(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110741637 B
(45)授权公告日 2023.10.03

(21)申请号 201880037616.2
(22)申请日 2018.04.26
(65)同一申请的已公布的文献号
申请公布号 CN 110741637 A
(43)申请公布日 2020.01.31
(30)优先权数据
62/491,864 2017.04.28 US
(85)PCT国际申请进入国家阶段日
2019.12.06
(86)PCT国际申请的申请数据
PCT/US2018/029636 2018.04.26
(87)PCT国际申请的公布数据
WO2018/200862 EN 2018.11.01
(73)专利权人 阿斯卡瓦公司
地址 美国加利福尼亚
(72)发明人 H·莎朗潘尼
(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
专利代理师 刘玉洁
(51)Int.Cl.
H04N 19/40(2006.01)
H03M 7/30(2006.01)
(56)对比文件
CN 102823245 A,2012.12.12
WO 2006055587 A2,2006.05.26
WO 2016205209 A1,2016.12.22
US 2013339323 A1,2013.12.19
审查员 于典

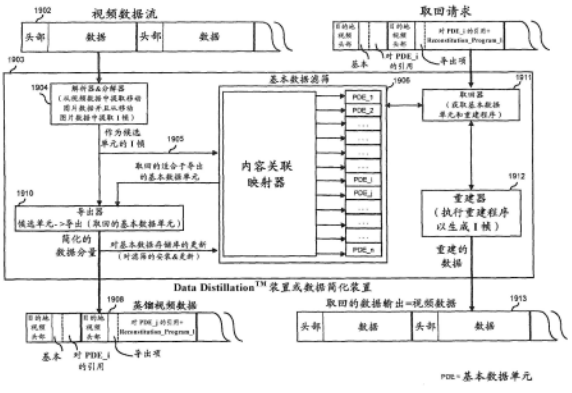
权利要求书6页 说明书56页 附图82页

(54)发明名称

简化视频数据的方法、计算机可读存储介质和电子装置

(57)摘要

通过使用基于基本数据单元的内容组织基本数据单元的数据结构，可以无损简化输入数据。替代地，数据结构可以基于从基本数据单元导出的名称的内容来组织基本数据单元。具体而言，视频数据可以通过(1)使用数据结构来识别一组基本数据单元，以及(2)使用该组基本数据单元来无损简化内帧来进行无损简化。可以基于数据结构的分量的存储器使用情况对输入数据进行动态分区。可以基于分区创建包裹，以促进数据的归档和移动。可以使用一组蒸馏文件和一组基本数据单元文件来存储无损简化的数据。



CN 110741637 B

1.一种用于简化视频数据以获得简化的视频数据的方法,所述方法包括:

从所述视频数据中提取压缩的移动图片数据和压缩的音频数据;

从所述压缩的移动图片数据中提取内帧(I帧);

从所述压缩的移动图片数据中提取预测帧;

维护组织基本数据单元的数据结构,每个基本数据单元包括字节序列;

无损简化I帧以获得无损简化的I帧,其中无损简化I帧包括对于每个I帧:

通过使用I帧对基于基本数据单元的内容组织基本数据单元的数据结构进行第一内容关联查找来识别第一组基本数据单元,以及

使用所述第一组基本数据单元来无损简化I帧,以便(i)如果所述I帧是第一组基本数据单元中的基本数据单元的副本,则获得对所述基本数据单元的引用,或者(ii)如果所述I帧不是第一组基本数据单元中的任何基本数据单元的副本,则获得对第一组基本数据单元中的一个或多个基本数据单元的引用以及从所述一个或多个基本数据单元导出所述I帧的变换序列;以及

其中所述简化的视频数据包括无损简化的I帧、所述无损简化的I帧引用的基本数据单元、预测帧和压缩的音频数据。

2.如权利要求1所述的方法,其中使用所述第一组基本数据单元来无损简化I帧包括:

响应于确定(i)对所述第一组基本数据单元的引用的大小和(ii)重建程序的描述的大小的总和小于I帧的大小的阈值比例,生成I帧的第一无损简化表示,其中所述第一无损简化表示包括对所述第一组基本数据单元中的每个基本数据单元的引用以及所述重建程序的描述;以及

响应于确定(i)对所述第一组基本数据单元的引用的大小和(ii)所述重建程序的描述的大小的总和大于或等于I帧的大小的阈值比例,

在所述数据结构中将I帧添加为新的基本数据单元,以及

生成I帧的第二无损简化表示,其中所述第二无损简化表示包括对所述新的基本数据单元的引用。

3.如权利要求2所述的方法,其中所述重建程序的描述指定变换序列,所述变换序列在应用于所述第一组基本数据单元时产生I帧。

4.如权利要求1所述的方法,其中所述方法还包括:

对所述压缩的音频数据进行解压缩以获得一组音频分量;以及

对于所述一组音频分量中的每个音频分量,

通过使用该音频分量对基于基本数据单元的内容组织基本数据单元的数据结构执行第二内容关联查找来识别第二组基本数据单元,以及

使用所述第二组基本数据单元来无损简化该音频分量;以及

其中所述简化的视频数据包括无损简化的I帧、所述无损简化的I帧引用的基本数据单元、预测帧、无损简化的音频分量和所述无损简化的音频分量引用的基本数据单元。

5.如权利要求4所述的方法,其中使用所述第二组基本数据单元来无损简化该音频分量包括:

响应于确定(i)对所述第二组基本数据单元的引用的大小和(ii)重建程序的描述的大小的总和小于所述音频分量的大小的阈值比例,生成所述音频分量的第一无损简化表示,

其中所述第一无损简化表示包括对所述第二组基本数据单元中的每个基本数据单元的引用以及所述重建程序的描述;以及

响应于确定(i)对所述第二组基本数据单元的引用的大小和(ii)所述重建程序的描述的大小的总和大于或等于所述音频分量的大小的阈值比例,

将所述音频分量作为新的基本数据单元添加到所述数据结构中,以及

生成所述音频分量的第二无损简化表示,其中所述第二无损简化表示包括对所述新的基本数据单元的引用。

6.如权利要求5所述的方法,其中所述重建程序的描述指定变换序列,所述变换序列在应用于所述第二组基本数据单元时,产生所述音频分量。

7.如权利要求1所述的方法,还包括:

响应于接收到取回视频数据的请求,

重新创建压缩的移动图片数据,这包括从所述无损简化的I帧和所述无损简化的I帧引用的基本数据单元重建所述I帧并且将所述I帧与所述预测帧组合,以及

将压缩的移动图片数据与压缩的音频数据组合以获得所述视频数据。

8.如权利要求1所述的方法,其中,从所述压缩的移动图片数据中提取I帧包括执行霍夫曼解码。

9.如权利要求8所述的方法,还包括对无损简化的I帧和所述无损简化的I帧引用的基本数据单元执行霍夫曼编码以分别获得霍夫曼编码的无损简化的I帧和霍夫曼编码的基本数据单元,其中简化的视频数据包括霍夫曼编码的无损简化的I帧、霍夫曼编码的基本数据单元、所述预测帧和压缩的音频数据。

10.如权利要求9所述的方法,还包括:

响应于接收到取回视频数据的请求,

通过(1)对霍夫曼编码的无损简化的I帧执行霍夫曼解码以获得无损简化的I帧,以及对霍夫曼编码的基本数据单元执行霍夫曼解码以获得基本数据单元,(2)从无损简化的I帧和所述基本数据单元重建I帧,以及(3)对I帧执行霍夫曼编码,并与所述预测帧组合,来重新创建压缩的移动图片数据,以及

将压缩的移动图片数据与压缩的音频数据组合以获得所述视频数据。

11.一种存储指令的计算机可读存储介质,所述指令在由计算机执行时,使得所述计算机执行用于简化视频数据以获得简化的视频数据的方法,所述方法包括:

从所述视频数据中提取压缩的移动图片数据和压缩的音频数据;

从所述压缩的移动图片数据中提取内帧(I帧);

从所述压缩的移动图片数据中提取预测帧;

维护组织基本数据单元的数据结构,每个基本数据单元包括字节序列;

无损简化I帧以获得无损简化的I帧,其中无损简化I帧包括对于每个I帧:

通过使用I帧对基于基本数据单元的内容组织基本数据单元的数据结构进行第一内容关联查找来识别第一组基本数据单元,以及

使用所述第一组基本数据单元来无损简化I帧,以便(i)如果所述I帧是第一组基本数据单元中的基本数据单元的副本,则获得对所述基本数据单元的引用,或者(ii)如果所述I帧不是第一组基本数据单元中的任何基本数据单元的副本,则获得对第一组基本数据单元

中的一个或多个基本数据单元的引用以及从所述一个或多个基本数据单元导出所述I帧的变换序列;以及

其中所述简化的视频数据包括无损简化的I帧、所述无损简化的I帧引用的基本数据单元、预测帧和压缩的音频数据。

12.如权利要求11所述的计算机可读存储介质,其中使用所述第一组基本数据单元来无损简化I帧包括:

响应于确定(i)对所述第一组基本数据单元的引用的大小和(ii)重建程序的描述的大小的总和小于I帧的大小的阈值比例,生成I帧的第一无损简化表示,其中所述第一无损简化表示包括对所述第一组基本数据单元中的每个基本数据单元的引用以及所述重建程序的描述;以及

响应于确定(i)对所述第一组基本数据单元的引用的大小和(ii)所述重建程序的描述的大小的总和大于或等于I帧的大小的阈值比例,

在所述数据结构中将I帧添加为新的基本数据单元,以及

生成I帧的第二无损简化表示,其中所述第二无损简化表示包括对所述新的基本数据单元的引用。

13.如权利要求12所述的计算机可读存储介质,其中所述重建程序的描述指定变换序列,所述变换序列在应用于所述第一组基本数据单元时产生I帧。

14.如权利要求11所述的计算机可读存储介质,其中所述方法还包括:

对所述压缩的音频数据进行解压缩以获得一组音频分量;以及

对于所述一组音频分量中的每个音频分量,

通过使用该音频分量对基于基本数据单元的内容组织基本数据单元的数据结构执行第二内容关联查找来识别第二组基本数据单元,以及

使用所述第二组基本数据单元来无损简化该音频分量;以及

其中所述简化的视频数据包括无损简化的I帧、所述无损简化的I帧引用的基本数据单元、预测帧、无损简化的音频分量和所述无损简化的音频分量引用的基本数据单元。

15.如权利要求14所述的计算机可读存储介质,其中使用所述第二组基本数据单元来无损简化该音频分量包括:

响应于确定(i)对所述第二组基本数据单元的引用的大小和(ii)重建程序的描述的大小的总和小于所述音频分量的大小的阈值比例,生成所述音频分量的第一无损简化表示,其中所述第一无损简化表示包括对所述第二组基本数据单元中的每个基本数据单元的引用以及所述重建程序的描述;以及

响应于确定(i)对所述第二组基本数据单元的引用的大小和(ii)所述重建程序的描述的大小的总和大于或等于所述音频分量的大小的阈值比例,

将所述音频分量作为新的基本数据单元添加到所述数据结构中,以及

生成所述音频分量的第二无损简化表示,其中所述第二无损简化表示包括对所述新的基本数据单元的引用。

16.如权利要求15所述的计算机可读存储介质,其中所述重建程序的描述指定变换序列,所述变换序列在应用于所述第二组基本数据单元时,产生所述音频分量。

17.如权利要求11所述的计算机可读存储介质,还包括:

响应于接收到取回视频数据的请求，

重新创建压缩的移动图片数据，这包括从所述无损简化的I帧和所述无损简化的I帧引用的基本数据单元重建所述I帧并且将所述I帧与所述预测帧组合，以及

将压缩的移动图片数据与压缩的音频数据组合以获得所述视频数据。

18.如权利要求11所述的计算机可读存储介质，其中，从所述压缩的移动图片数据中提取I帧包括执行霍夫曼解码。

19.如权利要求18所述的计算机可读存储介质，还包括对无损简化的I帧和所述无损简化的I帧引用的基本数据单元执行霍夫曼编码以分别获得霍夫曼编码的无损简化的I帧和霍夫曼编码的基本数据单元，其中简化的视频数据包括霍夫曼编码的无损简化的I帧、霍夫曼编码的基本数据单元、所述预测帧和压缩的音频数据。

20.如权利要求19所述的计算机可读存储介质，还包括：

响应于接收到取回视频数据的请求，

通过(1)对霍夫曼编码的无损简化的I帧执行霍夫曼解码以获得无损简化的I帧，以及对霍夫曼编码的基本数据单元执行霍夫曼解码以获得基本数据单元，(2)从无损简化的I帧和所述基本数据单元重建I帧，以及(3)对I帧执行霍夫曼编码，并与所述预测帧组合，来重新创建压缩的移动图片数据，以及

将压缩的移动图片数据与压缩的音频数据组合以获得所述视频数据。

21.一种电子装置，包括：

处理器；和

存储指令的存储器，所述指令在由所述处理器执行时，使得所述处理器执行用于简化视频数据以获得简化的视频数据的方法，所述方法包括：

从所述视频数据中提取压缩的移动图片数据和压缩的音频数据；

从所述压缩的移动图片数据中提取内帧(I帧)；

从所述压缩的移动图片数据中提取预测帧；

维护组织基本数据单元的数据结构，每个基本数据单元包括字节序列；

无损简化I帧以获得无损简化的I帧，其中无损简化I帧包括对于每个I帧：

通过使用I帧对基于基本数据单元的内容组织基本数据单元的数据结构进行第一内容关联查找来识别第一组基本数据单元，以及

使用所述第一组基本数据单元来无损简化I帧，以便(i)如果所述I帧是第一组基本数据单元中的基本数据单元的副本，则获得对所述基本数据单元的引用，或者(ii)如果所述I帧不是第一组基本数据单元中的任何基本数据单元的副本，则获得对第一组基本数据单元中的一个或多个基本数据单元的引用以及从所述一个或多个基本数据单元导出所述I帧的变换序列；以及

其中所述简化的视频数据包括无损简化的I帧、所述无损简化的I帧引用的基本数据单元、预测帧和压缩的音频数据。

22.如权利要求21所述的电子装置，其中使用所述第一组基本数据单元来无损简化I帧包括：

响应于确定(i)对所述第一组基本数据单元的引用的大小和(ii)重建程序的描述的大小的总和小于I帧的大小的阈值比例，生成I帧的第一无损简化表示，其中所述第一无损简

化表示包括对所述第一组基本数据单元中的每个基本数据单元的引用以及所述重建程序的描述;以及

响应于确定(i)对所述第一组基本数据单元的引用的大小和(ii)所述重建程序的描述的大小的总和大于或等于I帧的大小的阈值比例,

在所述数据结构中将I帧添加为新的基本数据单元,以及

生成I帧的第二无损简化表示,其中所述第二无损简化表示包括对所述新的基本数据单元的引用。

23.如权利要求22所述的电子装置,其中所述重建程序的描述指定变换序列,所述变换序列在应用于所述第一组基本数据单元时产生I帧。

24.如权利要求21所述的电子装置,其中所述方法还包括:

对所述压缩的音频数据进行解压缩以获得一组音频分量;以及

对于所述一组音频分量中的每个音频分量执行以下操作:

通过使用该音频分量对基于基本数据单元的内容组织基本数据单元的数据结构执行第二内容关联查找来识别第二组基本数据单元,以及

使用所述第二组基本数据单元来无损简化该音频分量;以及

其中所述简化的视频数据包括无损简化的I帧、所述无损简化的I帧引用的基本数据单元、预测帧、无损简化的音频分量和所述无损简化的音频分量引用的基本数据单元。

25.如权利要求24所述的电子装置,其中使用所述第二组基本数据单元来无损简化该音频分量包括:

响应于确定(i)对所述第二组基本数据单元的引用的大小和(ii)重建程序的描述的大小的总和小于所述音频分量的大小的阈值比例,生成所述音频分量的第一无损简化表示,其中所述第一无损简化表示包括对所述第二组基本数据单元中的每个基本数据单元的引用以及所述重建程序的描述;以及

响应于确定(i)对所述第二组基本数据单元的引用的大小和(ii)所述重建程序的描述的大小的总和大于或等于所述音频分量的大小的阈值比例,

将所述音频分量作为新的基本数据单元添加到所述数据结构中,以及

生成所述音频分量的第二无损简化表示,其中所述第二无损简化表示包括对所述新的基本数据单元的引用。

26.如权利要求25所述的电子装置,其中所述重建程序的描述指定变换序列,所述变换序列在应用于所述第二组基本数据单元时,产生所述音频分量。

27.如权利要求21所述的电子装置,还包括:

响应于接收到取回视频数据的请求,

重新创建压缩的移动图片数据,这包括从所述无损简化的I帧和所述无损简化的I帧引用的基本数据单元重建所述I帧并且将所述I帧与所述预测帧组合,以及

将压缩的移动图片数据与压缩的音频数据组合以获得所述视频数据。

28.如权利要求21所述的电子装置,其中,从所述压缩的移动图片数据中提取I帧包括执行霍夫曼解码。

29.如权利要求28所述的电子装置,还包括对无损简化的I帧和所述无损简化的I帧引用的基本数据单元执行霍夫曼编码以分别获得霍夫曼编码的无损简化的I帧和霍夫曼编码

的基本数据单元，其中简化的视频数据包括霍夫曼编码的无损简化的I帧、霍夫曼编码的基本数据单元、所述预测帧和压缩的音频数据。

30.如权利要求29所述的电子装置，还包括：

响应于接收到取回视频数据的请求，

通过(1)对霍夫曼编码的无损简化的I帧执行霍夫曼解码以获得无损简化的I帧，以及对霍夫曼编码的基本数据单元执行霍夫曼解码以获得基本数据单元，(2)从无损简化的I帧和所述基本数据单元重建I帧，以及(3)对I帧执行霍夫曼编码，并与所述预测帧组合，来重新创建压缩的移动图片数据，以及

将压缩的移动图片数据与压缩的音频数据组合以获得所述视频数据。

简化视频数据的方法、计算机可读存储介质和电子装置

技术领域

[0001] 本公开涉及数据存储、取回和通信。更具体来说，本公开涉及对已经使用基本数据滤筛进行无损简化的数据执行多维搜索和与内容关联的取回。

背景技术

[0002] 当今的信息时代以巨量数据的产生、捕获和分析为标志。新的数据从多样的来源产生，这方面的实例包括购买交易记录、企业及政府记录和通信、电子邮件、社交媒体发帖、数字图片和视频、机器日志、来自嵌入式设备的信号、数字传感器、蜂窝电话全球定位卫星、航天卫星、科学计算以及大挑战科学。数据以多样的格式生成，其中得许多数据是无结构的，并且不适合输入到传统的数据库中。企业、政府和个人以前所未有的速度生成数据，并且在存储、分析和传送该数据方面遇到困难。为了保存累积的数据，每年在购买存储系统方面要花费数百亿美元。在用以处理数据的计算机系统上也要花费类似地巨大金额。

[0003] 在最现代的计算机和存储系统中，在被组织成存储分级结构的多层存储上容纳和部署数据。需要被经常并且快速地存取的数据被放置在最快速但是也最昂贵的层级，大多数数据(包括用于备份的拷贝)则优选地被存储在最密集并且最便宜的存储介质中。最快速并且最昂贵的数据存储层级是计算机系统的非易失性随机存取存储器或RAM，其驻留在紧邻微处理器核心的位置并且为随机数据存取给出最低等待时间和最高带宽。逐渐地更密集并且更便宜但是也更慢的各层(其对于随机存取具有逐渐地更高的等待时间和更低的带宽)包括非易失性固态存储器或闪存存储装置、硬盘驱动器(HDD)并且最后是磁带驱动器。

[0004] 为了更加有效地存储和处理不断增加的数据，计算机行业持续对数据存储介质的密度和速度以及对计算机的处理能力作出改进。但是数据量的增加速度远远超出计算和数据存储系统的容量和密度的改进。来自2014年的数据存储行业的统计数据表明，在过去的几年里所产生并捕获的新数据构成全世界至今所捕获的数据的一大部分。全世界至今为止所产生的数据的数量估计超出多个泽字节(一个泽字节是10^{21}个字节)。数据的大量增加对于必须可靠地存储、处理和传送该数据的数据存储、计算和通信系统提出了高要求。这就促使更多地使用无损数据简化或压缩技术来紧缩(compact)数据，从而能够以更低的成本来存储并且同样高效地处理和传送数据。

[0005] 已经出现了多种无损数据简化(reduction)或压缩技术，并且近年来发生了演进。这些技术对数据进行检查以寻找数据中的某种形式的冗余，并且利用该冗余在没有任何信息损失的情况下实现数据足迹(data footprint)的简化。对于期望利用数据中的特定形式的冗余的给定技术，所实现的数据简化的程度取决于在数据中找到该特定形式的冗余的频度。所希望的是数据简化技术能够灵活地发现并且利用数据中的任何可用的冗余。由于数据源自多种来源和环境并且具有多种格式，因此对于用以应对这一多样数据的通用无损数据简化技术的开发和采用的兴趣很大。除了字母表之外通用数据简化技术不需要关于输入数据的先验知识；因此通用数据简化技术一般可以被应用于任何和所有数据，而不需要事先知道数据的结构和统计分布特性。

[0006] 可以被用来比较数据压缩技术的不同实现方式的优度(goodness)量度包括在目标数据集上实现的数据简化的程度,实现压缩或简化的效率,以及解压缩并取回数据以供未来使用的效率。效率量度评估解决方案的性能和成本有效性。性能量度包括新数据可以被消耗并简化的吞吐量或摄取速率,对输入数据进行简化所需要的等待时间或时间,数据可以被解压缩并取回的吞吐量或速率,以及解压缩并取回数据所需要的等待时间或时间。成本量度包括任何所需的专用硬件组件的成本,比如微处理器核心或微处理器利用(中央处理单元利用),专用暂时存储器的数量和存储器带宽,以及对于保存数据的各个存储层级所需要的存取次数和带宽。应当提到的是,在简化数据足迹的同时提供高效且快速的压缩以及解压缩和取回不仅具有降低存储和传送数据的总体成本的好处,而且还具有高效地允许对于数据的后续处理的好处。

[0007] 当前在业内所使用的许多通用数据压缩技术是从在1977年由Abraham Lempel和Jacob Ziv开发的Lempel-Ziv压缩方法导出的,例如参见Jacob Ziv和Abraham Lempel的“A Universal Algorithm for Sequential Data Compression(用于顺序数据压缩的通用算法)”,IEEE transactions on information theory,Vol.IT-23,No.3,1977年5月。这种方法成为允许通过因特网的高效数据传输的基础。Lempel-Ziv方法(也就是LZ77、LZ78及其变体)通过用引用替换串的重复出现而简化数据足迹,其中所述引用是针对在顺序呈现的输入数据流的滑动窗口内所见到的所述串的先前的出现。在消耗来自输入数据流的给定数据块的新鲜串时,这些技术搜索过先前在直到窗口长度的当前和先前块内所见到的所有串。如果所述新鲜串是重复,则用对原始串的后向引用将其替换。如果通过重复串所消除的字节数目大于后向引用所需的字节数目,则实现了数据的简化。为了搜索过在窗口中所见到的所有串,并且为了提供最大串匹配,这些技术的实现方式采用多种方案,其中包括迭代扫描以及建立包含在窗口中见到的所有串的字典的临时簿记结构。在消耗新的输入字节以组装新鲜串时,这些技术或者扫描过现有窗口中的所有字节,或者对串的字典进行引用(随后是一些计算)以便判定是否找到重复并且用后向引用将其替换(或者判定是否需要对字典进行添加)。

[0008] Lempel-Ziv压缩方法常常伴随有应用于数据的第二优化,其中基于其在正被压缩的数据块中的出现频率或概率对源符号进行动态重编码,所述动态重编码常常采用可变宽度编码方案从而对于频率更高的符号使用长度更短的代码,从而导致数据的简化。对于这种基于熵的重编码方法的示例,参见David A.Huffman的“A Method for the Construction of Minimum-Redundancy Codes(用于构造最小冗余代码的方法)”,Proceedings of the IRE-Institute of Radio Engineers,1952年9月,pp.1098-1101。这种技术被称作Huffman重编码,并且通常需要第一遍经过数据以计算频率以及第二遍经过数据以实际编码数据。围绕这一主题的几种变型也在使用之中。

[0009] 使用这些技术的一个实例是一种被称作“Deflate”的方案,该方案将Lempel-Ziv LZ77压缩方法与Huffman重编码相组合。Deflate提供了压缩流数据格式规范,所述规范规定一种用于把字节序列表示成(通常更短的)比特序列的方法,以及一种用于把所述比特序列打包成字节的方法。Deflate方案最初由PKWARE,Inc.的Phillip W.Katz设计用于PKZIP归档实用程序。例如参见Phillip W.Katz的标题为“String searcher,and compressor using same(串搜索器以及使用串搜索器的压缩器)”的美国专利5,051,745,1991年9月24

日。美国专利5,051,745描述了一种用于针对预定目标串(输入串)搜索符号矢量(窗口)的方法。所述解决方案采用具有针对窗口中的每一个符号的指针的指针阵列,并且使用一种散列方法对窗口中的可能位置进行过滤,其中需要在所述可能位置处搜索输入串的完全相同的拷贝。随后是在这些位置处进行扫描和串匹配。

[0010] Deflate方案被实施在用于数据压缩的zlib库中。zlib是作为例如Linux、Mac OS X、iOS之类的几种软件平台以及多种游戏机的关键组件的软件库。zlib库提供Deflate压缩和解压缩代码以供zip(文件归档)、gzip(单文件压缩)、png(用于无损压缩图像的便携式网络图形格式)以及许多其他应用来使用。zlib现在被广泛用于数据传输和存储。服务器和浏览器的大多数HTTP事务使用zlib来压缩和解压缩数据。类似的实现方式正越来越多地被数据存储系统使用。

[0011] 由Intel Corp.在2014年4月公布的一篇标题为“High Performance ZLIB Compression on **Intel®** Architecture Processors(**Intel®**架构处理器上的高性能ZLIB压缩)”的文章描述了运行在当代Intel处理器(Core I7 4770处理器,3.4GHz,8MB高速缓存)上并且在Calgary数据资料库上操作的zlib库的优化版本的压缩和性能。在zlib中所使用的Deflate格式把用于匹配的最小串长度设定成3个字符,把最大匹配长度设定成256个字符,并且把窗口的大小设定成32千字节。所述实现方式提供对于9个优化等级的控制,其中第9级提供最高压缩但是使用最多计算并且实施最详尽的串匹配,第1级是最快的等级并且采用贪婪(greedy)串匹配。该文章报告,在使用单线程处理器的情况下,使用第1级(最快等级)zlib获得51%的压缩比并且对于输入数据平均花费17.66时钟/字节。在3.4GHz的时钟频率下,这意味着在用尽单一处理器核心的同时达到192MB/秒的摄取速率。所述报告还描述了在使用第6级优化获得压缩中的适度增益时其性能如何快速地下降到38MB/秒的摄取速率(平均88.1时钟/字节),并且在使用第9级优化时下降到16MB/秒的摄取速率(平均209.5时钟/字节)。

[0012] 现有的数据压缩方案通常使用当代微处理器上的单一处理器核心操作在从10MB/秒到200MB/秒的摄取速率下。为了进一步提升摄取速率,采用多个核心,或者减小窗口大小。使用定制硬件加速器会实现摄取速率的进一步改进,但是成本会增加。

[0013] 前面所描述的现有数据压缩方法在通常具有单一消息或文件或者几个文件的大小的本地窗口中能够有效地利用较短的串和符号等级的细粒度冗余。但是当在操作于较大或极大的数据集上并且需要高数据摄取和数据取回速率的应用中使用这些方法时则存在严重的限制和缺陷。

[0014] 一个重要的限制是这些方法的实用实现方式只能在本地窗口内高效地利用冗余。虽然这些实现方式可以接受任意长的输入数据流,但是效率决定在将于其中发现细粒度冗余的窗口的大小方面存在限制。这些方法是高度计算密集型的,并且需要对于窗口中的所有数据的频繁和快速的存取。在消耗产生新鲜输入串的输入数据的每一个新鲜字节(或几个字节)时触发各种簿记结构的串匹配和查找。为了达到所期望的摄取速率,用于串匹配的窗口和相关联的机器必须主要驻留在处理器高速缓存子系统中,从而在实践中对窗口大小构成约束。

[0015] 例如为了在单一处理器核心上达到200MB/秒的摄取速率,每个所摄取字节的平均可用时间预算(包括所有数据存取和计算)是5ns,这在使用具有3.4GHz操作频率的当代处

理器的情况下意味着17个时钟。这一预算容许对于芯片上高速缓存的存取(花费少量循环)以及随后的一些串匹配。当前的处理器具有几兆字节容量的芯片上高速缓存。对于主存储器的存取花费超出200个循环(～70ns),因此主要驻留在存储器中的更大窗口将使得摄取速率进一步变慢。此外,随着窗口大小增大以及去到重复串的距离增大,规定后向引用的长度的成本也增加,从而只能致使在更大的范围内搜索更长的串的重复。

[0016] 在大多数当代数据存储系统上,存储在存储分级结构的各个层级上的数据的足迹比系统中的存储器容量大几个数量级。举例来说,虽然系统可以提供数百吉字节的存储器,但是驻留在闪存存储装置中的活跃数据的数据足迹可以是数十太字节,并且存储系统中的总数据可以处于数百太字节到多个拍字节的范围。此外,对于每一个相继层级,对于后续存储层级的可实现的数据存取吞吐量下降一个数量级或更多。当滑动窗口变大到无法容纳在存储器中时,这些技术受到对于接下来的数据存储等级的随机IO(输入或输出操作)存取的显著更低的带宽和更高等待时间的节制。

[0017] 例如考虑具有4千字节的传入数据的一个文件或页面,所述文件或页面可以通过对已经存在于数据中并且分散在256太字节足迹上的例如100个平均长度为40字节的串进行引用而从现有数据组装。每一项引用将花费6个字节来规定其地址以及用于串长度的1个字节,同时有望节省40个字节。虽然在本例中描述的页面可以被压缩多于五倍,但是对应于该页面的摄取速率将受到获取并验证100个重复串所需的对于存储系统的100次或更多次IO存取的限制(即使可以完美地以低成本预测这些串驻留在何处)。在用尽存储系统的所有带宽的情况下,给出250000次随机IO存取/秒(这意味着对于4KB页面的1GB/秒的随机存取带宽)的存储系统只能以10MB/秒的摄取速率每秒压缩2500个这样的4KB大小的页面,从而使其不可用作存储系统。

[0018] 具有太字节或拍字节量级的大窗口大小的传统压缩方法的实现方式将受到对存储系统的减小的数据存取带宽的困扰,并且将是不可接受地缓慢。因此,这些技术的实用实现方式只有在能够容纳于处理器高速缓存或系统存储器中的窗口大小上本地存在冗余的情况下才能高效地发现和利用所述冗余。如果冗余数据在空间上或时间上与传入数据分开多个太字节、拍字节或艾字节,这些实现方式由于受到存储存取带宽的限制将无法以可接受的速度发现冗余。

[0019] 传统方法的另一个限制在于其不合适于随机数据存取。跨越被压缩的整个窗口的各个数据块需要在能够对任何块内的任何组块进行存取之前被解压缩。这就对窗口的大小构成了实用限制。此外,传统上在未压缩数据上实施的操作(例如搜索操作)无法高效地在已压缩数据上实施。

[0020] 传统方法(特别是基于Lempel-Ziv的方法)的另一个限制在于其仅仅沿着一个维度搜索冗余——也就是用后向引用替换完全相同的串。Huffman重编码方案的限制在于其需要经过数据两遍以便计算频率并且随后进行重编码。这样在更大的块上就变得较慢。

[0021] 在全局数据存储库上检测长重复串的数据压缩方法使用数字指纹处理与散列方案的组合。这种压缩处理被称作数据去重复(deduplication)。最基本的数据去重复技术把文件分解成固定大小块,并且在数据储存库中寻找重复块。如果创建了文件的拷贝,则第一文件中的每一个块将在第二文件中具有重复,并且可以用针对原始块的引用替换所述重复。为了加快潜在地重复块的匹配,采用一种散列方法。散列函数是把一个串转换成被称作

