第5章 大数据的压缩计算方法

本章介绍第二种把大数据计算问题转换为小数据计算问题的方法，即基于大数据压缩的计算方法，简称为压缩计算方法。本章介绍的压缩计算方法是一种精确计算方法，不同于第 4 章介绍的抽样计算方法。5.1 节介绍该方法的基本思想、适用范围、需要解决的问题。5.2 节探讨支持压缩计算的数据压缩方法。5.3 节至 5.7 节以不同的大数据计算问题为例，介绍基于压缩计算方法的算法设计与分析原理。

5.1 压缩计算方法概述

压缩计算方法是把大数据计算问题转换为小数据计算问题的有效方法。压缩计算方法首先以预处理的方式压缩数据集合，然后直接在压缩数据上无解压地求解大数据计算问题。本章介绍的压缩计算方法是精确计算方法，而基于抽样的计算方法是近似计算方法。当然，如果应用压缩计算方法设计大数据计算问题的近似求解算法，可能会更大幅度地降低大数据计算的时间复杂性。

很多大数据计算问题可以使用压缩计算方法实现亚线性时间精确求解。对于那些使用压缩计算方法很难实现亚线性时间精确求解的大数据计算问题，也可以使用压缩计算方法提高大数据计算问题求解的效率。当问题的输入大数据集合不可压缩或压缩比很低时，压缩计算方法就无能为力了。本章既考虑如何使用压缩计算方法实现亚线性时间算法，也考虑如何使用压缩计算方法最大化大数据计算的效率。

压缩计算方法非常适用于科学大数据、统计大数据、数据仓库等具有如下特点的大数据计算问题。

（1）多维性。科学大数据、统计大数据、数据仓库等都具有多维的特点。科学大数据主要用于各种分析，目标是发现新的科学现象和规律。统计大数据主要用于统计分析，目标是支持决策制订。数据仓库主要用于支持联机分析处理，目的是支持实时决策。这些大数据

中的每个数据项通常具有两类属性值：一类属性值称为描述属性值，描述用于各种分析的数值型数据的语义；另一类属性值称为度量属性值，是直接用于分析的数值数据。例如，在科学实验大数据集合中，每个数据项包括有关实验环境设置的描述数据、有关实验设备的描述数据和实验结果数据。实验环境设置的描述数据和实验设备的描述数据是描述属性值，而实验结果数据则是度量属性值。又如，在一个人口统计数据库中，每个数据项可能包括省、市、年、性别、年龄段、人口数据，其中省、市、年、性别、年龄段数据就是描述属性值，人口数据就是度量属性值。这些大数据的模式可以抽象为 $MS(c_1,c_2,\cdots,c_d;m_1,m_2,\cdots,m_k)$，其中，$c_i$ 是定义域为 $Dom(c_i)$ 的描述属性，m_j 是定义域为 $Dom(m_j)$ 的度量属性。多维数据的实例可以定义为

$$D\subseteq Dom(c_1)\times Dom(c_2)\times\cdots\times Dom(c_d)\times Dom(m_1)\times Dom(m_2)\times\cdots\times Dom(m_k)$$

$D(c_1,c_2,\cdots,c_d;m_1,m_2,\cdots,m_k)$ 的每个数据项 $(x_1,x_2,\cdots,x_d;y_1,y_2,\cdots,y_k)$ 可视为 d 维空间中坐标为 $(x_1,x_2,\cdots,x_d)$ 的点 $(y_1,y_2,\cdots,y_k)$。

(2) 稀疏性。统计大数据、科学大数据和数据仓库等多维数据的多维性产生了数据的稀疏性，即并非所有描述属性值的组合都有对应的度量属性值。例如，令

$$D(\text{性别，年龄段，癌症种类，患者人数})$$

是一个癌症患者统计数据集合，每种特定的癌症可能仅与一种性别或少数年龄段有关，从而由{性别，年龄段，癌症种类}构成的空间是一个稀疏空间。

(3) 冗余性。多维数据的多维性会引起大量数据的重复出现，使得科学大数据、统计大数据和数据仓库等多维数据具有很高的数据冗余性。这是因为，当把每组描述属性值及其描述的度量属性值作为单个数据项存储时，不同的数据项可能具有某些相同的描述数据，从而描述属性值被重复存储。

(4) 空值性。由于数据采集方法、测量设备的限制等原因，科学大数据、统计大数据和数据仓库等多维数据通常包含大量空值。例如，在高能物理实验中，当测量数据低于给定的界限值时，则被视为噪声，通常用空值来记录。如果实验时间较长，自动测量装置将记录大量的噪声。

