



[12] 发明专利申请公开说明书

[21] 申请号 95194055.4

[43]公开日 1997 年 6 月 18 日

[11] 公开号 CN 1152365A

[22]申请日 95.6.5

[30]优先权

[32]94.6.6 [33]FI[31]942663

[32]94.6.6 [33]FI[31]942664

[86]国际申请 PCT/FI95/00319 95.6.5

[87]国际公布 WO95/34155 英 95.12.14

[85]进入国家阶段日期 97.1.9

[71]申请人 诺基亚电信公司

地址 芬兰埃斯波

[72]发明人 马蒂·迪卡南

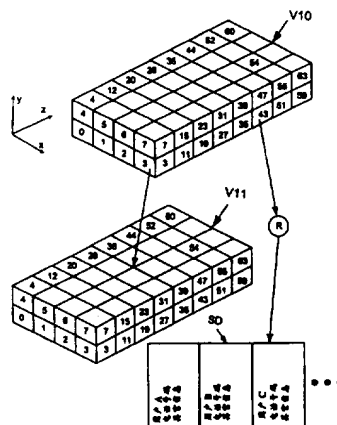
[74]专利代理机构 中国国际贸易促进委员会专利商标
事务所
代理人 杨晓光

权利要求书 6 页 说明书 12 页 附图页数 3 页

[54]发明名称 一种存储和检索数据的方法和一种存储器配置

[57]摘要

本发明涉及一种存储和检索方法以及一种存储器配置。通过在具有不同层次节点的目录结构上进行关键字搜索来执行存储。单个节点可以是 (i) 一个包括多维数组的内部节点，或者 (ii) 一个包含数据单元或数据存储单元的指针的叶节点。为了最小化所需的存储空间，存储通过下述方式实现：(a) 选择预定数量 (k_i) 的比特形成搜索字，基于该搜索字，从树形结构的根层次上的内部节点开始搜索下一个节点的地址，一直进行到所述的节点，(b) 从对应于每一维的搜索关键字未被选中的比特中，选出预定数量的比特 (k_i) 形成搜索字，基于该搜索字，从该已访问节点的数组开始搜索低层次上进一步的新节点的地址，(c) 重复步骤(b)直到遇到一个空元素或低层次新节点的地址是一个叶节点的地址，(d) 在叶节点中存储该数据单元或指向该数据单元的指针。



权 利 要 求 书

1.一种在存储器中存储由搜索关键字标识的数据的方法,数据作为数据单元存储在为每一个数据单元分配的专用存储空间内,其中基于需要存储的数据单元的相关的搜索关键字,在一个目录结构中进行存储,该目录结构包括一个具有不同层次节点的树形层次结构,其中单个节点可以是

- 一个内部节点,包括一个多维数组,其中单个元素可以包含树形层次结构的低层次节点的地址,和其中单个元素可以是空的,或

- 一个叶节点,包括至少一个指向存储数据单元的指针,其特征在于以下步骤:

- (a) 从对应于每一维的搜索关键字中选择预定特定维数 (k_i) 的比特并形成搜索字,基于该搜索字,从树形层次结构的根层次上的内部节点搜索下一个节点的地址,并进行到所述的节点,

- (b) 从对应于每一维的搜索关键字未被选中的比特中,选出预定特定维数 (k_i) 比特并形成搜索字,基于该搜索字,从该已访问节点的数组搜索低层次上进一步的新节点的地址,

- (c) 重复步骤 (b) 直到遇到一个空元素或直到低层次新节点的地址是一个叶节点的地址,

- (d) 在叶节点中存储一个指针,该指针指向所述数据单元的存储单元。

2.权利要求 1 中的方法,其特征在于:如果需要,执行存储的新的叶节点在步骤 (c) 后以一种众所周知的方式创建。

3.权利要求 1 中的方法,其特征在于:所述特定维数 (k_i) 以下述方式选出:该结构在所有维的方向上具有相同深度,因而在所有方向上部分关键字检索具有同样效果。

4.权利要求 1 中的方法,其特征在于:所述特定维数 (k_i) 以下述方式选出:该结构在给定维的方向上具有较大的深度,因而在该方向上部分关键字检索具有同样效果。

5.权利要求 1 中的方法,其特征在于:搜索字由从每一维的搜索关键字

中相继选出的比特组成，将获取的比特串解释成指示内部节点数组的数字，从该内部节点的元素继续搜索低层次节点的地址。

6.一种在存储器中存储由搜索关键字标识的数据的方法，数据作为数据单元存储在为每一个数据单元分配的专用存储空间内，其中基于需要存储的数据单元的相关的搜索关键字，在一个目录结构中进行存储，该目录结构包括一个具有不同层次节点的树形层次结构，其中单个节点可以是

- 一个内部节点，包括一个多维数组，其中单个元素可以包含树形层次结构的低层次节点的地址，和其中单个元素可以是空的，或

- 一个叶节点，其中至少存储一个数据单元，其特征在于以下步骤：

- (a) 从对应于每一维的搜索关键字中选择预定特定维数 (k_i) 的比特并形成搜索字，基于该搜索字，从树形分级结构的根层次上的内部节点搜索下一个节点的地址，并进行到所述的节点，

- (b) 从对应于每一维的搜索关键字未被选中的比特中，选出预定特定维数的 (k_i) 比特并形成搜索字，基于该搜索字，从该已访问节点的数组开始搜索低层次上进一步的新节点的地址，

- (c) 重复步骤 (b) 直到遇到一个空元素或直到低层次新节点的地址是一个叶节点的地址，

- (d) 在叶节点中存储所述数据单元。

7.权利要求 6 中的方法，其特征在于：如果需要，执行存储的新的叶节点在步骤 (c) 后以一种众所周知的方式创建。

8.权利要求 6 中的方法，其特征在于：所述特定维数 (k_i) 以下述方式选出：该结构在所有维的方向上具有相同深度，因而在所有方向上部分关键字检索具有同样效果。

9.权利要求 6 中的方法，其特征在于：所述特定维数 (k_i) 以下述方式选出：该结构在给定维的方向上具有较大的深度，因而在该方向上部分关键字检索具有同样效果。

10.权利要求 6 中的方法，其特征在于：搜索字由从每一维的搜索关键字中相继选出的比特组成，将获取的比特串解释成指示内部节点数组的数字，从该内部节点的元素继续搜索低层次节点的地址。

11.一种在存储器中检索由搜索关键字标识的数据的方法，数据作为数

据单元存储在为每一个数据单元分配的专用存储空间内，其中基于搜索关键字，在一个目录结构中进行检索，该目录结构包括一个具有不同层次节点的树形层次结构，其中单个节点可以是

- 一个内部节点，包括一个多维数组，其中单个元素可以包含树形层次结构的低层次节点的地址，和其中单个元素可以是空的，或

- 一个叶节点，包括至少一个指向存储数据单元的指针，其特征在于以下步骤：

- (a) 从对应于每一维的搜索关键字中选择预定特定数 (k_i) 的比特并形成搜索字，基于该搜索字，从树形层次结构的根层次上的内部节点搜索下一个节点的地址，并进行到所述的节点，