其散列值的数字值的函数。如果两个串相等,其散列值也相等。散列函数把多个串映射到给定的散列值,从而可以把长串简化成长度短得多的散列值。散列值的匹配将比两个长串的匹配快得多;因此首先进行散列值的匹配以便过滤掉可能是重复的可能串。如果输入串或块的散列值匹配存在于储存库中的串或块的散列值,随后则可以把输入串与储存库中的具有相同散列值的每一个串进行比较以便证实重复的存在。

[0022] 把文件分解成固定大小块是简单且方便的,并且固定大小块在高性能存储系统中是高度期望的。但是这种技术在其所能够发现的冗余的数量方面存在限制,这意味着这些技术的压缩等级较低。举例来说,如果对第一文件进行拷贝以创建第二文件,并且即使如果只把单一字节的数据插入到第二文件中,则所有下游块的对准都将改变,每一个新块的散列值将被重新计算,并且所述数据去重复方法将不再能找到所有重复。

[0023] 为了解决数据去重复方法中的这一限制,行业内采用了使用指纹处理在匹配内容的位置处同步和对准数据流。后面的这种方案导致基于指纹的可变大小块。Michael Rabin展示出如何能够使用随机选择的不可约多项式对比特串进行指纹处理,例如参见Michael O.Rabin的“Fingerprinting by Random Polynomials(通过随机多项式进行指纹处理)”,Center for Research in Computing Technology,Harvard University,TR-15-81,1981年。在这种方案中,随机选择的素数p被用来对长字符串进行指纹处理,这是通过计算被视为大整数对p取模的该串的余数。这种方案需要在£比特整数上实施整数运算,其中$k=\log_2(p)$。或者可以使用k次随机不可约多项式,指纹则是数据的多项式表示对素多项式取模。

[0024] 这种指纹处理方法被使用在数据去重复系统中以便识别将在该处建立组块边界的适当位置,从而使得系统可以在全局储存库中寻找这些组块的重复。可以在找到特定值的指纹时设定组块边界。作为这种用法的一个实例,通过采用32次或更低次多项式,可以对于输入数据中的每一个48字节串计算指纹(在输入的第一字节处开始,并且在随后的每一个相继字节处进行)。随后可以检查32比特指纹的13个低位比特,并且每当这13个比特的值是预先规定的值(例如值1)时则设定断点。对于随机数据,所述13个比特具有该特定值的概率将是2^{13}分之1,因此对于每8KB可能会遇到近似一个这样的断点,从而导致平均大小为8KB的可变大小组块。所述断点或组块边界将会有效地对准到取决于数据内容的指纹。当很久没有找到指纹时,可以在某一预先规定的阈值处强制断点,从而使得系统确保为储存库创建短于预先规定的大小的组块。例如参见Athicha Muthitacharoen、Benjie Chen和David Mazières的“A Low-bandwidth Network File System(低带宽网络文件系统)”,SOSP‘01,Proceedings of the eighteenth ACM symposium on Operating Systems Principles,10/21/2001,pp.174-187。

[0025] 由Michael Rabin和Richard Karp开发的Rabin-Karp串匹配技术提供了对于指纹处理和串匹配的效率的进一步改进(例如参见Michael O.Rabin和R.Karp的“Efficient Randomized Pattern-Matching Algorithms(高效的随机化模式匹配算法)”,IBM Jour.of Res.and Dev.,vol.31,1987年,pp.249-260)。应当提到的是,检查m字节子串的指纹的指纹处理方法可以在O(m)的时间内评估指纹处理多项式函数。由于这种方法将需要被应用在开始于例如n字节输入流的每一个字节的子串上,因此在整个数据流上实施指纹处理所需的总工作量将是O(n×m)。Rabin-Karp识别出被称作滚动散列(Rolling Hash)的散列函数,在

所述滚动散列上，通过独立于子串的长度仅仅进行恒定次数的运算，有可能从前一个子串计算下一个子串的散列值。因此，在向右移位一个字节之后，可以在新的m字节串上递增进行指纹计算。这样就把用以计算指纹的工作量减少到O(1)，并且把用于对整个数据流进行指纹处理的总工作量减少到O(n)，从而与数据的大小成线性。这样就大大加快了指纹的计算和识别。

[0026] 对于前面描述的数据去重复方法的典型的数据存取和计算要求可以被如下描述。对于给定的输入，一旦完成指纹处理从而创建组块，并且在计算出用于该组块的散列值之后，这些方法首先需要针对存储器和后续存储层级的一个存取集合，以便搜索并且查找保持储存库中的所有组块的散列值的全局散列表。这通常将需要针对存储的第一IO存取。在散列表中找到匹配之后是第二存储IO集合(取决于在储存库中存在多少具有相同散列值的组块，这通常是一次但是也可以多于一次)，以便获取具有相同散列值的实际数据组块。最后实施逐字节匹配以便把输入组块与所获取的潜在匹配组块进行比较，从而确认并且识别重复。随后是用对原始块的引用替换新的重复块的第三存储IO存取(针对元数据空间)。如果在全局散列表中没有匹配(或者如果没有找到复制)，系统需要一次IO以把新的块输入到储存库中，并且需要另一次IO来更新全局散列表以便输入新的散列值。因此，对于较大的数据集(其中元数据和全局散列表无法容纳在存储器中，因此需要存储IO对其进行存取)，这样的系统对于每个输入组块可能需要平均三次IO。通过采用多种过滤器可能实现进一步的改进，从而常常可以在无需用以对全局散列表进行存取的第一存储IO的情况下检测到全局散列表中的缺失，从而把对其中一些组块进行处理所需的IO次数减少到两次。

[0027] 给出250000次随机IO存取/秒(这意味着对于4KB页面的1GB/秒的随机存取带宽)的存储系统每秒可以摄取大约83333(250000除以每个输入组块3次IO)个平均大小为4KB的输入组块并且对其进行去重复，从而在用尽存储系统的所有带宽的情况下允许333MB/秒的摄取速率。如果仅使用存储系统的一半带宽(从而使得另一半可用于对所存储的数据进行存取)，这样的去重复系统仍然可以给出166MB/秒的摄取速率。如果在系统中有足够的处理能力可用，则(受到I/O带宽限制的)这些摄取速率是可以实现的。因此，在给定足够处理能力的情况下，数据去重复系统能够以经济的IO在全局数据范围内找到较大的数据重复，并且在当代存储系统上以每秒数百兆字节的摄取速率给出数据简化。

[0028] 基于前面的描述应当清楚的是，虽然这些去重复方法在全局范围内找到长串的重复方面是有效的，但是其主要在找到大的重复方面是有效的。如果数据在更细的粒度上存在变化或修改，则使用这种方法将不会找到可用的冗余。这大大减小了这些方法对其有效的数据集的广度。这些方法已被使用在特定的数据存储系统和应用中，例如对于数据的定期备份，其中正被备份的新数据只有几个文件被修改，其余部分都是已被保存在先前的备份中的文件的重复。同样地，基于数据去重复的系统常常被部署在其中产生数据或代码的多份精确拷贝的环境中，比如数据中心中的虚拟化环境。但是随着数据演进并且更加一般地或者在更细的粒度上被修改，基于数据去重复的技术则失去其有效性。

[0029] 一些方法(其通常被采用在数据备份应用中)不实施输入数据与其散列值匹配输入的串之间的实际的逐字节比较。这样的解决方案依赖于使用例如SHA-1之类的强散列函数的低冲突概率。但是由于冲突(其中多个不同的串可以映射到相同的散列值)的有限非零概率，这样的方法不能被视为提供无损数据简化，因此将不满足主存储和通信的高数据完

整性要求。

[0030] 一些方法组合多种现有的数据压缩技术。在这样的设置中，通常首先对数据应用全局数据去重复方法。随后在经过去重复的数据集上并且采用小窗口，应用与Huffman重编码相组合的Lempel-Ziv串压缩方法以实现进一步的数据简化。

[0031] 但是尽管采用了所有至此已知的技术，在不断增长和累积的数据的需求与世界经济使用最佳可用现代存储系统所能可负担地适应的情况之间仍然存在几个数量级的差距。在给定不断增长的数据所需要的非常高的存储容量需求的情况下，仍然需要进一步简化数据足迹的改进的方式。仍然需要开发解决现有技术的限制或者沿着尚未被现有技术解决的维度利用数据中的可用冗余的方法。与此同时，能够以可接受的速度并且以可接受的处理成本高效地存取和取回数据仍然非常重要。仍然需要能够直接对简化的数据高效地执行搜索操作。

[0032] 总而言之，长期以来一直需要能够利用较大和极大的数据集中的冗余并且提供高数据摄取、数据搜索和数据取回速率的无损数据减损解决方案。

发明内容

[0033] 这里描述的实施例的特征在于可在提供高数据摄取和数据取回速率的同时，对较大和极大的数据集进行无损数据简化，并且不存在现有数据压缩系统的缺陷和限制的技术和系统。

[0034] 具体而言，一些实施例可以从视频数据中提取压缩的移动图片数据和压缩的音频数据。接下来，实施例可以从压缩的移动图片数据提取内帧(I帧)(intra-frame)。实施例然后可以无损简化I帧以获得无损简化的I帧。无损简化I帧可以包括，对于每个I帧，(1)通过使用I帧对基于基本数据单元的内容组织基本数据单元的数据结构执行第一内容关联查找来识别第一组基本数据单元，以及(2)使用第一组基本数据单元来无损简化I帧。实施例可以附加地对压缩的音频数据解压缩来获得一组音频分量。接下来，对于所述一组音频分量中的每个音频分量，实施例可以(1)通过使用音频分量对基于基本数据单元的内容组织基本数据单元的数据结构执行第二内容关联查找来识别第二组基本数据单元，以及(2)使用第二组基本数据单元来无损简化音频分量。

[0035] 一些实施例可以初始化存储在第一存储器设备中并且被配置成基于基本数据单元的内容来组织基本数据单元的数据结构。接下来，实施例可以将输入数据分解成候选单元序列。对于每个候选单元，实施例可以(1)通过使用候选单元对数据结构执行内容关联查找来识别一组基本数据单元，以及(2)通过使用所述一组基本数据单元来无损简化候选单元，其中如果候选单元的大小未被充分简化，则候选单元作为新的基本数据单元被添加到数据结构中。接下来，实施例可以将无损简化的候选单元存储在第二存储器设备中。在检测到数据结构的一个或多个分量的大小大于阈值时，实施例可以(1)将数据结构的一个或多个分量移动到第二存储器设备，以及(2)初始化数据结构中被移动到第二存储器设备的一个或多个分量。无损简化的数据批次可以包括(1)在时间相邻的初始化之间存储在第二存储器设备上的无损简化的候选单元，以及(2)在时间相邻的初始化之间移动到第二存储器设备的数据结构的分量。在变型中，实施例可以基于存储在第二存储器设备上的无损简化的数据批次来创建一组包裹，其中所述一组包裹有助于数据从一个计算机到另一个计算机

的归档和移动。

[0036] 一些实施例可以将输入数据分解成候选单元序列。接下来,对于每个候选单元,实施例可以(1)将候选单元划分为一个或多个字段,(2)对于每个字段,用素多项式(prime polynomial)除以该字段以获得商和余数对,(3)基于一个或多个商和余数对确定名称,(4)通过使用名称对基于基本数据单元的相应名称的内容组织基本数据单元的数据结构执行内容关联查找来识别一组基本数据单元,以及(5)通过使用所述一组基本数据单元来无损简化候选单元。

[0037] 一些实施例可以将输入数据分解成候选单元序列。接下来,对于每个候选单元,实施例可以(1)通过使用候选单元对基于基本数据单元的内容组织基本数据单元的数据结构执行内容关联查找来识别一组基本数据单元,以及(2)通过使用所述一组基本数据单元来无损简化候选单元。实施例然后可以将无损简化的候选单元存储在一组蒸馏文件中。接下来,实施例可以将基本数据单元存储在一组基本数据单元文件中。在一些实施例中,每个无损简化的候选单元为用于简化候选单元的每个基本数据单元指定基本数据单元文件,该基本数据单元文件包含基本数据单元和可以在基本数据单元文件中找到该基本数据单元的位置的偏移量。在一些实施例中,每个蒸馏文件存储基本数据单元文件的列表,这些基本数据单元文件包含用于无损简化存储在蒸馏文件中的候选单元的基本数据单元。

[0038] 使用一组基本数据单元来无损简化数据单元(例如,I帧、音频分量、候选单元等)可以包括:(1)响应于确定(i)对所述一组基本数据单元的引用的大小以及(ii)重建程序的描述的大小的总和小于数据单元的大小的阈值比例,生成数据单元的第一无损简化表示,其中第一无损简化表示包括对所述一组基本数据单元中的每个基本数据单元的引用以及重建程序的描述;以及(2)响应于确定(i)对所述一组基本数据单元的引用的大小以及(ii)重建程序的描述的大小的总和大于或等于数据单元的大小的阈值比例,将数据单元添加为数据结构中的新基本数据单元,并生成数据单元的第二无损简化表示,其中第二无损简化表示包括对新基本数据单元的引用。注意的是,重建程序的描述可以指定变换序列,该变换序列在应用于所述一组基本数据单元(即,用于无损简化数据单元的一个或多个基本数据单元)时,产生数据单元。

附图说明

[0039] 图1A示出了根据这里所描述的一些实施例的用于数据简化的方法和装置,其把输入数据因式分解成各个单元并且从驻留在基本数据滤筛中的基本数据单元导出这些单元。

[0040] 图1B-1G示出了根据这里所描述的一些实施例的图1A中所示的方法和装置的各种变型。

[0041] 图1H给出了根据这里所描述的一些实施例的描述蒸馏数据(Distilled Data)的结构的格式和规范的一个实例。

[0042] 图1I到1P示出了对应于图1A到图1G中示出的用于数据简化的方法和装置的各种变型的输入数据到无损简化形式的概念性变换。

[0043] 图2示出了根据这里所描述的一些实施例的通过把输入数据因式分解成各个单元并且从驻留在基本数据滤筛中的基本数据单元导出这些单元而进行数据简化的处理。

[0044] 图3A、3B、3C、3D和3E示出了根据这里所描述的一些实施例的可以被用来基于其名

称对基本数据单元进行组织的不同的数据组织系统。

[0045] 图3F给出了根据这里所描述的一些实施例的自描述树节点数据结构。

[0046] 图3G给出了根据这里所描述的一些实施例的自描述叶节点数据结构。

[0047] 图3H给出了根据这里所描述的一些实施例的包括导航前瞻字段的自描述叶节点数据结构。

[0048] 图4示出了根据这里所描述的一些实施例的如何可以把256TB的基本数据组织成树形式的一个实例,并且呈现出如何可以把树布置在存储器和存储装置中。

[0049] 图5A-5C示出了关于如何可以使用这里所描述的实施例组织数据的一个实际的实例。

[0050] 图6A-6C分别示出了根据这里所描述的一些实施例的如何可以把树数据结构用于参照图1A-1C描述的内容关联映射器。

[0051] 图7A提供了根据这里所描述的一些实施例的可以在重建程序中规定的变换的一个实例。

[0052] 图7B示出了根据这里所描述的一些实施例的从基本数据单元导出候选单元的结果的实例。

[0053] 图8A-8E示出了根据这里所描述的一些实施例的如何通过把输入数据因式分解成固定大小单元并且把所述单元组织在参照图3D和3E描述的树数据结构中而实施数据简化。

[0054] 图9A-9C示出了根据这里所描述的一些实施例的基于图1C中示出的系统的Data Distillation™(数据蒸馏)方案的一个实例。

[0055] 图10A提供了根据这里所描述的一些实施例的关于如何对基本数据单元应用在重建程序中规定的变换以产生导出单元(Derivative Element)的一个实例。

[0056] 图10B-10C示出了根据这里所描述的一些实施例的数据取回处理。

[0057] 图11A-11G示出了根据这里所描述的一些实施例的包括Data Distillation™机制(可以利用软件、硬件或者其组合来实施)的系统。

[0058] 图11H示出了根据这里所描述的一些实施例的Data Distillation™装置如何可以与范例通用计算平台进行接口。

[0059] 图11I图解说明在块处理存储系统中,如何把Data Distillation™装置用于数据简化。

[0060] 图12A-12B示出了根据这里所描述的一些实施例的使用Data Distillation™装置在受到带宽约束的通信介质上传送数据。

[0061] 图12C-12K示出了根据这里所描述的一些实施例的由Data Distillation™装置对于各种使用模型所产生的简化数据的各个分量。

[0062] 图12L-12R示出了根据本文描述的一些实施例如何在分布式系统上部署和执行蒸馏处理以能够以非常高的摄取速率适应显著更大的数据集。

[0063] 图13-17图示了根据本文描述的一些实施例如何可以对简化的数据执行多维搜索和数据取回。

[0064] 图18A-18B表示用于按照MPEG 1,层3标准(也被称为MP3)的音频数据的压缩和解压缩的编码器和解码器的方框图。

[0065] 图18C表示如何增强首次示于图1A中的Data Distillation(数据蒸馏)装置,以对

MP3数据进行数据简化。

[0066] 图19表示可以如何增强首次示于图1A中的数据蒸馏装置，以对视频数据进行数据简化。

具体实施方式

[0067] 给出后面的描述是为了使得本领域技术人员能够制作和使用本发明，并且是在特定应用及其需求的情境中所提供的。本领域技术人员将很容易认识到针对所公开的实施例的各种修改，并且这里所定义的一般原理可以被应用于其他实施例和应用而不会背离本发明的精神和范围。因此，本发明不限于所示出的实施例，而是应当符合与这里所公开的原理和特征相一致的最宽泛的范围。在本公开内容中，当某一短语对于一个实体集合使用术语“和/或”时，除非另行表明，否则所述短语涵盖所述实体集合的所有可能组合。举例来说，短语“X、Y和/或Z”涵盖以下其中组合：“只有X”，“只有Y”，“只有Z”，“X和Y，但是没有Z”，“X和Z，但是没有Y”，“Y和Z，但是没有X”，以及“X、Y和Z”。

[0068] 使用基本数据滤筛的数据的高效无损简化

[0069] 在这里所描述的一些实施例中，数据被组织和存储，以便在整个数据集的全局范围内高效地发现和利用冗余。输入数据流被分解成被称作单元的构成片段或组块，并且以比单元本身更细的粒度检测和利用各个单元当中的冗余，从而减缩所存储的数据的总体足迹。识别出被称作基本数据单元的一个单元集合并且将其用作数据集的共同的共享构建块，并且将其存储在被称作基本数据存储库或基本数据滤筛的结构中。基本数据单元简单地是具有特定大小的比特、字节或数位的序列。取决于实现方式，基本数据单元可以是固定大小或可变大小。输入数据的其他构成单元从基本数据单元导出，并且被称作导出单元。因此，输入数据被因式分解成基本数据单元和导出单元。

[0070] 基本数据滤筛对基本数据单元进行排序和组织，从而使得可以按照内容关联方式对基本数据滤筛进行搜索和存取。在给定一些输入内容和一些限制的情况下可以对基本数据滤筛进行查询以取回包含该内容的基本数据单元。在给定输入单元的情况下，可以使用所述单元的值或者所述单元中的特定字段的值对基本数据滤筛进行搜索，以便快速地提供一个基本数据单元或者较小的基本数据单元集合，从中可以导出输入单元并且只利用规定所述导出所需的最小存储。在一些实施例中，基本数据滤筛中的单元被组织成树形式。通过在基本数据单元上实施变换从基本数据单元导出导出单元，这样的变换被规定在重建程序中，所述重建程序描述如何从一个或多个基本数据单元生成导出单元。距离阈值规定关于导出单元的所存储足迹的大小的限制。该阈值有效地规定导出单元与基本数据单元的最大可允许距离，并且还对可以被用来生成导出单元的重建程序的大小作出限制。

[0071] 导出数据的取回是通过在由所述导出规定的一个或多个基本数据单元上执行重建程序而实现的。

[0072] 在本公开内容中，前面描述的通用无损数据简化技术可以被称作Data Distillation™处理。所述处理实施类似于化学中的蒸馏的功能——把混合物分离成其构成单元。基本数据滤筛也被称作滤筛或Data Distillation™滤筛或基本数据存储库。

[0073] 在这种方案中，输入数据流被因式分解成一个单元序列，其中每一个单元是基本数据单元或者从一个或多个基本数据单元导出的导出单元。每一个单元被变换成无损简化

表示,所述无损简化表示在基本数据单元的情况下包括对基本数据单元的引用,并且在导出单元的情况下包括对所述导出中所涉及的一个或多个基本数据单元的引用,以及关于重建程序的描述。因此,输入数据流被因式分解成处于无损简化表示中的单元序列。(出现在无损简化表示中的)该单元序列被称作蒸馏数据流或蒸馏数据。蒸馏数据中的单元序列与输入数据中的单元序列具有一一对应关系,也就是说蒸馏数据中的单元序列中的第n个单元对应于输入数据中的单元序列中的第n个单元。

[0074] 在本公开内容中描述的通用无损数据简化技术接收输入数据流并且将其转换成蒸馏数据流与基本数据滤筛的组合,从而使得蒸馏数据流和基本数据滤筛的足迹的总和通常小于输入数据流的足迹。在本公开内容中,蒸馏数据流和基本数据滤筛被统称作无损简化数据,并且将被可互换地称作“简化数据流”或“简化数据”或“Reduced Data(简化数据)”。同样地,对于通过本公开内容中描述的无损数据简化技术所产生的出现在无损简化格式中的单元序列,可互换地使用以下术语:“简化输出数据流”、“简化输出数据”、“蒸馏数据流”、“蒸馏数据”以及“Distilled Data(蒸馏数据)”。

[0075] 图1A示出了根据这里所描述的一些实施例的用于数据简化的方法和装置,其把输入数据因式分解成各个单元并且从驻留在基本数据滤筛中的基本数据单元导出这些单元。该图示出了数据简化或Data Distillation™方法和装置的总体方块图,并且提供了功能组件、结构和操作的总览。图1A中示出的组件和/或操作可以使用软件、硬件或其组合来实现。

[0076] 从输入数据流接收字节序列并且将其作为输入数据102给出到数据简化装置103,其也被称作Data Distillation™装置。解析器和因式分解器104对传入数据进行解析并且将其分解成组块或候选单元。因式分解器决定将在输入流中的何处插入中断以便把该流切分成候选单元。一旦识别出数据中的两处接连的中断,则由解析器和因式分解器创建候选单元105并且将其给出到基本数据滤筛106,所述基本数据滤筛也被称作Data Distillation™滤筛。

[0077] Data Distillation™滤筛或基本数据滤筛106包含所有基本数据单元(在图1A中被标记成PDE),并且基于其值或内容对其进行排序和组织。所述滤筛提供对于两种存取的支持。首先,可以通过对基本数据单元驻留在滤筛中的位置的位置的引用对每一个基本数据单元进行直接存取。其次,可以通过使用内容关联映射器121按照内容关联方式对各个单元进行存取,所述内容关联映射器121可以通过软件、硬件或其组合来实施。针对滤筛的这种第二存取形式是一项重要的特征,并且由所公开的实施例使用来识别与候选单元105精确地匹配的基本数据单元,或者用来识别可以从中导出候选单元的基本数据单元。具体来说,在给定候选单元(例如候选单元105)的情况下,可以对基本数据滤筛106进行搜索(基于候选单元105的值或者基于候选单元105中的特定字段的值),以便快速地提供一个基本数据单元107或者基本数据单元107的较小集合,从中可以导出候选单元并且只利用规定所述导出所需的最小存储。

[0078] 可以利用其值分散在数据空间内的一个基本数据单元集合对所述滤筛或基本数据滤筛106进行初始化。或者根据这里参照图1A-1C和图2所描述的Data Distillation™处理,所述滤筛最初可以是空的,并且可以随着摄取数据将基本数据单元动态地添加到所述滤筛。

[0079] 导出器110接收候选单元105以及所取回的适合于导出的基本数据单元107(从基

本数据滤筛106相关联地取回的内容)，确定是否可以从这些基本数据单元当中的一个或多个导出候选单元105，生成简化数据分量115(由对相关的基本数据单元的引用和重建程序构成)，并且向基本数据滤筛提供更新114。如果候选单元是所取回的基本数据单元的重复，则导出器在蒸馏数据108中放入对位于基本数据滤筛中的基本数据单元的引用(或指针)，并且还有表明这是基本数据单元的指示。如果没有找到重复，则导出器把候选单元表达成在一个或多个所取回的基本数据单元上实施的一项或多项变换的结果，其中所述变换序列被统称作重建程序，例如重建程序119A。每一项导出可能需要由导出器构造该项导出自身所独有的程序。重建程序规定可以对基本数据单元应用的例如插入、删除、替换、串联、算术以及逻辑运算之类的变换。如果导出单元的足迹(被计算成重建程序的大小加上针对所需的基本数据单元的引用的大小)处在关于候选单元的特定的指定距离阈值之内(以便允许数据简化)，则把候选单元改订成导出单元并且由重建程序与对(多个)相关基本数据单元的引用的组合替换——这些形成本例中的简化数据分量115。如果超出所述阈值，或者如果没有从基本数据滤筛取回适当的基本数据单元，则可以指示基本数据滤筛把所述候选安装成新鲜基本数据单元。在这种情况下，导出器在蒸馏数据中放入对新添加的基本数据单元的引用，并且还有表明这是基本数据单元的指示。

[0080] 对数据取回的请求(例如取回请求109)可以采取对基本数据滤筛中的包含基本数据单元的位置的引用的形式，或者在导出项的情况下可以采取对基本数据单元的此类引用与相关联的重建程序的组合的形式(或者在基于多个基本数据单元的导出项的情况下是对多个基本数据单元的引用与相关联的重建程序的组合)。通过使用对基本数据滤筛中的基本数据单元的一项或多项引用，取回器111可以对基本数据滤筛进行存取以便取回一个或多个基本数据单元，并且把所述一个或多个基本数据单元以及重建程序提供到重建器112，所述重建器112在所述一个或多个基本数据单元上执行(在重建程序中规定的)变换以便生成重建数据116(也就是所请求的数据)，并且响应于数据取回请求将其递送到取回数据输出113。

[0081] 在该实施例的一种变型中，基本数据单元可以通过压缩形式(使用本领域内已知的技术，包括Huffman编码和Lempel Ziv方法)被存储在滤筛中，并且在需要时被解压缩。这样做的优点是简化了基本数据滤筛的总体足迹。唯一的约束在于内容关联映射器121必须像以前一样继续提供对于基本数据单元的内容关联存取。

[0082] 图1B和1C示出了根据这里所描述的一些实施例的图1A中所示的方法和装置的变型。在图1B中，重建程序可以被存储在基本数据滤筛中，并且像基本数据单元那样被对待。对重建程序的引用或指针119B被提供在蒸馏数据108中，而不是提供重建程序119A本身。如果重建程序由其他导出项共享，并且如果对重建程序的引用或指针(加上在重建程序与对重建程序的引用之间作出区分所需的任何元数据)所需的存储空间小于重建程序本身，则实现进一步的数据简化。

[0083] 在图1B中，重建程序可以像基本数据单元那样被对待和存取，并且作为基本数据单元被存储在基本数据滤筛中，从而允许从基本数据滤筛对重建程序进行内容关联搜索和取回。在用以创建导出单元的导出处理期间，一旦导出器110确定对于导出所需要的重建程序，其随后可以确定该候选重建程序是否已经存在于基本数据滤筛中，或者确定是否可以从已经存在于基本数据滤筛中的另一个条目导出该候选重建程序。如果候选重建程序已经

存在于基本数据滤筛中，则导出器110可以确定对所述预先存在的条目的引用，并且把所述引用包括在蒸馏数据108中。如果可以从已经驻留在基本数据滤筛中的现有条目导出候选重建程序，则导出器可以把候选重建程序的导出项或改订递送到蒸馏数据，也就是说导出器在蒸馏数据中放入对预先存在于基本数据滤筛中的条目的引用连同从所述预先存在的条目导出候选重建程序的增量重建程序。如果候选重建程序既不存在于基本数据滤筛中也无法从基本数据滤筛中的条目导出，则导出器110可以把重建程序添加到基本数据滤筛中(把重建程序添加到滤筛的操作可以返回对新添加的条目的引用)，并且把对重建程序的引用包括在蒸馏数据108中。

[0084] 图1C给出了根据这里所描述的一些实施例的图1B中所示的方法和装置的一种变型。具体来说，图1C中的被用来存储和查询重建程序的机制类似于被用来存储和查询基本数据单元的机制，但是重建程序被保持在与包含基本数据单元的结构分开的结构(称为基本重建程序滤筛)中。这样的结构中的条目被称作基本重建程序(在图1C中被标记为PRP)。回想到基本数据滤筛106包括支持快速内容关联查找操作的内容关联映射器121。图1C中示出的实施例包括类似于内容关联映射器121的内容关联映射器122。在图1C中，内容关联映射器122和内容关联映射器121被显示成基本数据滤筛或基本数据存储库106的一部分。在其他实施例中，内容关联映射器122和重建程序可以在称为基本重建程序滤筛的结构中与基本数据滤筛或基本数据存储库106分开存储。

[0085] 在该实施例的一种变型中，基本数据单元可以通过压缩形式(使用本领域内已知的技术，包括Huffman编码和Lempel Ziv方法)被存储在滤筛中，并且在需要时被解压缩。同样地，基本重建程序可以通过压缩形式(使用本领域内已知的技术，包括Huffman编码和Lempel Ziv方法)被存储在基本重建程序滤筛中，并且在需要时被解压缩。这样做的优点是减缩了基本数据滤筛和基本重建程序滤筛的总体足迹。唯一的约束在于内容关联映射器121和122必须像以前一样继续提供对于基本数据单元和基本重建程序的内容关联存取。

[0086] 图1D给出了根据这里所描述的一些实施例的图1A中所示的方法和装置的一种变型。具体来说，在图1D所描述的实施例中，基本数据单元被内联存储在蒸馏数据中。基本数据滤筛或基本数据存储库106继续提供对于基本数据单元的内容关联存取，并且继续在逻辑上包含基本数据单元。其保持对内联位于蒸馏数据中的基本数据单元的引用或链接。例如在图1D中，基本数据单元130内联位于蒸馏数据108中。基本数据滤筛或基本数据存储库106保持对基本数据单元130的引用131。同样地，在这种设置中，导出单元的无损简化表示将包含对所需的基本数据单元的引用。在数据取回期间，取回器111将从所需的基本数据单元所处的位置获取所述基本数据单元。

[0087] 图1E给出了根据这里所描述的一些实施例的图1D中所示的方法和装置的一种变型。具体来说，在图1E所描述的实施例中，与图1B中所示出的设置一样，重建程序可以从其他基本重建程序导出，并且被规定为增量重建程序加上对基本重建程序的引用。这样的基本重建程序像基本数据单元一样被对待，并且在逻辑上被安装在基本数据滤筛中。此外，在这种设置中，基本数据单元和基本重建程序都被内联存储在蒸馏数据中。基本数据滤筛或基本数据存储库106继续提供对于基本数据单元和基本重建程序的内容关联存取，并且继续在逻辑上包含这些基本数据单元和基本重建程序，同时保持对这些基本数据单元和基本重建程序内联位于蒸馏数据中的位置的引用或链接。例如在图1E中，基本数据单元130内联

位于蒸馏数据108中。同样地在图1E中，基本重建程序132内联位于蒸馏数据中。基本数据滤筛或基本数据存储库106保持对基本数据单元130(即PDE_i)的引用131(即Reference_to_PDE_i)，以及对基本重建程序132(即Prime_Recon_Program_1)的引用133(即Reference_to_PDEj)。同样地，在这种设置中，导出单元的无损简化表示将包含对所需的基本数据单元和所需的基本重建程序的引用。在数据取回期间，取回器111将从所需的分量在相应的蒸馏数据中所处的位置获取所述分量。

