华 中 科 技 大 学

《数据结构》设计型实验报告

摘要

BOM（物料清单）是企业生产及管理的核心，在每个生产部门都需要对应的 BOM 指导生产上级物料。由多种原材料生产上级材料，再有该类生成材料去生产更高一级的物料，直至最后一个根结点即产品的完成。此一系列的过程的数据存储实现，我们组采用了具有层次结构的多路树。该方法能够形象地展示物料与物料之间的关系，同时也能够动态地进行删除插入等操作。

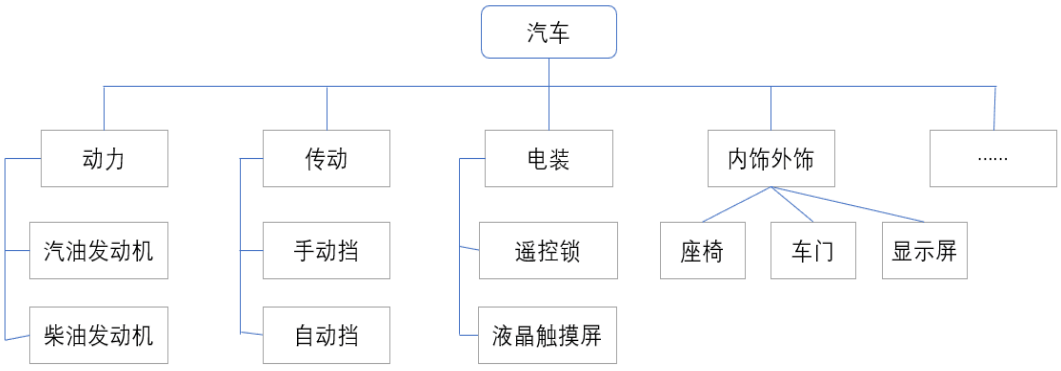
问题一

首先定义配置 BOM（如图 1 以制造汽车为例），在获得客户的订单需求数据并分类 BOM 模块后，从数据库中查找并提取相应信息，构成客户 BOM。

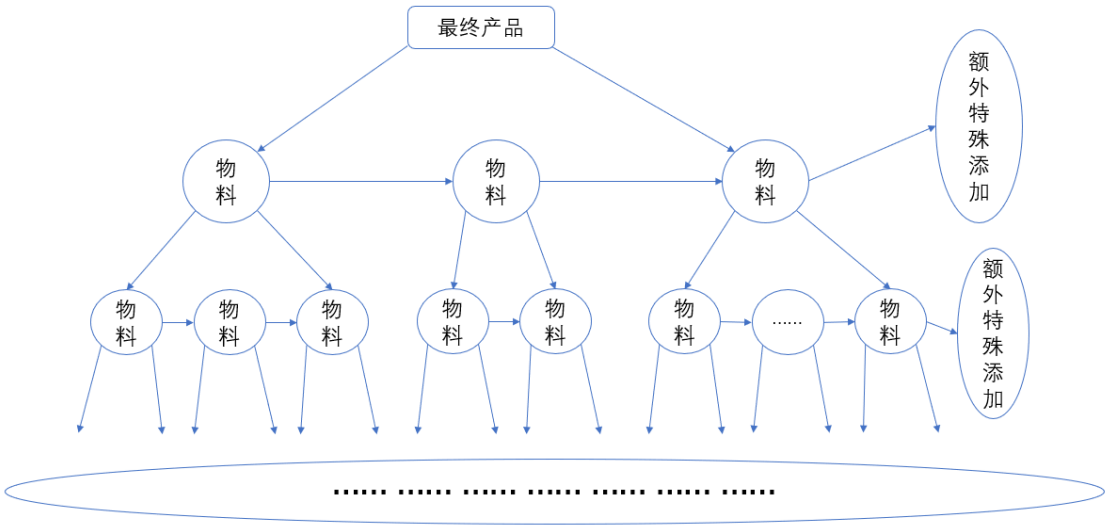
新生成的客户 BOM 以“单父-多子”的树形结构为基础，增加与兄弟节点之间的关系：每一个节点结构体中设置配件编码，单台数量，以及 2 个指向子件的指针域，最后还有指向一个兄弟节点的指针域。最终同一父节点以下同一层次的子节点横向形成一个链表，而父节点的两个指向子件的指针分别指向链表的头和尾部。