- (b) 从对应于每一维的搜索关键字未被选中的比特中，选出预定特定维数 (k_i) 比特并形成搜索字，基于该搜索字，从该已访问节点的数组开始搜索低层次上进一步的新节点的地址，

- (c) 重复步骤 (b) 直到遇到一个空元素或直至低层次新节点的地址是一个叶节点的地址，或直至已经选出搜索关键字的所有比特；以及

如果在步骤 (c) 中没有碰到一个空元素，

- (d) 从叶节点读出包含的指针，从该指针所指向的单元中读取数据单元。

12. 权利要求 11 中的方法，其特征在于：所述特定维数 (k_i) 以下述方式选出：该结构在所有维的方向上具有相同深度，因而在所有方向上部分关键字检索具有同样效果。

13. 权利要求 11 中的方法，其特征在于：所述特定维数 (k_i) 以下述方式选出：该结构在给定维的方向上具有较大的深度，因而在该方向上部分关键字检索具有同样效果。

14. 权利要求 11 中的方法，其特征在于：搜索字由从每一维的搜索关键字中相继选出的比特组成，将获取的比特串解释成指示内部节点数组的数字，从该内部节点的元素继续搜索低层次节点的地址。

15. 一种在存储器中检索由搜索关键字标识的数据的方法，数据作为数据单元存储在为每一个数据单元分配的专用存储空间内，其中基于搜索关键字，在一个目录结构中进行检索，该目录结构包括一个具有不同层次节

点的树形层次结构，其中单个节点可以是

- 一个内部节点，包括一个多维数组，其中单个元素可以包含树形层次结构的低层次节点的地址，和其中单个元素可以是空的，或

- 一个叶节点，其中至少存储一个数据单元，其特征不在于以下步骤：

- (a) 从对应于每一维的搜索关键字中选择预定特定维数 (k_i) 的比特并形成搜索字，基于该搜索字，从树形层次结构的根层次上的内部节点搜索下一个节点的地址，并进行到所述的节点，

- (b) 从对应于每一维的搜索关键字未被选中的比特中，选出预定特定维数 (k_i) 比特并形成搜索字，基于该搜索字，从该已访问节点的数组开始搜索低层次上进一步的新节点的地址，

- (c) 重复步骤 (b) 直到遇到一个空元素或直至低层次新节点的地址是一个叶节点的地址，或直至还经选出搜索关键字的所有比特；以及

- 如果在步骤 (c) 中没有碰到一个空元素，

- (d) 从叶节点读出一个或多个数据单元。

16. 权利要求 15 中的方法，其特征在于：所述特定维数 (k_i) 以下述方式选出：该结构在所有方向上具有相同深度，因而在所有方向上部分关键字检索具有同样效果。

17. 权利要求 15 中的方法，其特征在于：所述特定维数 (k_i) 以下述方式选出：该结构在给定维方向上具有较大的深度，因而在该方向上部分关键字检索具有同样效果。

18. 一种在存储器中存储由搜索关键字标识的数据的方法，数据作为数据单元存储在为每一个数据单元分配的专用存储空间内，其中基于需要存储的数据单元的相关的搜索关键字，在一个目录结构中进行存储，该目录结构具有一个包括多维数组的节点，该多维数组的单个元素可以包含指向数据单元的指针，或该多维数组的单个元素也可以是空的，其特征在于：

从对应于每一维的搜索关键字中选择预定特定维数 (k_i) 的比特并形成搜索字，基于该搜索字，在该节点的数组中搜索一个元素，如果所述元素是空的，在该元素中存储一个指针，该指针指向所述元素的存储地址。

19. 权利要求 18 中的方法，其特征在于：搜索字由从每一维的搜索关键字中相继选出的比特组成，将获取的比特串解释成指示所述元素的数字。

20.一种在存储器中存储由搜索关键字标识的数据的方法，数据作为数据单元存储在为每一个数据单元分配的专用存储空间内，其中基于需要存储的数据单元的相关的搜索关键字，在一个目录结构中进行存储，该目录结构具有一个包括多维数组的节点，该多维数组的单个元素可以包含一个数据单元，或该多维数组的单个元素也可以是空的，其特征在于：

从对应于每一维的搜索关键字中选择预定特定维数（ k_i ）的比特并形成搜索字，基于该搜索字，在该节点的数组中搜索一个元素，如果所述元素是空的，在所述元素中存储该数据单元。

21.权利要求 20 中的方法，其特征在于：搜索字由从每一维的搜索关键字中相继选出的比特组成，将获取的比特串解释成指示所述元素的数字。

22.一种在存储器中检索由搜索关键字标识的数据的方法，数据作为数据单元存储在为每一个数据单元分配的专用存储空间内，其中基于需要存储的数据单元的相关的搜索关键字，在一个目录结构中进行检索，该目录结构具有一个包括多维数组的节点，该多维数组的单个元素可以包含指向所述数据单元的指针，或该多维数组的单个元素也可以是空的，其特征在于：

从对应于每一维的搜索关键字中选择预定特定维数（ k_i ）的比特并形成搜索字，基于该搜索字，在该节点的数组中搜索一个元素，如果所述元素是空的，在所述元素中存储一个指针，该指针指向数据单元的存储地址。

23.权利要求 22 中的方法，其特征在于：搜索字由从每一维的搜索关键字中相继选出的比特组成，将获取的比特串解释成指示所述元素的数字。

24.一种在存储器中检索由搜索关键字标识的数据的方法，数据作为数据单元存储在为每一个数据单元分配的专用存储空间内，其中基于需要存储的数据单元的相关的搜索关键字，在一个目录结构中进行存储，该目录结构具有一个包括多维数组的节点，其中单个元素可以包含一个数据单元，或单个元素也可以是空的，其特征在于：

从对应于每一维的搜索关键字中选择预定特定维数（ k_i ）的比特并形成搜索字，基于该搜索字，在该节点的数组中搜索一个元素，如果所述元素是空的，在所述元素中存储该数据单元。

25.一种以电气形式存储数据的相关联存储器配置，数据作为数据单元

存储在存储器中为每一个数据单元分配的专用存储空间内，存储器配置包括一个目录结构，该目录结构是一个多维数字检索结构，包括一个具有不同层次节点的树形层次结构，其中一个节点可以是

- 一个内部节点，包括一个多维数组，其中单个非空元素包含一个指针（31）引导在目录结构中的处理，或

- 一个叶节点，包括一个或多个指向存储器中存储数据单元的指针，

每一个需要在存储器中存储的数据单元与一个多维搜索关键字相关联，该搜索关键字包括每一维上的一个搜索关键字，在此基础上，在目录结构上进行关键字处理，在每一种情况下，目录结构返回与所用的多维搜索关键字相关联的数据单元，其特征不在于，通过在独立于其它维的每一维中，预定需要搜索的搜索关键字比特的数量 k_i ，固定该目录结构中单个多维内部节点的数量，每一维上内部节点的数量是 2^{k_i} 。