[0088] 图1F给出了根据这里所描述的一些实施例的图1E中所示的方法和装置的一种变型。具体来说，在图1F所描述的实施例中，与图1C中所示出的设置一样，基本数据滤筛108包含分开的映射器——用于基本数据单元的内容关联映射器121和用于基本重建程序的内容关联映射器122。

[0089] 图1G给出了图1A到1F中所示的方法和装置的一种更加一般化的变型。具体来说，在图1G所描述的实施例中，基本数据单元可以位于基本数据滤筛中或者内联位于蒸馏数据中。一些基本数据单元可以位于基本数据滤筛中，其他的基本数据单元则内联位于蒸馏数据中。同样地，基本重建程序可以位于基本数据滤筛中或者内联位于蒸馏数据中。一些基本重建程序可以位于基本数据滤筛中，其他的基本重建程序则内联位于蒸馏数据中。基本数据滤筛在逻辑上包含所有基本数据单元和基本重建程序，并且在基本数据单元或基本重建程序内联位于蒸馏数据中的情况下，基本数据滤筛提供对其位置的引用。

[0090] 前面对于把输入数据因式分解成各个单元并且从驻留在基本数据滤筛中的基本数据单元导出这些单元的用于数据简化的方法和装置的描述仅仅是出于说明和描述的目的而给出的。所述描述不意图进行穷举或者把本发明限制到所公开的形式。因此，本领域技术人员将会想到许多修改和变型。

[0091] 图1H给出了根据这里所描述的一些实施例的描述用于Data Distillation™处理的方法和装置的图1A-1G中的蒸馏数据119A的结构的格式和规范的一个实例。由于Data Distillation™处理把输入数据因式分解成基本数据单元和导出单元，因此用于数据的无损简化表示的格式在蒸馏数据中标识这些单元并且描述这些单元的各个分量。自描述格式标识蒸馏数据中的每一个单元，表明其是基本数据单元还是导出单元，并且描述该单元的各个分量，也就是对安装在滤筛中的一个或多个基本数据单元的引用，对安装在基本数据滤筛中的重建程序的引用(如图1B的119B)，或者对存储在基本重建程序(PRP)滤筛中的重建程序的引用(如图1C的119C)，以及内联重建程序(RP)。基本重建程序(PRP)滤筛也被可互换地称作基本重建程序(PRP)存储库。图1H中的格式通过在多个基本数据单元上执行重建程序而规定导出，其中导出单元和每一个基本数据单元的大小是独立地可规定的。图1H中的格式还规定内联位于蒸馏数据中而不是位于基本数据滤筛内的基本数据单元。这是通过操作码编码7规定的，其规定单元的类型是内联位于蒸馏数据中的基本数据单元。蒸馏数据使用该格式被存储在数据存储系统中。该格式中的数据被数据取回器111消耗，从而可以获取并且随后重建数据的各个分量。

[0092] 图1I到1P示出了对应于图1A到图1G中示出的用于数据简化的方法和装置的各种变型的输入数据到无损简化形式的概念性变换。图1I示出了输入数据流如何被因式分解成候选单元，并且随后候选单元被视为基本数据单元或导出单元。最后，数据被变换成无损简化形式。图1I到1N示出了对应于各个实施例的无损简化形式的各种变型。

[0093] 图1I和图1J示出了通过图1A中所示的方法和装置所产生的数据的无损简化形式的实例。图1I中的无损简化形式包括内容关联映射器，并且是允许连续的进一步数据摄取以及针对现有的基本数据单元简化该数据的形式，与此同时，图1J中的无损简化形式不再保留内容关联映射器，从而导致更小的数据足迹。图1K和图1L示出了通过图1C中所示的方法和装置所产生的数据的无损简化形式的实例。图1K中的无损简化形式包括内容关联映射器，并且是允许连续的进一步数据摄取以及针对现有的基本数据单元和基本重建程序简化该数据的形式，与此同时，图1L中的无损简化形式不再保留内容关联映射器，从而导致更小的数据足迹。

[0094] 图1M和图1N示出了通过图1F中所示的方法和装置所产生的数据的无损简化形式的实例，其中基本数据单元和基本重建程序内联位于蒸馏数据中。图1M中的无损简化形式包括内容关联映射器，并且是允许连续的进一步数据摄取以及针对现有的基本数据单元和基本重建程序简化该数据的形式，与此同时，图1N中的无损简化形式不再保留内容关联映射器，从而导致更小的数据足迹。图1O和图1P示出了通过图1G中所示的方法和装置所产生的数据的无损简化形式的实例，其中基本数据单元和基本重建程序可以内联位于蒸馏数据中或者位于基本数据滤筛中。图1O中的无损简化形式包括内容关联映射器，并且是允许连续的进一步数据摄取以及针对现有的基本数据单元和基本重建程序简化该数据的形式，与此同时，图1P中的无损简化形式不再保留内容关联映射器，从而导致更小的数据足迹。

[0095] 在图1A到P所示出的实施例的变型中，简化数据的各个分量可以使用本领域内已知的技术(比如Huffman编码和Lempel Ziv方法)被进一步简化或压缩，并且通过该压缩形式被存储。这些分量可以随后在需要被使用在数据蒸馏装置中时被解压缩。这样做的好处是进一步简化了数据的总体足迹。

[0096] 图2示出了根据这里所描述的一些实施例的通过把输入数据因式分解成各个单元并且从驻留在基本数据滤筛中的基本数据单元导出这些单元而进行数据简化的处理。随着输入数据到达，其可以被解析和因式分解或者分解成一系列候选单元(操作202)。从输入消耗下一个候选单元(操作204)，并且基于候选单元的内容对基本数据滤筛实施内容关联查找，以便查看是否存在可以从中导出候选单元的任何适当的单元(操作206)。如果基本数据滤筛没有找到任何这样的单元(操作208的“否”分支)，则候选单元将作为新的基本数据单元被分配并且输入到滤筛中，并且在蒸馏数据中为候选单元创建的条目将是对新创建的基本数据单元的引用(操作216)。如果对基本数据滤筛的内容关联查找确实产生可以潜在地从中导出候选单元的一个或多个适当的单元(操作208的“是”分支)，则在所取回的基本数据单元上实施分析和计算以便从中导出候选单元。应当提到的是，在一些实施例中，首先仅获取用于适当的基本数据单元的元数据并且在所述元数据上实施分析，并且只有在认为有用的情况下才随后获取适当的基本数据单元(在这些实施例中，用于基本数据单元的元数据提供关于基本数据单元的内容的一些信息，从而允许系统基于元数据快速地排除匹配或者评估可导出性)。在其他实施例中，基本数据滤筛直接取回基本数据单元(也就是说在取回基本数据单元之前并不首先取回元数据以便对元数据进行分析)，从而在所取回的基本数据单元上实施分析和计算。

[0097] 实施第一检查以便查看候选是否任何这些单元的重复(操作210)。可以使用任何适当的散列技术加速这一检查。如果候选与从基本数据滤筛取回的基本数据单元完全相同

(操作210的“是”分支),则蒸馏数据中的为候选单元创建的条目由对该基本数据单元的引用以及表明该条目是基本数据单元的指示所替换(操作220)。如果没有找到重复(操作210的“否”分支),则基于候选单元从基本数据滤筛取回的条目被视为潜在地可以从中导出候选单元的条目。以下是基本数据滤筛的重要、新颖而且并非是显而易见的特征:当没有在基本数据滤筛中找到重复时,基本数据滤筛可以返回基本数据单元,所述基本数据单元虽然并非与候选单元完全相同,却是可以潜在地通过对(多个)基本数据单元应用一项或多项变换而导出候选单元的单元。所述处理随后可以实施分析和计算,以便从最适当的基本数据单元或者适当的基本数据单元的集合导出候选单元(操作212)。在一些实施例中,所述导出把候选单元表达成在一个或多个基本数据单元上实施的变换的结果,这样的变换被统称作重建程序。每一项导出可能需要构造其自身独有的程序。除了构造重建程序之外,所述处理还可以计算通常表明存储候选单元的改订以及从所述改订重建候选单元所需要的存储资源和/或计算资源的水平的距离量度。在一些实施例中,导出单元的足迹被用作从(多个)基本数据单元到候选的距离度量——具体来说,距离量度可以被定义成重建程序的大小加上对在导出中所涉及的一个或多个基本数据单元的引用的大小的总和。可以选择具有最短距离的导出。把对应于该导出的距离与距离阈值进行比较(操作214),如果该距离没有超出距离阈值,则接受该导出(操作214的“是”分支)。为了产生数据简化,所述距离阈值必须总是小于候选单元的大小。举例来说,距离阈值可以被设定到候选单元大小的50%,从而使得只有在导出项的足迹小于或等于候选单元足迹的一半时才接收导出项,从而对于为之存在适当导出的每一个候选单元确保2x或更大的简化。距离阈值可以是预定的百分比或比例,其或者是基于用户规定的输入或者是由系统选择。距离阈值可以由系统基于系统的静态或动态参数确定。一旦导出被接收,候选单元被改订并且被重建程序与对一个或多个基本数据单元的引用的组合所替换。蒸馏数据中的为候选单元创建的条目被所述导出所替换,也就是说被表明这是导出单元的指示连同重建程序加上对在导出中所涉及的一个或多个基本数据单元的引用所替换(操作218)。另一方面,如果对应于最佳导出的距离超出距离阈值(操作214中的“否”分支),则将不会接收任何可能的导出项。在这种情况下,候选单元可以作为新的基本数据单元被分配并且输入到滤筛中,并且在蒸馏数据中为候选单元创建的条目将是对新创建的基本数据单元的引用连同表明这是基本数据单元的指示(操作216)。

[0098] 最后,所述处理可以检查是否存在任何附加的候选单元(操作222),并且如果还有更多候选单元则返回操作204(操作222的“是”分支),或者如果没有更多候选单元则终止处理(操作222的“否”分支)。

[0099] 可以采用多种方法来实施图2中的操作202,也就是对传入数据进行解析并且将其分解成候选单元。因式分解算法需要决定将在字节流中的何处插入中断以便把该流切分成候选单元。可能的技术包括(而不限于)把流分解成固定大小块(比如4096字节的页面),或者应用指纹处理方法(比如对输入流的子串应用随机素多项式的技术)以便在数据流中定位变成单元边界的指纹(这种技术可以导致可变大小单元),或者对输入进行解析以便检测报头或者某种预先声明的结构并且基于该结构来界定单元。可以对输入进行解析以便检测通过图式(schema)声明的特定结构。可以对输入进行解析以便在数据中检测预先声明的模式、语法或规则表达法的存在。一旦识别出数据中的两处接连的中断,则创建候选单元(所述候选单元是位于所述两处接连的中断之间的数据)并且将其呈现到基本数据滤筛以供内

容关联查找。如果创建了可变大小单元，则需要规定候选单元的长度并且作为元数据与候选单元一起携带。

[0100] 基本数据滤筛的一项重要功能是基于为之给出的候选单元而提供内容关联查找并且快速地提供一个基本数据单元或者较小的基本数据单元集合，从中可以导出候选单元并且只利用规定所述导出所需的最小存储。在给定较大数据集的情况下，这是一个困难的问题。在给定以太字节计的数据的情况下，即使对于千字节大小的单元，仍然要搜索数以十亿计的单元并且从中作出选择。这一问题在更大的数据集上甚至会更加严重。因此变得很重要的是使用适当的技术对单元进行组织和排序，并且随后在单元的该组织内检测相似性和可导出性，以便能够快速地提供适当的基本数据单元的较小集合。

[0101] 可以基于每一个单元(也就是基本数据单元)的值对滤筛中的条目进行排序，从而可以按照升序或降序通过值来安排所有的条目。或者可以沿着基于单元中的特定字段的值的主轴对条目进行排序，随后是使用单元的其余内容的次要轴。在本上下文中，字段是来自单元的内容的邻接字节的集合。可以通过对单元的内容应用指纹处理方法来定位字段，从而使得指纹的位置标识字段的位置。或者可以选择单元内容内部的特定的固定偏移量以便定位字段。还可以采用其他方法以定位字段，其中包括而不限于对单元进行解析以便检测所声明的特定结构并且定位该结构内的字段。

[0102] 在另一种形式的组织中，单元内的特定字段或字段组合可以被视为维度，从而可以使用这些维度的串联以及随后的每一个单元的剩余内容对数据单元进行排序和组织。一般来说，字段与维度之间的对应性或映射可以是任意地复杂。例如在一些实施例中，确切地一个字段可以映射到确切地一个维度。在其他实施例中，多个字段的组合(例如F1、F2和F3)可以映射到一个维度。可以通过串联两个字段或者通过对其应用任何其他适当的功能而实现字段的组合。重要的要求是被用来对单元进行组织的字段、维度以及单元的剩余内容的安排必须允许通过其内容唯一地识别所有基本数据单元并且在滤筛中对其进行排序。

[0103] 在又一个实施例中，可以将某个合适的函数(诸如代数或算术变换)应用于单元，其中该函数具有以下属性：函数的结果唯一地识别每个单元。在一个这样的实施例中，每个单元被素多项式或某个选择的数或值除，并且除的结果(其包括商和余数对)被用作在基本数据滤筛中组织和排序单元的函数。例如，包含余数的位可以形成函数的结果的前导字节，后面跟着包含商的位。或替代地，包含商的位可以用于形成函数的结果的前导字节，后面跟着包含余数的位。对于用于除输入单元的给定除数，商和余数对将唯一地识别该单元，因此该对可以用于形成函数的结果，该结果用于在基本数据滤筛中组织和排序单元。通过将该函数应用于每个单元，可以基于函数的结果在滤筛中组织基本数据单元。该函数仍将唯一地识别每个基本数据单元，并将提供在基本数据滤筛中排序和组织基本数据单元的一种替代方法。

[0104] 在又一个实施例中，可以将某些合适的函数(诸如代数或算术变换)应用于单元的每个字段，其中该函数具有以下属性：函数的结果唯一地识别该字段。例如，可以对每个单元的内容的连续字段或连续部分执行诸如除以合适的多项式或数或值之类的函数，使得可以将连续函数的结果的串联用于排序和组织基本数据滤筛中的单元。注意的是，在每个字段上，可以将不同的多项式用于除法。每个函数将根据该部分或字段的除法运算得出的商和余数提供适当排序的位的串联。通过使用应用于单元的字段的函数的这种串联，可以在

滤筛中对每个基本数据单元进行排序和组织。函数的串联仍将唯一地识别每个基本数据单元，并将提供在基本数据滤筛中排序和组织基本数据单元的一种替代方法。

[0105] 在一些实施例中，单元的内容可以被表示成下面的表达法：单元＝头部.*sig1.*sig2.*…sigI.*…sigN.*尾部，其中“头部”是包括单元的开头字节的字节序列，“尾部”是包括单元的结尾字节的字节序列，“sig1”、“sig2”、“sigI”和“sigN”是表征单元的单元内容主体内的特定长度的各个签名或模式或规则表达法或字节序列。各个签名之间的表达法“.*”是通配符表达法，也就是允许除了表达法“.*”之后的签名之外的其他任何值的任意数目的中间字节的规则表达法标记。在一些实施例中，N元组(sig1,sig2,…sigI,…sigN)被称作单元的骨架数据结构或骨架，并且可以被视为单元的简化实质子集或实质。在其他实施例中，(N+2)元组(头部,sig1,sig2,…sigI,…sigN,尾部)被称作单元的骨架数据结构或骨架。或者可以采用头部或尾部连同其余签名的N+1元组。

[0106] 可以对单元的内容应用指纹处理方法，以便确定单元内容内的骨架数据结构的各个分量(或签名)的位置。或者可以选择单元内容内部的特定的固定偏移量以便定位分量。还可以采用其他方法来定位骨架数据结构的分量，其中包括而不限于对单元进行解析以便检测所声明的特定结构并且定位该结构内的分量。可以基于其骨架数据结构在滤筛中对基本数据单元进行排序。换句话说，单元的骨架数据结构的各个分量可以被视为维度，从而可以使用这些维度的串联以及随后的每一个单元的剩余内容对滤筛中的基本数据单元进行排序和组织。

[0107] 一些实施例把输入数据因式分解成候选单元，其中每一个候选单元的大小显著大于对全局数据集中的所有此类单元进行存取所需要的引用的大小。关于被分解成此类数据组块(并且按照内容关联方式被存取)的数据的一项观察是，实际的数据关于数据组块所能规定的全部可能值是非常稀疏的。例如考虑1泽字节数据集。需要大约70个比特对数据集中的每一个字节进行寻址。在128字节(1024比特)的组块大小下，在1泽字节数据集中近似有2^{63}个组块，因此需要63个比特(少于8个字节)对所有组块进行寻址。应当提到的是，1024比特的单元或组块可以具有2^{1024}个可能值当中的一个，而数据集中的给定组块的实际值的数目最多是2^{63}个(如果所有组块都不同的话)。这表明实际的数据关于通过一个单元的内容所能达到或命名的值的数目是极为稀疏的。这就允许使用非常适合于组织非常稀疏的数据的树结构，从而允许高效的基于内容的查找，允许把新的单元高效地添加到树结构，并且在对于树结构本身所需要的增量存储方面是成本有效的。虽然在1泽字节数据集中仅有2^{63}个不同的组块，从而只需要63个区分信息比特将其区别开，但是相关的区分比特可能分散在单元的整个1024个比特上，并且对于每一个单元出现在不同的位置处。因此，为了完全区分所有单元，仅仅检查来自内容的固定的63比特是不够的，相反，整个单元内容都需要参与单元的分拣，特别在提供对于数据集中的任一个和每一个单元的真实内容关联存取的解决方案中尤其是如此。在Data Distillation™框架中，希望能够在被用来对数据进行排序和组织的框架内检测可导出性。考虑到所有前述内容，基于内容的树结构(其随着检查更多内容逐渐地区分数据)是对经过因式分解的数据集中的所有单元进行排序和区分的适当组织。这样的结构提供了可以被作为可导出单元的分组或者具有类似的可导出性属性的单元分组来对待的许多中间子树等级。这样的结构可以利用表征每一个子树的元数据或者利用表征每一个数据单元的元数据通过分级方式被加强。这样的结构可以有效地传达其所包含的整

个数据的构成，包括数据中的实际值的密度、邻近性和分布。

[0108] 一些实施例把基本数据单元按照树形式组织在滤筛中。每一个基本数据单元具有从该基本数据单元的整个内容构造的独特“名称”。该名称被设计成足以唯一地标识基本数据单元，并且将其与树中的所有其他单元作出区分。可以通过几种方式从基本数据单元的内容构造名称。名称可以简单地由基本数据单元的所有字节构成，这些字节按照其存在于基本数据单元中的相同顺序出现在名称中。在另一个实施例中，被称作维度的特定字段或字段组合(其中字段和维度在前面作了描述)被用来形成名称的开头字节，基本数据单元的其余内容则形成名称的其余部分，从而使得基本数据单元的整个内容都参与创建单元的完整并且唯一的名称。在另一个实施例中，单元的骨架数据结构的字段被选择成维度(其中字段和维度在前面作了描述)，并且被用来形成名称的开头字节，基本数据单元的其余内容形成名称的其余部分，从而使得基本数据单元的整个内容都参与创建单元的完整并且唯一的名称。

[0109] 在一些实施例中，可以通过对单元执行代数或算术变换来计算单元的名称，同时保留每个名称唯一地识别每个单元的属性。在一个这样的实施例中，将每个单元除以素多项式或某个所选择的数字或值，并将除法的结果(即商和余数对)用于形成单元的名称。例如，包含余数的位可以形成名称的前导字节，后面跟着包含商的位。或者替代地，包含商的位可以用于形成名称的前导字节，后面跟着包含余数的位。对于用于除输入单元的给定除数，商和余数对将唯一地识别该单元，因此该对可以用于形成每个单元的名称。通过使用名称的这种表示形式，可以基于基本数据单元的名称在滤筛中组织基本数据单元。名称仍将唯一地识别每个基本数据单元，并将提供在基本数据滤筛中排序和组织基本数据单元的一种替代方法。

[0110] 在另一个实施例中，可以采用这种生成名称的方法的变体(其涉及除法和提取商/余数对)，其中可以在每个单元的内容的连续字段或连续部分上执行除以合适的多项式或数或值的除法，从而产生每个单元的名称的连续部分(每个部分是根据该部分或字段的除法运算得到的商和余数的适当排序的位的串联)。注意的是，在每个字段上，可以将不同的多项式用于除法。使用名称的这种表示形式，可以根据基本数据单元的名称在滤筛中组织基本数据单元。名称仍将唯一地识别每个基本数据单元，并将提供在基本数据滤筛中排序和组织基本数据单元的一种替代方法。

[0111] 每一个基本数据单元的名称被用来在树中对基本数据单元进行排序和组织。对于大多数实际的数据集，即使是大小非常大的那些数据集(比如由2^{58}个4KB大小单元构成的1泽字节数据集)，也预期到名称的较小字节子集将常常可以用来对树中的大部分基本数据单元进行分拣和排序。

[0112] 图3A、3B、3C、3D和3E示出了根据这里所描述的一些实施例的可以被用来基于其名称对基本数据单元进行组织的不同的数据组织系统。

[0113] 图3A示出了前缀树(trie)数据结构，其中基于来自每一个基本数据单元的相继字节的值把基本数据单元组织在逐渐更小的群组中。在图3A所示的实例中，每一个基本数据单元具有从该基本数据单元的整个内容构造的独特名称，该名称简单地由基本数据单元的所有字节构成，这些字节按照其存在于基本数据单元中的相同顺序出现在名称中。前缀树的根节点表示所有基本数据单元。前缀树的其他节点表示基本数据单元的子集或群组。从

前缀树的根节点或第1级(在图3A中被标记成根302)开始,基于其名称的最高有效字节(在图3A中被标记成N1)的值把基本数据单元分组到子树中。在其名称的最高有效字节中具有相同值的所有基本数据单元将被一起分组到共同的子树中,并且通过该值标示的链接将从根节点存在到表示该子树的节点。例如在图3A中,节点303表示分别在其对应名称的最高有效字节N1中具有相同的值2的基本数据单元的子树或群组。在图3A中,该群组包括基本数据单元305、306和307。

[0114] 在前缀树的第二级,每一个基本数据单元的名称的第二最高有效字节被用来把每一个基本数据单元群组进一步划分成更小的子组。例如在图3A中,使用第二最高有效字节N2把由节点303表示的基本数据单元群组进一步细分成各个子组。节点304表示在其对应名称的最高有效字节N1中具有值2并且在第二最高有效字节N2中具有值1的基本数据单元的子组。该子组包括基本数据单元305和306。

[0115] 所述细分处理在前缀树的每一级继续,从而创建从亲代节点到每一个子代节点的链接,其中子代节点表示通过亲代节点表示的基本数据单元的一个子集。这一处理继续到在前缀树的叶子处只有单独的基本数据单元为止。叶节点表示叶子的群组。在图3A中,节点304是叶节点。由节点304表示的基本数据单元的群组包括基本数据单元305和306。在图3A中,通过使用其名称的第三最高有效字节,该群组被进一步细分成单独的基本数据单元305和306。值N3＝3导致基本数据单元305,值N3＝5则导致基本数据单元306。在该例中,其完整名称当中的仅仅3个有效字节就足以完全标识基本数据单元305和306。同样地,来自名称的仅仅两个有效字节就足以标识基本数据单元307。

[0116] 该例示出了在基本数据单元的给定混合中如何只有名称的一个字节子集用来在树中标识基本数据单元,并且不需要整个名称以到达独有的基本数据单元。此外,基本数据单元或者基本数据单元的群组可能分别需要不同数目的有效字节以便能够对其进行唯一标识。因此,从根节点到基本数据单元的前缀树深度对于不同的基本数据单元可以是不同的。此外,在前缀树中,每一个节点可能具有下降到下方的子树的不同数目的链接。

[0117] 在这样的前缀树中,每一个节点具有由规定如何到达该节点的字节序列构成的名称。举例来说,对应于节点304的名称是“21”。此外,在树中的当前单元分布中唯一地标识单元的来自单元名称的字节子集是从根节点去到该基本数据单元的“路径”。例如在图3A中,具有值213的路径301标识基本数据单元305。

[0118] 这里所描述的前缀树结构可能会产生很深的树(也就是具有许多等级的树),这是因为树中的单元名称的每一个区分字节都为前缀树增加了一级深度。

[0119] 应当提到的是,图3A-3E中的树数据结构是从左向右绘制的。因此当我们从图的左侧向图的右侧移动时,我们从树的更高等级移动到树的较低等级。在给定节点的下方(也就是说朝向图3A-3E中的给定节点的右侧),对于通过来自名称的区分字节的特定值所选择的任何子代,驻留在该子代下方的各个子树中的所有单元在单元名称中的该相应字节中都将具有相同的值。

[0120] 现在我们将描述一种在给定输入候选单元的情况下对前缀树进行内容关联查找的方法。此方法涉及使用候选单元的名称对前缀树结构进行导航,随后进行后续的分析和筛选以便决定作为总体内容关联查找的结构将返回什么。换句话说,前缀树导航处理返回第一结果,随后在该结果上实施分析和筛选以便确定总体内容关联查找的结果。

[0121] 为了开始前缀树导航处理，来自候选单元的名称的最高有效字节的值将被用来选择从根节点到一个后续节点的链接(由该值表示)，所述后续节点表示在其名称的最高有效字节中具有该相同值的基本数据单元的子树。从该节点继续，检查来自候选单元的名称的第二字节并且选择由该值标示的链接，从而前进到前缀树中的更深(或更低)一级，并且选择现在与候选单元共享来自其名称的至少两个有效字节的基本数据单元的更小子组。这一处理继续到到达单一基本数据单元为止，或者继续到没有链接与来自候选单元的名称的相应字节的值相匹配为止。在这些条件当中的任一条件下，树导航处理终止。如果到达单一基本数据单元，则可以将其返回以作为前缀树导航处理的结果。如果没有，一种替换方案是报告“错失(miss)”。另一种替换方案是返回以导航终止的节点为根部的子树中的多个基本数据单元。

[0122] 一旦前缀树导航处理终止，可以使用其他标准或要求对前缀树导航处理的结果进行分析和筛选，以便确定作为内容关联查找的结果应当返回什么。举例来说，当由前缀树导航处理返回单一基本数据单元或多个基本数据单元时，在有资格作为内容关联查找的结果被返回之前，可以附加地要求其与候选单元的名称共享特定最小数目的字节(否则内容关联查找返回错失)。筛选要求的另一个实例可以是如果前缀树导航处理在没有到达单一基本数据单元的情况下终止从而返回多个基本数据单元(以前缀树导航终止的节点为根部)以作为前缀树导航处理的结果，则只有在所述多个基本数据单元的数目少于所规定的特定限制的情况下，这些单元才将有资格作为总体内容关联查找的结果被返回(否则内容关联查找返回错失)。可以采用多项要求的组合来确定内容关联查找的结果。通过这种方式，查找处理将报告“错失”或返回单一基本数据单元，或者如果不是单一基本数据单元，则是可能作为用于导出候选单元的良好起点的基本数据单元的集合。

[0123] 下面描述的图3B-3E涉及图3A中示出的树数据结构的变型和修改。虽然这些变型提供了优于图3A中示出的前缀树数据结构的改进和优点，但是用于对所述数据结构进行导航的处理类似于前面参照图3A所描述的处理。也就是说，在对于图3B-3E中示出的树数据结构的树导航终止之后，实施后续分析和筛选以便确定总体内容关联查找的结果，所述总体处理返回错失、单一基本数据单元或者可能作为用于导出候选单元的良好起点的基本数据单元的集合。

[0124] 图3B示出了可以被用来基于其名称对基本数据单元进行组织的另一种数据组织系统。在图3B所示出的实例中，每一个基本数据单元具有从该基本数据单元的整个内容构造的独特名称，该名称简单地由基本数据单元的所有字节构成，这些字节按照其存在于基本数据单元中的相同顺序出现在名称中。图3B示出了一种更加紧凑的结构，其中单一链接采用来自下方子树中的基本数据单元的名称的多个字节(而不是使用在图3A的前缀树中的单一字节)，以便产生细分或下一级的分组。从亲代节点到子代节点的链接现在由多个字节标示。此外，来自任何给定亲代节点的每一个链接可以采用不同数目的字节以便区分和标识与该链接相关联的子树。例如在图3B中，从根节点到节点308的链接通过使用来自名称的4个字节($N_1N_2N_3N_4$＝9845)来区分，从根节点到节点309的链接则通过使用来自名称的3个字节($N_1N_2N_3$＝347)来区分。

[0125] 应当提到的是，在(使用来自给定候选单元的)树导航期间，当到达树中的任何亲代节点时，树导航处理需要确保检查来自候选单元的名称的足够多的字节以便明确地决定

将要选择哪一个链接。为了选择给定的链接，来自候选项的名称的字节必须与标示去到该特定链接的过渡的所有字节相匹配。同样地，在这样的树中，树的每一个节点具有由规定如何到达该节点的字节序列构成的名称。举例来说，节点309的名称可以是“347”，这是因为其表示一个基本数据单元的群组(例如单元311和312)，其中所述基本数据单元的名称的3个开头字节是“347”。在使用其名称的开头3个字节是347的候选单元对树进行查找时，该数据模式导致树导航处理到达如图3B中示出的节点309。同样地，在树中的当前单元混合当中唯一地标识单元的来自该单元的字节子集是从根节点去到该基本数据单元的“路径”。例如在图3B中，字节序列3475导致基本数据单元312，并且在该例中示出的基本数据单元混合当中唯一地标识基本数据单元312。

[0126] 对于多样并且稀疏的数据，可以证明图3B中的树结构比图3A的前缀树结构更加灵活和紧凑。

[0127] 图3C示出了可以被用来基于其名称对基本数据单元进行组织的另一种数据组织系统。在图3C所示出的实例中，每一个基本数据单元具有从该基本数据单元的整个内容构造的独特名称，该名称简单地由基本数据单元的所有字节构成，这些字节按照其存在于基本数据单元中的相同顺序出现在名称中。图3C示出了(针对图3B中所描述的组织的)另一种变型，其使得树更加紧凑，并且(在必要和/或有用时)通过使用规则表达法来规定导致各种链接的来自基本数据单元名称的值而对子树中的单元进行分组。通过使用规则表达法允许对相同子树下的在相应字节上共享相同表达法的单元进行高效的分组；随后可以是对于子树内的不同基本数据单元的更加局部的歧义消除。此外，通过使用规则表达法允许以更加紧凑的方式来描述把单元映射到任何下方的子树所需要的字节的值。这样进一步减少了规定树所需要的字节数目。举例来说，规则表达法318规定28个接连的“F”的模式；如果在树导航期间遵循该链接，我们可以到达包括模式320的单元314，所述模式320具有依照规则表达法318的28个接连的“F”。同样地，到达单元316的路径具有使用规定具有16个接连的“0”的模式的规则表达法的链接或分支。对于这样的树，树导航处理需要检测并执行这样的规则表达法以便确定将要选择哪一个链接。

[0128] 图3D示出了可以被用来基于其名称对基本数据单元进行组织的另一种数据组织系统。在图3D所示出的实例中，每一个基本数据单元具有从该基本数据单元的整个内容构造的独特名称。对每一个单元应用指纹处理方法，以便识别包含评估到所选指纹的内容的字段的位置。在单元中找到的第一指纹的位置处的字段被作为维度对待，来自该字段的特定数目的字节(例如x个字节，其中x显著小于单元中的字节数目)被提取出来并且被用作单元名称的开头字节，名称的其余字节由基本数据单元的其余字节构成，并且按照其存在于基本数据单元中的相同循环顺序出现。该名称被用来对树中的基本数据单元进行组织。在该例中，当在单元中没有检测到指纹时，通过简单地按照其存在于单元中的顺序使用单元的所有字节而制订名称。一个单独的子树(通过表明没有找到指纹的指示标示)基于其名称保持并组织所有这样的单元。