(5) 偏斜性。在科学大数据、统计大数据与数据仓库等多维数据中，数据的分布经常具有偏斜型。例如，在很多实际领域，数值型数据可能趋向于绝对值小(或绝对值大)的数据，即绝对值小的数据多于绝对值大的数据(或绝对值大的数据多于绝对值小的数据)。通常，字符型数据中的不同字符出现的频率差异也会很大。

(6) 静态性。科学大数据、统计大数据和数据仓库一旦完成了数据采集以及质量检验和校正以后，就没有理由进行数据更新了。于是，科学大数据、统计大数据和数据仓库可以被视为相对静态的大数据集合。这种静态性数据管理系统的设计要比动态数据管理系统的设计容易，不需要考虑并发控制等问题。数据的静态性也使得数据一旦压缩，压缩数据不需要动态更新，可以长久使用。

科学大数据、统计大数据和数据仓库等多维数据的多维性、稀疏性、冗余性、空值性、偏斜性、静态性使其具有很高的可压缩性。可以使用数组线性化等方法，充分利用多维性、稀疏性和冗余性，压缩描述属性值或维属性值；可以使用 Head 方法等数据压缩方法，充分利用空值性压缩空值；可以使用哈夫曼编码等方法，充分利用数据分布的偏斜性实现高效的数

据压缩。值得庆幸的是，数据的静态性为数据压缩作为大数据计算的预处理提供了充分理由。

压缩计算方法需要解决两个关键问题。

第一个关键问题是支持压缩计算的数据压缩方法的设计问题。传统的数据压缩方法的目标是最大化压缩比，节省计算机存储空间或降低网上数据传输量，多数方法不能支持压缩计算。因此，我们面临的第一个问题是支持压缩计算的数据压缩方法的设计问题。设 D 是问题的输入大数据集合，$c(D)$ 是 D 的压缩结果。通常，D 称为逻辑数据集合，$c(D)$ 则称为物理数据集合。为了支持压缩计算，数据压缩方法需要具有映射完整性，即存在两个映射 $f_1: D \to c(D)$ 和 $f_2: c(D) \to D$。于是，支持压缩计算的数据压缩的目标具有两个：第一个目标是最大化压缩比，既节省存储空间，也提高压缩计算效率；第二个目标是最小化映射 f_1 和 f_2 的时间复杂性。第二个目标对于压缩计算的效率具有重要影响。总之，压缩计算方法需要解决的第一个关键问题是如何设计映射完全的最大化压缩比且最小化映射时间的数据压缩方法。

压缩计算方法需要解决的第二个关键问题是压缩计算的算法设计问题。设问题$\mathcal{P}$的输入数据集合为 D，$c(D)$ 是 D 的压缩结果。使用压缩计算方法求解问题$\mathcal{P}$的算法设计的关键是如何直接在 $c(D)$ 上应用两个映射函数无解压地求解问题$\mathcal{P}$。值得注意的是，并非所有大数据计算问题都可以使用压缩计算方法设计出亚线性时间算法。在这种情况下，人们将放弃亚线性时间这个目标，而把最小化求解问题$\mathcal{P}$的时间复杂性作为使用压缩计算方法设计算法的目标。需要注意的是，不同的大数据计算问题可能需要不同的数据压缩方法。

5.2 节介绍几种常用的支持压缩计算的数据压缩方法，讨论支持压缩计算的数据压缩方法的设计原理。5.3 节至 5.7 节以 5 个大数据计算问题为例，介绍使用压缩计算方法求解大数据计算问题的算法的设计与分析方法。

5.2 数据压缩方法

多数传统的数据压缩方法不支持压缩计算。这是因为，在这样的压缩数据集合上实施计算之前，必须通过解压缩得到原始数据集合。设 D 是一个大数据集合，c 是数据压缩方法，$c(D)$ 是 D 的压缩结果。如上面谈到的，数据压缩方法 c 能够支持压缩计算的关键是支持两种映射：第一种映射称为向前映射 $f_F: D \to c(D)$，即 $\forall x \in D, f_F(x) \in c(D)$；第二种映射称为向后映射 $f_B: c(D) \to D$，即 $\forall y \in c(D), f_B(y) \in D$。通常，称 D 为逻辑数据集，并称 $c(D)$ 为物理数据集。$\forall x \in D$，x 在 D 中的地址称为 x 的逻辑地址，记作 $\mathrm{ladd}(x)$。$\forall y \in c(D)$，y 在 $c(D)$ 中的地址称为 y 的物理地址，记作 $\mathrm{padd}(x)$。令 $\mathrm{LADD}=\{\mathrm{ladd}(x) \mid x \in D\}$，$\mathrm{PADD}=\{\mathrm{padd}(y) \mid y \in c(D)\}$。向前映射经常被定义为 $f_F: \mathrm{LADD} \to \mathrm{PADD}$，向后映射也经常被定义为 $f_B: \mathrm{PADD} \to \mathrm{LADD}$。

