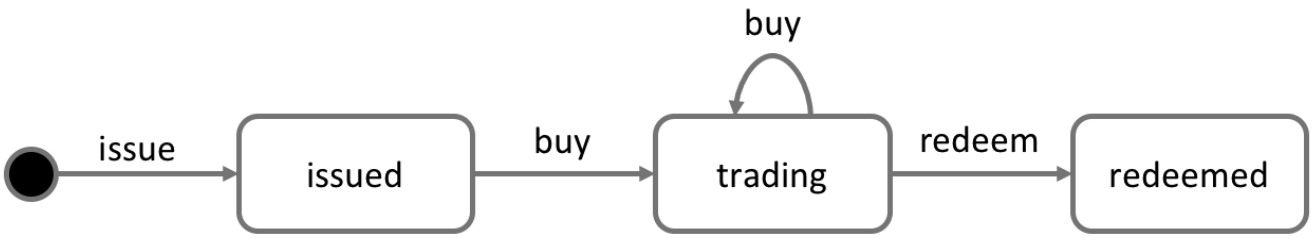
受众：架构师，应用程序和智能合约开发者，业务专家

本主题主要讨论在 PaperNet 中如何设计商业票据的流程和与它相关的数据结构。我们在 分析章节 中已经强调使用状态和交易对 PaperNet 建模提供了一种精确的方法来了解正在发生的事情。我们现在将详细阐述这两个强烈相关的概念，以帮助我们随后设计 PaperNet 的智能合约和应用程序。

生命周期

正如我们所见，在处理商业票据时有两个重要的概念：**状态**和**交易**。实际上，*所有*区块链用例都是如此; 状态建模是重要的概念对象，其生命周期转换由交易描述。对状态和交易的有效分析是成功实施的重要起点。

我们可以用状态转移表来表示商业票据的生命周期：



商业票据的状态转移表。商业票据通过发行，购买和兑换交易在已发行、交易中和已兑换之间进行状态转移。

了解状态图如何描述商业票据随着时间如何变化，以及特定交易如何控制生命周期转换。在 Hypledger Fabric 中，智能合约实现了在不同状态之间转换商业票据的交易逻辑。商业票据状态实际上是保存在帐本的世界状态中; 让我们来深入了解一下。

账本状态

回想一下商业票据的结构：

```
Issuer: MagnetoCorp
Paper: 00001
Owner: DigiBank
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: trading
```

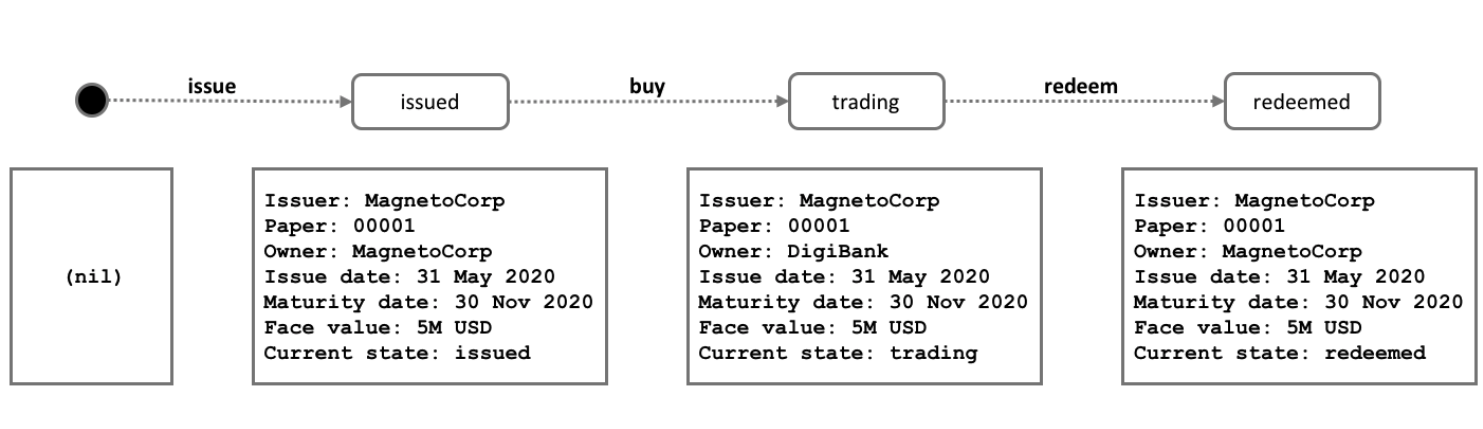
商业票据可以被表示为属性集，每个属性都对应一个值。通常，这些属性的组合会为每个票据提供一个