[0129] 如图3D中所示，可以对单元338(其包含t个字节的数据，也就是$B_1B_2B_3\cdots B_t$)应用指纹处理技术，以便在标识将被选择成“维度1”的字节B_{i+1}处获得指纹位置“指纹1”。接下来，可以提取来自由“指纹1”标识的位置的x个字节以形成“维度1”，并且这x个字节可以被用作图3D中的每一个单元的名称的开头字节$N_1N_2\cdots N_x$。随后串联来自单元338的t-x个字节(从

B_{i+x+1}开始并且随后绕回到$B_1B_2B_3\cdots B_i$)并且将其用作名称的其余字节$N_{x+1}N_{x+2}\cdots N_t$。当在单元中没有找到指纹时，名称$N_1N_2\cdots N_t$简单地是来自单元338的$B_1B_2B_3\cdots B_t$。使用其名称在树中对基本数据单元进行分拣和组织。举例来说，在使用路径13654…06经历树的两个等级之后识别出并且到达基本数据单元(PDE)330，其中字节13654…0是作为来自维度1的字节的$N_1N_2\cdots N_x$。从根部沿着链接334(通过表明没有找到指纹的指示标示)到达的节点335处的单独子树保持并组织其内容未评估到所选指纹的所有基本数据单元。因此，在这种组织中，一些链接(例如链接336)可以使用由按照与单元中相同的顺序出现的单元的字节构成的名称对单元进行组织，其他链接(例如链接340)则可以使用利用指纹制订的名称对单元进行组织。

[0130] 在接收到候选单元时，所述处理应用前面描述的相同技术来确定候选单元的名称，并且使用该名称对树进行导航以进行内容关联查找。因此，对基本数据单元(在其被安装到树中时)和候选单元(在从解析器和因式分解器接收到候选单元时)应用相同并且一直的处理，以便创建其名称。树导航处理使用候选单元的名称对树进行导航。在该实施例中，如果在候选单元中没有找到指纹，则树导航处理沿着组织并包含其内容未评估到指纹的基本数据单元的子树向下导航。

[0131] 图3E示出了可以被用来基于其名称对基本数据单元进行组织的另一种数据组织系统。在图3E所示出的实例中，每一个基本数据单元具有从该基本数据单元的整个内容构造的独特名称。对每一个单元应用指纹处理方法，以便识别包含评估到两个指纹当中的任一个的内容的字段的位置。单元中的第一指纹(图3E中的指纹1)的第一次出现位置处的字段被作为第一维度(维度1)对待，第二指纹(图3E中的指纹2)的第一次出现位置处的字段被作为第二维度(维度2)对待。使用指纹处理寻找单元上的两个不同指纹导致四种可能的情形：(1)在单元中找到全部两个指纹，(2)找到指纹1但是没有找到指纹2，(3)找到指纹2但是没有找到指纹1，以及(4)没有找到指纹。基本数据单元可以被分组到对应于每一种情形的4个子树中。在图3E中，“FP1”标示指纹1的存在，“FP2”标示指纹2的存在，“～FP1”标示指纹1的缺失，并且“～FP2”标示指纹2的缺失。

[0132] 对于所述4种情形当中的每一种，如下创建单元的名称：(1)当找到全部两个指纹时，可以提取来自由“指纹1”标识的位置的x个字节以形成“维度1”，并且可以提取来自由“指纹2”标识的位置的y个字节以形成“维度2”，这x+y个字节可以被用作图3E中的每一个这样的单元的名称的开头字节$N_1N_2\cdots N_{x+y}$。随后通过循环方式提取来自单元348的其余的t-(x+y)个字节(在来自第一维度的字节之后开始)，并且将其串联并且用作名称的其余字节$N_{x+y+1}N_{x+y+2}\cdots N_t$。(2)当找到指纹1但是没有找到指纹2时，可以提取来自由“指纹1”标识的位置的x个字节以形成在前维度，并且这x个字节可以被用作每一个这样的单元的名称的开头字节$N_1N_2\cdots N_x$。随后串联来自单元348的其余的t-x个字节(从B_{i+x+1}开始并且随后绕回到$B_1B_2B_3\cdots B_i$)，并且将其用作名称的其余字节$N_{x+1}N_{x+2}\cdots N_t$。(3)当找到指纹2但是没有找到指纹1时，可以提取来自由“指纹2”标识的位置的y个字节以形成在前维度，并且这y个字节可以被用作每一个这样的单元的名称的开头字节$N_1N_2\cdots N_y$。随后串联来自单元348的其余的t-y个字节(从B_{j+y+1}开始并且随后绕回到$B_1B_2B_3\cdots B_j$)，并且将其用作名称的其余字节$N_{y+1}N_{y+2}\cdots N_t$。(4)当在单元中没有找到指纹时，名称$N_1N_2\cdots N_t$简单地是来自单元348的$B_1B_2B_3\cdots B_t$。因此对于这4种情形当中的每一种存在单独的子树。对于所述四种情形可以如下概括用以提取针对单元348的名称($N_1N_2N_3\cdots N_t$)的处理：

[0133] (1)指纹1和指纹2都被找到：

[0134] N_1-N_x←B_{i+1}-B_{i+x}＝来自维度1的x个字节

[0135] N_{x+1}-N_{x+y}←B_{j+1}-B_{j+y}＝来自维度2的y个字节

[0136] N_{x+y+1}…N_t＝其余的字节(来自大小为t个字节的候选单元)＝$B_{i+x+1}B_{i+x+2}B_{i+x+3}...B_jB_{j+y+1}B_{j+y+2}B_{j+y+3}...B_tB_1B_2B_3...B_i$

[0137] (2)找到指纹1但是没有找到指纹2：

[0138] N_1-N_x←B_{i+1}-B_{i+x}＝来自维度1的x个字节

[0139] N_{x+1}…N_t＝其余的字节(来自大小为t个字节的候选单元)＝$B_{i+x+1}B_{i+x+2}B_{i+x+3}...B_tB_1B_2B_3...B_i$

[0140] (3)找到指纹2但是没有找到指纹1：

[0141] N_1-N_y←B_{j+1}-B_{j+y}＝来自维度2的y个字节

[0142] N_{y+1}…N_t＝其余的字节(来自大小为t个字节的候选单元)＝$B_{j+y+1}B_{j+y+2}B_{j+y+3}...B_tB_1B_2B_3...B_j$

[0143] (4)没有找到指纹：

[0144] N_1-N_x←B_1-B_t

[0145] 在接收到候选单元时，所述处理应用前面描述的相同技术以确定候选单元的名称。在该实施例中，对候选单元应用前面描述的4种名称构造方法(取决于是否找到指纹1和指纹2)，正如在被输入到滤筛中时对其所应用的那样。因此，对基本数据单元(在其被安装到树中时)和候选单元(在从解析器和因式分解器接收到候选单元时)应用相同并且一致的处理，以便创建其名称。树导航处理使用候选单元的名称对树进行导航以便进行内容关联查找。

[0146] 如果内容关联查找是成功的，则将产生在特定维度的位置处于候选单元具有相同模式的基本数据单元。举例来说，如果在候选单元中找到全部两个指纹，则树导航处理将从根节点开始沿着树的链接354向下。如果候选单元具有作为“维度1”的模式“99…3”和作为“维度2”的模式“7…5”，则树导航处理将到达节点334。这样就到达包含可能是导出目标的两个基本数据单元(PDE 352和PDE 353)的子树。实施附加的分析和筛选(通过首先检查元数据，并且需要的话通过随后获取和检查实际的基本数据单元)以便确定哪一个基本数据单元最适合于导出。因此，这里所描述的实施例识别出可以被使用在滤筛中的多种树结构。可以采用这样的结构或者其变型的组合以便对基本数据单元进行组织。一些实施例通过树形式来组织基本数据单元，其中单元的整个内容被用作该单元的名称。但是各个字节出现在单元名称中的序列不一定是所述字节出现在单元中的序列。单元的特定字段被作为维度提取出来，并且被用来形成名称的开头字节，单元的其余字节构成名称的其余部分。使用这些名称在滤筛中通过树形式对单元进行排序。名称的开头数位被用来区分树的更高分支(或链接)，其余数位被用来逐渐地区分树的所有分支(或链接)。树的每一个节点可以具有从该节点发出的不同数目的链接。此外，来自一个节点的每一个链接可以通过不同数目的字节被区分和标示，并且通过使用规则表达法以及用以表达其规范的其他有力方式可以实现对于这些字节的描述。所有这些特征导致紧凑的树结构。对各个单元基本数据单元的引用驻留在树的叶节点处。

[0147] 在一个实施例中，可以对构成基本数据单元的字节应用指纹处理方法。驻留在通

过指纹识别出的位置处的一定数目的字节可以被用来构成单元名称的一个分量。可以组合一个或多个分量以便提供一个维度。多个指纹可以被用来识别多个维度。这些维度被串联并且被用作单元名称的开头字节，单元的其余字节构成单元名称的其余部分。由于维度位于通过指纹识别出的位置处，因此提高了从来自每一个单元的一致的内容形成名称的可能性。在通过指纹定位的字段处具有相同内容值的单元将沿着树的相同枝干被分组在一起。通过这种方式，类似的单元将在树数据结构中被分组在一起。通过使用其名称的替换制订，可以把没有在其中找到指纹的单元一起分组在单独的子树中。

[0148] 在一个实施例中，可以对单元的内容应用指纹处理方法，以便确定单元内容内的(前面所描述的)骨架数据结构的各个分量(或签名)的位置。或者可以选择单元内容内部的特定的固定偏移量以定位分量。还可以采用其他方法来定位单元的骨架数据结构的分量，其中包括而不限于对单元进行解析以便检测所声明的特定结构并且定位该结构内的分量。骨架数据结构的各个分量可以被视为维度，从而使用这些维度的串联以及随后的每一个单元的其余内容来创建每一个单元的名称。名称被用来对树中的基本数据单元进行排序和组织。

[0149] 在另一个实施例中，对单元进行解析以便检测单元中的特定结构。该结构中的特定字段被识别成维度。多个这样的维度被串联并且被用作名称的开头字节，单元的其余字节构成单元名称的其余部分。由于维度位于通过对单元进行解析并且检测其结构而识别出的位置处，因此提高了从来自每一个单元的一致的内容形成名称的可能性。在通过所述解析定位的字段处具有相同内容值的单元将沿着树的相同枝干被分组在一起。通过这种方式，同样地，类似的单元将在树数据结构中被分组在一起。

[0150] 在一些实施例中，树数据结构中的每一个节点包括自描述规范。树节点具有一个或多个子代。每一个子代条目包含关于去到该子代的链接上的区分字节的信息以及对该子代节点的引用。子代节点可以是树节点或叶节点。图3F给出了根据这里所描述的一些实施例的自描述树节点数据结构。图3F中示出的树节点数据结构规定(A)关于从根节点到该树节点的路径的信息，包括所有以下分量或者其中的一个子集：用以到达该树节点的来自名称的实际字节序列，从根节点到达该节点所消耗的名称的字节数目，关于所消耗的该字节数目是否大于某一预先规定的阈值的指示，以及描述去到该节点的路径并且对于树的内容关联搜索以及对于涉及树的构造的决定是有用的其他元数据，(B)该节点所具有的子代的数目，以及(C)对于每一个子代(其中每一个子代对应于树的一个分支)规定(1)子代ID，(2)为了沿着树的该链接向下过渡所需要的来自名称的后继字节的区分字节的数目，(3)对于沿着该链接向下的来自名称的字节的实际值的规定，以及(4)对该子代节点的引用。

[0151] 图3G给出了根据这里所描述的一些实施例的自描述叶节点数据结构。叶节点具有一个或多个子代。每一个子代是去到一个基本数据单元的链接。每一个子代条目包含关于去到该基本数据单元的链接上的区分字节的信息，对该基本数据单元的引用，重复和导出项的计数，以及关于该基本数据单元的其他元数据。图3G中示出的叶节点数据结构规定(A)关于从根节点到该叶节点的路径的信息，包括所有以下组成部分或者其中的一个子集：用以到达该叶节点的来自名称的实际字节序列，从根节点到达该节点所消耗的名称的字节数目，关于所消耗的该字节数目是否大于某一预先规定的阈值的指示，以及描述去到该节点的路径并且对于树的内容关联搜索以及对于涉及树的构造的决定是有用的其他元数据，

(B)该节点所具有的子代的数目,以及(C)对于每一个子代(其中每一个子代对应于该叶节点下方的一个基本数据单元)规定(1)子代ID,(2)为了沿着树的该链接向下过渡到一个基本数据单元所需要的来自名称的后继字节的区分字节的数目,(3)对于沿着该枝干向下的来自名称的字节的实际值的规定,(4)对在树的该路径上终止树的基本数据单元的引用,(5)关于多少重复和导出项指向该基本数据单元的计数(这被用来确定在删除存储系统中的数据时是否可以从滤筛中删除条目),以及(6)包括基本数据单元的大小等等的对应于基本数据单元的其他元数据。

[0152] 为了提高新鲜的基本数据单元被安装到树中的效率,一些实施例把一个附加的字段合并到被保持在树的叶节点处的对应于每一个基本数据单元的叶节点数据结构中。应当提到的是,当必须把新鲜的单元插入到树中时,可能需要所讨论的子树中的每一个基本数据单元的名称或内容的附加字节以便决定将把所述新鲜单元插入到子树中的何处,或者是否触发子树的进一步分割。对于这些附加字节的需求可能需要获取其中几个所讨论的基本数据单元,以便对于这些单元当中的每一个提取出关于所述新鲜单元的相关区分字节。为了减少并且优化(并且在某些情况下完全消除)对于这一任务所需的IO的次数,叶节点中的数据结构包括来自该叶节点下方的每一个基本数据单元的名称的特定数目的附加字节。这些附加字节被称作导航前瞻字节,并且帮助关于新鲜的传入单元对基本数据单元进行分拣。对应于给定的基本数据单元的导航前瞻字节在该基本数据单元被安装到滤筛中时被安装到叶节点结构中。可以使用多种标准静态地或者动态地选择将为此目的保留的字节数目,其中包括所涉及的子树的深度以及该子树中的基本数据单元的密度。例如对于正被安装在树的较浅等级的基本数据单元,解决方案可以添加比驻留在非常深的树中的基本数据单元更长的导航前瞻字节。此外,当新鲜的单元正被安装到滤筛中时,并且如果在现有的目标子树中已经有许多导航前瞻字节(从而提高了即将发生再分割的可能性),则在所述新鲜的基本数据单元被安装到子树中时可以为之保留附加的导航前瞻字节。

[0153] 图3H给出了包括导航前瞻字段的对应于叶节点的叶节点数据结构。该数据结构规定(A)关于从根节点到该叶节点的路径的信息,包括所有以下组成部分或者其中的一个子集:用以到达该叶节点的来自名称的实际字节序列,从根节点到达该节点所消耗的名称的字节数目,关于所消耗的该字节数目是否大于某一预先规定的阈值的指示,以及描述去到该节点的路径并且对于树的内容关联搜索以及对于涉及树的构造的决定是有用的其他元数据,(B)该节点所具有的子代的数目,以及(C)对于每一个子代(其中每一个子代对应于该叶节点下方的一个基本数据单元)规定(1)子代ID,(2)为了沿着树的该链接向下过渡到一个基本数据单元所需要的来自名称的后继字节的区分字节的数目,(3)对于沿着该枝干向下的字节的实际值的规定,(4)对在树的该路径上终止树的基本数据单元的引用,(5)规定为所述基本数据单元保留多少导航前瞻字节的导航前瞻字段以及这些字节的实际值,(6)关于多少重复和导出项指向该基本数据单元的计数(这被用来确定在删除存储系统中的数据时是否可以从滤筛中删除条目),以及(7)包括基本数据单元的大小等等的对应于基本数据单元的其他元数据。

[0154] 在一些实施例中,树的各个分支被用来把各个数据单元映射到各个群组或范围中,其中所述群组或范围是通过对沿着导向作为范围定界符的子代子树的链接的区分字节进行解释而形成的。该子代子树中的所有单元将使得单元中的相应字节的值小于或等于对

于去到所述特定子代子树的链接所规定的区分字节的值。因此，每一个子树现在将表示其值落在特定范围内的一组单元。在给定的子树内，该树的每一个后续等级将逐渐地把单元集合划分成更小的范围。该实施例为图3F中示出的自描述树节点结构的组成部分提供了不同的解释。图3F中的N个子代通过其在树节点数据结构中的区分字节的值被排序，并且表示非重叠范围的有序序列。对于N个节点存在N+1个范围，最低的或第1个范围由小于或等于最小条目的值构成，并且第N+1个范围由大于第N个条目的值构成。第N+1个范围将被作为超出范围而对待，因此N个链接导向下方的N个子树或范围。

[0155] 例如在图3F中，子代1定义最低范围并且使用6个字节(值为abef12d6743a)来区分其范围——对应于子代1的范围是从00000000到abef12d6743a。如果候选单元的相应的6个字节落在该范围内(包括端值)，则将选择对应于该子代的链接。如果候选单元的相应的6个开头字节大于范围定界符abef12d6743a，则子代1将不会被选择。为了检查候选单元是否落在对应于子代2的范围内必须满足两个条件——首先候选必须处于紧接着的前一个子代的范围之外(在该例中是子代1)，其次其名称中的相应字节必须小于或等于对应于子代2的范围定界符。在该例中，对应于子代2的范围定界符由值为dcfa的2个字节描述。因此，对应于候选单元的2个相应字节必须小于或等于dcfa。使用这种方法，可以对候选单元和树节点中的所有子代进行检查，以便检查候选单元落在所述N+1个范围当中的哪一个范围内。对于图3F中示出的实例，如果候选单元的名称的4个相应字节大于对应于子代N的链接的区分字节的值f3231929，则将检测到错失状况。

[0156] 可以对树导航处理进行修改以便容纳这一新的范围节点。在到达范围节点时，为了选择从该节点发出的给定链接，来自候选的名称的字节必须落在对于该特定链接所定义的范围内。如果来自候选的名称的字节的值大于所有链接中的相应字节的值，则候选单元落在下方子树所跨越的所有范围之外——在这种情况下(其被称作“超出范围状况”)检测到错失状况，并且树导航处理终止。如果候选单元的名称的开头字节落在由沿着导向子代子树的链接的相应的区分字节所确定的范围之内，树导航继续到下方的该子树。除非由于“超出范围状况”而终止，否则树导航可以逐渐地沿着树继续下到更深处，直到其到达叶节点数据结构为止。

[0157] 这种范围节点可以结合在图3A-3E中描述的前缀树节点被采用在树结构中。在一些实施例中，树结构的特定数目的等级的上方节点可以是前缀树节点，其中树遍历是基于候选单元的名称的开头字节与沿着树的链接的相应字节之间的精确匹配。后续节点可以是具有通过候选的名称的相应字节落在其中的范围所决定的树遍历的范围节点。当树导航处理终止时，正如在本文献中早前所描述的那样，多种标准可以被用来决定作为总体内容关联查找侧结果将返回什么。

[0158] 前面对用于表示和使用树节点和叶节点的方法和装置的描述仅仅是出于说明和描述的目的而给出的。其并不意图作出穷举或者把本发明限制到所公开的形式。因此，本领域技术人员将认识到许多修改和变型。

[0159] 在给出候选单元以作为输入时，可以对前面描述的树节点和叶节点结构进行遍历，并且可以基于候选单元的内容对树实施内容关联查找。将从候选单元的字节构造候选单元的名称，正如基本数据单元的名称是当基本数据单元被安装在滤筛中时从其内容构造的。在给定输入候选单元的情况下，对树进行内容关联查找的方法涉及使用候选单元的名

称对树结构进行导航，随后进行分析和筛选以便决定作为总体内容关联查找的结果将返回什么。换句话说，树导航处理返回第一输出结果，随后在该结果上实施分析和筛选，以便确定总体内容关联查找的结果。

[0160] 如果存在具有与候选相同的名称开头字节(或者落在相同范围内的字节)的任何基本数据单元，树将通过由链接标示的单元子树的形式来标识基本数据单元的该子集。一般来说，每一个树节点或叶节点可以存储允许树导航处理决定将选择哪一个外出链接(如果存在的话)以便导航到树中的下一个更低等级的信息，这是基于输入单元的名称的相应字节以及在沿着所选链接对树进行导航时所到达的节点的身份。如果每一个节点都包含该信息，则树导航处理可以通过递归方式向下导航树中的每一个等级，直到没有找到匹配(此时树导航处理可以返回存在于以当前节点为根部的子树中的一个基本数据单元集合)或者到达一个基本数据单元(此时树导航处理可以返回该基本数据单元以及任何相关联的元数据)为止。

[0161] 一旦树导航处理终止，可以使用其他标准和要求对树导航处理的结果进行分析和筛选，以便确定作为总体内容关联查找的结果应当返回什么。首先，可以挑选在其名称中具有最多数目的与候选相同的开头字节的基本数据单元。其次，当由树导航处理返回单一基本数据单元或多个基本数据单元时，在有资格作为内容关联查找的结果被返回之前，可以附加地要求其与候选单元的名称共享特定的最少数目的字节(否则内容关联查找返回错失)。筛选要求的另一个实例可以是，如果树导航处理在没有到达单一基本数据单元的情况下终止并且从而作为树导航处理的结果返回多个基本数据单元(以树导航终止的节点为根部)，则只有在这些单元的数目小于所规定的特定限制(比如4-16个单元)的情况下，所述多个基本数据单元才将有资格作为总体内容关联查找的结果被返回(否则内容关联查找返回错失)。可以采用多项要求的组合来确定内容关联查找的结果。如果仍有多个候选，则可以检查导航前瞻字节并且还有相关联的元数据，以便决定哪些基本数据单元是最适当的。如果仍然无法把选择收窄到单一基本数据单元，则可以把多个基本数据单元提供到导出功能。通过这种方式，查找处理将报告“错失”或返回单一基本数据单元，或者如果不是单一基本数据单元，则是可能作为用于导出候选单元的良好起点的基本数据单元的集合。

[0162] 树需要被设计成用于高效的内容关联存取。具有良好平衡的树对于大部分数据将提供可比的存取深度。预期树的几个上方等级将常常驻留在处理器高速缓存中，接下来的几个等级驻留在快速存储器中，并且后续等级驻留在闪存存储装置中。对于非常大的数据集，可能有一个或多个等级需要驻留在闪存存储装置或者甚至是盘中。

[0163] 图4示出了根据这里所描述的一些实施例的如何可以把256TB的基本数据组织成树形式的一个实例，并且呈现出如何可以把树布置在存储器和存储装置中。假设每个节点平均展开64(2^6)个子代，则可以通过到达(平均)驻留在树的第6级(也就是说在5次链接遍历或跳跃之后)的叶节点数据结构(例如在图3H中描述)而存取对某一基本数据单元的引用。因此，在5次跳跃之后，树的第6级处的此类结构将与另外的2^{30}个此类节点并列驻留，每一个节点平均具有64个子代(这些子代是对基本数据单元的引用)，从而容纳近似640亿个基本数据单元。在4KB的单元大小下，这样就容纳256TB个基本数据单元。

[0164] 树可以被布置成使得可以如下遍历树的6个等级：3个等级驻留在芯片上高速缓存中(其中包含规定对应于去到近似256K个节点的链接的过渡的近似四千个“上方等级”树节

点数据结构)，存储器中的2个等级(其中包含规定对应于去到近似10亿个叶节点的链接的过渡的1600万个“中间等级”树节点数据结构)，以及闪存存储装置中的第6级(容纳10亿个叶节点数据结构)。驻留在闪存存储装置中的树的该第6级的10亿个叶节点数据结构提供对640亿个基本数据单元的引用(每个叶节点平均64个单元)。

[0165] 在图4所示的实例中，在第4和第5级，每一个节点对于每个单元专用平均16个字节(对应于子代ID的1个字节，例如对PDE的6字节引用，加上用于字节计数的一个字节，加上用以规定实际过渡字节以及一些元数据的平均8个字节)。在第6级，每一个叶节点对于每个单元专用平均48个字节(对应于子代ID的1个字节，用于字节计数的1个字节，用以规定实际过渡字节的8个字节，对基本数据单元的6字节引用，用于来自该基本数据单元的导出项计数的1个字节，16个字节的导航前瞻，对应于基本数据单元的大小的2个字节，以及13个字节的其他元数据)，因此对于树所需的闪存存储装置中的总容量(包括对基本数据单元的引用并且包括任何元数据)是大约3个太字节。对于树的上方节点所需的总容量是这一大小的一小部分(这是因为节点更少，规定对子代节点的更严格的引用所需的字节更少，并且每个节点所需的元数据更少)。在该例中，上方树节点对于每个单元专用平均8个字节(对应于子代ID的1个字节，用于字节计数的1个字节，加上用以规定实际过渡字节的平均3-4个字节，以及对子代节点的2-3字节引用)。在该例中，总体上使用3TB(或者256TB的1.17％)的附加装置把具有256TB的基本数据的合成数据集分拣到10亿个群组中。

[0166] 在图4所示出的实例中，256TB的基本数据包含640亿个4KB基本数据单元，为了完全区分所述640亿个基本数据单元需要少于5字节(或36比特)的地址。从内容关联角度来看，如果数据的混合使得在前3个等级当中的每一个等级处消耗平均4字节的渐进式名称，并且在接下来的3个等级当中的每一个等级处消耗8个字节，从而总共(平均)36字节(288比特)的名称将区分所有640亿个基本数据单元。这36个字节将少于构成每一个单元的4KB的1％。如果可以通过其字节的1％(或者甚至5-10％)来标识4KB的基本数据单元，则(构成大部分字节的)其余字节可以容许微扰，并且具有此类微扰的候选仍然可以到达该基本数据单元并且可以考虑从该基本数据单元导出。

[0167] 应当提到的是，(为了区分下方的各个子树)在任何给定的链接上所需的字节数目将由构成数据集的单元混合中的实际数据决定。同样地，从给定节点发出的链接数目也将随着数据而改变。自描述树节点和叶节点数据结构将生命对于每一个链接所需的字节的实际数目和值，以及从任何节点发出的链接的数目。

[0168] 可以施加进一步的控制以便限制在树的各个等级处专用的高速缓存、存储器和存储装置的数量，以便在增量存储的已分配预算内把输入分拣到尽可能多的已区分群组中。为了应对其中存在需要非常深的子树来完全区分单元的数据密度和口袋(pocket)的情况，可以通过以下步骤高效地应对这样的密度：把相关单元的更大集合分组到树的特定深度(例如第6级)处的平坦群组中，并且在其上实施流线式搜索和导出(这是通过首先检查导航前瞻和元数据以确定最佳基本数据单元，或者(作为回退)对于其余的数据仅仅寻找重复而不是由所述方法提供的完全导出)。这样将避免产生非常深的树。另一种替换方案是允许(具有许多等级的)很深的树，只要这些等级能够容纳在可用的存储器中。当更深的等级溢出到闪存或盘时，可以采取一些步骤以使得树从该等级往后变平坦，从而最小化原本将由于针对存储在闪存或盘中的更深等级的树节点的多次相继存取而招致的等待时间。

[0169] 预期来自单元名称的全部字节当中的相对较小的一部分将常常足以标识每一个基本数据单元。使用这里所描述的实施例在多种真实世界数据集上实施的研究证实，基本数据单元的一个较小的字节子集可以用来对大部分单元进行排序从而允许所述解决方案。因此，这样的解决方案在对于其操作所需的存储的数量方面是高效的。

[0170] 在对于来自图4的实例所需的存取方面，对于每一个传入的4KB输入组块(或候选单元)，所述方法将需要实施一次以下存取以便对树结构进行查询并且到达叶节点：三个高速缓存引用、两个存储器引用(或者可能有多个存储器引用)加上来自闪存存储装置的单一IO，以便对叶节点数据结构进行存取。来自存储装置的该单一IO将获取一个4KB页面，其将保有对应于一组近似64个单元的叶节点数据结构的信息，从而将包括专用于所讨论的基本数据单元的48个字节。这48个字节将包括关于所讨论的基本数据单元的元数据。这将结束树查找处理。随后所需要的IO的次数将取决于候选单元结果是重复、导出项还是将被安装在滤筛中的新鲜基本数据单元。

[0171] 作为某一基本数据单元的重复的候选单元将需要1次IO来获取该基本数据单元，以便验证重复。一旦验证了重复，将再需要一次IO以更新树中的元数据。因此，重复单元的摄取在树查找之后将需要两次IO，从而一共是3次IO。

[0172] 没有通过树查找并且既不是重复也不是导出项的候选单元需要另外的1次IO以把该单元作为新的基本数据单元存储在滤筛中，并且需要另一次IO以更新树中的元数据。因此，没有通过树查找的候选单元的摄取在树查找之后将需要2次IO，从而导致一共3次IO。但是对于其中树查找处理在不需要存储IO的情况下终止的候选单元，对于摄取这样的候选单元一共只需要2次IO。

[0173] 作为导出项(而非重复)的候选单元将首先需要1次IO以获取计算导出所需的基本数据单元。由于预期最经常的导出将是源自单一基本数据单元(而不是多个基本数据单元)，因此对于获取该基本数据单元将只需要单一IO。在成功完成导出之后，将再需要1次IO以把重建程序和导出细节存储在为该单元在存储装置中创建的条目中，并且需要另一次IO以更新树中的元数据(比如计数等等)以便反映出新的导出项。因此，变成导出项的候选单元的摄取在第一树查找之后需要3次附加的IO，从而一共是4次IO。

[0174] 总而言之，为了摄取候选单元并且对其应用Data Distillation™方法(同时在非常大的数据集上全局利用冗余)需要大致3到4次IO。与传统的数据去重复技术所需要的情况相比，对于每个候选单元通常只有另外的一次IO，其回报是能够以比所述单元本身更细的粒度在数据集上全局地利用冗余。

[0175] 给出250000次随机IO存取/秒(这意味着对于4KB页面的1GB/秒的随机存取带宽)的存储系统每秒可以摄取大约62500(250000除以平均大小分别为4KB的每个输入组块4次IO)个输入组块并且对其实施Data Distillation™方法。这在用尽存储系统的所有带宽的情况下允许250MB/秒的摄取速率。如果仅使用存储系统的一半带宽(从而使得另一半可用于对所存储的数据进行存取)，这样的Data Distillation™系统仍然可以给出125MB/秒的摄取速率。因此，在给定足够处理能力的情况下，Data Distillation™系统能够以经济的IO(在比所述单元本身更细的粒度上)在数据集上全局地利用冗余，并且在当代存储系统上以每秒数百兆字节的摄取速率给出数据简化。

[0176] 因此，正如测试结果所证实的那样，这里所描述的实施例实现了以经济的IO存取

并且利用对于装置所需的最小增量存储从大容量数据存储库搜索单元(从中可以导出输入单元并且只利用规定所述导出所需的最小存储)的复杂任务。如此构造的这一框架使得使用全部单元字节的更小百分比找到适合于导出的单元成为可行,从而留下大部分字节可用于微扰和导出。解释这一方案为何对于大部分数据能够有效工作的一项重要洞察在于,树提供了易于使用的细粒度结构,从而允许定位在滤筛中标识单元的区分和区别字节,并且尽管这些字节分别处于数据中的不同深度和位置,在树结构中可以高效地对其进行隔离和存储。