26.一种以电气形式存储数据的相关联存储器配置，在存储器中，数据作为数据单元存储为每一个数据单元分配的专用存储空间内，存储器配置包括一个目录结构，该目录结构是一个多维数字检索结构，包括一个具有不同层次节点的树形层次结构，其中一个节点可以是

- 一个内部节点，包括一个多维数组，其中单个非空元素包含一个指针（31），引导在目录结构中的处理，或

- 一个叶节点，至少包含一个数据单元，

每一个需要在存储器中存储的数据单元与一个多维搜索关键字相关联，该搜索关键字包括每一维上的一个搜索关键字，在此基础上在目录结构上进行关键字处理，在每一种情况下，目录结构返回与所用的多维搜索关键字相关联的数据单元，其特征不在于，通过在独立于其它维的每一维中，预定需要搜索的搜索关键字比特的数量 k_i ，固定该目录结构中单个多维内部节点的数量，每一维上内部节点的数量是 2^{k_i} 。

说明书

一种存储和检索数据的方法和一种存储器配置

本发明涉及根据后附权利要求 1, 6, 18 和 20 的前序的一种数据存储方法, 以及根据后附权利要求 11, 15, 22 和 24 的前序的一种数据检索方法。本发明也涉及根据后附权利要求 25 和 26 的前序的相联存储器配置。

本发明的实现方法主要用于集中存储数据库的连接。一种合适的应用是在电话交换机中用户数据库的维护。

现有技术的单维目录结构, 称为数字检索 (Digital trie) (单词 “trie” 是从英语单词 “retrieval” 中派生的), 是本发明基本原则的潜在基础。数字检索是一个树状结构, 由两类节点组成: 叶节点, 包括一个记录, 和引导搜索的内部节点。一个内部节点是一个有 2^k 个元素的数组。如果一个元素被使用, 它或者指向目录树下一层的内部节点, 或者指向包含一个记录的叶节点。在其它情况下, 该元素是空的 (空)。对数据库的搜索是通过一次检查搜索关键字 (例如, 在电话交换机用户数据库的情况下, 一般是对应于该用户电话号码的二进制数字) 的 k 个比特进行的。需要搜索的比特以下述方式选出: 在树形结构的根层次 (在第一个内部节点), 搜索最左的 k 个比特; 在结构的第二层次, 搜索这些最左比特的下一组 k 个比特, 以此类推。需要搜索的比特被解释成无符号二进制整数, 直接用作元素数组的索引 (该索引指示了数组中的一个给定元素)。如果该索引指示的元素是空的, 搜索以不成功结束。如果该元素指向下一层的一个内部节点, 从搜索关键字中抽取下一组 k 个比特在该层按上述方式进行搜索。经过比较之后, 寻径或者向下一层的一个内部节点分支, 或者指向通常包含一个关键字—指针对的叶节点。如果该元素指向一个包含记录的叶节点, 则比较搜索关键字和该记录中存储的关键字。这样, 仅在搜索发现一个叶节点之后才比较整个搜索关键字。当关键字相等时, 搜索成功, 所需的数据单元从该叶节点的指针所指示的存储地址中获取。当关

键字不等时，搜索以不成功终止。

图 1 说明了一个数字检索结构的示例，其中关键字长 4 比特， $k=2$ ，这样每一个内部节点有 $2^2=4$ 个元素，在每一层搜索从关键字中抽取出的 2 个比特。包含记录的叶节点以参数 A, B, C, D...H...M, N, O 和 P 标明。这样，一个叶节点是一个不指向树中低层的节点。图 1 中内部节点以参数 V1...V5 标明，内部节点元素由参数 VA 标明。

在图 1 所示的例示性例子中，叶节点的搜索关键字按下述方式示出：A=0000, B=0001, C=0010, ..., H=0111, ...和 P=1111。在这种情况下，每一个叶节点中存储的指针指向数据库 SD 的某个存储单元，在该存储单元中可以查找到实际数据，例如，有关用户的电话号码，以及该用户的其他相关信息。实际用户数据可以是，例如图 1 示出的顺序文件类型，存储在数据库中。搜索基于记录 H 的关键字进行，例如，首先抽取关键字最左的两位比特（01），并予以解释，向节点 V1 的第二个元素传送，该元素含有指向下一层节点 V3 的指针。在这一层上，抽取搜索关键字紧接着的两位比特（11），然后得出该节点的第四个元素，指向记录 H。

除了指针，叶节点也可以包含（除搜索关键字之外）一个实际的数据文件。例如，与用户 A（图 1）的相关数据可以放置于叶节点 A，与用户 B 的相关数据可以放置于叶节点 B，等等。这样，在第一种实施例（其中叶节点存储了一个关键字 - 指针对）中，数字检索结构仅作为一个目录结构，在第二种实施例（其中叶节点存储了关键字和实际数据）中，数字检索结构作为一块相联存储器。（在相联存储器中，存储单元基于它们的内容标识。）

在本发明的实现中，与上述情况不同，搜索关键字是多维的，也就是说，搜索关键字包括许多属性（例如，用户的姓氏和一个或多个家族名），这样在每一个属性中，可以一次搜索一个属性的 k_i 个搜索关键字比特。（ i 是对应于一个属性的索引）。

以前，上述的单维检索结构被扩展成一个多维“EXCELL”结构（英语词汇“extendible cell”的缩写），以支持与几何应用相关搜索的最大效率。EXCELL 结构包括单元划分搜索空间，和一个独立目录部分。单元划分在搜索空间的任一个轴线方向上将搜索空间递归划分成两个相等的部

分。单元划分的结果是多维检索结构。目录部分是表格形式的栅格，其中每一个栅格等于搜索空间中的最小单元。栅格元素指向搜索空间中的一个单元，并且多于一个栅格元素可以指向同一个单元。（EXCELL 结构在 Tamminen, M.: The EXCELL method for efficient geometric access to data, Helsinki 1981, Acta Polytechnica Scandinavica, Mathematics and Compute Science Series #34 中描述，感兴趣的读者可以参考该资料以获取更详尽的说明。）

这样一种多维检索结构，其中单元大小是动态的（当数据进入该结构时目录部分加倍），伴随着由特定类型（不统一的）关键字分布导致的目录部分快速（指数级）增长。其结果是，存储占有率（存储器电路要求）将过度增长，存储管理效率（速度）将受到影响。