在该结构中，按照单机，部件，分部件，零件等顺序在树形结构中存储产品客户订单要求下的生产信息。若客户订单在原本 BOM 的基础上还有额外的附加要求时（即客户要求的是附加部件和零件，并不修改普通情况下的所有部分的属性），则在该层子节点形成链表之后加入新节点存储客户的附加要求。（如图 2）举例：客户 A 订购一辆跑车，根据配置 BOM 选择发动机功率，挡风玻璃类型，制动装置等属性，但是对车的外形装饰有特殊要求比如 A 想要在跑车车门上镶钻，但原本的车门部件下方的子部件没有镶钻这个属性，就在所有车门的孩子节点链表后加上一个新节点加入该要求。

该方法的优点为增加了兄弟节点间的联系同时也没有破坏部件的父子结构。可以根据客户订单自由选择 BOM 中部件的各种属性，添加新属性方便。并且浪费的空间较少。存在的不足为一个父节点只有两个指针指向子结点链表的头尾。当客户 BOM 没有附加要求时，执行创建时要注意层次间的关系，这比普通树型结构要复杂；并且在同一父亲节点的同一层次子结点中，访问就有严格的顺序性；没有办法从子结点找到其父节点。



图片 1



图片 2

问题二

BOM 的配置的规则分为：基于约束的方法、基于实例的方法。

基于约束的方法：

如今企业往往是以模块化的形式制定订单，在独立的模块内产品部件的逻辑关系会影响到该模块是否合乎常理。由此派生出了基于约束的 BOM 配置方法。

约束是产品部件间的组合关系。不是所有产品部件之间都能任意搭配，因为有的之间存在着逻辑上的互斥关系。因此在转换过程中需建立起相对应的互斥约束。同理，部分产品部件之间也存在着依赖的逻辑关系，所以也应建立相对立的依赖关系，以保证互相依赖的部件能被同时选择。当存在新添加的部件时意味着新的约束也应对应建立。在可选件和必选件之中在约束的条件下分别成立模块、模板，以便利客户的合理选择。经过建立约束后，系统会自动根据已选对象过滤掉不合理的模板，以提高组合新产品的效率

优点突出，可以较灵活地响应客户对产品需求的变化，对承接大规模定制任务有一定的参考价值。劣势为产生新的约束或移除已有约束时须人工分清楚逻辑关系，在此过程中可能会出现较大纰漏，智能程度待进一步提高。

基于实例推理的方法：

企业可以参考市面上已有同类别产品实例的部件详细属性信息，再拿顾客提出的属性要求与其对比，可以依据实例得出和顾客需求最贴切的配置实例。实际运用层面，可将各实例属性值与顾客提交的属性值作差并将得到的新矩阵归一化并乘以个属性对应的权重，加和，得到最终的偏差值，偏差值最小的配置案例即为和顾客需求最为相似的实例，即企业可按照此实例生产。

优点为对人工操作部分要求较低，无需企业或是顾客自行搭配选择产品的各个配件；可以满足顾客较为抽象的需求。缺点也十分明显，该方法缺乏创新

性，只生产市面上已有的产品，企业会缺乏一定的竞争力；对顾客具体要求的响应灵活程度不高，不能满足各式各样的顾客需求。

问题三

BOM 物料的查询：

首先是根据节点给定的信息在需要编辑的 BOM 中查询出需要编辑的节点，这过程需要运用到 BOM 遍历算法。查询算法的核心步骤是每遍历一个节点，需要判断该节点信息是否与给定的节点信息一致。如果一致说明节点是需要编辑的节点，然后结束遍历；如果不一致则继续遍历 BOM。