[0177] 图5A-5C示出了关于如何可以使用这里所描述的实施例组织数据的一个实际的实例。图5A示出了512字节的输入数据以及因式分解的结果(例如实施图2中的操作202的结果)。在该例中应用指纹处理以确定数据中的中断,从而使得接连的中断标识候选单元。使用粗体和常规字体示出了交替的候选单元。举例来说,第一候选单元是“b8ac83d9dc7caf18f2f2e3f783a0ec69774bb50bbe1d3ef1ef8a82436ec43283bc1c0f6a82e19c224b22f9b2”,下一个候选单元是“ac83d9619ae5571ad2bbcc15d3e493eef62054b05b2dbccce933483a6d3daab3cb19567dedbe33e952a966c49f3297191cf22aa31b98b9dcd0fb54a7f761415e”,后面以此类推。如图所示,图5A中的输入被因式分解成12个可变大小候选单元。每一个组块的开头字节被用来在滤筛中对单元进行排序和组织。图5B示出了如何可以使用其名称并且使用图3B中所描述的树结构按照树形式把图5A中示出的12个候选单元组织成滤筛中的基本数据单元。每一个单元具有从该单元的整个内容构造的独特名称。在该例中,由于应用了指纹处理以确定12个候选单元之间的中断,因此每一个候选单元的开头字节将已经被对准到锚指纹:因此,每一个名称的开头字节将已经是从锚定在该指纹处的内容的第一维度构造的。名称的开头字节组织各个单元。举例来说,如果单元名称中的第一字节等于“0x22”,则取得顶部链接以选择基本数据单元#1。应当提到的是,使用不同数目的字节对图5B中的各个链接进行区分,正如参照图3B中示出的树数据结构所解释的那样。

[0178] 图5C示出了如何可以使用参照图3D描述的树数据结构对图5A中示出的12个候选单元进行组织。进一步对每一个单元的内容应用指纹处理,以便识别单元内容内的第二指纹。把从第一指纹(其已经存在于每一个单元的边界处)的位置处提取出的内容字节与第二指纹串联,从而形成被用来对单元进行组织的名称的开头字节。换句话说,单元名称被如下构造:来自(分别通过锚指纹和第二指纹定位的)两个维度或字段的数据字节被串联形成名称的开头字节,随后是其余的字节。作为针对名称构造的这一选择的结果,(相比于图5B)不同的字节序列导向图5C中的各个基本数据单元。例如为了到达基本数据单元#4,树导航处理首先取得对应于作为第一维度(即第一指纹)处的字段的开头字节的“46093f9d”的链接,并且随后取得对应于作为位于第二维度(即第二指纹)处的字段的开头字节的“c4”的链接。

[0179] 图6A-6C分别示出了根据这里所描述的一些实施例如何可以把树数据结构用于参照图1A-1C描述的内容关联映射器121和122。

[0180] 一旦解决了找到(从中尝试导出候选单元的)适当的基本数据单元的困难问题,所述问题就被收窄到检查一个基本数据单元或者基本数据单元的较小子集并且从中最优地导出候选单元,其中只利用规定所述导出所需的最小存储。其他目标包括把对于存储系统的存取次数保持到最低程度,并且把导出时间和重建时间保持到可以接受。

[0181] 导出器必须把候选单元表达成在一个或多个基本数据单元上实施的变换的结果,

并且必须把这些变换规定成将被用来在取回数据时重新生成导出项的重建程序。每一项导出可能需要构造其自身所独有的程序。导出器的功能是识别这些变换，并且创建具有最小足迹的重建程序。可以采用多种变换，其中包括在一个或多个基本数据单元上或者在每一个单元的特定字段上实施的算术、代数或逻辑运算。此外还可以使用字节操纵变换，比如串联、插入、替换和删除一个或多个基本数据单元中的字节。

[0182] 图7A提供了根据这里所描述的一些实施例的可以在重建程序中规定的变换的一个实例。在该例中规程的变换的词汇表包括在单元中的规定长度的字段上实施的算术运算，以及在基本数据单元中的规定偏移量处插入、删除、附加和替换所声明的长度的字节。导出器可以采用多种技术和操作来检测候选单元与一个或多个基本数据单元之间的相似性和差异并且用来构造重建程序。导出器可以利用在底层硬件中可用的词汇表来实施其功能。所述工作的最终结果是在对于重建程序所规定的词汇表中规定变换，并且在这样做时使用最小数量的增量存储并且采取还允许快速数据取回的方式。

[0183] 导出器可以利用底层机器的处理能力并且在为之分配的处理预算内工作，以便在系统的成本-性能约束内提供尽可能最佳的性能。鉴于微处理器核心更容易获得，并且鉴于针对存储装置的IO存取较为昂贵，因此Data Distillation™解决方案被设计成利用当代微处理器的处理能力，以便高效地实施本地分析以及从少数几个基本数据单元导出候选单元的内容。预期Data Distillation™解决方案(在非常大的数据上)的性能将不受计算处理的速率限制(rate-limited)，而是受到典型存储系统的IO带宽的速率限制。举例来说，预期几个微处理器核心就将足以实施所需的计算和分析，从而在支持250000次IO/秒的典型的基于闪存的存储系统上支持每秒几百兆字节的摄取速率。应当提到的是，来自当代微处理器，比如Intel Xeon处理器E5-2687W(10核，3.1GHz，25MB高速缓存)的两个这样的微处理器核心是可以从处理器获得的全部计算能力的一部分(十分之二)。

[0184] 图7B示出了根据这里所描述的一些实施例的从基本数据单元导出候选单元的结果的实例。具体来说，数据模式“Elem”是存储在基本数据滤筛中的基本数据单元，并且数据模式“Cand”是将从基本数据单元导出的候选单元。已经突出显示出“Cand”与“Elem”之间的18个共同的字节。重建程序702规定如何可以从数据模式“Elem”导出数据模式“Cand”。如图7B中所示，重建程序702示出了如何从“Elem”导出“Cand”，这是通过使用1字节替换、6字节插入、3字节删除、7字节批量替换。用以规定导出项的成本是20字节+3字节引用＝23字节，从而是原始大小的65.71％。应当提到的是，所示出的重建程序702是所述程序的人类可读表示，并且可能不是所述程序被这里所描述的实施例实际存储的方式。同样地，在图7B中还示出了基于算术运算(比如乘法和加法)的其他重建程序。举例来说，如果“Elem”是bc1c0f6a790c82e19c224b22f900ac83d9619ae5571ad2bbec152054ffffff83并且“Cand”是bc1c0f6a790c82e19c224b22f91c4da1aa0369a0461ad2bbec152054ffffff83，则如图所示可以使用乘法(00ac83d9619ae557)*2a＝[00]1c4da1aa0369a046导出8字节差异。用以规定导出项的成本是4字节+3字节引用＝7字节，从而是原始大小的20.00％。或者如果“Elem”是bc1c0f6a790c82e19c224b22f9b2ac83ffffffffffffffffffffffffffffb283并且“Cand”是bc1c0f6a790c82e19c224b22f9b2ac830000000000000000000000000000002426，则如图所示可以使用加法导出16字节差异，例如通过把0x71a3加到开始于偏移量16的16字节区段并且丢弃进位。用以规定导出项的成本是5字节+3字节引用＝8字节，从而是原始大小的22.85％。应当

提到的是，图7A中的范例编码仅仅是出于说明的目的而选择的。图7B中的实例具有32字节的数据大小，因此对于单元内的长度和偏移量字段有5比特就足够了。对于较大的单元(例如4KB单元)，这些字段的大小将需要被增加到12比特。同样地，所述范例编码容许3字节或24比特的引用大小。这应当允许引用1600万个基本数据单元。如果引用需要能够对例如256TB的数据中的任何位置进行寻址，则引用的大小将需要是6个字节。当这样的数据集被因式分解成4KB单元时，规定引用所需要的6个字节将是4KB单元的大小的一小部分。

[0185] 规定(从一个或多个基本数据单元导出的)导出单元所需的信息的大小是重建程序的大小与规定所需的(一个或多个)基本数据单元所需的引用大小之和。把候选单元规定为导出单元所需的信息的大小被称作候选与基本数据单元的距离。当可以从多个基本数据单元集合当中的任一个集合可行地导出候选时，则把具有最短距离的基本数据单元集合选择成目标。

[0186] 当需要从多于一个基本数据单元导出候选单元时(通过组装从这些基本数据单元当中的每一个基本数据单元导出的提取项)，导出器需要考虑到针对存储系统的附加存取的成本，并且将该成本与更小的重建程序和更小的距离的益处进行权衡。一旦对于候选创建了最优的重建程序，将其距离与距离阈值进行比较；如果没有超出阈值，则接受导出。一旦接受导出，就把候选单元改订为导出单元，并且由基本数据单元于重建程序的组合替换。为候选单元创建的蒸馏数据中的条目被重建程序加上对相关基本数据单元的一项或多项引用替换。如果对应于最佳导出的距离超出距离阈值，则导出项将不被接受。

[0187] 为了产生数据简化，所述距离阈值必须总是小于候选单元的大小。举例来说，距离阈值可以被设定到候选单元大小的50％，从而使得只有在导出项的足迹小于或等于候选单元足迹的一半时才接受导出项，从而对于为之存在适当导出的每一个候选单元确保2x或更大的简化。距离阈值可以是预定的百分比或比例，其或者是基于用户规定的输入或者是由系统选择。距离阈值可以由系统基于系统的静态或动态参数确定。

[0188] 图8A-8E示出了根据这里所描述的一些实施例如何通过把输入数据因式分解成固定大小单元并且把所述单元组织在参照图3D和3E描述的树数据结构中而实施数据简化。图8A示出了如何可以把输入数据简单地因式分解成32字节组块。具体来说，图8A示出了前10个组块，并且随后示出了例如出现在4200万个组块之后的另外几个组块。图8B示出了使用名称把基本数据单元组织在滤筛中，所述名称被构造成使得名称的开头字节由来自单元内容中的3个维度(对应于锚指纹、第二指纹和第三指纹的位置)的内容构成。具体来说，在图8B中，每一个32字节组块成为具有32个字节的候选单元(固定大小块)。对单元的内容应用指纹处理方法。每一个单元具有如下构造的名称：来自单元的三个维度或字段(分别通过锚指纹、第二指纹和第三指纹定位)的数据的字节被串联形成名称的开头字节，随后是单元的其余字节。名称被用来在滤筛中组织单元。如图8B中所示，前10个组块不包含重复或导出项，并且作为单元相继被安装在滤筛中。图8B示出了在第10个组块被消耗之后的滤筛。图8C示出了在消耗了数据输入的附加的几百万个单元之后(例如在给出了接下来的4200万个组块之后)的某一后续时间点处的滤筛内容。针对重复或导出项检查滤筛。无法从单元导出的组块被安装在滤筛中。图8C示出了在消耗了4200万个组块之后的滤筛，其中例如包含16000010个单元(可以利用3个字节的引用地址进行逻辑寻址)，其余的26000000个组块则成为导出项。图8D示出了随后被呈现到滤筛并且被识别为滤筛中的某一条目(被示出为单

元编号24789)的重复的新鲜输入的一个实例。在该例中,滤筛把单元24789(组块9)识别成对应于组块42000011的最适当的单元。导出功能确定新的组块是确切的重复,并且将其替换成对单元24789的引用。用以表示导出项的成本是3字节引用相比于35B的原始大小,从而是原始大小的8.57%。图8D示出了被转换成滤筛中的某一条目(被示出为单元编号187126)的导出项的输入(组块42000012)的第二实例。在该例中,滤筛确定不存在确切的匹配。其把单元187125和187126(组块8和1)识别成最适当的单元。从所述最适当的单元导出新的单元。在图8D中示出了导出相比于单元187125和导出相比于单元187126。用以表示导出项相比于单元187125的成本是39字节+3字节引用=42字节,从而是原始大小的120.00%。用以表示导出项相比于单元187126的成本是12字节+3字节引用=15字节,从而是原始大小的42.85%。选择最佳导出(相比于单元187126)。将重建大小与阈值进行比较。举例来说,如果阈值是50%,则该导出项(42.85%)被接受。图8E提供了从基本数据单元导出的数据组块的两个附加的实例,其中包括通过从两个基本数据单元导出而实际创建导出项的一个实例。在第一实例中给出组块42000013。滤筛把单元9299998(组块10)识别成最适当的单元。在图8E中示出了导出相比于单元9299998。用以表示导出项的成本是4字节+3字节引用=7字节,从而是原始大小的20.00%。将重建大小与阈值进行比较。举例来说,如果阈值是50%,则该导出项(20.00%)被接受。在第二实例中给出组块42000014。在该例中,组块42000014使得该组块的一般可以从单元9299997最佳地导出,该组块的另一半则可以从单元9299998最佳地导出。因此,创建多单元导出项以产生进一步的数据简化。在图8E中示出了多单元导出。用以表示该多单元导出项的成本是3字节引用+3字节+3字节引用=9字节,从而是原始大小的25.71%。将重建大小与阈值进行比较,例如如果阈值是50%,则该导出项(25.71%)被接受。应当提到的是,来自单一单元导出项的最佳结果将是45.71%。

[0189] 图8A-8E示出了Data Distillation™系统的一个重要优点:可以在消耗和产生固定大小块的同时有效地实施数据简化。应当提到的是,固定大小块在高性能存储系统中是高度期望的。通过使用Data Distillation™装置,由许多固定大小的块构成的较大的传入输入文件可以被因式分解成许多固定大小的单元,从而使得所有基本数据单元都具有固定的大小。对应于每一个导出单元的潜在的可变大小重建程序被包装在一起并且被内联保持在蒸馏数据文件中,其随后可以被组块成固定大小块。因此,出于所有实际的目的,可以在存储系统中消耗和产生固定大小块的同时实施强力的数据简化。

[0190] 图9A-9C示出了首先在图1C中示出的系统的Data Distillation™方案的一个实例:这种方案采用可以通过内容关联方式被存取的单独的基本重建程序滤筛。这样的结构允许检测到构成已经存在于基本重建程序滤筛中的重建程序。这样的导出项可以被改订成引用现有的重建程序。这允许检测重建程序当中的冗余。在图9A中摄取输入数据。对所述数据应用指纹处理方法,并且在指纹位置处设定组块边界。如图所示,输入被因式分解成8个候选单元(在图9A中通过粗体和常规字体示出了交替的组块)。在图9B中,所述8个候选单元被示出为组织在滤筛中。每一个单元具有从该单元的整个内容构造的独特名称。在该例中,单元名称被如下构造:来自两个维度或字段(分别通过锚指纹和第二指纹定位)的数据的字节被串联形成名称的开头字节,随后是其余字节。名称被用来在滤筛中对单元进行排序,并且还通过树结构提供对滤筛的内容关联存取。图9B还示出了包含基本重建程序的第二内容关联结构。图9C示出了重复重建。假设所到来的55字节候选单元(图9C中示出)并非任何基

本数据单元的重复。单元3被选择成最适当的单元——前2个维度对于PDE 2和3是相同的，但是开始于88a7的其余字节与单元3匹配。利用一个12字节重建程序(RP)从单元3导出所述新的输入。编码如图7A中所示。对于该例应当注意到的是，最大单元大小是64比特，并且所有偏移量和长度都被编码成6比特值，而不是图7A中所示的5比特长度和偏移量。对基本重建程序滤筛进行搜索并且没有找到这一新RP。该RP被插入到基本重建程序滤筛中并且基于其值被排序。所述新单元被改订成对基本数据单元3的引用以及对基本重建程序滤筛中的引用4处的新创建的基本重建程序的引用。对应于该导出单元的总存储大小是:3字节PDE引用、3字节RP引用、12字节RP＝18字节，从而是相比于将其存储为PDE的大小的31.0％。后面假设所述55字节候选单元的一份拷贝到达。与前面一样，基于单元3创建一个12比特RP。对基本重建程序滤筛进行搜索，并且找到具有基本RP ID＝3、RP引用＝4的RP。该候选单元在系统中被表示成针对基本数据单元3的引用和针对重建程序4的引用。对于该导出单元所增加的总存储大小现在是:3字节PDE引用、3字节RP引用＝6字节，从而是相比于将其存储为PDE的大小的10.3％。

[0191] 图10A提供了根据这里所描述的一些实施例的关于如何对基本数据单元应用在重建程序中规定的变换以产生导出单元的一个实例。该例示出了被规定从编号为187126的基本数据单元(该基本数据单元也被示出在图8C的滤筛中)导出的导出单元，这是通过对该基本数据单元应用由所示出的重建程序规程的四种变换(插入、替换、删除和附加)。如图10A中所示，从滤筛加载单元187126，并且执行重建程序以便从单元187126导出组块42000012。图10B-10C示出了根据这里所描述的一些实施例的数据取回处理。每一项数据取回请求实质上采取蒸馏数据中的一个单元的形式，并且在无损简化格式中被呈现到取回引擎。对应于每一个单元的无损简化格式包含对相关联的(多个)基本数据单元的引用以及重建程序。Data Distillation™装置的取回器获取基本数据单元和重建程序，并且将其提供到重建器以供重建。在获取了对应于蒸馏数据的某一单元的相关基本数据单元和重建程序之后，重建器执行重建程序以便生成处于其原始未简化形式的所述单元。数据取回处理执行重建所需的工作量关于重建程序的大小和基本数据单元的大小成线性。因此，通过所述系统可以实现高数据取回速率。

[0192] 显而易见的是，为了把一个单元从蒸馏数据中的无损简化形式重建到其原始未简化形式，只需要获取对于该单元所规定的(多个)基本数据单元和重建程序。因此，为了重建给定的单元，不需要对其他单元进行存取或重建。这就使得Data Distillation™装置即使在为针对重建和取回的随机请求序列服务时仍然是高效的。应当提到的是，例如Lempel Ziv方法之类的传统压缩方法需要获取并且解压缩包含所期望的块的整个数据窗口。举例来说，如果存储系统采用Lempel-Ziv方法使用32KB的窗口压缩4KB数据库，则为了获取和解压缩给定的4KB块，需要获取和解压缩整个32KB窗口。由于为了给出所期望的数据需要消耗更多带宽并且需要解压缩更多数据，这样就构成了性能惩罚。Data Distillation™装置则不会招致这样的惩罚。

[0193] Data Distillation™装置可以通过多种方式被集成到计算机系统中，以便通过高效地在系统中的整个数据上全局发现并利用冗余的方式对数据进行组织和存储。图11A-11G示出了根据这里所描述的一些实施例的包括Data Distillation™机制(可以利用软件、硬件或者其组合来实施)的系统。图11A给出了通用计算平台，其中软件应用运行在系统软

件上，系统软件执行在硬件平台上，硬件平台由处理器、存储器和数据存储组件构成。图11B示出了被集成到平台的应用层中的Data Distillation™装置，每一个特定应用使用所述装置利用对应于该应用的数据集内的冗余。图11C示出了被采用来为在其上方应用的所有应用提供数据虚拟化层或服务的Data Distillation™装置。图11D和11E示出了Data Distillation™装置与范例计算平台的操作系统、文件系统和数据管理服务的两种不同形式的集成。其他集成方法包括(而不限于)与硬件平台中的嵌入式计算堆栈集成，比如采用在如图11F中所示的基于闪存的数据存储子系统中的嵌入式计算堆栈。

[0194] 图11G给出了Data Distillation™装置与图11D中示出的范例计算平台的集成的附加细节。图11G示出了Data Distillation™装置的组件，其中解析器和因式分解器、导出器、取回器和重建器作为软件在通用处理器上执行，并且内容关联映射结构驻留在存储分级结构的几个等级上。基本数据滤筛可以驻留在存储介质(比如基于闪存的存储驱动器)中。

[0195] 图11H示出了Data Distillation™装置如何可以与范例通用计算平台进行接口。

[0196] 文件系统(或档案系统(filesystem))把文件(例如文本文档、电子数据表、可执行文件、多媒体文件等等)与标识符(例如文件名、文件句柄等等)相关联，并且允许通过使用与文件相关联的标识符在文件上实施操作(例如读取、写入、插入、附加、删除等等)。由文件系统实施的命名空间可以是平坦的或分级的。此外，命名空间可以被分层，例如顶层标识符可以被解析成相继的更低层处的一个或多个标识符，直到顶层标识符被完全解析为止。通过这种方式，文件系统提供对于物理地存储文件内容的(多个)物理数据存储设备和/或存储介质(例如计算机存储器、闪存驱动器、盘驱动器、网络存储设备、CD-ROM、DVD等等)的抽象。

[0197] 被用于在文件系统中存储信息的物理存储设备和/或存储介质可以使用一种或多种存储技术，并且可以位于相同的网络位置处或者可以分布在不同的网络位置处。在给定与文件相关联的标识符以及被请求在文件上实施的一项或多项操作的情况下，文件系统可以(1)识别一个或多个物理存储设备和/或存储介质，并且(2)使得由文件系统识别出的物理存储设备和/或存储介质实施被请求在与所述标识符相关联的文件上实施的操作。

[0198] 每当在系统中实施读取或写入操作时，可能涉及不同的软件和/或硬件组件。术语“读取器”可以指代在系统中实施给定的读取操作时在所述系统中涉及的软件和/或硬件组件的选集，并且术语“写入器”可以指代在系统中实施给定的写入操作时在所述系统中涉及的软件和/或硬件组件的选集。这里所描述的数据简化方法和装置的一些实施例可以由在实施给定的读取或写入操作时所涉及的系统的一个或多个软件和/或硬件组件利用，或者可以被合并到其中。不同的读取器和写入器可以利用或合并不同的数据简化实现方式。但是，利用或合并特定数据简化实现方式的每一个写入器将对应于同样利用或合并相同的数据简化实现方式的读取器。应当提到的是，在系统中实施的一些读取和写入操作可能不会利用或合并数据简化装置。举例来说，当Data Distillation™装置或数据简化装置103取回基本数据单元或者把新的基本数据单元添加到基本数据滤筛时，其可以在没有数据简化的情况下直接实施读取和写入操作。

[0199] 具体来说，在图11H中，写入器150W可以总体上涉及在实施给定的写入操作时所涉及的系统的软件和/或硬件组件，并且读取器150R可以总体上涉及在实施给定的读取操作

时所涉及的系统的软件和/或硬件组件。如图11H中所示，写入器150W向Data Distillation™装置或数据简化装置103提供输入数据，并且从Data Distillation™装置或数据简化装置103接收蒸馏数据108。读取器150R向Data Distillation™装置或数据简化装置103提供取回请求109，并且从Data Distillation™装置或数据简化装置103接收所取回的数据输出113。

[0200] 对应于图11H的实现方式实例包括而不限于在应用、操作系统内核、文件系统、数据管理模块、设备驱动程序或者闪存或盘驱动器的固件中合并或利用Data Distillation™装置或数据简化装置103。这跨越了在图11B-11F描述的多种配置和使用。

[0201] 图11I图解说明在块处理存储系统中，如何把Data Distillation™装置用于数据简化。在这样的块处理系统中，数据被保存在块中，每个块由逻辑块地址或者说LBA识别。块不断被更改和重写，以致新的数据可能被重写到由特定LBA识别的块中。系统中的每个块被视为候选单元，Data Distillation™装置可用于把所述候选单元简化成由对(保存在特定基本数据单元块中的)基本数据单元的引用，以及在导出单元的情况下，对(保存在特定重建程序块中的)重建程序的引用构成的无损简化形式。图11I介绍了把利用LBA识别的块的内容映射到无损简化形式的对应单元的数据结构1151。而关于每个LBA将驻留关联单元的规范。对于采用固定大小的块的系统，便利的是使传入的块，基本数据单元块1152，以及重建程序块1153都是固定大小的。在该系统中，每个基本数据单元可被保存为单独的块。多个重建程序可被包装到也具有相同的固定大小的重建程序块中。对于各个基本数据单元和重建程序，数据结构还包含对驻留在叶节点数据结构的计数字段和关联元数据的引用，以致当用新的数据重写块时，可有效地管理驻留在LBA处的以前数据-(正在被重写的)现有基本数据单元和重建程序的计数字段必须被递减，并且同样地，进入到LBA中的由传入数据引用的基本数据单元的计数必须被递增。通过把对计数字段的引用保持在数据结构1151中，可快捷地管理重写，从而能够实现充分利用Data Distillation™装置提供的数据简化的高性能块处理存储系统。

[0202] 图12A示出了根据这里所描述的一些实施例使用Data Distillation™装置在受到带宽约束的通信介质上传送数据。在所示出的设置中，通信节点A创建将被发送到通信节点B的文件集合。节点A采用Data Distillation™装置把输入文件转换成蒸馏数据或蒸馏文件，其中包含对安装在基本数据滤筛中的基本数据单元的引用以及用于导出单元的重建程序。节点A随后把蒸馏文件连同基本数据滤筛发送到节点B(可以在发送蒸馏文件之前、同时或之后发送基本数据滤筛;此外，可以通过相同的通信信道或者通过与被用于发送蒸馏文件的通信信道不同的通信信道发送基本数据滤筛)。节点B把基本数据滤筛安装在相应结构的末端，并且随后通过驻留在Data Distillation™装置中的取回器和重建器馈送蒸馏文件，以便产生由节点A创建的原始文件集合。因此，通过在受到带宽约束的通信介质的全部两端采用Data Distillation™装置仅发送简化数据，使得对于所述介质的使用更加高效。应当提到的是，使用Data Distillation™允许利用更大范围内的冗余(超出使用例如Lempel-Ziv之类的传统技术的可行范围)，从而可以高效地传送非常大的文件或文件群组。

[0203] 我们现在将讨论在广域网设置中使用Data Distillation™装置，其中各个工作组协作共享分散在多个节点上的数据。当数据被初次创建时，可以如图12A中所示对其进行简化和传送。广域网在每一个站点处保持数据的拷贝，以便允许对于数据的快速本地存取。使

用Data Distillation™装置可以简化每一个站点处的足迹。此外，在任何站点处进行新鲜数据的后续摄取时，可以利用新鲜数据与已有的基本数据滤筛的内容之间的任何冗余以便对新鲜数据进行简化。

[0204] 在这样的设置中，需要把针对任何给定站点处的数据的任何修改传送到所有其他站点，从而把每一个站点处的基本数据滤筛保持一致。因此，如图12B中所示，根据这里所描述的一些实施例，可以把例如基本数据单元的安装和删除之类的更新以及元数据更新传送到每一个站点处的基本数据滤筛。举例来说，在把新鲜的基本数据单元安装到给定站点处的滤筛中时，所述基本数据单元需要被传送到所有其他站点。每一个站点可以使用该基本数据单元的值按照内容关联方式对滤筛进行存取，并且确定需要把新的条目添加在滤筛中的何处。同样地，在从给定站点处的滤筛中删除某一基本数据单元时，需要更新所有其他站点以反映出所述删除。可以实现这一点的一种方式是通过把所述基本数据单元传送到所有站点，从而使得每一个站点可以使用所述基本数据单元对滤筛进行内容关联存取，以便确定需要删除叶节点中的哪一个条目，连同针对树中的相关链接的必要更新以及从滤筛中删除该基本数据单元。另一种方法是向所有站点传送对所述基本数据单元所驻留的叶节点中的用于所述基本数据单元的条目的引用。

[0205] 因此，Data Distillation™装置可以被用来简化存储在广域网的各个站点处的数据的足迹，以及对网络的通信链接进行高效的使用。

[0206] 图12C-12K示出了根据这里所描述的一些实施例的由Data Distillation™装置对于各种使用模型所产生的简化数据的各个分量。

[0207] 图12C示出了Data Distillation™装置1203如何摄取输入文件1201，并且在完成蒸馏处理之后生成蒸馏文件集合1205和基本数据滤筛或基本数据存储库1206。图12C的基本数据滤筛或基本数据存储库1206本身由两个组成部分构成，即如图12D中所示的映射器1207和基本数据单元(或PDE)1208。

[0208] 映射器1207本身之内具有两个组成部分，即定义总体的树的树节点数据结构集合以及叶节点数据结构集合。树节点数据结构集合可以被放置到一个或多个文件中。同样地，叶节点数据结构集合可以被放置到一个或多个文件中。在一些实施例中，被称作树节点文件的单一文件保持对应于为给定数据集(输入文件1201)的基本数据单元创建的树的整个树节点数据结构集合，并且被称作叶节点文件的另一个单一文件保持对应于为该数据集的基本数据单元创建的树的整个叶节点数据结构集合。

[0209] 在图12D中，基本数据单元1208包含为给定的数据集(输入文件1201)创建的基本数据单元集合。所述基本数据单元集合可以被放置到一个或多个文件中。在一些实施例中，被称作PDE文件的单一文件保持为所述给定数据集创建的整个基本数据单元集合。

[0210] 树节点文件中的树节点将包含对树节点文件内的其他树节点的引用。树节点文件中的最深等级(或最低等级)的树节点将包含对叶节点文件中的叶节点数据结构中的条目的引用。叶节点文件中的叶节点数据结构中的条目将包含对PDE文件中的基本数据单元的引用。

[0211] 在图12E中示出了树节点文件、叶节点文件和PDE文件，该图示出了由所述装置创建的所有分量的细节。图12E示出了包括名称为file1、file2、file3…fileN的N个文件的输入文件集合1201，所述文件被Data Distillation™装置简化从而产生蒸馏文件集合1205和

基本数据滤筛的各个分量，也就是树节点文件1209、叶节点文件1210和PDE文件1211。蒸馏文件1205包括名称为file1.dist、file2.dist、file3.dist…fileN.dist的N个文件。Data Distillation™装置把输入数据因式分解成其构成单元，并且创建两个类别的数据单元——基本数据单元和导出单元。蒸馏文件包含处于无损简化格式中的数据单元的描述，并且包含对PDE文件中的基本数据单元的引用。输入文件1201中的每一个文件在蒸馏文件1205中具有相应的蒸馏文件。举例来说，输入文件1201中的file1 1212对应于蒸馏文件1205中的名称为file1.dist 1213的蒸馏文件。图12R示出了被指定为一组输入文件和目录或文件夹的输入数据集的替代表示。

[0212] 应当提到的是，图12E示出了由数据蒸馏装置基于根据图1A的蒸馏数据和基本数据滤筛的组织而创建的各个组成部分，其中重建程序被放置在蒸馏文件中的单元的无损简化表示中。应当提到的是，(根据图1B的)一些实施例可以把重建程序放置在基本数据滤筛中，并且将其像基本数据单元一样对待。蒸馏文件中的单元的无损简化表示将包含对基本数据滤筛中的重建程序的引用(而不是包含重建程序本身)。在这些实施例中，重建程序将像基本数据单元一样被对待，并且在PDE文件1211中产生。在另一个实施例中，根据图1C，重建程序与基本数据单元分开存储并且被存储在称作重建程序存储库的结构中。在这样的实施例中，蒸馏文件中的单元的无损简化表示将包含对重建程序存储库中的重建程序的引用。在这样的实施例中，如图12F中所示，除了产生对应于基本数据单元的树组织的树节点文件1209、叶节点文件1210和PDE文件1211之外，所述装置还将产生被称作重建树节点文件1219和重建叶节点文件1220的树和叶节点文件的第二集合，连同被称作RP文件1221的包含所有重建程序的文件。

[0213] 图12E中所示的Data Distillation™装置还把管理其操作的配置和控制信息存储在树节点文件1209、叶节点文件1210、PDE文件1211和蒸馏文件1205当中的一项或多项中。或者可以生成包含该信息的第五分量。类似地对于图12F中示出的装置，所述配置和控制信息可以被存储在图12F所示的各个分量当中的一个或多个分量中，或者可以被存储在为此目的生成的另一个分量中。

[0214] 图12G示出了Data Distillation™装置的使用的总览，其中给定的数据集(输入文件1221)被馈送到Data Distillation™装置1203，并且被处理以产生无损简化数据集(无损简化数据集1224)。输入数据集1221可以由文件、对象、块、组块或来自数据流的提取项的选集构成。应当提到的是，图12E示出了其中所述数据集由文件构成的实例。图12G的输入数据集1221对应于图12E的输入文件1201，图12G的无损简化数据集1224则包括图12E中示出的四个组成部分，即图12E的蒸馏文件1205、树节点文件1209、叶节点文件1210和PDE文件1211。在图12G中，Data Distillation™装置利用为之给出的输入数据集的整个范围内的数据单元当中的冗余。

[0215] Data Distillation™装置可以被配置成利用输入数据集的一个子集中的冗余，并且提供对应于为之给出的每一个数据子集的无损简化。举例来说，如图12H中所示，输入数据集1221可以被分割成许多更小的数据选集，每一个选集在本公开内容中被称作“批次”或“数据的批次”或“数据批次”。图12H示出了被配置成摄取输入数据批次1224并且产生无损简化数据批次1225的Data Distillation™装置。图12H示出输入数据集1221由若干数据选集构成，也就是数据批次1…数据批次i…数据批次n。数据被每次一个数据批次呈现到Data