本发明的目标是消除上述问题，并提供一种存储器，其中目录部分可以容纳在比现有技术要小的存储空间，而且比以前更容易管理。这个目标由本发明的存储和检索方法实现，存储方法的基本特征在于，（a）从对应于每一维的搜索关键字中选择预定数量(k_i)的比特，组成一个搜索关键字，基于该搜索关键字从树状层次结构的根层次的内部节点查找下一个节点的地址，进行到所述节点，（b）从对应于每一维的搜索关键字未被选中的比特中选择预定数量(k_i)的比特，组成一个搜索关键字，通过该搜索关键字从已被访问的节点数组中查找低层次进一步新节点的地址，（c）重复步骤(b)直到遇到了一个空元素或直到低层次新节点的地址是一个叶节点的地址，和（d）在叶节点中存储一个指针，将所述数据单元存储在该指针指示的存储单元或者将所述数据单元存储在该叶节点。根据本发明的检索方法的基本形式，其特征在后附权利要求 11 和 15 的特征部分提出。对上述基本形式而言，目录结构仅包括一个节点的特定例子也适用。根据该实施例的存储方法，其特征在后附权利要求 18 和 20 的特征部分提出，其检索方法的特征相应在后附权利要求 22 和 24 的特征部分提出。本发明的相联存储器配置相应地在后附权利要求 25 和 26 的特征部分提出。

本发明的想法是在多维检索结构中以下述方式执行地址运算：在每一维中独立于其它维选择预定数量的比特，一种典型实现是为了计算一个元素数组的索引，从不同维的搜索关键字中选择不等数量的比特。因此，通

过预先给定在每一维中需要搜索的搜索关键字比特数量，检索结构中的单个内部节点的每一维都设置一个独立于其它维的固定限制。这样，在每一维中的选择独立于其它维中的选择。这种方式一方面避免了横向上结构的过分增长，另一方面避免了在任意维深度方向上结构的过分增长（深度，也就是说，每一维上的层次数量，等于搜索关键字总长度的比特数除以一次搜索的比特数量，也就是说，深度受决定内部节点多少的参数影响）。这样，通过在宽度和深度方向上预先给目录加以特定限制，存储要求得到控制。

通过本发明的实现，硬件中存储器电路的要求将比以前更小，存储管理将是高效的，尤其在预先得知搜索关键字值分布的情况下（例如，电话交换机通常是这种情况）。

本发明的另一个优点在于，可以设计一种结构（通过选择参数）以使完全关键字检索和部分关键字检索在所有方向上都是相等的。这种方法也允许仅在特定维上进行的检索，也就是说，部分关键字检索，具有最大效率。这些特性是非常重要的，尤其在需要高速的实时数据库中。在速度方面的一个进一步优点在于，这种实现允许高效地并行处理（换句话说，如果例如查找分支成两个子树，每一个可以在独立处理器中进行处理）。

以下参照图 2 到 6，通过相应图片的例子，解释本发明和它的优选实施例，其中

图 1 说明了在电话交换机中用以维护用户数据的一种单维数字检索结构，

图 2 说明了一种多维数字检索结构，

图 3 说明了根据本发明的检索结构的内部节点中元素的结构，

图 4 说明了根据本发明的存储器所使用的目录结构的节点，

图 5 说明了判定内部节点的元素（元素数组的索引）的方法，内部节点与图 4 一致，以及

图 6 在框图层次上说明了根据本发明的一种存储器配置。

如前面所述，在本发明中检索结构有多维实现（一般 n 维）。这样一种多维结构在最初时与所描述的单维结构完全类似，只是在内部节点中包括的数组元素是多维的。图 2 示例给出了一种二维 $2^2 \times 2^1$ 结构，其中数组元

素的一维包括四个元素，另一维包括 2 个元素。内部节点元素指向的叶节点在图中以圆示出。

在多维情况下的地址运算的原则与单维情况下相同。然而，基本的差别在于，对元素数组（ n 索引）中的每一维计算索引，而不再是一个元素数组索引。这样，每一维有它自己的搜索关键字空间 $\{0, 1, \dots, 2^{v_i}-1\}$ （ v_i 是每一维数上搜索关键字长度的比特数， $i \in \{1, \dots, n\}$ ）。

为了表示内部节点中单个元素 VA 的状态，首先定义术语溢出元素的意义。当向目录树增加一个新指针时（作为数据插入存储器的结果），可能在该内部节点的位置上已经有一个指向某个叶节点的指针。这种情况称为冲突。有许多方法可以解决冲突。从控制存储器管理的角度来看，在目录树的下一层为冲突指针创建一个新的内部节点是没有用的，但是如果该节点含有空元素，该指针可置于发生冲突的内部节点的元素中。这样的元素称为溢出元素，这样它们包括根据上述直接地址运算机制所无法放置的指针。为了能检索到这样的记录，内部节点的元素必须具有一个独立的标识符，以指示在所述元素中包含的指针类型，换句话说，说明它是指向下一层次的内部节点还是一个溢出元素（在同一个内部节点中）。

图 3 例示了根据本发明的存储器中目录结构中内部节点元素的结构。内部节点可以是目录节点或溢出节点类型。如果内部节点没有包含溢出元素，则该内部节点类型是目录节点，如果包含即使一个溢出元素，则该内部节点类型是溢出节点。相应地，一个元素可以是一个目录元素，一个溢出元素或一个空元素（未使用）。一个目录元素可以指向一个目录节点，溢出节点或叶节点。另一方面，一个溢出节点仅指向一个溢出节点或一个叶节点。

在这个例子中的每一个元素长度为 40 比特（5 字节），包括下列字段（括号中为字段长度）：指针字段 31（32 比特），指针类型字段 32（2 比特），和元素类型字段 33（2 比特）。进一步，元素中保留了一个空闲字段 34。指针类型可以通过指针类型字段 32 的两位比特进行编码，例如以下述方式进行：

00：指针未用，

01：指针指向一个目录节点，

10：指针指向一个溢出节点，和

11：指针指向一个叶节点。

元素类型可以通过元素类型字段的两位比特进行编码，例如以下述方式进行：

00：空元素，

01：目录元素，

10：溢出元素，和

11：未用。

在根据本发明的多维检索存储器结构中，基于每一维方向上单个内部节点所需大小，另一方面，基于该维方向上结构的所需深度，通过固定每一维的参数， k_i （该维数需要搜索的搜索关键字比特数量），从而预先决定了每一个内部节点的大小（元素数量）。这样，在每一维数方向上内部节点的大小是 2^{k_i} 个元素，内部节点元素总数 S 也是2的幂：

$$S = \prod 2^{K_i} = 2^{K_1} \times 2^{K_2} \times 2^{K_3} \times \dots = 2^N \quad (1)$$

可以用 n 个（ $n \geq 2$ ）整数指向有 n 个维内部节点中的所有元素，每一个整数值可以在 $\{0, 1 \dots 2^{K_i} - 1\}$ 之间。这样，预定参数包括每一维搜索关键字的总长度和每一维搜索的比特数量。例如，如果在某一维上的搜索关键字最多有256个属性（例如姓氏），搜索关键字的总长度有8比特。如果选择这些比特用于实例1进行一次搜索，在该维方向上结构的深度为 $8/1=8$ 。通过恰当地选取每一维上的搜索比特数量，可以将例如所有维上的深度设成相等。