以后会看到，这两个映射的效率直接影响压缩计算的效率。因此，支持压缩计算的数据压缩方法的设计具有两个优化目标：第一个目标是最小化 $c(D)$ 的规模 $|c(D)|$；第二个目标是最小化 f_F 和 f_B 的时间复杂性。不同的大数据计算问题需要不同的数据压缩方法。本节

仅介绍几种常用的支持压缩计算的数据压缩方法，其他数据压缩方法将在介绍具体大数据计算问题的压缩计算方法时讨论。

5.2.1 数据编码方法

数据编码是数据压缩的基础，本身也具有压缩的能力。下面介绍几种常用的支持压缩计算的数据编码方法。以后，用 $D(A_1,A_2,\cdots,A_m)$ 表示具有 m 个属性的数据集合，D 的每个元组或数据项表示为 $d=(a_1,a_2,\cdots,a_m)$，其中 a_i 属于 A_i 的值域 $\mathrm{Dom}(A_i)$。下面介绍的编码方法是对 D 在每个属性 A_i 上的投影集合的编码。以下，用 $\mathrm{Proj}(D,A_i)$ 表示 D 在任意属性 A_i 上的投影集合。对于任意 i，$\mathrm{Proj}(D,A_i)$ 可以使用不同的编码方法。

1. 二进制编码方法

二进制编码是最简单的编码方法。设 $D(A_1,A_2,\cdots,A_m)$ 是一个数据集合，$\mathrm{Proj}(D,A_i)$ 具有 n 个不同的数据。$\mathrm{Proj}(D,A_i)$ 二进制编码如下实现：

(1) 定义一个函数 f：$\mathrm{Proj}(D,A_i)\to\{0,1,\cdots,n\}$，使得 $\mathrm{Proj}(D,A_i)$ 中的每个数据对应一个整数。

(2) 用 $\lceil\log n\rceil$ 个二进制位表示 $\mathrm{Proj}(D,A_i)$ 中的每个数据，即 $\forall x\in\mathrm{Proj}(D,A_i)$，$x$ 的编码为 $f(x)$ 的 $\lceil\log n\rceil$ 位二进制数表示。

二进制编码是需要存储空间最小的编码方法。但是，当在 $\mathrm{Proj}(D,A_i)$ 中搜索一个数据时，需要考察每个数据的所有 $\lceil\log n\rceil$ 位。

例 5.2.1 设 EXP(Obj,Equ,Err,Pos,Tim,Res)是一个科学实验数据集合，其中，Obj 为实验对象，Equ 为实验设备类型，Err 为设备误差，Pos 为实验地点，Tim 为实验时间，Res 为实验结果。如果 Proj(EXP, Equ)包含 10 个实验设备类型，则设备类型数据的编码需要 4 位二进制数，即 0000～1001。

2. k-of-N 编码方法

设 $D(A_1,A_2,\cdots,A_m)$ 是一个数据集合，$\mathrm{Proj}(D,A_i)$ 具有 n 个数据。$\mathrm{Proj}(D,A_i)$ 的 k-of-N 编码如下实现：

(1) N 和 k 满足 $n=\binom{N}{k}$。

(2) 使用 N 个二进制位编码表示 $\mathrm{Proj}(D,A_i)$ 中的每个数据。

(3) 令 N 位中不同的 k 位为 1 表示 $\mathrm{Proj}(D,A_i)$ 中的不同数据。

例 5.2.2 仍然来看实例 EXP(Obj,Equ,Err,Pos,Tim,Res)。如果使用 1-of-10 编码对具有 10 个实验设备类型的 Proj(EXP, Equ)进行编码，则需要如下 10 个 10 位的编码：

0000000001
0000000010
⋮
1000000000

如果使用 2-of-10 编码对具有 10 个实验对象类的 Proj(EXP, Obj)进行编码，则需要如下 10 个 5 位的编码：

00011
00101
00110
01001
01010
01100
10001
10010
10100
11000

不同于二进制编码，当使用 k-of-N 编码在 Proj(S,A_i)中搜索一个数据时，只需要考察每个数据的 k 位。

3. 一进制编码方法

设 $D(A_1,A_2,\cdots,A_m)$是一个数据集合，Proj(D,A_i)具有 n 个数据。Proj(D,A_i)一进制编码如下实现：

(1) 定义一个函数 f：Proj(S,A_i)→{0, 1,…,n}，使得 Proj(S,A_i)中的每个数据对应一个整数。

(2) 用 n 个一进制数表示 Proj(S,A_i)中的每个数据，即 $\forall x \in$ Proj(S,A_i)，x 的编码为 $f(x)$的 n 位一进制数表示。