Distillation™装置，并且利用每一个数据批次的范围内的冗余以生成无损简化数据批次。举例来说，来自输入数据集1221的数据批次i1226被馈送到所述装置，并且无损简化数据批次i 1228被递送到无损简化数据集1227。来自输入数据集1221的每一个数据批次被馈送到所述装置，并且相应的无损简化数据批次被递送到无损简化数据集1227。在消耗并且简化了所有数据批次1…数据批次i…数据批次n之后，输入数据集1221被简化到无损简化数据集1227。

[0216] 虽然Data Distillation™装置的设计在利用全局数据范围内的冗余方面已经是高效的，但是前面的技术可以被用来进一步加快数据简化处理并且进一步改进其效率。通过把数据批次的大小限制到能够容纳到系统的可用存储器中，可以提高数据简化处理的吞吐量。举例来说，其大小是许多太字节或甚至拍字节的输入数据集可以被分解成分别具有例如256GB大小的许多数据批次，并且每一个数据批次可以被快速地简化。通过使用具有256GB的存储器的单一处理器核心(Intel Xeon E5-1650 V3,Haswell 3.5Ghz处理器)，在我们的实验室中已经实施了利用256GB的范围内的冗余的此类解决方案，从而给出每秒几百兆字节的数据摄取速率，同时在各种数据集上给出2-3x的简化水平。应当提到的是，256GB的范围比32KB大几百万倍，而32KB是Lempel Ziv方法在现今的处理器上给出10MB/秒到200MB/秒之间的摄取性能的窗口大小。因此，通过适当地限制冗余的范围，通过潜在地牺牲一些简化可以实现数据蒸馏处理的速度方面的改进。

[0217] 图12I示出了图12H中的设置的一种变型，并且示出了运行在多个处理器上以便显著提升输入数据集的数据简化(并且还有数据重建/取回)吞吐量的多个数据蒸馏处理。图12I示出了被分割成x个数据批次的输入数据集1201，所述x个独立数据批次被馈送到运行在独立处理器核心上的j个独立处理中(每一个处理被分配足够的存储器以容纳将被馈送到该处的任何数据批次)以得到并行执行，并且对于数据简化以及重建/取回都产生近似j倍的加速。

[0218] 图12H示出了被配置成摄取输入数据批次1224并且产生无损简化数据批次1225的Data Distillation™(数据蒸馏)装置。图12H示出了输入数据集1221由多个数据选集构成，也就是数据批次1…数据批次i…数据批次n。在一些实施例中，可以采用将输入数据集划分为多个数据批次的替代性分区方案，其中动态地确定数据批次边界以便最好地利用可用存储器。可用存储器可以用于首先保存所有树节点，或者可以用于保存数据批次的所有树节点和所有叶节点，或者最后可以用于保存所有树节点、叶节点和所有基本数据单元。这三种不同的选择使装置能够具有替代的操作点。例如，使可用存储器专用于树节点可以使更大范围的数据容纳在数据批次中，但这要求装置必须在需要时从存储装置中获取叶节点以及相关的基本数据单元，从而导致附加的延迟。替代地，使可用存储器专用于容纳树节点和叶节点两者加快了蒸馏速度，但会减少树的有效大小，从而减少可容纳在数据批次中的数据的范围。最后，使用可用存储器来保持所有树节点、叶节点和基本数据单元将实现最快的蒸馏，但是可以作为单个范围支持的数据批次的大小将最小。在所有这些实施例中，将在达到存储器限制时动态关闭数据批次，并且来自输入数据集的后续文件将成为新数据批次的一部分。

[0219] 存在可以提高装置的效率并加快重建过程的进一步的改进。在一些实施例中，单个统一映射器用于蒸馏，但是不是将基本数据单元保持在单个PDE文件中，而是基本数据单

元跨N个PDE文件保持。因此，以前的单个PDE文件被划分为n个PDE文件，每个PDE文件小于某个阈值大小，并且当PDE文件超过该阈值大小时(由于基本数据单元的安装而增长时)，每个分区在蒸馏过程中被创建。通过咨询映射器以内容关联地选择适合于导出的适当基本数据单元，然后导出从其所在的适当PDE文件中获取的适当基本数据单元，来对每个输入文件进行蒸馏。每个蒸馏文件被进一步增强，以列出(n个PDE文件中)包含特定蒸馏文件引用的基本数据单元的所有PDE文件。为了重建特定的蒸馏文件，只有那些列出的PDE文件将需要被加载或打开以被访问以用于重建。这样做的好处在于，对于单个或几个蒸馏文件的重建，仅包含该特定蒸馏文件所需的基本数据单元的那些PDE文件需要被访问或保持活动，而其它PDE文件无需被保留或加载到快速的存储器或存储装置层中。因此，重建可以加快并且更高效。

[0220] 将PDE文件分区为n个PDE文件还可以通过标准进行指导，该标准将在数据集中的任何给定文件的简化过程中对基本数据进行的引用模式本地化。装置可以通过计数器进行增强，该计数器对当前PDE文件中对单元的引用的密度进行计数和估计。如果该密度高，则PDE文件不会被分区或拆分，并且将在后续单元的安装时不断增长。一旦给定蒸馏文件中的引用的密度逐渐降低，就可以允许PDE文件在后续增长超过某个阈值时被拆分和分区。一旦被分区，就将打开新的PDE文件，并将使得来自后续蒸馏的后续安装进行到该新的PDE文件中。如果只需要重建数据批次中的文件的子集，那么这种布置将进一步加快重建。

[0221] 图12J示出了由Data Distillation™装置对于一个使用模型产生的简化数据的各个分量，其中在输入数据集的简化之后不再需要保留映射器。这样的使用模型的实例是特定种类的数据备份和数据归档应用。在这样的使用模型中，对于简化数据的后续使用是从简化数据集重建和取回输入数据集。在这样的情形中，通过在数据简化完成之后不再存储映射器可以把简化数据的足迹进一步简化。图12J示出了被馈送到所述装置的输入文件1201，从而产生蒸馏文件1205和PDE文件1211——这些组成部分(或分量)构成这种情形中的简化数据。应当提到的是，仅使用蒸馏文件1205和PDE文件1211可以完全重新生成并恢复输入文件1201。回想到对应于蒸馏文件中的每一个单元的无损简化表示包含重建程序(在需要时)以及对PDE文件中的基本数据单元的引用。与PDE文件相耦合，这就是执行重建所需的全部信息。还需要指出的是，这种布置对输入数据集的重建和取回的性能效率具有重要的益处。在该实施例中，装置将输入数据集分解为包含在单独的PDE文件中的蒸馏文件和基本数据单元。在重建期间，可以先将PDE文件从存储装置加载到可用存储器中，然后可以从存储装置中连续读取蒸馏文件进行重建。在重建每个蒸馏文件期间，将快速从存储器中取回重建蒸馏文件所需的任何基本数据单元，而不会因读取基本数据单元而引起任何附加的存储装置访问延迟。重建的蒸馏文件可以在其完成时被写出到存储装置中。这种布置排除了执行随机存储访问的需要，否则这种随机存储访问将对性能产生有害的影响。在这种解决方案中，来自存储装置的PDE文件的负载是对顺序的连续字节块的一组访问，每个蒸馏文件的读取也是对顺序的连续字节块的一组访问，最后每个重建的输入文件作为对一组顺序的连续字节块的访问被写出到存储装置中。这种布置的存储性能更紧密地跟踪顺序地读取和写入连续字节块的性能，而不是引起多个随机存储访问的解决方案的性能。

[0222] 应当提到的是，图12J示出了由数据蒸馏装置基于根据图1A的蒸馏数据和基本数据滤筛的组织而创建的各个分量，其中重建程序被放置在蒸馏文件中的单元的无损简化表

示中。应当提到的是,(根据图1B的)一些实施例可以把重建程序放置在基本数据滤筛中,并且将其像基本数据单元一样对待。蒸馏文件中的单元的无损简化表示将包含对基本数据滤筛中的重建程序的引用(而不是包含重建程序本身)。在这些实施例中,重建程序将像基本数据单元一样被对待,并且在PDE文件1211中产生。在另一个实施例中,根据图1C,重建程序与基本数据单元分开存储并且被存储在称作重建程序存储库的结构中。在这样的实施例中,蒸馏文件中的单元的无损简化表示将包含对重建程序存储库中的重建程序的引用。在这样的实施例中,除了产生对应于基本数据单元的PDE文件之外,所述装置还将产生被称作RP文件的包含所有重建程序的文件。这在图12K中示出,该图对于其中不再需要保留映射器的使用模型示出了简化数据的分量。图12K示出了包括蒸馏文件1205、PDE文件1211和RP文件1221的简化数据分量。

[0223] 图12L-12P示出了根据本文描述的一些实施例如何在分布式系统上部署和执行蒸馏过程以能够以非常高的摄取速率适应非常大的数据集。

[0224] 分布式计算范例需要通过在多个计算机上运行的程序对大数据集进行分布式处理。图12L示出了被称为分布式计算集群的组织中联网在一起的多个计算机。图12L示出了计算机之间的点对点链路,但是将会理解的是,可以使用任何通信拓扑结构(例如,轴辐式(hub-and-spoke)拓扑结构或者网状拓扑结构)来代替图12L所示的拓扑结构。在给定的集群中,一个节点被指定为将任务分配给从节点、并控制和协调整个操作的主节点。从节点按照主节点的指示执行任务。

[0225] 数据蒸馏过程可以跨分布式计算集群的多个节点以分布式方式执行,以利用集群中的多个计算机的总计算、存储器和存储容量。在此设置中,主节点上的主蒸馏模块与从节点上运行的从蒸馏模块相互作用,以分布式方式实现数据蒸馏。为了便于这种分配,可以将装置的基本数据滤筛分割成多个独立的子集或子树,这些子集或子树可以分布在从节点上运行的多个从模块上。回想一下,在数据蒸馏装置中,基本数据单元基于其名称以树形式进行组织,并且它们的名称来源于其内容。基本数据滤筛可以根据基本数据滤筛中单元名称的前导字节划分为多个独立的子集或子筛。可以有多种方法在多个子树上划分名称空间。例如,可以将单元名称的前导字节的值分割成多个子范围,并将每个子范围分配给子筛。可以创建多少个子集或分区,因为集群中有从模块,因此每个独立分区都部署在特定的从模块上。使用部署的子筛,每个从模块被设计为对其接收的候选单元执行数据蒸馏处理。

[0226] 图12M示出了将基本数据滤筛分成将被部署在运行于4个节点上的4个从模块上的4个基本数据滤筛或子筛,标记为PDS_1、PDS_2、PDS_3和PDS_4。分区基于基本数据单元的名称的前导字节。在所示的示例中,PDS_1中所有单元的名称的前导字节将位于范围A到I中,并且滤筛PDS_1将具有名称A_I,该名称由引导到它的值的范围标记。同样,PDS_2中的所有单元的名称的前导字节将在J到O的范围内,并且子筛PDS_2将具有由引导到其的值的范围标记的名称J_O。同样,PDS_3中的所有单元的名称的前导字节将在P到S的范围内,并且子筛PDS_3将具有由引导到其的值的范围标记的名称P_S。最后,PDS_4中所有单元的名称的前导字节将在T到Z的范围内,并且子筛PDS_4将具有由引导到它的值的范围标记的名称T_Z。

[0227] 在此设置中,在主节点上运行的主模块接收输入文件并执行输入文件的轻量级解析和因式分解以将输入文件分解为候选单元序列,并且随后将每个候选单元引导向合适的从模块进行进一步处理。轻量级解析可以包括针对图式解析每个候选单元,或者可以包括

在候选单元上应用指纹，以确定构成候选单元的名称的前导字节的维度。主设备处的解析仅限于识别足以确定哪个从模块应该接收候选单元的一样多的字节。基于候选单元的名称的前导字节中的值，候选者被转发到保持与该特定值相对应的子筛的从节点处的从模块。

[0228] 当数据累积到滤筛中时，可以间歇地重新访问分区并重新平衡分区。分区和重新平衡功能可以由主模块执行。

[0229] 在接收到候选单元之后，每个从模块执行数据蒸馏处理，从对候选单元的完整解析和检查开始创建其名称。使用该名称，从模块执行子筛的内容关联查询，并且执行蒸馏处理以将候选单元转换为关于该子筛的无损简化表示中的单元。通过用以识别从模块的被称为从编号(SlaveNumber)的字段和关于其简化了该单元的对应的子筛增强蒸馏文件中的单元的无损简化表示。单元的无损简化表示被发送回主模块。如果候选单元在子筛中未找到，或者不能从子筛中的基本数据单元中导出，则将新的基本数据单元识别为分配给子筛。

[0230] 主模块继续将来自输入文件的所有候选单元引导至适当的从模块并且累积进入的单元描述(以无损简化表示)，直到其已经接收到输入文件的所有单元为止。此时，可以向所有的从模块发出全局提交通信，以便根据其各自的蒸馏处理的结果来更新其各自的子筛。针对输入的蒸馏文件存储在主模块中。

[0231] 在一些实施例中，不是在任何从设备可以用新的基本数据单元或元数据来更新其子筛之前等待整个蒸馏文件被准备，而是可以在从模块处处理候选单元时完成对子筛的更新。

[0232] 在一些实施例中，根据图1B和1C的描述，每个子筛包含基本数据单元以及重建程序。在这样的实施例中，重建程序被存储在子筛中，并且无损简化表示包含对子筛中的基本数据单元以及重建程序(如果需要)的引用。这进一步减小了单元的大小，并因此减小了需要存储在主模块中的蒸馏文件的大小。在一些实施例中，每个子筛中的基本重建程序滤筛包含用于从驻留在该子筛中的基本数据单元创建导出项的那些重建程序。在这种情况下，基本重建程序在从节点本地可用，并且使得能够进行快速导出和重建，而不会产生以其它方式会从远程节点获取基本重建程序带来的任何延迟。在其它实施例中，基本重建程序滤筛跨所有节点全局分布，以利用分布式系统的总容量。通过第二字段增强无损简化表示，该第二字段识别包含基本重建程序的从节点或子筛。在这样的实施例中，该解决方案导致从远程节点获取基本重建程序的附加延迟，以便通过导出生成最终重建程序或者重建单元。整个方法利用所有从节点的组合存储容量来基于每个文件中的每个块或候选单元的内容在所有节点上分发文件。

[0233] 数据取回由主模块类似地协调。主模块接收蒸馏文件并检查蒸馏文件中每个单元的无损简化规格。其提取字段“SlaveNumber”，指示哪个从模块将重建该单元。该单元然后被发送到适当的从模块以进行重建。该重建的单元然后被发送回主模块。主模块汇编从所有从模块重建的单元，并将重建的文件转发给要求该文件的消费者。

[0234] 图12N图示了可以如何在分布式系统中部署和执行数据蒸馏装置。输入文件1251被馈送到主模块，该模块解析并识别文件中每个候选单元的名称的前导字节。主模块将候选单元引导到4个从模块之一。持有具有名称A_I的包含具有在范围A至I中的名称承载值(Name bearing value)的前导字节的基本数据单元的PDS_1或子筛的从节点1处的从模块1接收带有名称BCD...的候选单元1252，其被确定为已经存在于名为A_I的子筛中的单元的

副本。从模块1返回无损简化表示1253，该表示包含该单元为基本的指示符并且驻留在地址为refPDE1的Slave1处。如图12N所示，主模块将所有候选单元发送到相关的从模块并汇编和收集并最后存储蒸馏文件。

[0235] 图12O示出了图12N所示的图示的变体。在该变体中，在蒸馏文件中每个单元的无损简化表示中，标识该单元已经相对其简化的特定Child_Sieve的字段包含该Child_Sieve的名称，而不是Child_Sieve驻留在其上的模块或节点的编号。因此，字段SlaveNumber被字段Child_Sieve_Name取代。这具有通过其虚拟地址而不是Child_Sieve所驻留的模块或物理节点的编号引用相关的Child_Sieve的益处。因此，如图12O所示，持有具有名称A_I的包含具有在范围A至I中的名称承载值的前导字节的基本数据单元的PDS_1或子筛的从节点1处的从模块1接收带有名称BCD...的候选单元1252，其被确定为已经存在于名为A_I的子筛中的单元的副本。从模块1返回无损简化表示1254，该表示包含该单元为基本的指示符并且驻留在地址为refPDE1的Slave1处。

[0236] 注意，通过采用图12L至图12O中描述的布置，可以提高数据蒸馏处理的整体吞吐率。主模块的吞吐量将受到轻量级解析和来自主模块的候选单元的调度的限制。对许多候选单元的蒸馏将并行执行，只要它们的内容将它们引导到不同的从模块。

[0237] 为了进一步提高整体吞吐量，可以并行化输入流的轻量级解析和因式分解以识别哪个Child_Sieve应当接收候选单元的任务。这个任务可以被主模块划分成多个并行任务，由多个从节点上运行的从模块并行执行。这可以通过预测数据流并将数据流分成多个部分重叠的段来完成。这些段由主模块发送到并行执行轻量级解析和因式分解的每个从模块，并将因式分解的结果发送给主模块。主模块分解跨越每个段的边界的因式分解，然后将候选单元路由到适当的从模块。

[0238] 图12L至图12O描述了其中数据蒸馏装置以分布式方式运行的布置，其中主蒸馏模块在主节点上运行，并且多个从蒸馏模块在从节点上运行。主模块负责执行基本数据单元跨各种子筛的分区。在所示的安排中，所有要摄取的输入文件都由主模块摄取，并且无损简化的蒸馏文件保留在主模块处，而所有基本数据单元(以及任何主要重建程序)驻留在各个从模块处的子筛中。对文件的数据取回请求也由主文件处理，并且对相应的蒸馏文件的重建由主文件协调。图12P示出了输入文件可以被任何从蒸馏模块(以及保留在那些模块中的相应蒸馏文件)摄取的变型，并且数据取回请求可以由任何从蒸馏模块处理。主模块继续以相同的方式在子筛上执行基本数据单元的分区，以便基本数据单元在各个子筛上的分布将与图12L到12O中所示的布置相同。但是，在图12P所示的新配置中，每个从模块都知道该分区，因为每个从模块都可以摄取和取回数据。另外，所有的模块都知道在由这些模块摄取数据时在每个模块上创建和存储的蒸馏文件的存在和位置。这允许任何从模块满足针对存储在整个系统中的任何文件的数据取回请求。

[0239] 如图12P所示，每个从模块可以从分布式存储系统摄取和取回数据。例如，从蒸馏模块1 1270摄入输入文件I 1271并且执行轻量级解析以将输入文件I因式分解并且将候选单元路由到包含与输入文件I中的每个候选单元的名称对应的子筛的模块。例如，来自输入文件I的候选单元1275被发送到从蒸馏模块2 1279。同样，从蒸馏模块21279摄取输入文件II并执行轻量级解析以将输入文件II因式分解并将候选单元路由到包含与输入文件II中的每个候选单元的名称对应的子筛的模块。例如，来自输入文件II的候选单元1277被发送

到从蒸馏模块1 1270。每个从蒸馏模块处理它们接收的候选单元，完成关于其子筛的蒸馏过程，并将候选单元的无损简化表示返回到摄入数据的发起模块。例如，响应于从从蒸馏模块1 1270接收来自输入文件I的候选单元1275，从蒸馏模块2 1279将无损简化单元1276返回至从蒸馏模块1 1270。同样，响应于从从蒸馏模块2 1279接收来自输入文件II的候选单元1277，从蒸馏模块1 1270将无损简化单元1278返回到从蒸馏模块2 1279。

[0240] 在该布置中，可以在任何从模块处满足数据的取回。接收取回请求的模块需要首先确定该请求的文件的蒸馏文件所驻留的位置，并从相应的从模块获取蒸馏文件。随后，发起的从模块需要协调该蒸馏文件中各个单元的分布式重建，以产生原始文件并将其传送给请求的应用。

[0241] 以这种方式，可以在分布式系统的多个节点上以分布式方式执行数据蒸馏处理，以更有效地利用集群中的多个计算机的总计算、存储器和存储容量。系统中的所有节点都可以用来摄取和取回数据。这应该能够在充分利用系统中的节点的总的组合存储容量的同时，实现非常高的数据摄取和取回速率。这还允许在系统中的任何节点上运行的应用在本地节点上查询存储在系统中任何位置的任何数据，并且使该查询高效无缝地满足。

[0242] 在图12M到12P所描述的布置中，跨驻留在系统的各个节点中的子筛的数据划分基于全局可见名称空间中的单元名称，其中各单元通过因式分解输入文件来提取。在另一种安排中，共享某些元数据的数据批次或整个文件组可以被分配并存储在特定的节点上。因此，整体数据的主分区基于数据批次，并且由主数据执行和管理。所有的从模块都保持知晓数据批次到模块的分配。数据批次将完全驻留在给定的从节点上。在该从节点上运行的蒸馏从模块上的子筛将包含属于该数据批次的所有基本数据单元。换句话说，给定数据批次的所有基本数据单元的整个树将完全驻留在单个从蒸馏模块内的单个子筛上。给定数据批次的所有蒸馏文件也将驻留在同一从蒸馏模块中。使用这种布置，输入文件仍然可以被任何从蒸馏模块摄取，并且数据取回请求仍然可以被任何从蒸馏模块处理。但是，给定数据批次的整个数据蒸馏处理在包含该数据批次的模块上完全执行。数据摄取和数据取回的请求从发起模块路由到指定用于容纳特定数据批次的特定从模块。该方案具有在因式分解和蒸馏数据批次时在分布式环境中具有降低的通信开销的优点。在整个全局数据足迹内不再采用冗余，而是在数据批次内本地非常有效地利用冗余。该方案仍然使用分布式系统的组合存储容量，并提供无缝的能力来从系统的任何节点查询、摄取和取回任何数据。

[0243] 因此，采用上述众多技术，高效利用分布式系统中的资源来以非常高的速度在非常大的数据集上执行数据蒸馏。

[0244] 可以进一步增强Data Distillation™方法和装置，以促进数据的高效移动和迁移。在一些实施例中，可以以多个容器或包裹的形式递送无损简化的数据集，以促进数据移动。在一些实施例中，一个或多个简化的数据批次可以装配到单个容器或包裹中，并且替代地，可以将单个简化的数据批次转换成多个包裹。在一些实施例中，单个简化的数据批次作为单个自描述包裹被递送。图12Q图示了这种包裹的示例结构。可以将图12Q中的包裹1280视为单个文件或字节的连续集合，其包含依次彼此串联的以下组成部分：(1)头部1281，它是包裹头部，包裹头部首先包含指定包裹的长度的包裹长度1282，其次包含识别蒸馏文件、PDE文件和各种清单在包裹中的位置的偏移量的偏移量标识符；(2)蒸馏文件1283，它是一个接一个地串联的数据批次的蒸馏文件，其中首先指定每个蒸馏文件的长度，然后是组成

该蒸馏文件的所有字节；(3)PDE文件1284，它是PDE文件，从PDE文件的长度标识符开始，然后是包含所有基本数据单元的PDE文件的主体；(4)源清单1285，它是源清单，描述输入数据集的结构并识别包裹中每个文件的唯一目录结构、路径名和文件名。源清单还包含输入数据批次中的每个节点的列表(其已被简化并变成包裹)以及与每个节点相关联的元数据；(5)目的地清单和映射器1286，它是目的地清单和映射器。目的地映射器提供了每个输入节点和文件到目的地目录和文件结构或云中的目标存储桶/容器和对象/二进制大对象(bolb)结构的预期映射。该清单有助于在数据移动后将包裹中的各个组成部分移动、重建和重新放置到最终目的地。注意的是，可以单独更改该目的地映射器部分，以重新定位包裹中的数据要传输到和被重建的目的地。

[0245] 以这种方式，数据批次的无损简化表示作为包裹以自描述的并且适合于数据的移动和重定位的格式递送。

[0246] 使用本文描述的实施例在各种真实世界数据集上执行数据简化以确定这些实施例的有效性。所研究的真实世界数据集包括公司电子邮件的Enron语料库、美国政府的各种记录和文件、进入MongoDB NOSQL数据库的美国运输部记录以及向公众公开的公司PowerPoint演示文稿。使用这里描述的实施例并且将输入数据因式分解成平均4KB的可变大小的单元(边界由指纹确定)，在这些数据集上实现了3.23x的平均数据简化。3.23x的简化意味着简化的数据的大小等于原始数据的大小除以3.23x，导致压缩比为31％的简化足迹。传统的重复数据删除技术被发现使用等效的参数在这些数据集上传递1.487x的数据简化。使用本文描述的实施例并将输入数据因式分解成4KB的固定大小的单元，在这些数据集上实现了1.86x的平均数据简化。传统的重复数据删除技术被发现使用等效的参数在这些数据集上传递1.08x的数据简化。因此，Data Distillation™方案被发现比传统的重复数据删除方案提供了更好的数据简化。

[0247] 测试运行还确认了基本数据单元的字节的小子集用于对滤筛中的大部分单元进行排序，从而实现对于其操作需要最小增量存储的方案。

[0248] 结果证实，Data Distillation™装置有效地实现了以比该单元自身更精细的粒度在整个数据集上的全局数据单元之间利用冗余。这种方法实现的无损数据简化是通过数据访问和IO的经济实现的，采用的数据结构本身只需要最小的增量存储，并且使用现代多核微处理器上可用的总计算处理能力的一小部分。前面部分中描述的实施例的特征在于在大的和极大的数据集上执行无损数据简化，同时提供高速率的数据摄取和数据取回的系统和技术，并且不具有传统技术的缺点和局限性。

[0249] 通过从驻留在基本数据滤筛中的基本数据单元导出数据，对已经无损简化的数据执行内容关联搜索和取回

[0250] 可以利用某些特征来增强在前面的文本中描述并且在图1A至图12P中示出的数据蒸馏装置，以便有效地对来自以无损简化格式存储的数据的信息执行多维搜索和内容关联取回。这种多维搜索和数据取回是分析或数据仓储应用的关键构件。现在将描述这些增强。

[0251] 图13示出了类似于图3H所示的结构的叶节点数据结构。但是，如图13所示，每个基本数据单元的叶节点数据结构中的条目被增强，以包含对蒸馏数据中的所有单元的引用(其也被称为反向引用或反向链接)，包括对该特定基本数据单元的引用。回想一下，数据蒸馏图式将来自输入文件的数据因式分解成使用诸如图1H中描述的规范以简化格式置于蒸

馏文件中的单元序列。蒸馏文件中有两种单元-基本数据单元和导出单元。蒸馏文件中每个单元的规范都将包含对驻留在基本数据滤筛中的基本数据单元的引用。对于这些引用中的每一个(从蒸馏文件中的单元到基本数据滤筛中的基本数据单元),将会有相应的反向链接或反向引用(从叶节点数据结构中的基本数据单元的条目到蒸馏文件中的单元)安置在叶节点数据结构中。反向引用确定蒸馏文件中的偏移量,该偏移量标记单元的无损简化表示的开始。在一些实施例中,反向引用包括蒸馏文件的名称和定位该单元的开始的该文件内的偏移量。如图13中所示,除了对蒸馏文件中的每个单元的反向引用,叶节点数据结构还保持标识在蒸馏文件中被引用的单元是基本数据单元(基本)还是导出单元(导出项deriv)的指示符。在蒸馏过程中,在将单元放入蒸馏文件时,将反向链接安置到叶节点数据结构中。

[0252] 反向参考或反向链接被设计为通用句柄,其可以到达共享基本数据滤筛的所有蒸馏文件中的所有单元。

[0253] 预期反向引用的添加不会显著影响所达到的数据简化,因为预期数据单元的大小被选择为使得每个引用是数据单元的大小的一部分。例如,考虑一个系统,其中导出单元被约束为每个导出项不超过1个基本数据单元(因此不允许多单元导出项)。跨所有叶节点数据结构的反向引用的总数将等于所有蒸馏文件中单元的总数。假设32GB大小的样本输入数据集简化到8GB的无损简化数据,采用1KB的平均单元大小,并产生4倍的简化率。输入数据中有32M个单元。如果每个反向引用大小为8B,则反向引用占用的总空间为256MB或0.25GB。这对8GB足迹的简化数据是小的增加。新的足迹将达到8.25GB,有效降幅将达到3.88倍,相当于3%的简化损失。对于简化数据的强大的内容关联数据取回的好处,这是一个小的代价。

[0254] 如本文前面所述,蒸馏装置可采用多种方法来确定候选单元的内容内的骨骼数据结构的各个分量的位置。单元的骨架数据结构的各个分量可以被认为是维度,所以后面紧跟着每个单元的其余内容的这些维度的级联被用来创建每个单元的名称。该名称用于排序和组织树中的基本数据单元。

[0255] 在已知输入数据的结构的使用模型中,图式定义了各个字段或维度。这种图式由正在使用该内容关联数据取回装置的分析应用提供,并通过该应用的接口提供给装置。基于图式中的声明,蒸馏装置的解析器能够解析候选单元的内容以检测和定位各个维度并创建候选单元的名称。如前所述,与维度对应的字段中具有相同内容的单元将沿着树的同一分支(leg)分组在一起。对于安置到滤筛中的每个基本数据单元,可以将维度上的信息作为元数据存储在叶节点数据结构中的基本数据单元的条目中。该信息可以包括在每个所声明的维度上的内容的位置、大小和值,并且可以存储在图13中被称为“基本数据单元的其它元数据”的字段中。

[0256] 图14A示出根据本文描述的一些实施例的提供输入数据集的结构的描述以及输入数据集的结构与维度之间的对应关系的描述的示例图式。结构描述1402是描述输入数据的完整结构的更完整图式的摘录或部分。结构描述1402包括键(key)列表(例如,“PROD_ID”、“MFG”、“MONTH”、“CUS_LOC”、“CATEGORY”和“PRICE”),后跟对应于该键的值的类型。冒号字符“:”被用作分隔符来分隔键和值的类型,分号字符“;”被用作分隔符来分隔不同的键对和值的相应类型。请注意,完整图式(结构1402是其中的一部分)可以指定附加字段来标识每个输入的开始和结束,也可能指定维度之外的其他字段。维度映射描述1404描述了用于组

织基本数据单元的维度如何映射到结构化输入数据集中的键值。例如，维度映射描述1404中的第一行指定对应于输入数据集中的键“MFG”的值的前四个字节(因为第一行以文本“prefix＝4”结束)用于创建维度1。维度映射描述1404中的其余行描述了如何基于结构化输入数据创建其他三个维度。在键到维度的该映射中，键在输入中出现的顺序不一定匹配维度的顺序。使用提供的图式描述，解析器可以在输入数据中识别这些维度以创建候选单元的名称。对于图14A中的例子，并且使用维度映射描述1404，将如下创建候选单元的名称-(1)名称的前4个字节将是与被声明为维度1的键“MFG”对应的值的前4个字节，(2)名称的接下来的4个字节将是与被声明为维度2的键“CATEGORY”对应的值的前4个字节，(3)名称的接下来的3个字节将是与被声明为维度3的键“CUS_LOC”对应的值的前3个字节，(4)名称的接下来3个字节将是与被声明为维度4的键“MONTH”对应的值的前3个字节，(5)名称的下一组字节将由维度的剩余字节的级联组成，(6)并且最后，在维度的所有字节用完之后，名称的剩余字节将从候选单元的其余字节的级联创建。

[0257] 由驱动该装置的应用提供的图式可以指定多个主维度以及多个第二维度。所有这些主维度和第二维度的信息都可以保留在叶节点数据结构的元数据中。主维度用于形成主轴，沿该主轴分类和组织滤筛中的单元。如果主维度耗尽且具有大量成员的子树仍然存在，则可以更深入树中使用第二维度，以将单元进一步细分为更小的组。有关第二维度的信息可以保留为元数据，也可以用作区分叶节点内单元的第二标准。在提供内容关联多维搜索和取回的一些实施例中，可以要求所有传入数据必须包含该图式所声明的每个维度的键和有效值。这允许系统确保只有有效数据进入滤筛中所需的子树。不包含被指定为维度的所有字段或者在与维度的字段对应的值中包含无效值的候选单元将被发送到如前面在图3E中图示的不同的子树中。