唯一键

商业票据的 `Paper` 属性的值是 `00001`， `Face value` 属性的值是 `5M USD`。更重要的是，`Current state` 属性表示了商业票据是 `issued` 状态， `trading` 状态还是 `redeemed` 状态。 结合来看，属性的完整集合构成了商业票据的**状态**。此外，这些商业票据的全部集合构成了账本的 世界状态。

所有的账本状态都是这种形式；每个状态都是一个属性集，每个都有不同的值。状态的*多属性*是一个强大的特性——允许我们把 Fabric 的状态看做是一个向量而不是一个简单的标量。然后，我们把整个实际的对象当做独立的状态，随后由交易逻辑控制状态转换。Fabric 的状态是由键值对实现的，其中值以捕获对象的多个属性的格式编码对象属性，通常是 JSON 格式。根据这些属性，账本数据库 可以支持高级的查询操作，这对于复杂的对象检索非常有帮助。

查看 MagnetoCorp 的票据 `00001` 如何表示为一个状态向量，根据不同的交易刺激进行转换:



商业票据状态是由于不同的交易而产生和过渡的。Hyperledger Fabric 状态拥有多个属性，使他们成为向量而不是标量。

注意每个独立的票据都起于空状态，技术上被称作 nil，来表示票据不存在！通过**发行**交易，票据 `00001` 问世，然后由于**购买**和**兑换**交易而更新状态。

注意每个状态是如何自描述的；每个属性都有一个名字和值。尽管目前所有的商业票据都有相同的属性，这种情况不一定总是如此，而 Hyperledger Fabric 支持不同的状态有不同的属性。这允许相同的帐本世界状态包含相同资产的不同形式以及不同类型的资产。同样使得更新状态结构成为可能；试想有一个新的规则需要一个额外的数据字段。灵活的状态属性集支持数据演化的基本需求。

状态键值

大多数的实际应用中，状态会有一个属性组合在给定的上下文中唯一识别它——它就是**主键**。PaperNet 商业票据的主键是通过 `Issuer` 属性和 `paper` 属性拼接得到的；所以 MagnetoCorp 的第一个票据的主键就是 `MagnetoCorp00001`。

状态的主键允许我们唯一识别一个票据；它是通过**发行**交易创建的，然后由**购买**和**兑换**更新。Hyperledger Fabric 需要账本中的每个状态都有唯一的主键。

当唯一主键在可用的属性集中不能获得，应用决定的唯一键会被指定为交易的输入来创建状态。这个唯一键的形式一般是 UUID，尽管可读性不好，但是是一个很好的实践。最重要的是账本中每个独立的状态对象都必须有一个唯一键。

