构建有效灵活的 CMDB

CMDB 是 Configuration Management Database 的简称。英国商务部出版的《ITIL 服务支持》一书这样定义 CMDB：“它是一种包含每一个配置项（Configuration Item，CI）全部关联细节，以及配置项之间重要关联细节的数据库”。CMDB 是真实环境中服务资产在软件系统中的逻辑体现，它的目的是有效管理资产，并对其他的服务管理流程实现支持。CMDB 也常被用于帮助服务提供商在产生故障时能够迅速定位到故障位置，并能分析出受影响服务的深度和广度。所以 CMDB 对 IT 服务提供商尤为重要。

CMDB 的建设在早期的管理软件中使用电子报表的形式，简单记录了 IT 资产的信息，缺点显而易见，很难表明配置项之间的关系，对于实际使用的意义也相当有限。如今，CMDB 摆脱了管理软件附属品的角色，成为独立的系统管理模块，是企业级集中式的配置管理数据库。这样就给了我们更大的自由度，但是建设的难度也就更大了，CMDB 构建已经成为了大家议论的热点。

既然 CMDB 是真实环境的逻辑体现，那他与真实环境的匹配程度将决定 CMDB 的可用程度。那 CMDB 如何建？哪些对象可以作为配置项？配置项识别的广度和深度如何确定。目前 CMDB 中的 CI 信息覆盖了企业网络中的应用、操作系统、补丁、硬件设备、生命周期成本以及用户链接。

ITIL 书中所介绍的 CMDB 模型有很多的局限性，并且这个模型只是一种指导而已，同时很多厂商所开发了 CMDB 软件，大多也是都是参照了 ITIL 书中的 CMDB 模型。这个模型是以层次结构为指导，但在现实的很多情况下 CMDB 中的配置项（CI）是以一种错综复杂的形式出现，每个 IT 服务提供商只能根据自身的情况打造出适合自身的 CMDB 模型。

能否有一种通用的方法能够帮助我们构建 CMDB？

从技术的角度出发，构建 CMDB 过程就是一种思维转换的过程，需要把现实的环境转换为逻辑的环境，根据以往 CMDB 设计的经验总结了以下的步骤：

一、 在现实环境中寻找 CI 对象

第一步的工作对于整体而言相当关键，将影响到后续的步骤以及最终 CMDB 能否能被有效地使用。并且这一个步骤需要有一定的抽象和总结能力。

哪些可以作为CI（配置项）对象呢？没有固定的做法，但总结一句话，影响到服务使用的以及影响到业务的都可以确定为配置型，其中包含到有形的或无形的。确定CI的原则：

1. 对服务产生影响的才作为CI。也就是基于“广度”的考虑。
2. CI项一定是可以被更新的，对于不可更新的项无法进行变更管理。
3. 确定的CI项一定要考虑粒度，是否值得作为CI，更多地由成本因素决定。

从经验来看可以包含以下一些配置项对象，如：服务、应用、服务器、工作站、外设、网络、人员等等。当然不局限于这些。

二、 确定CI对象的属性

每一个配置项都会有很多属性，需要对属性进行筛选，也是考验构建者的抽象能力。另一方面有些CI还可以再进行分解，可以分解成更多的CI对象。可分解的内容是作为属性还是另一个CI对象也是需要反复考虑的问题。

确定CI对象的原则同样适用于确定CI对象的属性。

1. 对服务产生影响的属性才需要被记录。
2. CI的属性是可以被更新的，很多配置项都会设有一个通用的属性——配置项的状态，该属性设置时属性要有互斥性。这样便于用户了解配置项当前的状态。
3. 查看所确定的某个属性是否还需要再进行分解。也就是需要考虑配置项设置的“深度”。如果不需要就直接作为CI的属性，否则就作为另一个CI对象来考虑。

此过程一般需要反复地讨论、斟酌和验证。传统的做法前期可以采用问卷调查或头脑风暴等形式，后期再进行反复测试和验证后再确定下来。

三、 确定CI对象之间的关系

此过程也相当具有挑战性，关系的设定决定了以后在使用过程中对配置项的快速定位与影响度分析的有效性。配置项之间的关系种类会有很多。不同的CI之间会有不同的关系类型，相同的两个CI对象之间还可能有不同的关系，而且关系的确定也需要有适当的粒度，关系的保留采用“有用性”原则，也就是对业务来说是有用的。忽略那些对业务不影响的关系类型。