例 5.2.3 来看实验数据集合 EXP(Obj,Equ,Err,Pos,Tim,Res)。如果 Proj(EXP, Equ)具有 10 个实验设备类型，则这些类型的编码需要如下 10 位一进制数：

0000000001
0000000011
⋮
1111111111

一进制编码适用于那些经常在区域查询条件或不等式比较查询条件中出现的属性值的编码。例如，要获取实验设备类型值大于 3 的所有数据，只需考察 Proj(EXP,Equ)中的右起第四位是否为 1；类似地，如果要获取实验设备类型值小于 3 的所有数据，只需考察 Proj(EXP,Equ)中的右数第三位是否为 0。对于区域查询：获取实验设备类型在(a,b)区间的数据，只需将其转化为计算实验设备类型值大于 a 的数据集合与实验设备类型值小于 b 的数据集合的交集。

4. 叠加编码方法

叠加编码适用于值为长文本的属性。设 $D(A_1,A_2,\cdots,A_m)$是一个数据集合，Proj(D,A_i)是一个具有长文本值的属性，每个长文本值具有多个关键词。令 keyProj(D,A_i)是 Proj(D,A_i)的关键词集合。Proj(D,A_i)的叠加编码如下实现：

(1) 定义一个 Hash 函数 H：keyProj(S,A_i)→{α | α 是 N 位的 0 和 1 串}，使得 keyProj(S,A_i)中每个关键词对应一个 N 位 0-1 串。

(2) $\forall t \in$ Proj(S,A_i)，设 t 中关键词为{$w_1,w_2,\cdots,w_k$}，则 t 的编码为 $H(w_1)$。

$H(w_2)\circ\cdots\circ H(w_k)$，其中$\circ$是字符串连接操作符。

叠加编码适用于长文本数据的部分匹配查询。例如，为了查找包含关键词$\{w_1,w_2,\cdots,w_k\}$的A_i属性值，首先计算$H=H(w_1)\circ H(w_2)\circ\cdots\circ H(w_k)$，然后在经过叠加编码的属性$A_i$上查找与$H$匹配的文本值，过滤不包含$\{w_1,w_2,\cdots,w_k\}$的文本值。但是，没有过滤的文本值不一定真包含$\{w_1,w_2,\cdots,w_m\}$。这种编码适用于过滤操作。

5. 复合编码方法

当为一个数据集合的属性值编码时，可以复合使用上述4种编码方法，这种编码方法称为复合编码方法。设$D(A_1,A_2,\cdots,A_m)$是一个数据集合。$\mathrm{Proj}(D,A_i)$的复合编码如下实现：

(1) 把$\mathrm{Proj}(D,A_i)$划分为d列$\{A_{i1},A_{i2},\cdots,A_{id}\}$。

(2) 对于$1\leqslant j\leqslant d$，使用编码方法E_j对A_{ij}进行编码，E_j可以是任何编码方法。

(3) $\forall x=x_1x_2\cdots x_d\in\mathrm{Proj}(D,A_i)$，其中$x_j$对应于$A_{ij}$的值，$x$的编码为$E_1(x_1)\circ E_2(x_2)\circ\cdots\circ E_d(x_d)$。

例 5.2.4 假设前面的科学实验数据集合EXP的实验设备类型属性Equ具有1000个不同值。如果使用1-of-1000编码方法对$\mathrm{Proj}(S,A_i)$进行编码，每个值的编码长度需要1000位。如果把$\mathrm{Proj}(S,A_i)$划分为3列$\{A_{i1},A_{i2},A_{i3}\}$，每列具有10个不同值，则可以分别使用1-of-10编码方法对A_{i1}、A_{i2}和A_{i3}进行编码。这样，$\mathrm{Proj}(S,A_i)$的每个值的复合编码长度只需要30位。为了查找一个实验设备类型，只需要考察3个二进制位。