[0258] 数据蒸馏装置以一种附加的方式被约束，以基于维度中的内容来全面地支持数据的内容关联搜索和取回。当从基本数据单元创建导出单元时，导出器被约束为确保基本数据单元和导出项在每个相应维度的值字段中具有完全相同的内容。因此，在创建导出项时，不允许重建程序干扰或修改与基本数据单元的任何维度相对应的值字段中的内容，以构造导出项。给定候选单元，在查找滤筛期间，如果候选单元与目标基本数据单元的相应维度相比在任何维度中具有不同的内容，则需要安置新的基本数据单元，而不接受导出项。例如，如果主维度的子集将单元充分分类到树中的不同组中，使得候选单元到达叶节点以找到在主维度的这个子集中具有相同内容但在剩余主维度或第二维度中具有不同内容的基本数据单元，那么，需要安置新的基本数据单元而不是创建导出项。该功能确保通过简单地查询基本数据滤筛可以使用维度搜索所有数据。

[0259] 导出器可以采用各种实现技术来实行如下约束：候选单元和基本数据单元必须在每个对应维度的值字段中具有完全相同的内容。导出器可以从基本数据单元的骨架数据结构中提取包括与维度相对应的字段的位置、长度和内容的信息。类似地，该信息是从解析器/分解器接收的，或针对候选单元计算出的。接下来，可以比较候选单元和基本数据单元的维度的对应字段是否相等。一旦确认是相等的，导出器就可以继续进行其余导出。如果不相等，则候选单元将作为新的基本数据单元被安装在滤筛中。

[0260] 上述限制预计不会显著妨碍大多数使用模型的数据简化程度。例如，如果输入数据由一组单元组成，这组单元是数据仓储事务，每个数据仓储事务的大小为1000个字节，并

且如果由图式指定了一组6个主维度和14个第二维度，每个维度8个字节的数据，内容在维度上占用的总字节数为160个字节。在创建导出项时，这些160个字节不允许有扰动。这仍然将留出候选单元数据的剩余的840个字节可用于扰动以创建导出项，从而为利用冗余留下充足的机会，同时使得能够使用维度以内容关联的方式来搜索和取回来自数据仓储的数据。

[0261] 为了执行对包含维度中的字段的特定值的数据的搜索查询，装置可以遍历树并到达与指定的维度匹配的树中的节点，并且可以返回该节点下面的所有叶节点数据结构作为查找的结果。对存在叶节点处的基本数据单元的引用可用于在需要时获取所需的基本数据单元。如果需要，反向链接可以从蒸馏文件中取回输入单元(以无损简化格式)。该单元可以随后被重建以产生原始输入数据。因此，增强的装置允许对基本数据存储库(其是整个数据的较小子集)中的数据完成所有搜索，同时能够根据需要到达并取回所有的导出单元。

[0262] 增强的装置可以用于执行搜索和查找查询以基于查询所指定的维度中的内容来强大的搜索和取回数据的相关子集。内容关联数据取回查询将具有“获取(维度1，维度1的值；维度2，维度2的值；...)”的形式。查询将指定搜索中涉及的维度以及用于内容关联搜索和查找的每个指定维度的值。查询可以指定所有维度，也可以只指定维度的子集。查询可以指定基于多个维度的复合条件作为搜索和取回的标准。具有指定维度的指定值的滤筛中的所有数据都将被取回。

[0263] 可以支持多种获取查询并使其可用于正在使用该内容关联数据取回装置的分析应用。这些查询将通过应用的接口提供给装置。接口向装置提供来自应用的查询，并将查询结果从装置返回给应用。首先，可以使用查询FetchRefs来为与查询相匹配的每个基本数据单元获取对图13中的叶节点数据结构(连同子ID或条目的索引)的引用或处理。查询FetchMetaData的第二种形式可用于为匹配查询的每个基本数据单元从图13中的叶节点数据结构中的条目获取元数据(包括骨架数据结构、关于维度的信息和对基本数据单元的引用)。查询FetchPDEs的第三种形式将获取匹配搜索条件的所有基本数据单元。另一种形式的查询FetchDistilledElements将获取与搜索条件匹配的蒸馏文件中的所有单元。另一种形式的查询FetchElements将获取输入数据中与搜索条件匹配的所有单元。请注意，对于FetchElements查询，装置将首先获取蒸馏单元，然后将相关的蒸馏单元重建到来自输入数据的单元中，并将其作为查询的结果返回。

[0264] 除了这样的多维内容关联获取原语之外，接口还可以向应用提供直接访问基本数据单元(使用对基本数据单元的引用)和蒸馏文件中的单元(使用对单元的反向引用)的能力。另外，接口可以向应用提供重建蒸馏文件中的蒸馏单元的能力(给定对蒸馏单元的引用)，并且因其存在于输入数据中而传递该单元。

[0265] 这些查询的明智组合可以被分析应用用来执行搜索，确定相关的联合和交叉点，并且收集重要的见解。

[0266] 下面解释的图14B示出了具有在结构描述1402中描述的结构的输入数据集的示例。在该示例中，包含在文件1405中的输入数据包含电子商务交易。该输入数据由数据蒸馏装置中的分析器使用图14A中的图式和维度声明转换成候选单元序列1406。请注意，每个候选单元的名称的前导字节是如何由维度的内容组成的。例如，候选单元1的名称1407的前导字节是PRINRACQNYCFEB。这些名称用于以树形式组织候选单元。数据简化完成后，蒸馏数据

被放置在蒸馏文件1408中。

[0267] 以下解释的图14C示出了如何使用维度映射描述1404来根据结构描述1402解析图14A中所示的输入数据集，根据维度映射描述1404来确定维度，并且基于所确定的维度在树中组织基本数据单元。在图14C中，使用来自4个维度的总共14个字符将基本数据单元组织在主树中。主树中示出的是各种基本数据单元的叶节点数据结构的一部分。注意，为了容易查看的目的，未示出图13的完整的叶节点数据结构。但是，图14C示出了叶节点数据结构中的每个条目的路径信息或名称，子ID，从基本数据单元到蒸馏文件中的单元的所有反向引用或反向链接，以及指示蒸馏文件中的单元是否是“基本(或素数:prime)“(用P表示)或“导出项”(用D表示)的指示符以及对基本数据单元的引用。图14C示出了映射到主树中的5个基本数据单元的蒸馏文件中的7个单元。在图14C中，具有名称PRINRACQNYCFEB的用于基本数据单元的反向链接A返回到蒸馏文件中的单元1。同时，名为NIKESHOELAHJUN的基本数据单元具有分别到单元2、单元3和单元58的3个反向链接B、C和E。请注意，单元3和单元58是单元2的导出项。

[0268] 图14D示出了从维度创建的辅助索引或辅助树，以提高搜索的效率。在此示例中，创建的辅助映射树基于维度2(即CATEGORY)。通过直接遍历该辅助树，可以找到输入数据中给定CATEGORY的所有单元，而不需要否则可能会发生的更昂贵的主树遍历。例如，向下穿过用“SHOE”表示的腿直接导向鞋的两个基本数据单元ADDIKHOESJCSEP和NIKESHOELAHJUN。

[0269] 或者，这样的辅助树可以基于第二维度并且用于利用维度来帮助搜索的快速收敛。

[0270] 现在将提供在图14D中所示的装置上执行的查询的示例。查询FetchPDEs(维度1，NIKE;)将返回两个名为NIKESHOELAHJUN和NIKEJERSLAHOCT的基本数据单元。查询FetchDistilledElements(维度1,NIKE;)将返回单元2、单元3、单元58和单元59，其为无损简化格式的蒸馏单元。查询FetchElements(维度1,NIKE;维度2,SHOE)将从输入数据文件1405返回交易2、交易3和交易58。查询FetchMetadata(维度2,SHOES)将为名为ADIDSHOESJCSEP和NIKESHOELAHJUN的两个基本数据单元中的每一个返回存储在叶节点数据结构条目中的元数据。

[0271] 到目前为止所描述的装置可以用于支持基于在称为维度的字段中指定的内容的搜索。另外，该装置可以用于支持基于未包括在维度列表中的关键字列表的搜索。这样的关键字可以由诸如驱动该装置的搜索引擎之类的应用提供给该装置。关键字可以通过图式声明指定给装置，或者通过包含所有关键字的关键字列表传递，每个关键字由已声明的分隔符(如空格、逗号或换行)分隔。或者，可以使用图式以及关键字列表来共同指定所有关键字。可以指定非常多的关键字-装置对关键字的数量没有限制。这些搜索关键字将被称为Keywords(关键字)。该装置可以使用这些关键字来保持搜索的倒置索引。对于每个关键字，倒置索引包含对包含此关键字的蒸馏文件中的单元的反向引用列表。

[0272] 基于图式或关键字列表中的关键字声明，蒸馏装置的解析器可以解析候选单元的内容，以检测和定位传入候选单元中的各个关键字(是否找到以及在哪儿找到)。随后，通过数据蒸馏装置将候选单元转换为基本数据单元或导出单元，并将其作为单元置于蒸馏文件中。在此单元中找到的关键字的倒置索引可以通过对蒸馏文件中的单元的反向引用来更新。对于在单元中找到的每个关键字，倒置索引被更新以包括对蒸馏文件中的该单元的反

向引用。回想一下，蒸馏文件中的单元为无损简化表示。

[0273] 在使用关键字对数据进行搜索查询之后，查询倒置索引以找到并提取对包含该关键字的蒸馏文件中的单元的反向引用。使用对这样的单元的反向引用，可以取回对该单元的无损简化表示，并且可以重建该单元。然后可以提供重建的单元作为搜索查询的结果。

[0274] 可以增强倒置索引以包含定位关键字在重建单元中的偏移量的信息。请注意，候选单元中检测到的每个关键字的偏移量或位置可由解析器确定，因此，当将对蒸馏文件中的单元的反向引用置入倒置索引中时，也可将此信息记录在倒置索引中。在搜索查询时，在查询倒置索引以取回对包含相关关键字的蒸馏文件中的单元的反向引用之后，并且在该单元被重建之后，记录的关键字在重建单元中的偏移量或位置(与原始输入候选单元相同)可用于确定输入数据或输入文件中关键字的存在位置。

[0275] 图15示出了便于根据关键字进行搜索的倒置索引。对于每个关键字，倒置索引包含成对的值-第一个是对包含关键字的蒸馏文件中的无损简化单元的反向引用，第二个值是关键字在重建单元中的偏移量。

[0276] 维度和关键字对数据蒸馏装置中的基本数据滤筛具有不同的含义。请注意，维度用作在滤筛中沿其组织基本数据单元的主轴。维度形成数据中每个单元的骨架数据结构。维度是根据传入数据结构的知识来声明的。导出器受到限制，使得创建的任何导出单元必须与每个相应维度的字段的值中的基本数据单元具有完全相同的内容。

[0277] 不需要为关键字保持这些属性。在一些实施例中，既不存在关键字甚至存在于数据中的先验要求，也不是要求基于关键字组织基本数据滤筛，也不关于涉及包含关键字的导出项约束导出器。如果需要，导出器可以通过修改关键字的值来从基本数据单元自由创建导出项。关键字的位置仅记录在扫描输入数据时发现的地方，并且倒置索引被更新。在基于关键词的内容关联搜索时，查询倒置索引并获得关键词的所有位置。

[0278] 在其它实施例中，不需要关键字存在于数据中(数据中不存在关键字不会使数据无效)，但是要求基本数据滤筛包含所有包含关键字的单元，并且导出器在涉及包含关键字的内容的导出方面受到约束-除简化重复项外，不允许任何导出。这些实施例的目的在于，包含任何关键字的所有不同单元必须存在于基本数据滤筛中。这是其中控制基本数据的选择的规则以关键字为条件的示例。在这些实施例中，可以创建修改的倒置索引，该倒置索引针对每个关键字包含对每个包含该关键字的基本数据单元的反向引用。在这些实施例中，实现了强大的基于关键字的搜索能力，其中仅搜索基本数据滤筛与搜索整个数据一样有效。

[0279] 可能存在其它实施例，其中导出器受到约束，使得不允许重建程序干扰或修改基本数据单元中找到的任何关键字的内容，以便将候选单元制定为该基本数据单元的导出单元。关键字需要从基本数据单元不变地传播到导出项。如果导出器需要修改在基本数据单元中找到的任何关键字的字节，以便成功地将候选项制定为该基本数据单元的导出项，那么导出项可以不被接受，并且该候选项必须作为新的基本数据单元被安装在滤筛中。

[0280] 关于涉及关键字的导出项，可以以各种方式来约束导出器，使得管理基本数据的选择的规则以关键字为条件。

[0281] 使用关键字搜索数据的装置可以接受对关键字的列表的更新。装置使得能够对关键字列表更新。可以添加关键字而不会对以无损简化形式存储的数据进行任何更改。当添

加新的关键字时，可以针对更新的关键字列表对新的输入数据进行解析，并且随着输入数据更新的倒置索引随后以无损简化形式被存储。如果现有数据(已经以无损简化形式存储)需要针对新的关键字进行索引，则装置可逐渐地读取蒸馏文件(一次一个或多个蒸馏文件，或者一次一个无损简化数据批次)、重建原始文件(但不会干扰无损简化存储的数据)，并解析重建的文件以更新倒置索引。在这期间(all this while)，整个数据存储库可以继续保持以无损简化形式存储。

[0282] 图16A图示了图14A中所示的图式的变型的图式声明。图16A中的图式包括第二维度1609的声明和关键字列表1610。图16B示出了具有在结构描述1602中描述的结构的输入数据集1611的示例，其被解析并且被转换成具有基于所声明的主维度的名称的一组候选单元。候选单元被转换成蒸馏文件1613中的单元。第二维度“PROD_ID”的声明在导出器上设置约束，使得可能不从基本数据单元“具有PROD_ID＝348的NIKESHOELAHJUN”导出候选单元58，并且因此在基本数据滤筛中创建另外一个基本数据单元“具有PROD_ID＝349的NIKESHOELAHJUN”。尽管输入数据集与图14B中所示的输入数据集相同，但是蒸馏的结果是产生7个蒸馏单元，但产生6个基本数据单元。图16C示出了作为蒸馏过程的结果创建的蒸馏文件、主树和基本数据单元。

[0283] 图16D示出了为第二维度“PROD_ID”创建的辅助树。使用特定的PROD_ID值遍历此树会导致具有该特定PROD_ID的基本数据单元。例如，查询FetchPDEs(维度5,251)或者请求PROD_ID＝251的基本数据单元的查询FetchPDEs(PROD_ID,251)产生基本数据单元WILSBALLLAHNOV。

[0284] 图16E示出了针对在图16A结构1610中声明的3个关键字创建的倒置索引(标记为关键字的倒置索引1631)。这些关键字是FEDERER、LAVER和SHARAPOVA。在解析和消费输入数据集1611之后更新倒置索引。查询FetchDistilledElements(关键字,Federer)将利用倒置索引(而不是主树或辅助树)来返回单元2、单元3和单元58。

[0285] 图17示出了针对内容关联数据取回增强的整个装置的框图。内容关联数据取回引擎1701向数据蒸馏装置提供图式1704或包括数据的维度的结构定义。其还向装置提供关键字列表1705。其发出查询1702用于从蒸馏装置搜索和取回数据，并接收查询的结果作为结果1703。导出器110被增强以知晓维度的声明从而在创建导出项时禁止在维度的位置修改内容。注意，从叶节点数据结构中的条目到蒸馏文件中的单元的反向引用被存储在基本数据滤筛106中的叶节点数据结构中。同样，辅助索引也被存储在基本数据滤筛106中。还示出了当单元被写入蒸馏数据时，由导出器110通过反向引用1709更新的倒置索引1707。该内容关联数据取回引擎与其他应用(例如分析，数据仓储和数据分析应用)进行交互，向他们提供执行的查询的结果。

[0286] 总而言之，增强的数据蒸馏装置能够对以无损简化形式存储的数据进行强大的多维内容关联搜索和取回。

[0287] Data Distillation™装置可用于音频和视频数据的无损简化的用途。利用该方法完成的数据简化是通过从驻留在内容关联滤筛中的基本数据单元中导出音频和视频数据的分量实现的。下面说明所述方法对于这类用途的应用。

[0288] 图18A-18B表示用于按照MPEG 1，层3标准(也被称为MP3)的音频数据的压缩和解压缩的编码器和解码器的方框图。MP3是利用有损和无损数据简化技术的组合来压缩传入

音频的数字音频的音频编码格式。它设法把紧致盘(CD)音频从1.4Mbps压缩到128Kbps。MP3利用人耳的局限性来抑制大多数人的人耳无法感知的音频的分量。为了实现这一点，采用统称为感知编码技术的一组技术，感知编码技术有损但察觉不到地减小一段音频数据的大小。感知编码技术是有损的，并且在这些步骤期间丢失的信息无法恢复。这些感知编码技术由霍夫曼编码补充，霍夫曼编码是本文中前面描述的无损数据简化技术。

[0289] 在MP3中，传入的音频流被压缩成一系列的几个较小的数据帧，每个数据帧包含帧头和压缩音频数据。原始音频流被定期采样，以产生一系列的音频片段，随后采用感知编码和霍夫曼编码压缩所述一系列的音频片段，从而产生一系列的MP3数据帧。感知编码和霍夫曼编码技术都是在音频数据的每个片段中局部应用的。霍夫曼编码技术在音频片段内局部地，但不横跨音频流全局地利用冗余。从而，MP3技术既不横跨单个音频流，又不在多个音频流之间全局地利用冗余。这代表超越MP3所能达到的进一步数据简化的机会。

[0290] 每个MP3数据帧表示26ms的音频片段。每个帧保存1152个样本，并被细分成均包含576个样本的2个颗粒(granule)。在图18A中的编码器方框图中，可以看出在数字音频信号的编码期间，通过滤波处理，并通过应用改进的离散余弦变换(MDCT)，获得时域样本，并变换成576个频域样本。应用感知编码技术，以减少包含在样本中的信息的量。感知编码的输出是非均匀量化颗粒1810，非均匀量化颗粒1810每个频率线包含减少的信息。随后利用霍夫曼编码进一步减小颗粒的大小。每个颗粒的576个频率线可把多个霍夫曼表用于它们的编码。霍夫曼编码的输出是包含缩放因子、霍夫曼编码比特和辅助数据的帧的主要数据分量。(用于表征和定位各个字段的)边信息被放入MP3头部中。编码的输出是MP3编码音频信号。在128Kbps的比特率下，MP3帧的大小为417或418字节。

[0291] 图18C表示如何增强首次示于图1A中的数据蒸馏装置，以对MP3数据进行数据简化。图18C中图解所示的方法把MP3数据因式分解成候选单元，并按比单元本身更细的粒度利用单元之间的冗余。对于MP3数据，选择颗粒作为单元。在一个实施例中，非均匀量化颗粒1810(如图18A中所示)可被视为单元。在另一个实施例中，单元可由量化频率线1854和缩放因子1855的串联构成。

[0292] 在图18C中，MP3编码数据流1862由数据蒸馏装置1863接收，并被简化成以无损简化形式保存的蒸馏MP3数据流1868。传入的MP3编码数据流1862由一系列的多对MP3头部和MP3数据构成。MP3数据包括CRC、边信息、主要数据和辅助数据。传出的由该装置创建的蒸馏MP3数据由相似的一系列的多对构成(每一对是DistMP3头部，和后面的无损简化格式的单元规范)。DistMP3头部包含原始帧的除主要数据以外的所有分量，即，它包含MP3头部、CRC、边信息和辅助数据。该蒸馏MP3数据中的单元字段包含以无损简化形式指定的颗粒。解析器/因式分解器1864进行传入的MP3编码流的第一次解码(包括进行霍夫曼解码)，以提取量化频率线1851和缩放因子1852(示于图18B中)，和生成作为候选单元的音频颗粒1865。解析器/因式分解器进行的第一次解码步骤与图18B的同步和错误检验1851、霍夫曼解码1852、及缩放因子解码1853的步骤相同-这些步骤是在任何标准MP3解码器中进行的，并且在现有技术中是公知的。基本数据滤筛1866包含被组织，以便按照内容关联方式存取的作为基本数据单元的颗粒。在把颗粒安装在基本数据滤筛中期间，颗粒的内容被用于确定该颗粒应被安装在滤筛中的什么地方，和更新滤筛的适当叶节点中的骨架数据结构和元数据。随后，颗粒被霍夫曼编码并被压缩，以致颗粒可以不比当驻留在MP3数据中时，它所占据的足迹更

大的足迹，被保存在滤筛中。每当导出器需要滤筛中的颗粒作为基本数据单元时，在被提供给导出器之前，该颗粒被解压缩。利用数据蒸馏装置，传入的音频颗粒由导出器1870从驻留在滤筛中的基本数据单元(所述基本数据单元也是音频颗粒)导出，从而创建颗粒的无损简化表示或者蒸馏表示，并将其放入蒸馏的MP3数据1868中。颗粒的这种蒸馏表示被放入替换最初存在于MP3帧的主要数据字段中的霍夫曼编码信息的单元字段中。利用图1H中所示的格式，编码每个单元或颗粒的蒸馏表示-蒸馏数据中的每个单元或者是基本数据单元(伴随有对滤筛中的基本数据单元或基本颗粒的引用)，或者是导出单元(伴随有对滤筛中的基本数据单元或基本颗粒的引用，加上从所引用的基本数据单元，生成导出单元的重建程序)。在导出步骤期间，用于接受导出项的阈值可被设定为驻留在正被简化的帧的主要数据字段中的原始霍夫曼编码信息的大小的一部分。从而，除非重建程序与对基本数据单元的引用之和小于(包含霍夫曼编码数据的)MP3编码帧的对应主要数据字段的大小的所述一部分，否则，导出项不会被接受。如果重建程序与对基本数据单元的引用之和小于(包含霍夫曼编码数据)的编码MP3帧的现有主要数据字段的大小的所述一部分，那么可以决定接受导出项。

[0293] 上述方法使跨越保存在装置中的多个音频颗粒，在全局范围利用冗余成为可能。MP3编码数据文件可被变换成蒸馏MP3数据，并以无损简化形式保存。当需要被取回时，可调用(采用取回器1871和重建器1872的)数据取回处理，以重建MP3编码数据1873。在图18C中所示的装置中，重建器负责执行重建程序，以生成期望的颗粒。此外，它被增强，以进行为生成MP3编码数据所需的霍夫曼编码步骤(被表示成图18A中的霍夫曼编码1811)。该数据随后可被馈送给标准MP3解码器，以播放音频。

[0294] 按照这种方式，可以修改和采用数据蒸馏装置，以进一步减小MP3音频文件的大小。

[0295] 在所述方案的另一种变型中，当收到MP3编码流时，解析器/因式分解器把整个主要数据字段视为导出用候选单元，或者视为用于安装到基本数据滤筛中的基本数据单元。在这种变型中，所有单元将继续保持霍夫曼编码状态，并且重建程序将作用于已被霍夫曼编码的单元。也可采用数据蒸馏装置的这种变型，以进一步减小MP3音频文件的大小。

[0296] 以与前述各部分中描述的和在图18A-18C中图示的方式相似的方式，可以采用Data Distillation™装置用于无损简化视频数据的目的。通过从驻留在内容关联滤筛中的基本数据单元导出视频数据的分量来实现通过该方法完成的数据简化。视频数据流包含音频和移动图片分量。已经描述了用于蒸馏音频分量的方法。现在将解决移动图片分量。移动图片分量通常被组织为一系列图片组。图片组以I帧开始，并且通常后面跟着多个预测帧(称为P帧和B帧)。I帧通常较大并且包含图片的完整快照，而预测帧是在采用诸如相对于I帧或相对于其它导出帧的运动估计之类的技术之后导出的。Data Distillation™装置的一些实施例从视频数据中提取I帧作为单元，并对它们执行数据蒸馏过程，从而保留某些I帧作为驻留在内容关联滤筛中的基本数据单元，而其余的I帧是从基本数据单元导出的。所描述的方法使得能够在跨视频文件内的多个I帧和跨多个视频文件的全局范围内利用冗余。由于I帧通常是移动图片数据的庞大分量，因此该方法将使得减少移动图片分量的足迹。将蒸馏技术应用于音频分量以及移动图片分量将有助于无损简化视频数据的整体大小。

[0297] 图19示出了首次示于图1A中的数据蒸馏装置如何可以被增强以对视频数据进行

数据简化。视频数据流1902由数据蒸馏装置1903接收，并被简化为以无损简化形式存储的蒸馏视频数据流1908。传入的视频数据流1902包括两个分量-压缩的移动图片数据和压缩的音频数据。由该装置创建的传出的蒸馏视频数据也包括两个分量，即，压缩的移动图片数据和压缩的音频数据；但是，这些分量的尺寸通过数据蒸馏装置1903被进一步减小。解析器/分解器1904从视频数据流1902中提取压缩的移动图片数据和压缩的音频数据，并从压缩的移动图片数据中提取(包括执行任何所需的霍夫曼解码)内帧(I帧)和预测帧。I帧被用作候选单元1905以在基本数据滤筛1906中执行内容关联的查找。导出器1910使用基本数据滤筛1906返回的基本数据单元集合(也就是I帧)来生成I帧的无损简化表示或蒸馏表示，并且无损简化的I帧被放置在蒸馏视频数据1908中。使用图1H中所示的格式对蒸馏表示进行编码-蒸馏数据中的每个单元要么是基本数据单元(伴随着对滤筛中的基本数据单元的引用)，要么是导出单元(伴随着对滤筛中的基本数据单元的引用以及从引用的基本数据单元生成导出单元的重建程序)。在导出步骤期间，可以将用于接受导出的阈值设置为原始I帧的大小的比例(fraction)。因此，除非重建程序和对基本数据单元的引用的总和小于对应I帧的大小的这一比例，否则将不接受导出。如果重建程序和对基本数据单元的引用的总和小于原始I帧的大小的这一比例，则可以做出接受导出的决定。

[0298] 上述方法使得能够跨装置中存储的多个视频数据集的多个I帧在全局范围内利用冗余。当需要取回时，可以调用数据取回过程(采用取回器1911和重建器1912)来重建视频数据1913。在图19中所示的装置中，重建器负责执行重建程序以生成期望的I帧。它还被增强以将压缩的音频数据与压缩的移动图片数据组合在一起(本质上是由解析器&分解器1904执行的提取操作的逆操作)来生成视频数据1913。然后，该数据可以被馈送到标准视频解码器中以播放视频。

[0299] 以这种方式，数据蒸馏装置可以适于和用于进一步减小视频文件的大小。

[0300] 给出前面的描述是为了使得本领域技术人员能够制作和使用所述实施例。在不背离本公开内容的精神和范围的情况下，本领域技术人员将很容易认识到针对所公开的实施例的各种修改，并且这里所定义的一般原理适用于其他实施例和应用。因此，本发明不限于所示出的实施例，而是应被给予与这里所公开的原理和特征一致的最广范围。

[0301] 在本公开内容中描述的数据结构和代码可以部分地或完全地被存储在计算机可读存储介质和/或硬件模块和/或硬件装置上。计算机可读存储介质包括而不限于易失性存储器、非易失性存储器、磁性和光学存储设备(比如盘驱动器、磁带、CD(紧致盘)、DVD(数字通用盘或数字视频盘))或者能够存储代码和/或数据的现在已知或后来开发的其他介质。在本公开内容中描述的硬件模块或装置包括而不限于专用集成电路(ASIC)、现场可编程门阵列(FPGA)、专用或共享处理器以及/或者现在已知或后来开发的其他硬件模块或装置。

[0302] 在本公开内容中描述的方法和处理可以部分地或完全地被具体实现为存储在计算机可读存储介质或设备中的代码和/或数据，从而当计算机系统读取并执行所述代码和/或数据时，所述计算机系统实施相关联的方法和处理。所述方法和处理还可以部分地或完全地被具体实现在硬件模块或装置中，从而当所述硬件模块或装置被激活时，其实施相关联的方法和处理。应当提到的是，所述方法或处理可以使用代码、数据以及硬件模块或装置的组合来具体实现。

[0303] 前面关于本发明的实施例的描述仅仅是出于说明和描述的目的而给出的。所述描

述不意图进行穷举或者把本发明限制到所公开的形式。因此，本领域技术人员将会想到许多修改和变型。此外，前面的公开内容不意图限制本发明。

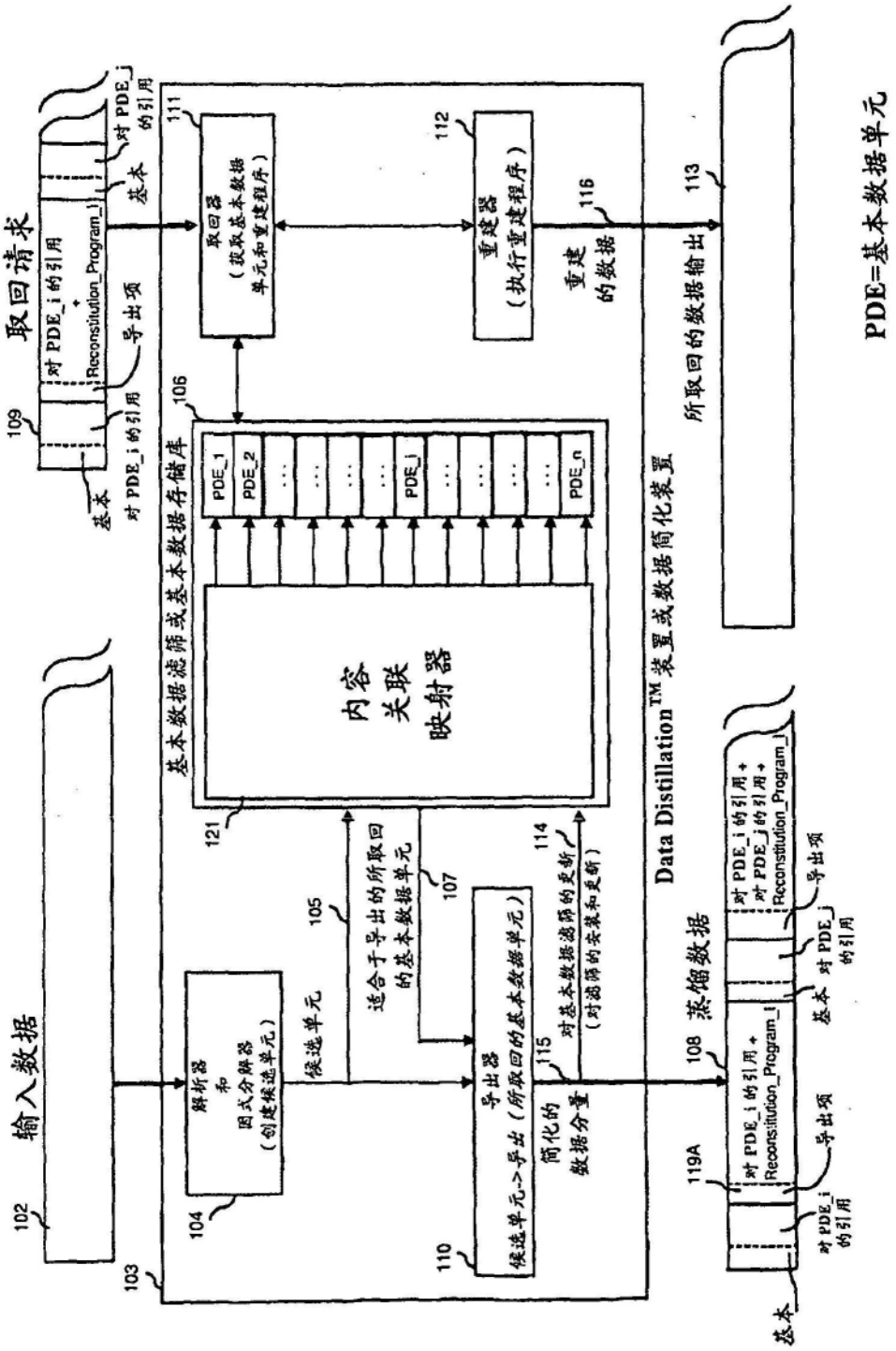

取回请求
对 PDE_i 的引用 + Reconstitution_Program_I
基本
导出项
对 PDE_j 的引用
对 PDE_i 的引用
基本
109
106
基本数据滤筛或基本数据存储库
PDE_1
PDE_2
PDE_i
PDE_n
121
内容关联映射器
111
取回器（获取基本数据单元和重建程序）
112
重建器（执行重建程序）
116
重建的数据
113
所取回的数据输出
PDE=基本数据单元
输入数据
102
103
解析器和因式分解器（创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据滤筛的更新（对滤筛的安装和更新）
114
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
对 PDE_i 的引用 + Reconstitution_Program_I
导出项
基本
对 PDE_j 的引用
119A
对 PDE_i 的引用 + 对 PDE_j 的引用 + Reconstitution_Program_I
导出项
对 PDE_j 的引用
基本
对 PDE_i 的引用