算法步骤：

1) 首先创建临时表和动态数组，然后将所需遍历产品的所有物料存储在该表中。
2) 查找产品的第一层物料，A、B 和 C，输出物料信息，条件是第一层节点右指针空为止。并将 ABC 物料编码存储在动态数组中。
3) 按顺序先判断 A 是否有子物料，然后将 A 的子物料 D 和 E 的信息输出，判断 D 和 E 是否是底层物料，否将 D 和 E 物料存储在动态数组中，是不需要将 D 和 E 添加存储在数组中，并将 A 从临时表中删除，同时将 A 从动态数组中移除。然后以同样方式操作 B 和 C 物料，此时第二层的物料遍历完成。
4) 因为此时数组中存储了第二层物料 D、E、F、G、H 的编码，所以可以根据编码按序以同样方法遍历第三层，遍历结束条件就是动态数组大小不在变化。

方案的优劣：优势是采用改进的分层算法，它将父节点的全部子节点存放在动态数组和临时表中，通过遍历动态数组、临时表完成 BOM 的一层遍历，这样按层遍历的同时可以体现产品的父子关系，遍历效率会比递归高，并且遍历一个节点会将节点从临时表中删除加快遍历效率，也方便对物料的添加和修改等。劣势是增加了额外的存储空间。

时间复杂度 O(n)

额外新增空间复杂度 O(m)，m 为遍历层的所有的节点个数

添加：

查询到添加的节点的位置，更改相关指针的指向

时间复杂度 O(n)

额外新增空间复杂度 O(k)，k 为新增节点的个数

删除：

进行节点批量删除，首先需在 BOM 中查询出需要删除子树的根节点，然后从子树的根节点作为起始节点遍历子树，对遍历的每个节点进行单节点删除操作。

时间复杂度 O(n)

替换：

首先需要节点批量删除操作来删除被替换的部件，即需要以被替换件 C 为根节点遍历 BOM；每遍历一个节点后使用 BOM 单节点删除操作，直到遍历完整个部件 C。查找出需要替换的产品部件 C1，将替换件添加到 BOM 中首先需

要以 C1 节点为根节点遍历“部件 BOM”，每遍历一个节点后使用 BOM 单节点添加操作，直到遍历完整个部件 C1。

时间复杂度 O(n)

总结与反思

在阅读了一定量的相关文献后，我们组成员对 BOM 有了一定的了解。BOM 在生产计划中起着相当大的作用，一个好的 BOM 结构设计以及存储设计可以直接指导生产人员进行物料的收集和匹配，省去了企业内部协调矛盾的时间以及人员浪费。我们组主要针对了 BOM 的存储结构进行了较长时间的讨论，虽然到最后的成果不一定适应于 BOM 的各个要求，但也存在一定的可行性和可取之处。我们的主要思想体现在第一题的图 2。使用一个节点两个指针而不是传统的多路树多指针的结构是因为要区分固定物料以及客户特殊需求所添加的物料。在父节点两个指针中间的为企业固定物料，在最后一个节点横向指向的为客户特殊添加的物料。而同层节点之间使用链表可以利用链表的动态添加以及删除等功能，更方便实现物料的更新。在第三题中使用了动态数组及临时表，利用了课程中排序算法的暂存单元思想，可以实现指针无法回退遍历的操作。

在算法结构设计的过程中，我们也出现了很多问题。刚开始想过使用递归以及矩阵的方法实现 BOM 的生成，但就递归的深度以及矩阵的维度分析后改变了我们的设计方向。对于 BOM，我们组的结构设计其实还有很大的不足，并没有非常明确地指出如何实现多客户的需求，局限在了小范围的结构设计中，没有考虑大的数据量如何与我们的数据存储方式进行匹配。