Note: 在主键中你应该避免使用 U+0000 (nil byte)。

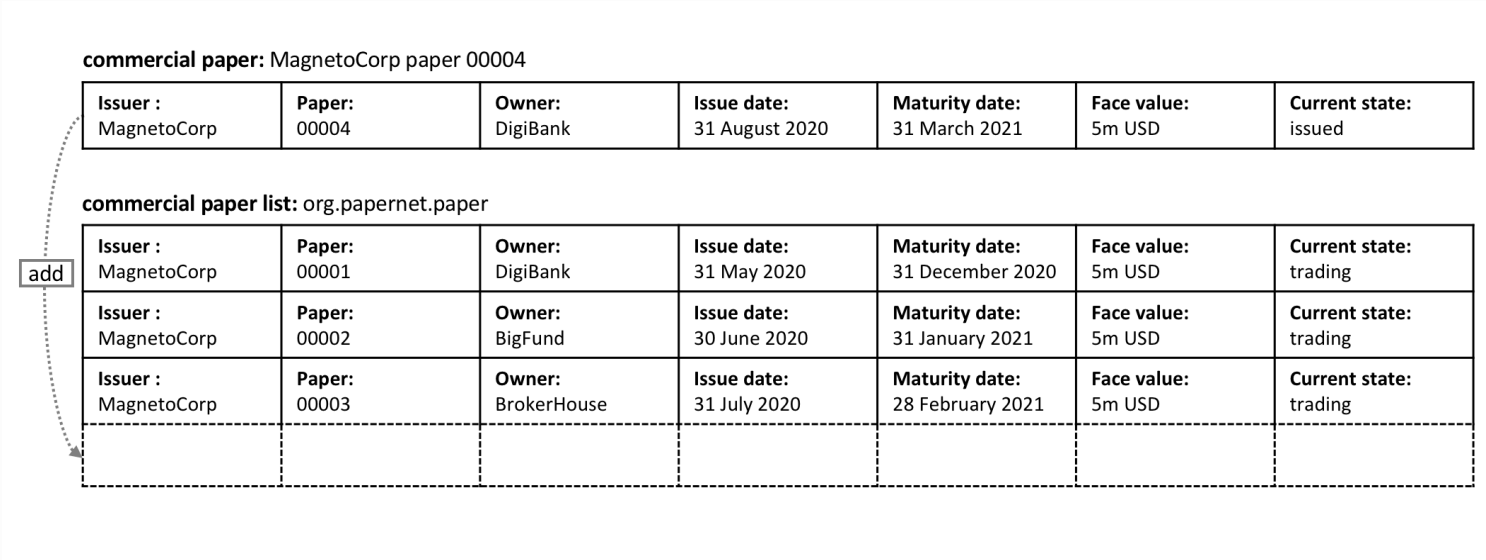
多个状态

正如我们所见，PaperNet 中的商业票据作为状态向量被存储在账本中。能够从账本中查询不同的商业票据是合理的需求；比如，查询所有由 MagnetoCorp 发行的的票据，或者查询所有由 MagnetoCorp 发行且处在 `redeemed` 状态的票据。

为了满足不同类型的查询任务，把所有相关的商业票据按逻辑顺序排列在一起是很有帮助的。PaperNet 的设计包含了商业票据列表的思想——一个逻辑容器，每当商业票据发行或发生其他更改时，该容器都会更新。

逻辑表示

把所有的 PaperNet 商业票据放在一个商业票据列表中是有帮助的:



MagnetoCorp 新增加的票据 00004 被加入到已有的商业票据列表中。

新票据由于**发行**交易被加入到列表中，然后列表中已存在的票据因为**购买**交易和**兑换**交易可以被更新状态。列表有一个描述性的名称：`org.papernet.papers`；使用这种 DNS 名真的是一个好主意，因为适当的名称会让你的区块链设计对其他人来说是直观的。这种想法同样也适用于智能合约的名字。

物理表现

我们可以正确地想到 PaperNet 中的单个票据列表—— `org.papernet.papers` ——列表最好作为一组单独的 Fabric 状态来实现，其复合键将状态与其列表关联起来。这样，每个状态的复合键都是惟一的，并支持有效的列表查询。

key	value
org.papernet.paperMagnetoCorp00001	**Issuer :** MagnetoCorp, **Paper:** 00001, **Owner:** DigiBank, **Issue date:** 31 May 2020, **Maturity date:** 31 December 2020, **Face value:** 5m USD, **Current state:** trading
org.papernet.paperMagnetoCorp00002	**Issuer :** MagnetoCorp, **Paper:** 00002, **Owner:** BigFund, **Issue date:,** 30 June 2020, **Maturity date:** 31 January 2021, **Face value:** 5m USD, **Current state:** trading
org.papernet.paperMagnetoCorp00003	**Issuer :** MagnetoCorp, **Paper:** 00003, **Owner:** BrokerHouse, **Issue date:** 31 July 2020,, **Maturity date:** 28 February 2021, **Face value:** 5m USD, **Current state:** trading
org.papernet.paperMagnetoCorp00004	**Issuer :** MagnetoCorp, **Paper:** 00004, **Owner:** DigiBank, **Issue date:** 31 August 2020, **Maturity date:** 31 March 2021, **Face value:** 5m USD, **Current state:** issued

将 PaperNet 商业票据列表表示为一组不同的 Hyperledger Fabric 状态

注意列表中的每个票据都是如何用向量状态表示的，其中唯一的**组合**键是由 `org.papernet.paper` 的属性 `Issuer` 和 `Paper` 连接而成的。这种结构有两个好处：

- 允许我们检查账本中的任意状态向量在哪个列表中，不用引用到不同的列表。这类似于观察一群体育迷，通过他们穿的衬衫的颜色来判断他们支持哪支球队。体育迷们自我宣布他们的忠诚；我们不需要粉丝名单。
- Hyperlegder Fabric 内部使用了一个并发控制机制来更新账本，所以把票据保存在不同的状态向量中大大减少了共享状态碰撞的机会。这种碰撞需要交易重新提交，复杂化了应用设计，而且降低了性能。

第二点是 Hyperledger Fabric 的关键；状态向量的物理设计对于优化性能和行为**非常重要**。保持状态的独立!

信任关系

我们已经讨论了网络中的不同角色，如发行者，交易员或评级机构，以及不同的商业利益如何决定谁需要签署交易。在 Fabric 中，这些规则由所谓的背书策略捕获。这些规则可以在链码粒度上设置，也可以为单个状态键设置。

这意味着在 PaperNet 中，我们可以为整个命名空间设置一个规则，以确定哪些组织可以发行新票据。然后，可以为单个票据设置和更新规则，以捕获购买和兑换交易的信任关系。

在下一个主题中，我们将向您展示如何结合这些设计概念来实现 PaperNet 商业票据智能合约，然后是应用程序来使用它!