因此，根据本发明，搜索比特的数量在每一维上可以不同，这样，每一维也可以单独控制所需的存储空间。

图4示出了根据本发明的存储器目录结构中所使用的内部节点V10的例子，该例使用了三维搜索关键字。在第一维方向上（ x ），内部节点有 $2^2=4$ 个元素，在第二维方向上（ y ），内部节点有 $2^1=2$ 个元素，在第三维方向上（ z ），内部节点有 $2^3=8$ 个元素，总共给出了 $2^6=64$ 个元素，编号为0...63。

因为实际硬件实现的存储器空间（例如计算机设备）是单维的，在地址运算操作中（也就是说，在搜索树中处理时），多维数组被线性化，也就是说转换成单维的。线性化是一个可以在各种不同大小数组上执行的算法操作。

在线性化过程中，数组中的元素从 0（如图 4 所示）开始排序，最后一个元素的编号比所有维大小小一。元素的编号是每一个坐标的乘积（例如在三维情况下， x ， y 和 z 坐标）加上它所处维中的编号。计算出的这个编号直接用于单维数组的索引。

在图 4 所示的内部节点的情况下，元素编号 VA_n 根据上述公式进行计算，

$$VA_n = x + y \times 4 + z \times 4 \times 2 \quad (2)$$

其中 $x \in \{0, 1, 2, 3\}$ ， $y \in \{0, 1\}$ ， $z \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ 。这样，例如对元素 54 来说，我们从坐标 $\{2, 1, 6\}$ 中获取该值： $2 + 1 \times 4 + 6 \times 4 \times 2 = 2 + 4 + 48 = 54$ 。

当对 n 维检索结构内部节点的（ n 维）元素数组进行线性化时，根据以上描述，每一维的大小是 2^{K_i} ，其中 k_i 是在所述维上一次需要搜索的比特数量。如果根据该维的坐标由参数 a_j 表示，（ $j \in \{0, 1, 2, \dots, n\}$ ），线性化可以以下述方法给出：

$$\sum_{j=1}^n a_j \prod_{i=0}^{j-1} 2^{K_i}, \forall j: a_j \in \{0, 1, 2, \dots, 2^{K_j}-1\} \wedge k_0 = 0 \quad (3)$$

可以根据公式（3）相乘进行线性化；但是通过从搜索关键字比特中组成一个比特串更为有利，对应数字指示的元素的内容使搜索可以在目录树中继续。相继设置从每一维搜索关键字中抽取的比特组成比特串，并且这样获取的最终比特串作为二进制数字来解释。其结果与公式（3）相同。

图 5 示出了通过相乘进行线性化和通过比特串格式连接图 4 中示出的内部节点的线性化方式。如上所述，在每一维中解释的搜索关键字比特的数量可以按以下方式选取： $k_1=2$ ， $k_2=1$ ， $k_3=3$ 。每一维方向上的搜索关

键字如下： $a_1=1001$ ， $a_2=10$ ， $a_3=110001$ 。在格式化比特串过程中，第一维（在第一维方向上开始对元素的编号，从零开始）中搜索关键字 a_1 的两位最左比特（10）首先被抽取，作为需要组成的比特串中最无关紧要的比特。这由线 51 标出。其次，从第二维的搜索关键字中抽取最左比特（线 52），该比特后缀在需要组成的比特串中最重要的比特（左边）旁。最后，从第三维的搜索关键字中抽取三个最左比特（110，线 53），这些比特作为最左比特插入到比特串中。结果比特串（110110）作为二进制数字解释，可以获取元素 54。根据公式（3）的乘法，也可以获取相同元素，但是在那种情况下，被选的搜索关键字比特首先解释成二进制数字，公式（3）的系数由通过图 5 所示方式获取的数字形成。在元素 54 所含指针的基础上，可以获取下一个内部节点，对未用过的搜索关键字比特重复相同的操作。（在例示性示例中，从该内部节点中选取元素 $9=(001001)_2$ ）。该内部节点的每一个元素可以进一步指向另一个内部节点（如元素 3 指向内部节点 V11）或指向包含记录的叶节点（如元素 43 指向记录 R）。其他情况下，元素未被使用（空）。

通过比特串格式化，乘法操作可以转换成加法操作和比特移位（或加法操作和比特相关 OR 操作）。在数字 2^{K_i} 上的互乘操作被替换成在成分 k_i 上的加法操作。通过比特串格式化，剩余的坐标 a_j 乘 2 的幂可以转换成一个比特移位操作（乘以 2^k 对应于向左移位 k 比特）。因为乘法总是占用至少 13 个时钟周期（例如，使用 Intel i486 处理器），而当操作数存储在处理器寄存器中时，比特移位可以在三个时钟周期中执行完毕，当操作数存储需要在处理器高速缓存中查找时，比特移位可以在四个时钟周期中执行完毕，甚至在操作数需要在存储器中检索时，也只用十个时钟周期。另外，另一方面，如果操作数在寄存器中，则仅占用一个时钟周期，否则 2 - 3 或 8 - 9 时钟周期，这取决于该操作数是否需要在处理器高速缓存中查寻。

通过上述方法形成比特串，其优点在于，它是有序的（换句话说，记录按照它们搜索关键字值的结构组织）。这是因为每次从搜索关键字抽取 k_i 个最重要（最左）比特。然而，原则上，从搜索关键字中抽取 k_i 个最无关紧要（最右）比特，以形成比特串也是可能的。

也可以通过不同于上述的顺序处理该维来形成比特串，但是在上述方

法中，从左边增加需要形成的比特串最为简单，因为在这种情况下不需知道比特串中需要为每一维的比特分配多少位。

根据公式（3）的乘法可以用于线性化（也就是说，计算元素数组索引），而与每一维的大小无关。另一方面，比特串格式化仅用于内部节点大小（元素数量）在每一维方向上都是2的幂的情况。因为根据本发明的目录的内部节点满足这个要求，在根据本发明的检索和存储方法中应用比特串格式化方法是优选的，它比根据公式（3）的乘法更有效（快速）。它的快速是因为比特串格式化将乘法转换成更易于执行的加法和比特移位。

以上提出的一种目录结构可以有所变化，例如在该结构中的叶节点包括多于一个关键字-指针对。其不同点在于叶节点中比较多个搜索关键字，而不是仅比较一个。

除了指针，叶节点中的记录可以包括（除必不可少的搜索关键字之外）实际数据。在这种实施例中，检索结构可以作为一种相关联存储器。

原则上，存储维护包括三种不同操作：检索，增加（存储）和删除。当向实际数据增加一个记录（例如，电话交换机上的用户数据）时，增加操作或者作用在一个叶节点上，或者作用在存储器中其它地方。在后一种情况中，叶节点中存储一个指针，该指针指向数据记录所在的存储地址。