6. 编码的优化设计

下面以k-of-N为例，介绍编码的优化设计方法。这种方法可以推广到其他编码方法中。

给定一个数据集合$D(A_1,A_2,\cdots,A_m)$和$\mathrm{Proj}(D,A_i)$，如果$\mathrm{Proj}(D,A_i)$具有v个可能的值，k-of-N编码方法使用N位二进制数表示每个$t\in\mathrm{Proj}(S,A_i)$，其中k位为1，$N-k$位为0。因为k-of-N编码只能表示$\binom{N}{k}$个不同的值，所以N和k必须满足约束条件$\binom{N}{k}\geqslant v$。为了满足这个约束条件，可以选择同时增加N和k，或者当k保持比较小时仅增加N，或者仅增加k。无论如何选择，k不能超过$N/2$。这是因为，当$k=N/2$或$k=(N-1)/2$时，$\binom{N}{k}$最大化。显然，增加k意味着在查询处理中需要更多的操作，而大的N则意味着需要大的空间存储$\mathrm{Proj}(D,A_i)$。于是，在k-of-N编码设计中，面临时间和空间均衡问题。我们要解决的k-of-N编码的优化设计问题可以叙述为：给定数据集合$D(A_1,A_2,\cdots,A_m)$和存储空间约束C，如何在m个属性之间分配存储空间C，使得平均查询处理时间最小化？下面将给出这个问题的形式化定义以及求解这个问题的动态规划方法。

设$D(A_1,A_2,\cdots,A_m)$是一个具有m个属性和n个元组的数据集合。可以用一组位向量存储$D(A_1,A_2,\cdots,A_m)$的所有元组，每个位向量存储所有元组的相同位的0或1值。如果限定$D(A_1,A_2,\cdots,A_m)$的所有m个属性的编码长度之和为C，则存储$D(A_1,A_2,\cdots,A_m)$的所有元组需要的存储空间为$C\times n$。假设属性A_i具有n_i个可能的值，A_i出现在一个查询中的概率是p_i。为了叙述简单，这里仅考虑准确匹配查询。要解决的问题是：在满

足给定约束的条件下，为每个 A_i 选择 k_i 和 N_i，并使用 k_i-of-N_i 编码方法对 A_i 的值集合进行编码，使得查询时间最小化。由于仅考虑准确匹配查询，当 A_i 出现在一个查询中时，查询处理所需要的布尔操作量正比于 k_i。于是，上述问题可以形式化地定义为如下的优化问题 Op-Encod：

$$\text{对于 } 1 \leqslant i \leqslant m \text{，为每个 } A_i \text{ 选择 } k_i \text{ 和 } N_i$$

$$\text{最小化查询代价均值 } \sum_{1 \leqslant i \leqslant m} p_i k_i$$

$$k_i \text{ 和 } N_i \text{ 满足 } \sum_{1 \leqslant i \leqslant m} N_i \leqslant C \text{ 且} \binom{N_i}{k_i} \geqslant n_i$$

下面给出求解 Op-Encod 问题的动态规划方法。

显然，N_i 的最小值是 $\log n_i$，k_i 的最大值是 $\log n_i$。令 $\mathrm{OPT}_w(1,2,\cdots,j)$ 是仅考虑属性集合 $\{A_1,A_2,\cdots,A_j\}$ 而且在查询处理中使用的所有这些属性的位数为 w 时 Op-Encod 问题优化查询代价的均值。于是，

$$\mathrm{OPT}_w(1,2,\cdots,j+1)=\min_y\{\mathrm{OPT}_y(1,2,\cdots,j)+\mathrm{OPT}_{w-y}(j+1)\}$$

依据这个递归公式，可以使用动态规划方法在循环中递归地求解 Op-Encod 问题。每次循环增加一个属性，并记录为相应的属性分配的 N_i 和 k_i，直到计算出 $\mathrm{OPT}_C(1,2,\cdots,m)$，完成了 C 个二进制位在 m 个属性之间的划分，即为每个属性 A_i 选择了 k_i 和 N_i，k_i 和 N_i 满足约束条件 $\left(\sum_{1 \leqslant i \leqslant m} N_i \leqslant C\right) \wedge \binom{N_i}{k_i} \geqslant n_i \wedge \left(\text{最小化} \sum_{1 \leqslant i \leqslant m} p_i k_i\right)$。最后，为每个 A_i 的 n_i 个值进行 k_i-of-N_i 编码，这样就完成了对 $D(A_1,A_2,\cdots,A_m)$ 的优化 k-of-N 编码设计。

5.2.2 Header 压缩方法

如 5.1 节所述，在很多数据集合（例如科学和统计数据库）中经常出现连续的重复值。例如，我国 2020 年新冠病毒病例统计数据库 2020-Covid19-Sum 中包括以下属性：

（Month，Day，Prov，N-cfm-cs，N-deaths，N-susp-cs）

其中，Month 表示月份，Day 表示日，Prov 表示省（或直辖市、自治区和特别行政区），N-cfm-cs 表示新增确诊病例，N-deaths 表示新增死亡病例，N-susp-cs 表示新增疑似病例。2020-Covid19-Sum 是一个统计数据库，Month、Day 和 Prov 是描述属性，N-cfm-cs、N-deaths、N-susp-cs 则是度量属性。表 5.2.1 给出了 2020-Covid19-Sum 的部分内容。基于多方面考虑，这里略去了具体病例数据。