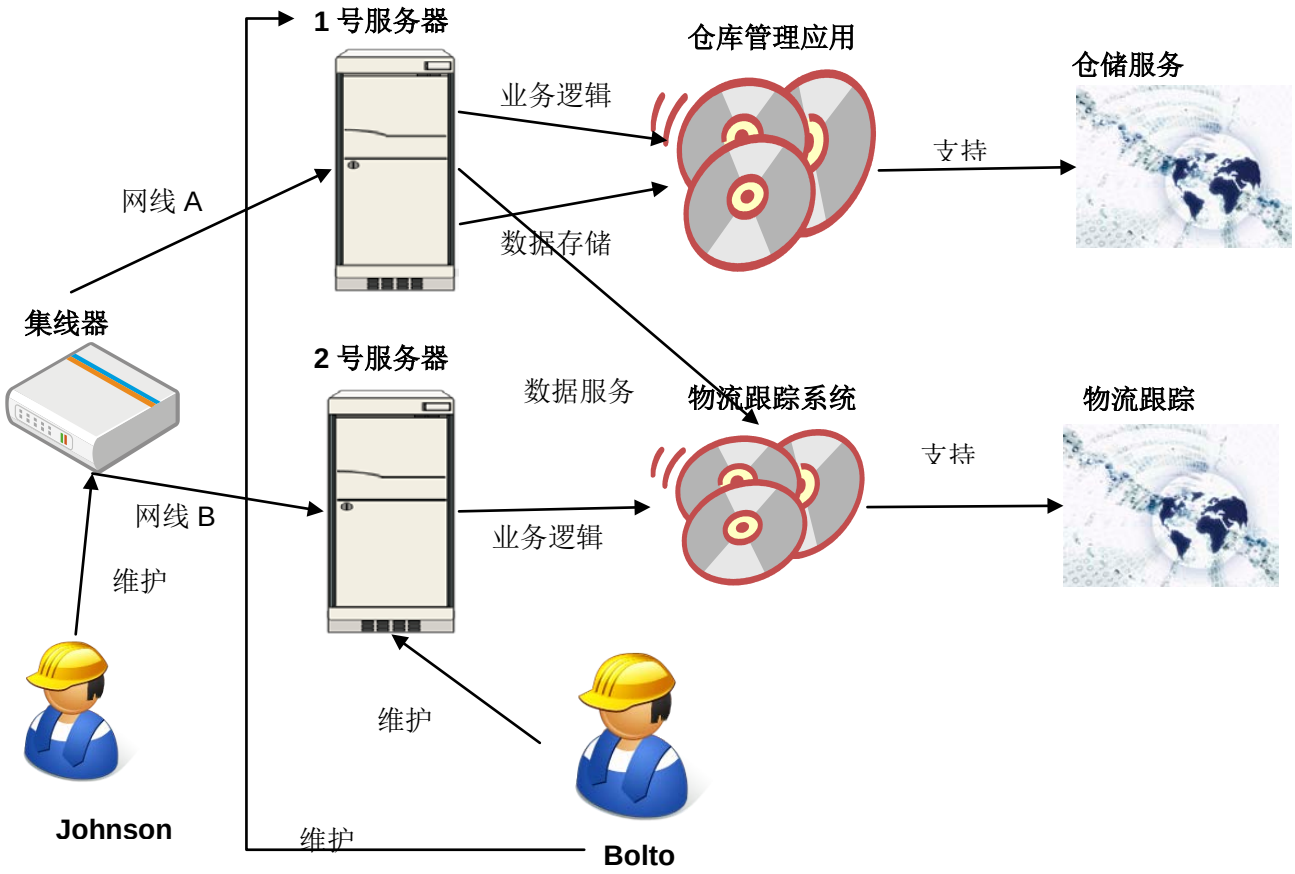
很多 ITIL 软件，包括 ITIL 本身都建议把 CI 建设成层次结构型。这种做法在建立 CMDB 时将失去灵活性，在很多情况下很难通过这些关系的设定迅速定位到某些相关的 CI 对象。面对复杂的对象和关系，层次结构型的 CMDB 模型就无能为力了。