因此搜索以下述方式进行：从对应于每一维的搜索关键字中选择预定数量（ k_i ）的比特，按上述方式形成搜索关键词，基于该搜索关键词，可以找到根层次上的内部节点的元素数组索引。对应于该索引的元素将指明下一个节点的地址，因而可以进行访问。在下一个节点中，从对应于每一维的搜索关键字未被选中的比特中，选出预定数量的比特（ k_i ）形成搜索关键词，用以寻找内部节点元素。搜索关键字比特的选取一直进行到该搜索关键字的所有比特都被选取，或者下一个低层次节点的地址是一个叶节点的地址，然后，或者（a）从该叶节点中读取指针，从该指针所指示的单元读取记录，或者（b）从该叶节点中读取一个或多个包含的记录。如果在目录结构中遇到一个空元素，搜索不成功终止（换句话说，在存储器中没有与关键字相关的数据）。

因此，根据本发明，从对应于每一维的搜索关键字未被选中的比特中，

选出预定数量的比特 (k_i)。在每一维中，选出的比特数量独立于其它维中所选取的比特数量，在同一维中，选择的比特数量保持不变。

当插入数据时，基于搜索关键字，在目录结构中以类似方式进行搜索，直到遇到一个空元素或者直到低层次的新节点的地址是一个叶节点的地址。

在前一种情况下，创建一个叶节点，空元素被设置成指向这个叶节点。然后，数据或者直接插入到叶节点，或者放置在存储器某处；在第二种情况下，在叶节点中存储一个指针，该指针指向增加的记录所在的存储单元。

在后一种情况下，叶节点已经存在，处理取决于该叶节点是否包含插入的空间（换句话说，处理取决于叶节点的大小）。如果，例如仅有一个记录适合插入到叶节点（或者叶节点满），目录结构的深度将增加。在这种情况下，发生冲突的元素被设置指向一个新节点，其中元素数组索引根据关键字计算。如果索引仍然相等，重复进行增加新节点操作。这样，重复增加新节点直至索引不相等。通过识别搜索关键字，在最坏情况下，搜索关键字的所有比特都已经用到，也就是说，该结构的深度已达到最大值，完成增加新节点。如果一个指针存储在溢出元素中，它将再次指向存储实际数据记录的存储地址。仅在目录结构的叶节点中可能出现溢出。

删除数据通过一种所众所周知的基于搜索关键字的查找方式进行，如上所述，包含某个指针和记录的节点将被删除。如果可能，该节点将被释放以保证所占用的存储空间始终保持最小。例如，在节点含有一个指向叶节点的指针，和节点不含指向节点的指针时，释放是可能的。

在每一种特定情况下选择的冲突防止策略通常不仅对插入操作有影响，而且对检索和删除操作如何进行有影响。这样，甚至在它们都采纳本发明的地址计算方法时，实现这些操作的细节仍可能不同。因为，这样的变化已与实际的发明思路无关，并且这种变化所采用的形式是本领域技术人员所熟知的，本文中不再对它进一步解释。

图 6 在框图层次上示出了根据本发明的一种存储器。每一维有一个专用的寄存器，因此有 n 个输入寄存器。每一维的搜索关键字存储在这些输入寄存器中，每一个关键字都存储在它自己的寄存器中。输入寄存器连接寄存器 TR，在 TR 中形成上述的关键词。寄存器 TR 通过加法器 61 连接

到存储器 MEM 的地址输入。存储器输出则连接地址寄存器 AR，地址寄存器 AR 的输出相应连接到加法器 61。最初，从每一个寄存器中选取的比特以正确的顺序读入到通用寄存器 TR。第一个内部节点的初始地址存储在地址寄存器 AR 内，从寄存器 TR 获取的偏移地址加入到加法器 61 中的初始地址中。结果地址提供给存储器 MEM 的地址输入，存储器的数据输出提供了下一个内部节点的初始地址，该地址写入到地址寄存器 AR，覆盖前一个存储的地址。然后，下一次选出的比特又以正确的顺序从输入寄存器装入通用寄存器 TR，获取的数组地址加入到从地址寄存器 AR 中获取的相关数组（也就是说，内部节点）的初始地址中。该地址提供给存储器 MEM 的地址输入，存储器的数据输出随后提供下一个内部节点的初始地址。重复上述过程直到获取所需的节点和记录，或者读到了所需的寄存器。

因为本发明的处理是通过上述比特操作，通过一次从每一维的搜索关键字中选择给定数量比特，所以本发明的快速性能受硬件配置选择的影响。可以通过将单处理器环境转换到多处理器环境以加速地址计算，多处理器环境可以同时进行多个并行处理。

以上存储器的结构的描述，是假定在典型实现的情况下给出的。然而，实际上首先从所有节点中创建一个叶节点，仅在第一个叶节点溢出之后，创建第一个内部节点，上述的目录结构才开始成型。这样，原则上，甚至可能在一个非常大的叶节点（永不会溢出）中实现上述类型的地址计算，该叶节点包括一个多维数组，该数组中每一个元素包含—如果该元素被使用—一个数据单元或一个指向数据单元的指针。因此，在这种情况下，每一维的搜索关键字中只有第一组 k_i 比特被选出。如果通过这些比特查找的元素是空的，那么在该元素中或者存储一个实际数据记录，或者存储一个指向存储数据记录的存储地址的指针。如果冲突情况在存储时发生了，可以通过众所周知的方法加以解决。如果对存储的数据进行适当的分布，这样一种单叶节点实际上是可行的。然而，相对于所公开的多维目录结构的基本实施例而言，其存储空间利用效率较差。

上述额外的单叶节点实施例，在某种意义上，是一种可能的特定实施例，它也具有多维目录结构，并且如前所述（只用仍存在未搜索的比特），地址运算也在叶节点继续。在这种情况下，一个叶节点的结构类似于一个