表 5.2.1 统计数据库 2020-Covid19-Sum 的部分内容

Month	Day	Prov	N-cfm-cs	N-deaths	N-susp-cs
2	1	黑龙江	$HC_{2\text{-}1}$	$HD_{2\text{-}1}$	$HS_{2\text{-}1}$
2	2	黑龙江	$HC_{2\text{-}2}$	$HD_{2\text{-}2}$	$HS_{2\text{-}2}$
2	3	黑龙江	$HC_{2\text{-}3}$	$HD_{2\text{-}3}$	$HS_{2\text{-}3}$
⋮	⋮	⋮	⋮	⋮	⋮
2	29	黑龙江	$HC_{2\text{-}29}$	$HD_{2\text{-}29}$	$HS_{2\text{-}29}$

续表

Month	Day	Prov	N-cfm-cs	N-deaths	N-susp-cs
2	1	辽宁	$LC_{2\text{-}1}$	$LD_{2\text{-}1}$	$LS_{2\text{-}1}$
2	2	辽宁	$LC_{2\text{-}2}$	$LD_{2\text{-}2}$	$LS_{2\text{-}2}$
2	3	辽宁	$LC_{2\text{-}3}$	$LD_{2\text{-}3}$	$LS_{2\text{-}3}$
⋮	⋮	⋮	⋮	⋮	⋮
2	29	辽宁	$C_{2\text{-}29}$	$D_{2\text{-}29}$	$S_{2\text{-}29}$

从表 5.2.1 给出的数据内容可以看到，在 Month 和 Prov 两个属性上具有很多连续重复的值。当月份大于 4 以后，N-cfm-cs、N-deaths、N-susp-cs 属性上将具有大量的连续的 0。实际上，多数科学数据库、统计数据库、数据仓库等多维数据集合都具有这种特点。下面介绍适用于这类数据集合并支持压缩计算的 Header 数据压缩方法。

设 $D(A_1,A_2,\cdots,A_m)$ 是一个具有 m 个属性的数据集合。使用列存储的方式存储 $D(A_1,A_2,\cdots,A_m)$，即 $D(A_1,A_2,\cdots,A_m)$ 的每一个属性的所有值保序地存储在一个向量或文件中。这样，$D(A_1,A_2,\cdots,A_m)$ 由 m 个向量 $VA_1,VA_2,\cdots,VA_m$ 存储，其中 VA_i 是 A_i 对应的向量。对每个向量使用 Header 压缩方法单独压缩。显然，这种列存储方式能够极大地提高压缩比。此外，列存储方式还可以极大地提高多维数据分析的效率。以下在不引起混淆的情况下使用 VA 表示任何一个 VA_i，$1\leqslant i\leqslant m$。称经常连续出现的值为可压缩值，称 VA 为逻辑数据集合，称 VA 的压缩结果 $c(VA)$ 为 VA 的物理数据集合。$\forall x\in VA$，称 x 在 VA 中的地址为 x 的逻辑地址，称 x 在 $c(VA)$ 中的地址为 x 的物理地址。

下面，首先考虑 VA 仅包含一个可压缩值的 Header 压缩方法，简称为单值 Header 压缩方法。然后，把单值 Header 压缩方法推广到 VA 包含多个可压缩值的情况，得到多值 Header 压缩方法。

1. 单值 Header 压缩方法

设 VA 只有一个可压缩值，简记作 c。在这种情况下，仅需使用单值 Header 压缩方法压缩 VA。下面以

$$VA=\langle v_1,v_2,v_3,c,c,c,c,c,v_4,v_5,c,c,c,c,v_6,v_7,v_8,v_9,c,c,c,c\rangle$$

为例，介绍单值 Header 压缩方法，其压缩过程分为如下 3 步：

(1) 从 VA 构造有序向量的 Header。首先把 VA 分成偶数段，设为 $2k$ 段。第 1 段是第一个 c 之前的连续的非可压缩值，第 2 段是下一个非可压缩值之前的连续的可压缩值 c，以此类推，奇数段为连续的非可压缩值段，偶数段为连续的可压缩值 c 段。如果 VA 的第 1 个值为 c，则第 1 段为空。如果 VA 的最后的值是非可压缩值，则最后一个连续 c 段为空。对于 $1\leqslant i\leqslant 2k$，令 VA[i]为 VA 的第 i 段。称奇数段 VA[$2i-1$]为第 i 个非 c 段，称偶数段 VA[$2i$]为第 i 个 c 段。Header 是长为 $2k$ 的向量 $Hd=\langle u_1,c_1,u_2,c_2,\cdots,u_k,c_k\rangle$，其中，$u_i$ 是前 i 个非 c 段的长度之和，c_i 是前 i 个 c 段的长度之和。