建议采用更灵活的方式来建立 CI 对象之间的关系。只要 CI 对象之间的关系对服务存在一种影响或者对帮助定位 CI 对象有帮助就可以定义一种关系。在很多情况下配置项之间的关系存在多重性，而传统的做法只考虑了一种关系，有局限性。例如：同一个软件产品（仓库管理应用）和同一个主机对象（服务器 1），就存在多种关系。此应用的数据服务层安装在 1 号服务器上，此应用的逻辑处理层安装在 1 号服务器上。而这台服务器对象上还可能安装有其他的应用软件。这种情况下，主机和软件之间就存在多重的关系。

建议的做法是这样的，可以用绘图工具，如 Viso 先把 CI 对象画好，不需要建立层次关系，因为很多关系都是不同层次间的，然后通过线条把关系再画出来，在每一个线条边上注明关系的类型，关系不能重复但相同的两个 CI 对象之间可以存在多重性。重复和多重性是两个概念，关系不要怕复杂，因为关系越多对以后的定位和查找原因更有帮助。完成后基本上就形成了一种网状图形。

四、 映射关系图到数据库

下面是某个物流公司简化的 CI 关系示意图：



根据以往为企业做咨询过程中的经验，CMDB 构建步骤如下：

1. 选择关系型数据库软件

为什么不使用对象型数据库软件来记录呢？因为对象型数据库技术还不成熟，而关系型数据库技术已经发展得很成熟了。不管在性能和容量方面都是非常优秀的，还有很多开源的关系型数据库产品可以使用，性能也非常优秀，而且这些关系型数据库还富含了很多客户端工具，使用时非常方便。随着 XML 技术的发展，在不同数据库产品之间也可以轻易地进行转换。

2. 把相同的 CI 对象归为一类实体，并把实体映射成数据表，属性映射成字段。

根据上图可以分析出有这样几个实体：服务、应用、服务器、人员、网线、集线器。每一项实体考虑的深度可以自己把握。

3. 确定每一个实体的属性，把握好 “深度”与“广度”原则。例如：服务器性能的关键因素会有，内存，硬盘，CPU 型号等，那这些属性就不能忽略。但有些情况下可以省略一些，

如上图，维护人员张三所维护的网络可能就是一个集线器和两条网线。这样就可以把集线器，网线 A 和网线 B 简化从而抽象成一个配置项——“网络”。

4. 把每一个 CI 对象映射成数据表中的记录。
5. 映射 CI 之间的关系。

简单的关系用数据表的外键可以实现。如果两个对象之间存在多重性关系，那只能通过关联表来实现，也就是把关系的类型映射到这张表中。此表中可以使用组合关键字用于区分不同的关系。上图中的服务器和应用之间就是这样的多重性关系，服务器支持某一个应用，既有逻辑处理支持，又有数据服务支持。这样的话就需要把他们之间的关系映射到一张独立的表中。

五、 如何使用户能够迅速地查询

CMDB 建立起来以后，接下来就是如何让用户能够方便地使用。

1. 从用户的角度来说。最好能提供一张直观技术地图，能够随着关系的变化而实时更新的。可以利用开发工具为用户开发一种图形化的跟踪界面，为用户带来更直观的感受和体验。
2. 对于 CMDB 软件提供商来说，数据库软件本身就提供了一些技术来帮助我们高效地查询信息。
   i. 存储过程：把一些常用的查询定义成存储过程，提高查询的性能。
   ii. 视图：把常用的关联性较强的实体之间关系建立成视图，视图呈现给用户的就已经是具有关联关系的对象了，方便使用者查找和定位。

六、 未来展望

随着国内 IT 服务管理应用地不断推进和深化，未来将有更多的 CMDB 软件推出。对于 CMDB 软件厂商来说，应该摆脱传统的 CMDB 模型来构建，从而提供给用户更为灵活的构建方式。