内部节点。

尽管以上参照实施例，根据相关图片对本发明进行解释，但显然不是非常严格的，在上面所公开的和后附权利要求书的发明思路范围内可以有所改动。

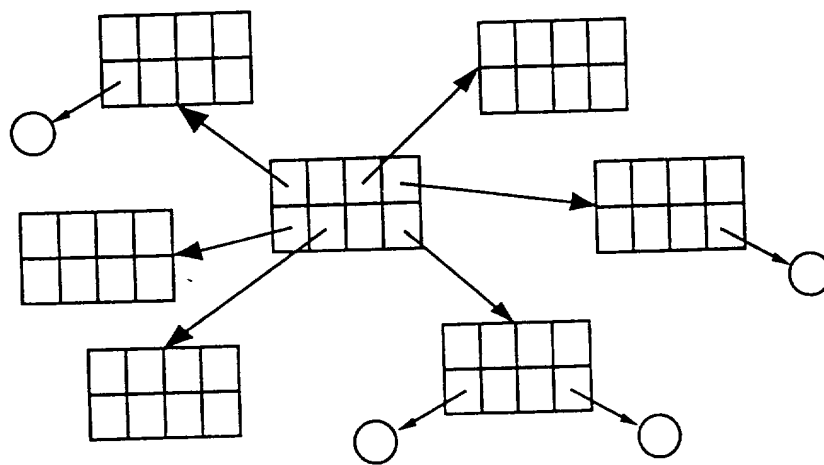
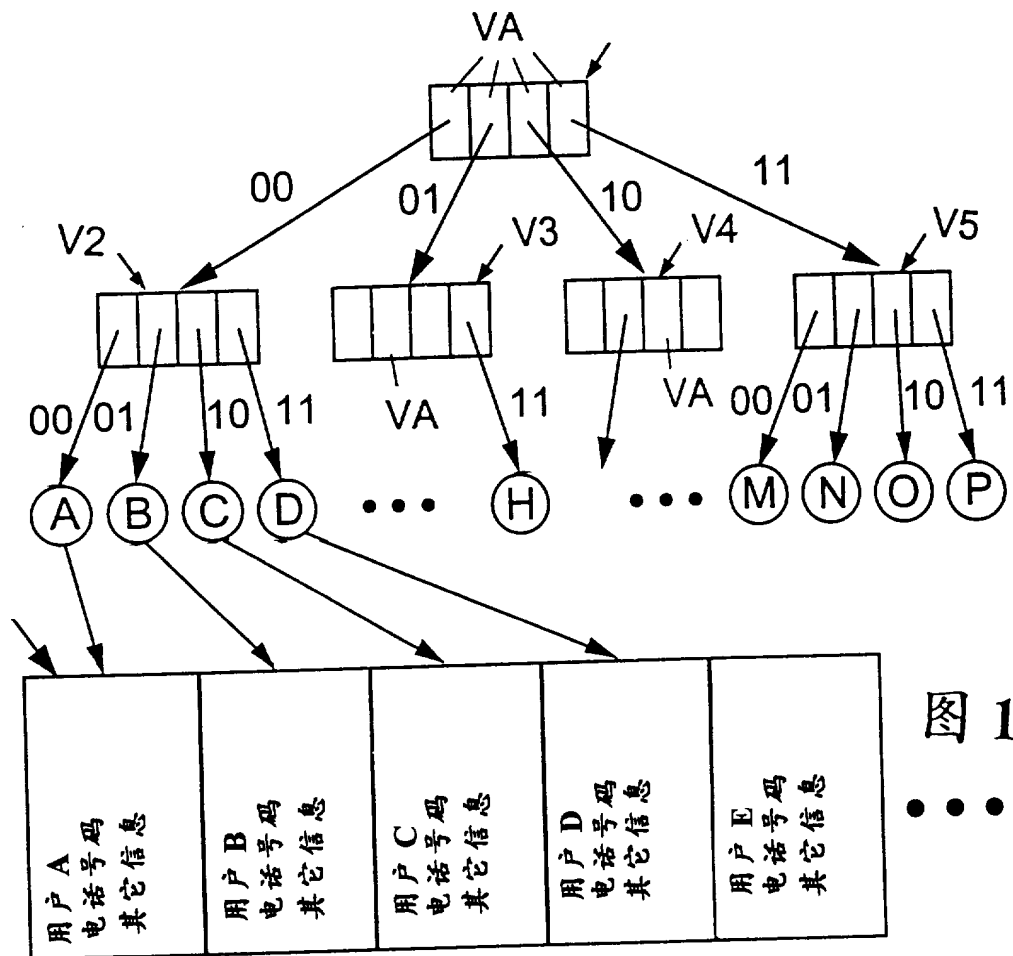