例 5.2.5 $VA=\langle v_1,v_2,v_3,c,c,c,c,c,v_4,v_5,c,c,c,c,v_6,v_7,v_8,v_9,c,c,c,c\rangle$ 的划分和相应的 Header 向量 Hd 如图 5.2.1 所示。

(2) 从逻辑数据集合 VA 构造物理数据集合 $c(VA)$。$c(VA)$ 包含 VA 的非可压缩值。

第1段	第2段	第3段	第4段	第5段	第6段
VA=⟨v_1, v_2, v_3 | c, c, c, c, c | v_4, v_5 | c, c, c, c | v_6, v_7, v_8, v_9 | c, c, c, c⟩					
Hd=⟨ u_1=3	c_1=5	u_2=5	c_2=9	u_3=9	c_3=13 ⟩

图 5.2.1　VA 的划分和 Header 向量 Hd

例如 VA=⟨v_1,v_2,v_3, c, c, c, c, c,v_4,v_5, c, c, c, c,v_6,v_7,v_8,v_9, c, c, c, c⟩的物理数据集合 c(VA)=⟨v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9⟩。

(3) 存储 VA 的 Header 压缩结果(c(VA), Hd)。

下面介绍单值 Header 压缩方法的向前映射和向后映射的实现算法。向前映射(forward mapping)算法如 Algorithm 5.2.1 所示。

Algorithm 5.2.1: Sig-c-f_F(l, Hd)
输入：x 的逻辑地址 l,Hd=⟨u_1,c_1,u_2,c_2,…,u_k,c_k⟩。
输出：若 $x=c$ 输出 $p=c$,否则输出 x 的物理地址 p。
1. 使用 Hd 和 l，应用插值搜索在 Hd 上查找 i，满足
$$u_i+c_i\leqslant l\leqslant c_i+u_{i+1}\text{ 或 }c_{i-1}+u_i\leqslant l\leqslant u_i+c_i$$
2. 如果 $u_i+c_i\leqslant l\leqslant c_i+u_{i+1}$，则 x 不是被压缩的 c 值,输出 $p=l-c_i$，停止；
3. 如果 $c_{i-1}+u_i\leqslant l\leqslant u_i+c_i$，则 x 是被压缩的 c,输出 $p=c$，停止.

类似于二叉搜索,算法第 1 步使用的插值搜索算法也是有序集合上的重要搜索算法,其平均时间复杂性为 $O(\log\log n)$,其中 n 是被搜索集合的大小。插值搜索的详细介绍参见本书作者的论文[1]。

例 5.2.6　用图 5.2.1 的例子说明向前映射算法。令 $x=v_7$,$l=16$,则算法第 1 步一定发现 $i=2$ 满足 $u_2+c_2=5+9=14\leqslant16\leqslant c_2+u_3=9+9=18$,于是,算法输出 $p=l-c_2=16-9=7$。如果 $x=c$,$l=12$,则算法第 1 步发现 $i=2$ 满足 $c_1+u_2=5+5=10\leqslant12\leqslant u_2+c_2=5+9=14$,于是算法输出 $p=c$。

命题 5.2.1　如果 Header 向量的大小为 k,则 Sig-c-f_F 算法的平均时间复杂性为 $O(\log\log k)$。

证明：因为插值搜索的平均时间复杂性为 $O(\log\log n)$,Sig-c-f_F 算法的第 1 步需要的时间的均值为 $O(\log\log k)$,其他步骤需要常数时间。于是,Sig-c-f_F 算法的平均时间复杂性为 $O(\log\log k)$。**证毕**。

如果在该算法的第 1 步使用二叉搜索算法,则该算法的时间复杂性为 $O(\log k)$。

向后映射(backward mapping)算法如 Algorithm 5.2.2 所示。

Algorithm 5.2.2: Sig-c-f_B(p, Hd)
输入：x 的物理地址 p,Hd=⟨u_1,c_1,u_2,c_2,…,u_k,c_k⟩;
输出：$l=x$ 的逻辑地址。
1. 使用 Hd 和 p，用插值搜索算法在$\{u_1,u_2,\cdots,u_k\}$中查找满足 $u_{i-1}<p\leqslant u_i$ 的 i;
2. 输出 $l=p+c_{i-1}$，停止.

例 5.2.7　还是用图 5.2.1 的例子说明向后映射算法。令 $x=v_7$,$p=7$,则算法第 1 步必然发现 $i=3$ 满足 $u_2=5\leqslant12\leqslant u_3=9$,于是,算法输出 $l=p+c_2=7+9=16$。

命题 5.2.2 如果 Header 向量的大小为 k，则 Sig-c-f_B 的平均时间复杂性为 $O(\log\log k)$。

证明：因为插值搜索的平均时间复杂性为 $O(\log\log n)$，Sig-c-f_B 算法的第 1 步需要的时间的均值为 $O(\log\log k)$，第 2 步需要常数时间。于是，Sig-c-f_B 算法的平均时间复杂性为 $O(\log\log k)$。**证毕**。

如果在该算法的第 1 步使用二叉搜索算法，该算法的时间复杂性为 $O(\log k)$。

从 Sig-c-f_F 和 Sig-c-f_B 的时间复杂性分析可以看出，由于其平均时间复杂性为 $O(\log\log k)$，单值 Header 压缩方法可以有效地支持大数据的压缩计算。本书作者在文献[2]中给出了另外一种单值压缩方法，当主存充分大时，其向前映射和向后映射的时间复杂性为常数。

下面分析单值 Header 压缩方法的压缩比。

定义 5.2.1 设 VA 是一个数据集合，$c(\mathrm{VA})$是使用数据压缩方法 C 对 VA 压缩后的物理文件，S 是 VA 压缩后需要保存的辅助数据集合。压缩方法 C 对 VA 的压缩比定义为 $|\mathrm{VA}|/(|c(\mathrm{VA})|+|S|)$。

定义 5.2.1 中$|\mathrm{VA}|$、$|c(\mathrm{VA})|$和$|S|$可以根据需要定义，可以是数据个数，也可以是字节数等其他度量。

下面的命题 5.2.3 给出了单值 Header 压缩方法的压缩比。

命题 5.2.3 设 $\mathrm{VA}=\langle x_1,x_2,\cdots,x_n\rangle$，VA 包含单个可压缩值且其数量为 m 个，VA 包含 k 个可压缩值段，VA 中每个数据的字节数为 β。如果 Header 向量的每个数据的字节数为 α，则单值 Header 压缩方法对于 VA 的压缩比的均值为 $n\beta/((n-m)\beta+2k\alpha)$。

证明：从 Header 压缩方法的定义可知，Header 向量包含 $2k$ 个正整数，所以 Header 向量的大小为 $2k\alpha$ 字节，而且$|c(\mathrm{VA})|=(n-m)\beta$。于是，单值 Header 压缩方法对于 VA 的压缩比为 $n\beta/((n-m)\beta+2k\alpha)$。**证毕**。

例 5.2.8 令 $n=10\,000$，$m=6000$，$k=20$，$\alpha=\beta$，则单值 Header 压缩方法的压缩比为 $n/((n-m)+2k)>2.48$。

2. 多值 Header 压缩方法

设 VA 具有 m 个可压缩值，分别为 $c_1,c_2,\cdots,c_m$。多值 Header 压缩方法反复应用单值 Header 压缩方法逐个压缩每个可压缩值，详细过程如下：

(1) 令 $\mathrm{VA}=\mathrm{PA}_0$。对于 $1\leqslant i\leqslant m$，使用单值 Header 压缩方法压缩 PA_{i-1} 中的 c_i，产生物理数据集合 $\mathrm{PA}_i=c(\mathrm{PA}_{i-1})$和 Header 向量 Hd_i。

(2) 存储 $c(\mathrm{VA})=(\mathrm{PA}_m,\mathrm{Hd}_1,\mathrm{Hd}_2,\cdots,\mathrm{Hd}_m)$。

下面看一个实例。设 VA 具有 3 个可压缩值 c_1、c_2、c_3，且

$$\mathrm{VA}=\langle v_1,v_2,v_3,c_1,c_1,c_1,v_4,v_5,c_2,c_2,c_2,v_6,v_7,v_8,c_1,c_1,c_1,v_9,$$
$$v_{10},c_2,c_2,v_{11},v_{12},c_3,c_3,c_3,v_{13},v_{14},c_3,c_3,c_3,c_2,c_2,c_2,c_1,c_1,c_1\rangle$$

在 $\mathrm{VA}=\mathrm{PA}_0$ 上压缩 c_1 以后的结果为

$$PA_1=\langle v_1,v_2,v_3,v_4,v_5,c_2,c_2,c_2,v_6,v_7,v_8,v_9,v_{10},c_2,c_2,v_{11},v_{12},c_3,c_3,c_3,v_{13},$$
$$v_{14},c_3,c_3,c_3,c_2,c_2,c_2\rangle$$

$$Hd_1=\langle 3,3,11,6,28,9\rangle$$

在 PA_1 上压缩 c_2 以后的结果为