图 2

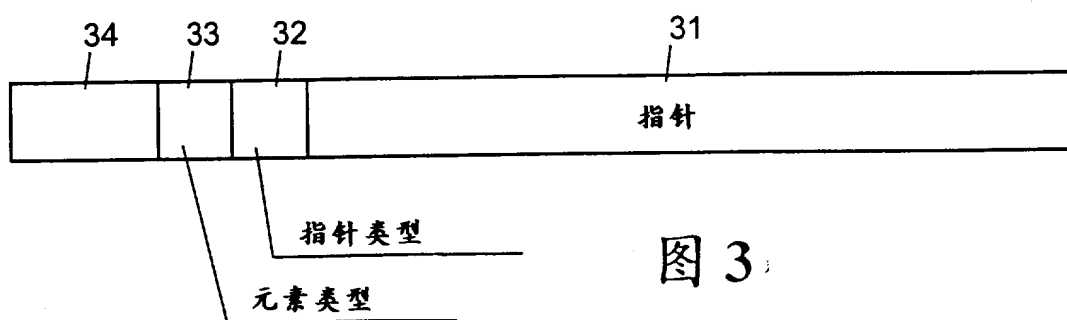


图 3

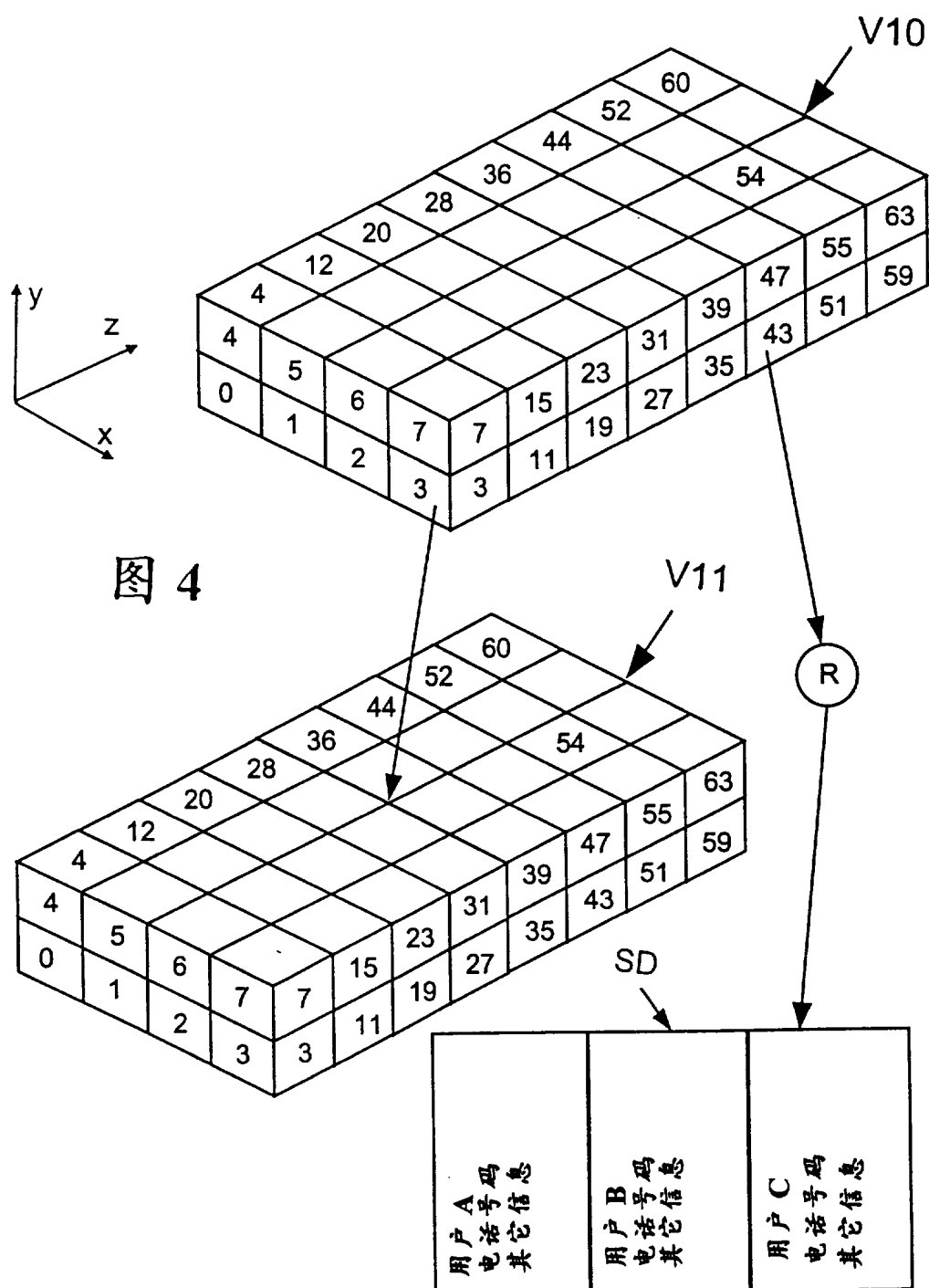


图 4

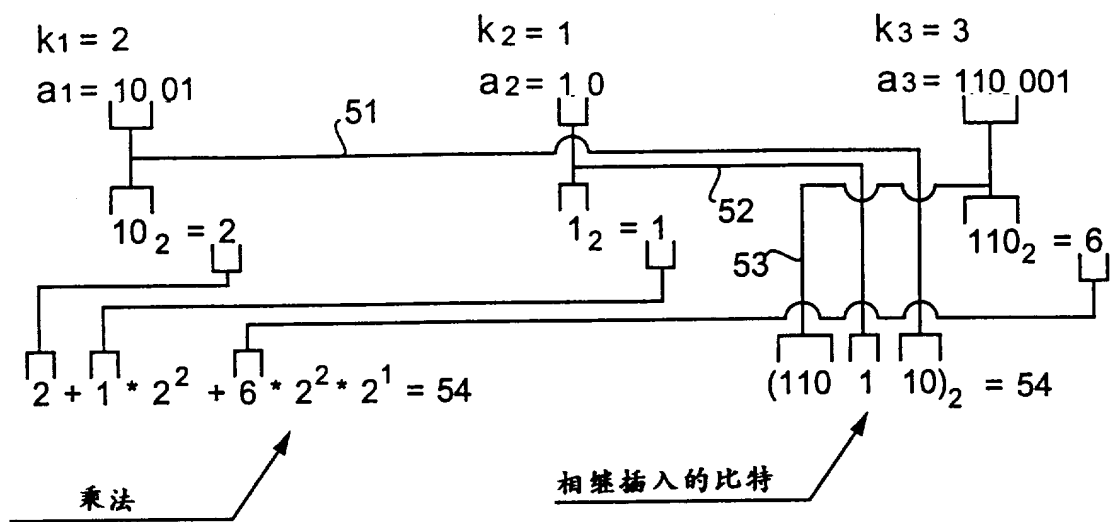


图 5

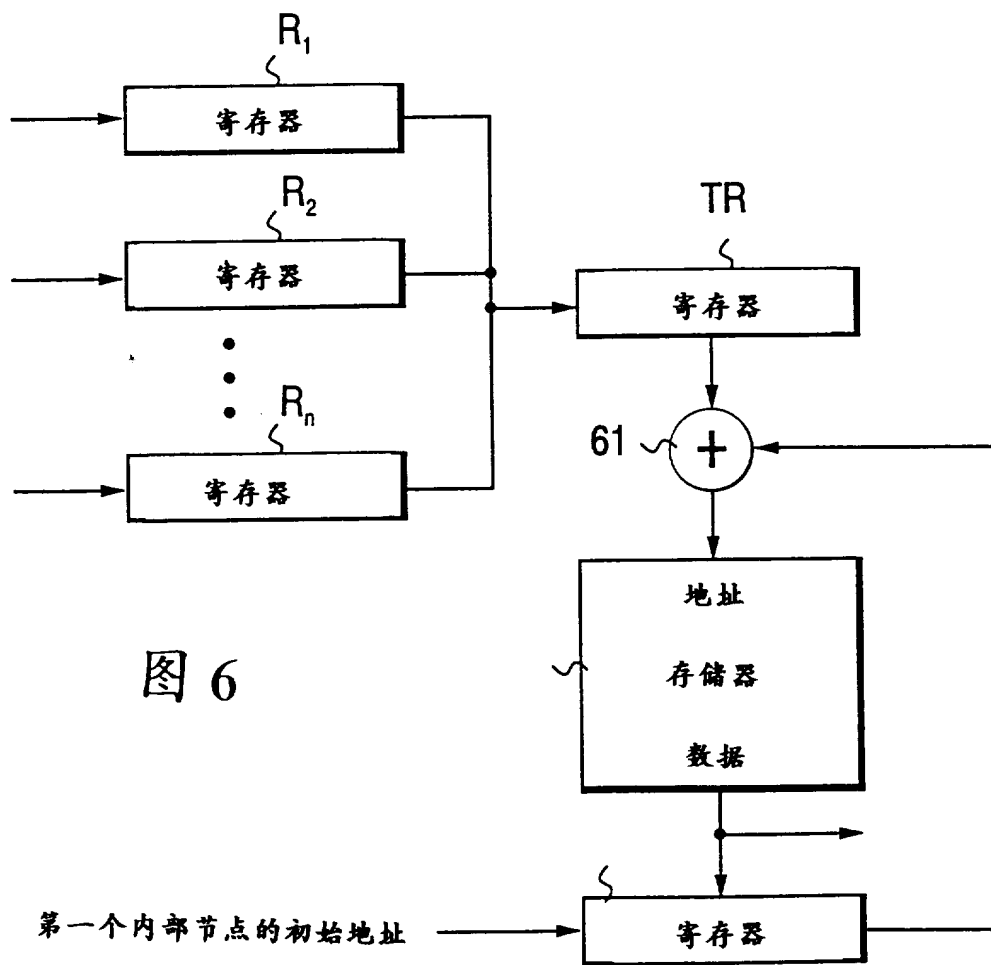


图 6