图1A

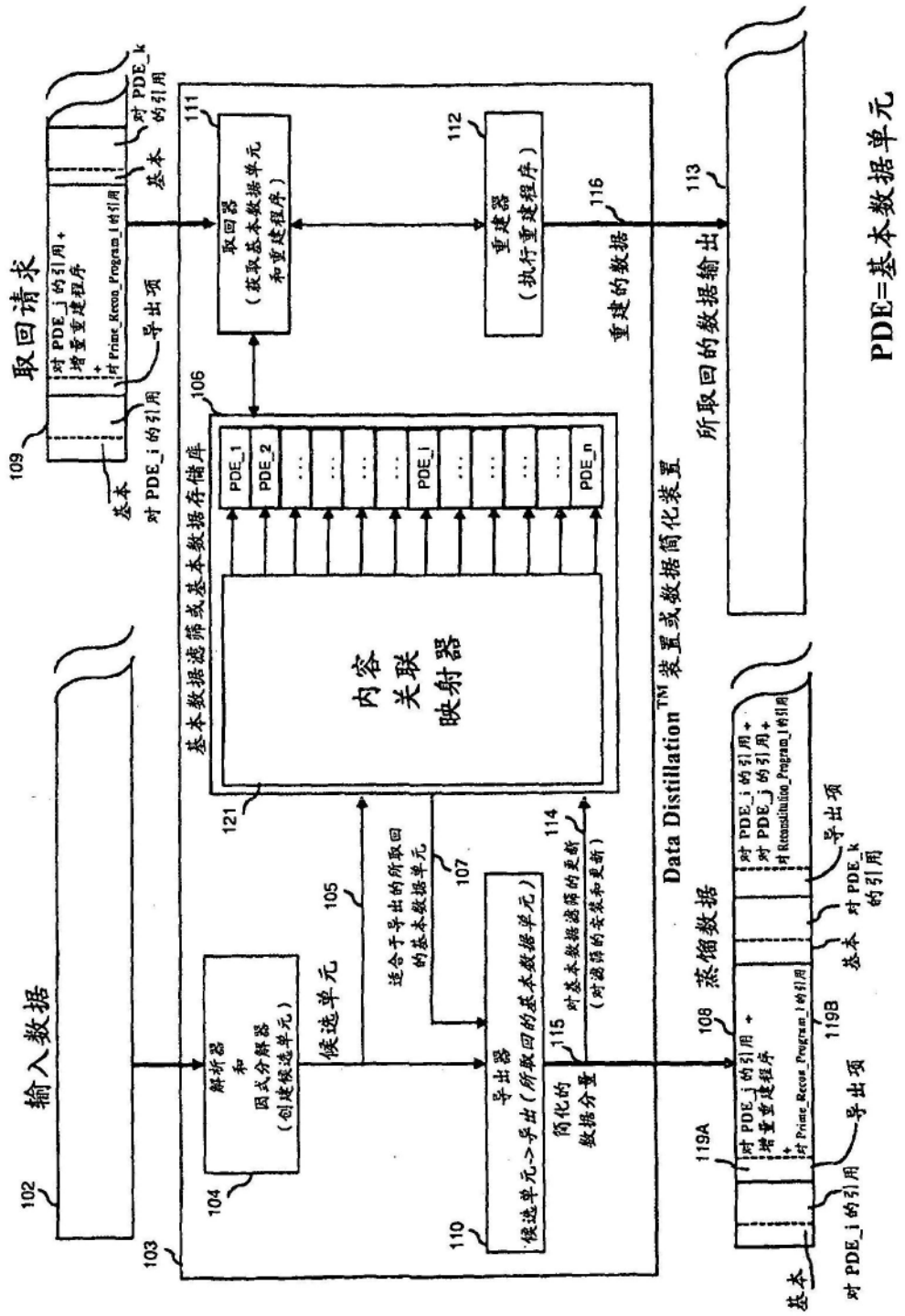

取回请求
109
对 PDE_j 的引用+
增量重建程序
+
对 Prime_Recon_Program_1 的引用
导出项
对 PDE_i 的引用
基本
对 PDE_k 的引用
基本
111
取回器
（获取基本数据单元和重建程序）
112
重建器
（执行重建程序）
116
113
重建的数据
所取回的数据输出
106
基本数据选择或基本数据存储库
PDE_1
PDE_2
PDE_i
PDE_n
内容关联映射器
121
输入数据
102
103
解析器和因式分解器（创建候选单元）
104
候选单元
105
连合于导出的所取回的基本数据单元
107
导出器
（候选单元->导出（所取回的基本数据单元））
110
简化的数据分量
115
对基本数据选择的更新
（对选择的安装和更新）
114
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
119A
对 PDE_j 的引用 +
增量重建程序
+
对 Prime_Recon_Program_1 的引用
导出项
对 PDE_i 的引用
基本
119B
对 PDE_j 的引用+
对 PDE_j 的引用+
对 Reconstitution_Program_1 的引用
导出项
对 PDE_k 的引用
基本
对 PDE_i 的引用
PDE=基本数据单元

图1B

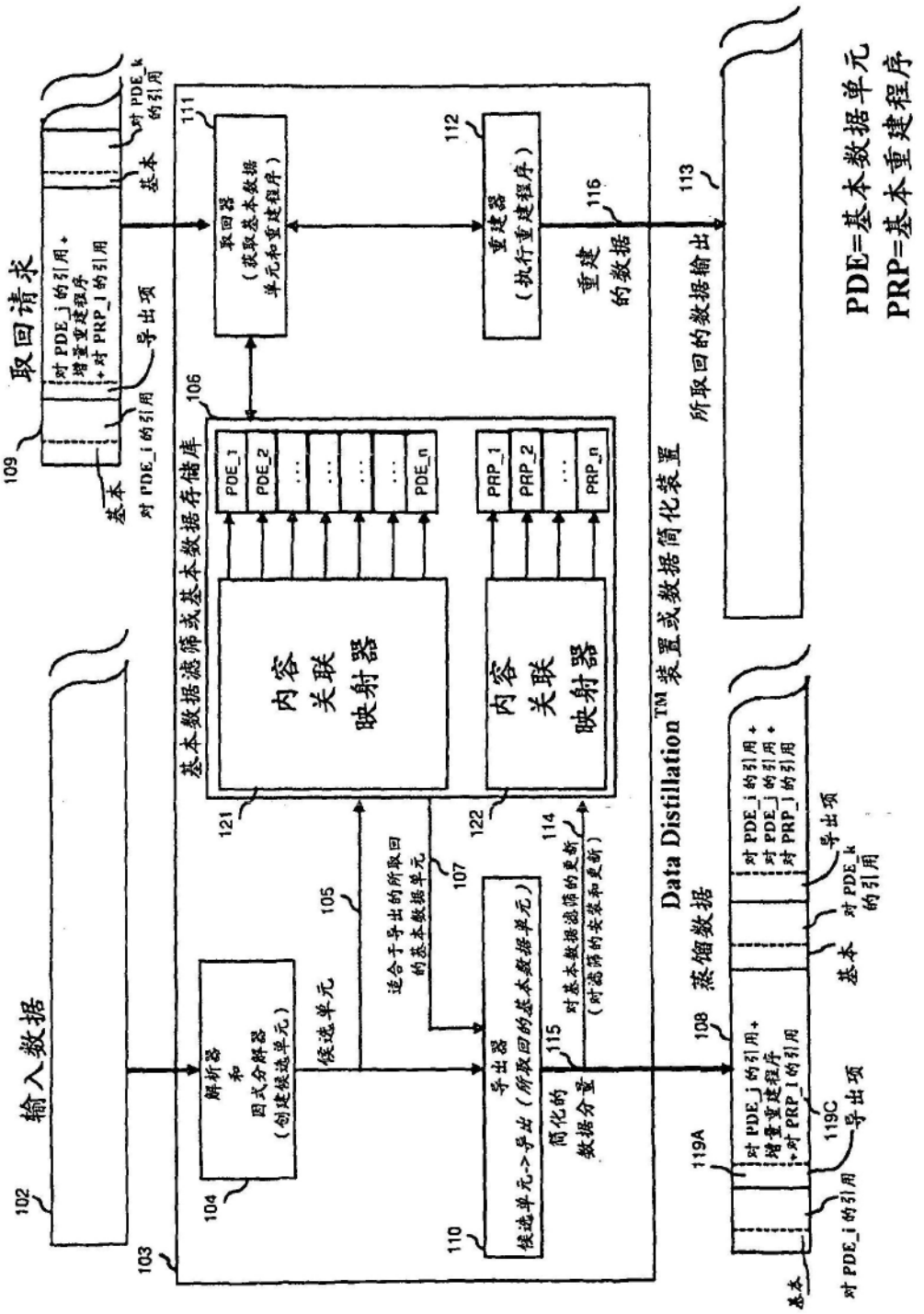

取回请求
对 PDE_j 的引用 + 增量重建程序 + 对 PRP_j 的引用
导出项
基本
对 PDE_i 的引用
对 PDE_k 的引用
109
输入数据
102
103
解析器 和 因式分解器 （创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器 候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据滤筛的更新 （对滤筛的安装和更新）
114
基本数据滤筛或基本数据存储库
106
121
内容关联映射器
PDE_1
PDE_2
PDE_n
122
内容关联映射器
PRP_1
PRP_2
PRP_n
取回器 （获取基本数据单元和重建程序）
111
重建器 （执行重建程序）
112
116
重建的数据
113
所取回的数据输出
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
对 PDE_j 的引用 + 增量重建程序 + 对 PRP_j 的引用
119A
119C
导出项
对 PDE_k 的引用
PDE=基本数据单元
PRP=基本重建程序

图1C

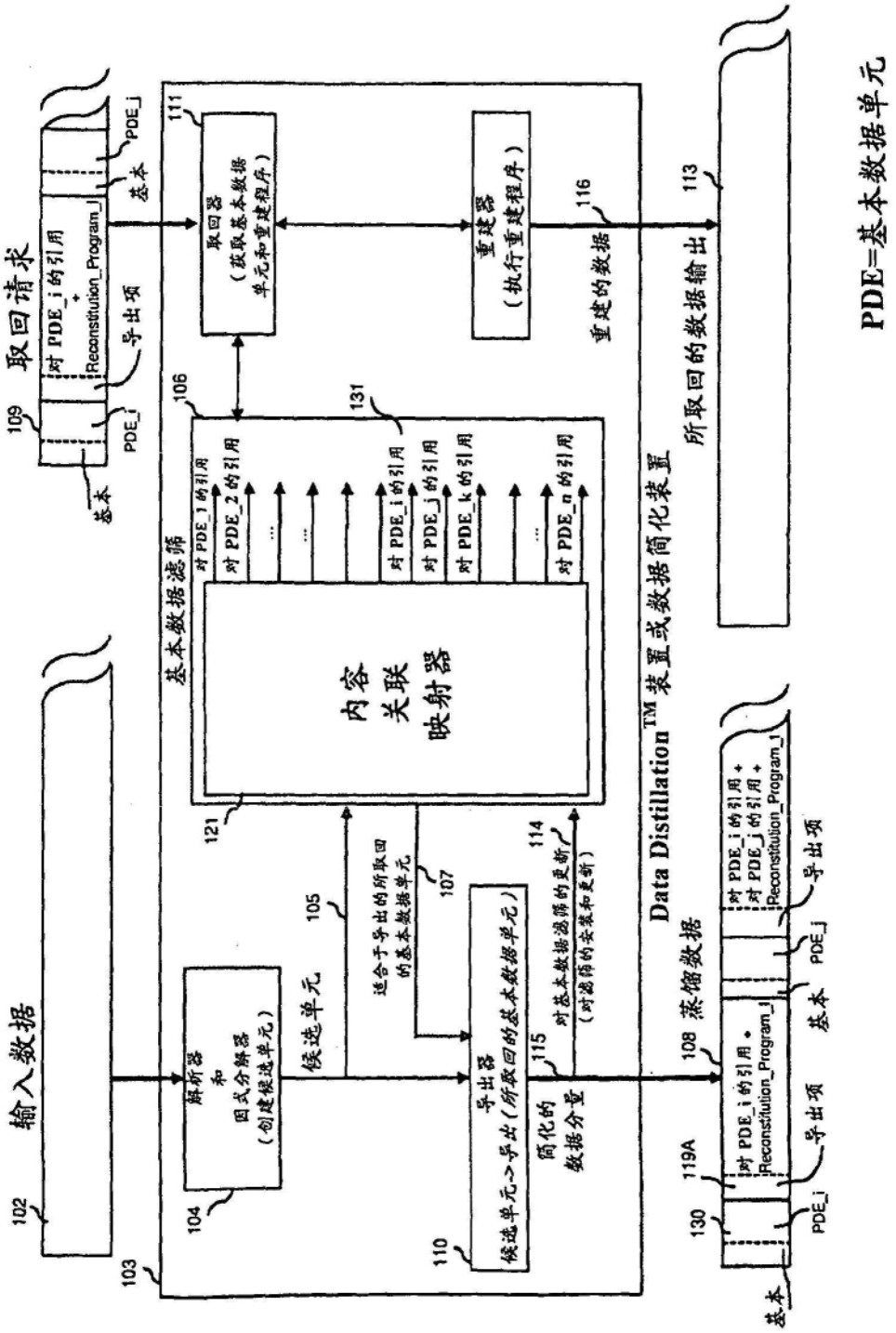

取回请求
109
对 PDE_i 的引用 + Reconstitution_Program_j
导出项
PDE_j
基本
PDE_j
基本
111
取回器
（获取基本数据单元和重建程序）
重建器
（执行重建程序）
116
重建的数据
113
所取回的数据输出
输入数据
102
103
基本数据滤筛
106
对 PDE_1 的引用
对 PDE_2 的引用
对 PDE_i 的引用
对 PDE_j 的引用
对 PDE_k 的引用
对 PDE_n 的引用
131
内容关联映射器
121
解析器和因式分解器（创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据滤筛的更新（对滤筛的安装和更新）
114
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
对 PDE_i 的引用 + 对 PDE_j 的引用 + Reconstitution_Program_j
导出项
PDE_j
基本
对 PDE_i 的引用 + Reconstitution_Program_j
导出项
PDE_j
基本
119A
130
PDE=基本数据单元

图1D

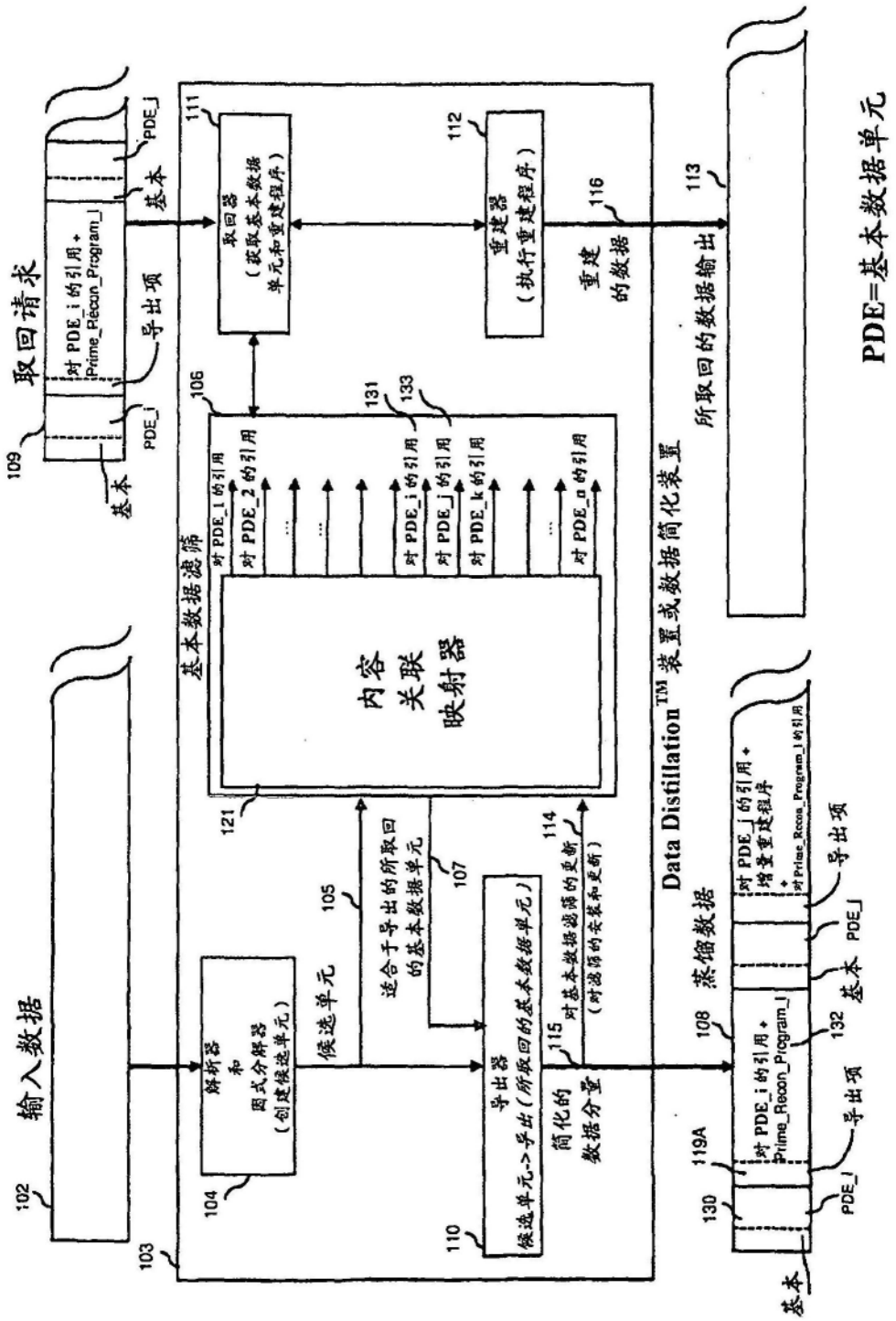

取回请求
109
对 PDE_i 的引用 + Prime_Recon_Program_I
PDE_I
基本
导出项
PDE_J
基本
111
取回器
（获取基本数据单元和重建程序）
112
重建器
（执行重建程序）
116
重建的数据
113
所取回的数据输出
106
131
133
基本数据滤筛
对 PDE_1 的引用
对 PDE_2 的引用
对 PDE_i 的引用
对 PDE_j 的引用
对 PDE_k 的引用
对 PDE_n 的引用
内容关联映射器
121
输入数据
102
103
解析器和图式分解器（创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据滤筛的更新（对滤筛的安装和更新）
114
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
对 PDE_j 的引用 + 增量重建程序 + 对 Prime_Recon_Program_I 的引用
导出项
PDE_J
基本
132
对 PDE_i 的引用 + Prime_Recon_Program_I
导出项
PDE_I
119A
130
基本
PDE=基本数据单元

图1E

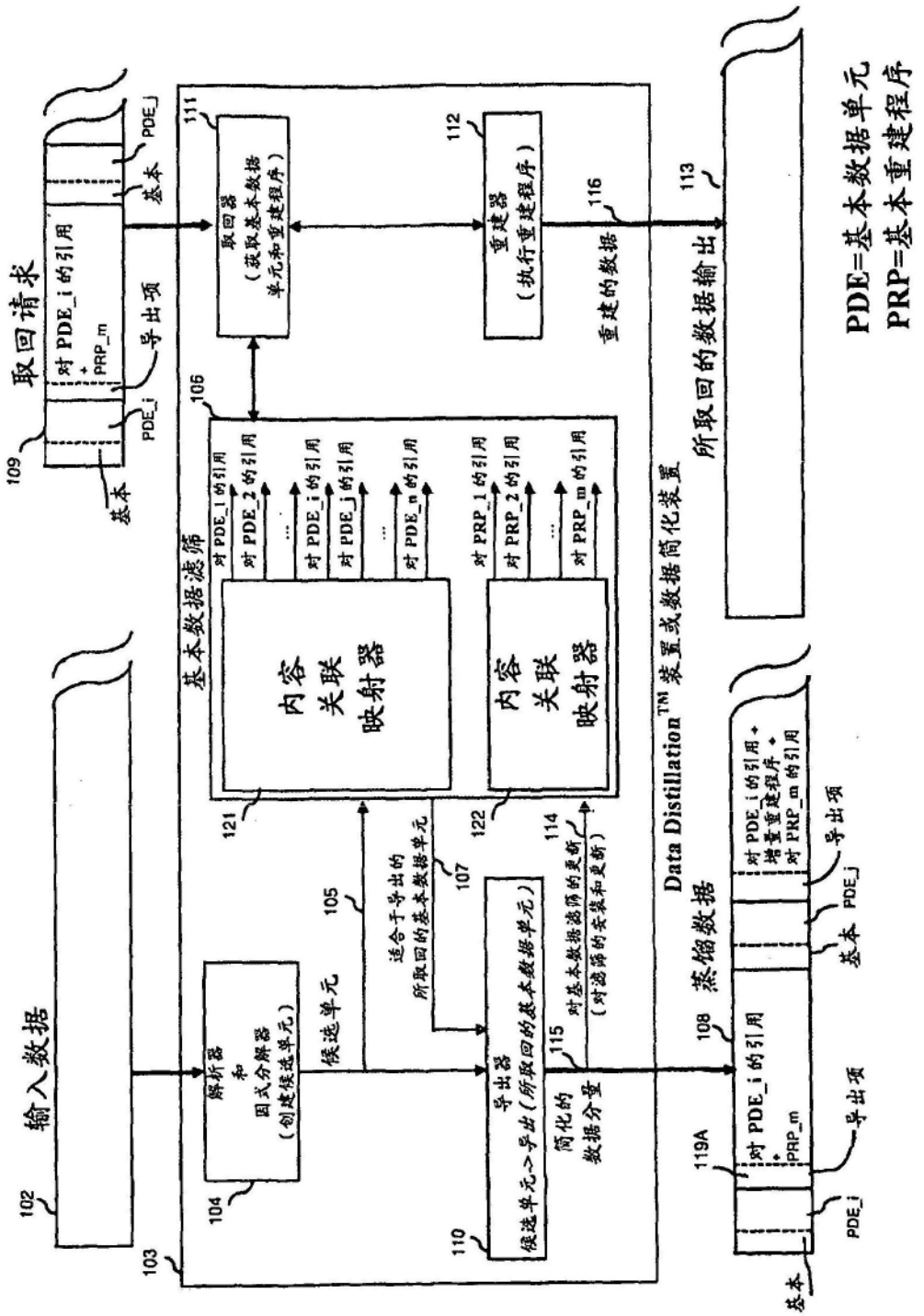

取回请求
对 PDE_i 的引用
+
PRP_m
基本
导出项
PDE_j
PDE_i
109
基本
导出项
基本数据滤筛
106
对 PDE_1 的引用
对 PDE_2 的引用
对 PDE_i 的引用
对 PDE_j 的引用
对 PDE_n 的引用
对 PRP_1 的引用
对 PRP_2 的引用
对 PRP_m 的引用
内容
关联
映射器
内容
关联
映射器
121
122
111
取回器
（获取基本数据
单元和重建程序）
112
重建器
（执行重建程序）
116
重建的数据
113
所取回的数据输出
输入数据
102
103
解析器
和
固式分解器
（创建候选单元）
104
候选单元
105
适合于导出的
所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的
数据分量
115
对基本数据滤筛的更新
（对滤筛的采纳和更新）
114
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
119A
对 PDE_i 的引用
+
PRP_m
导出项
PDE_i
基本
对 PDE_i 的引用 +
增量重建程序 +
对 PRP_m 的引用
导出项
PDE_j
基本
PDE=基本数据单元
PRP=基本重建程序

图1F

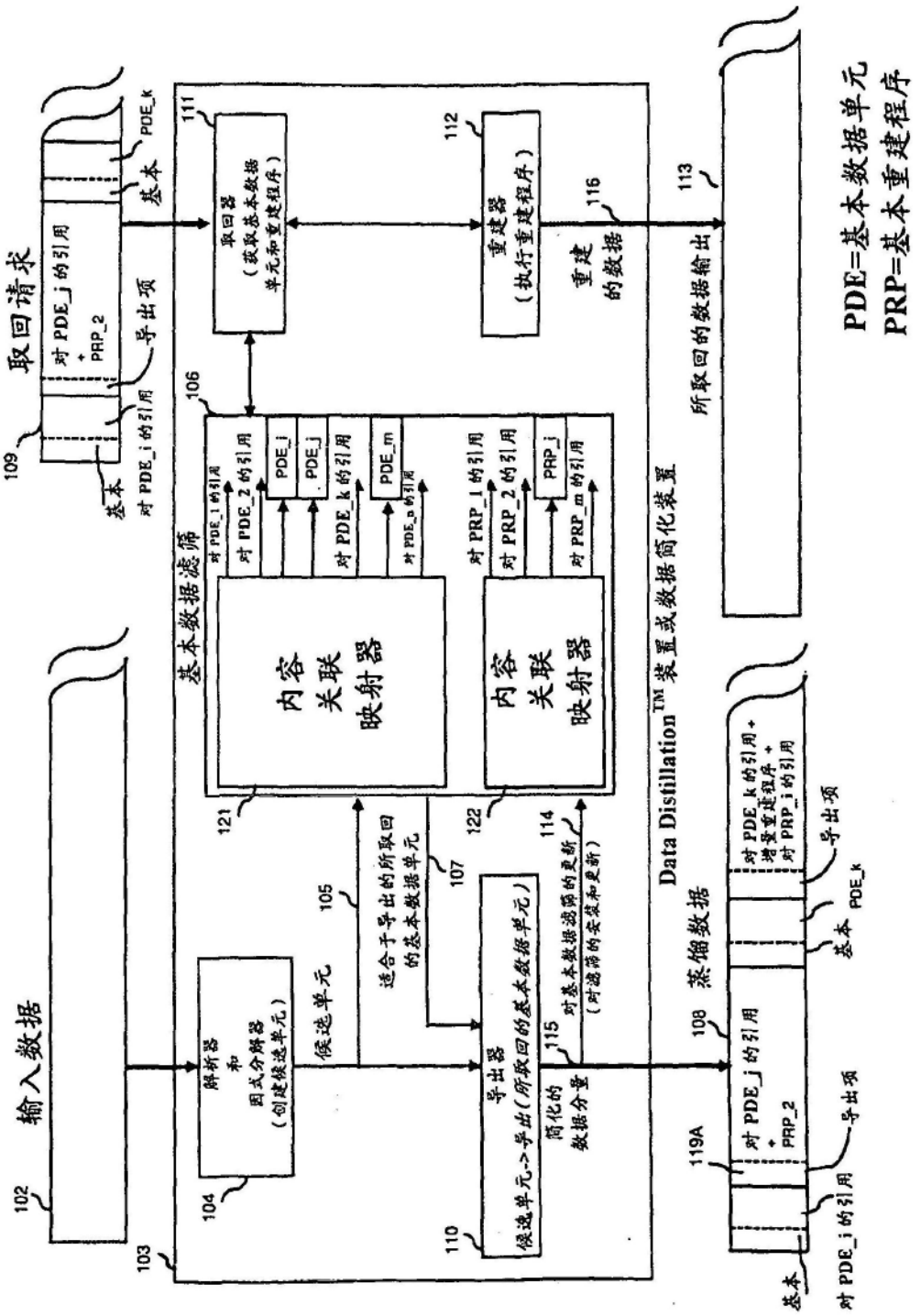

取回请求
对 PDE_j 的引用 + PRP_2
导出项
基本
PDE_k
对 PDE_i 的引用
109
111
取回器
（获取基本数据单元和重建程序）
112
重建器
（执行重建程序）
116
重建的数据
113
所取回的数据输出
106
基本数据滤筛
对 PDE_1 的引用
对 PDE_2 的引用
PDE_i
PDE_j
对 PDE_k 的引用
PDE_m
对 PDE_n 的引用
对 PRP_1 的引用
对 PRP_2 的引用
PRP_j
对 PRP_m 的引用
内容关联映射器
内容关联映射器
121
122
114
输入数据
102
103
解析器和因式分解器（创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据滤筛的更新
（对滤筛的实施和更新）
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
119A
对 PDE_j 的引用 + PRP_2
导出项
基本
对 PDE_i 的引用
对 PDE_k 的引用 + 增量重建程序 + 对 PRP_i 的引用
导出项
基本
PDE_k
PDE=基本数据单元
PRP=基本重建程序

图1G

图 1H：针对蒸馏数据中的单元的格式的范例规范

字段名称	大小	描述	编码
单元类型	3 比特	表明蒸馏单元的类型	0：基本数据筛中的基本数据单元 1：利用单 PDE 重建的导出项（蒸馏数据中的内联 RP） 2：利用单 PDE 重建的导出项（基本数据筛选筛中的 PRP） 3：利用单 PDE 重建的导出项（PRP 选筛中的 PRP） 4：利用多 PDE 重建的导出项（蒸馏数据中的内联 RP） 5：利用多 PDE 重建的导出项（基本数据筛选筛中的 PRP） 6：利用多 PDE 重建的导出项（PRP 选筛中的 PRP） 7：（蒸馏数据中的）内联基本数据单元 注意：PRP=基本重建程序，PDE=基本数据单元
单元大小	2 字节	单元的大小 （原始输入中的单元的大小）	编码为(大小-1)。0=1 字节，.....65535=64K 字节
提取 PDE 计数	2 比特	仅对于多 PDE 重建： 额外基本数据单元的数目	编码为(计数-1)。0=1 单元，...3=4 单元
引用大小	3 比特	对基本数据单元或重建程序 （RP）的以字节计的引用大小	编码为(大小-1)。0=1 字节，1=2 字节，...7=8 字节
（多个） PDE 大小	2 字节	基本数据单元（PDE）的大小	编码为(大小-1)。0=1 字节，.....65535=64K 字节
（多个） PDE 引用	（引用大小）	对默认基本数据单元以及 任何额外基本数据单元的引用	单元引用
RP 大小	2 字节	重建程序（RP）的大小	编码为(大小-1)。0=1 字节，.....65535=64K 字节
重建程序 引用	（引用大小）	仅对于基本数据筛选筛或重建 程序选筛中的重建程序： 引用重建程序	重建程序引用
重建程序 （RP）	可变	仅对于内联重建程序： 重建程序	参见图 7A

实例 1：重复或基本数据单元

记录类型：0	单元大小：2 字节	引用大小：5（6 字节）	PDE 大小：2 字节	PDE 引用：6 字节

实例 2：利用单一单元重建的导出项，内联 RP（13 字节）

记录类型：1	单元大小：2 字节	引用大小：5（6 字节）	PDE 大小：2 字节	PDE 引用：6 字节	RP 大小：2 字节	重建程序：13 字节

实例 3：利用多单元重建的导出项，2 个基本数据单元，单元存储库中的 RP（注意 B=字节）

记录类型：5	单元大小：2B	额外 PDE 计数：1	引用大小：5	PDE1 大小：2B	PDE1 引用：6B	PDE2 大小：2B	PDE2 引用：6B	RP 大小：6B	RP 引用：2B

图1H

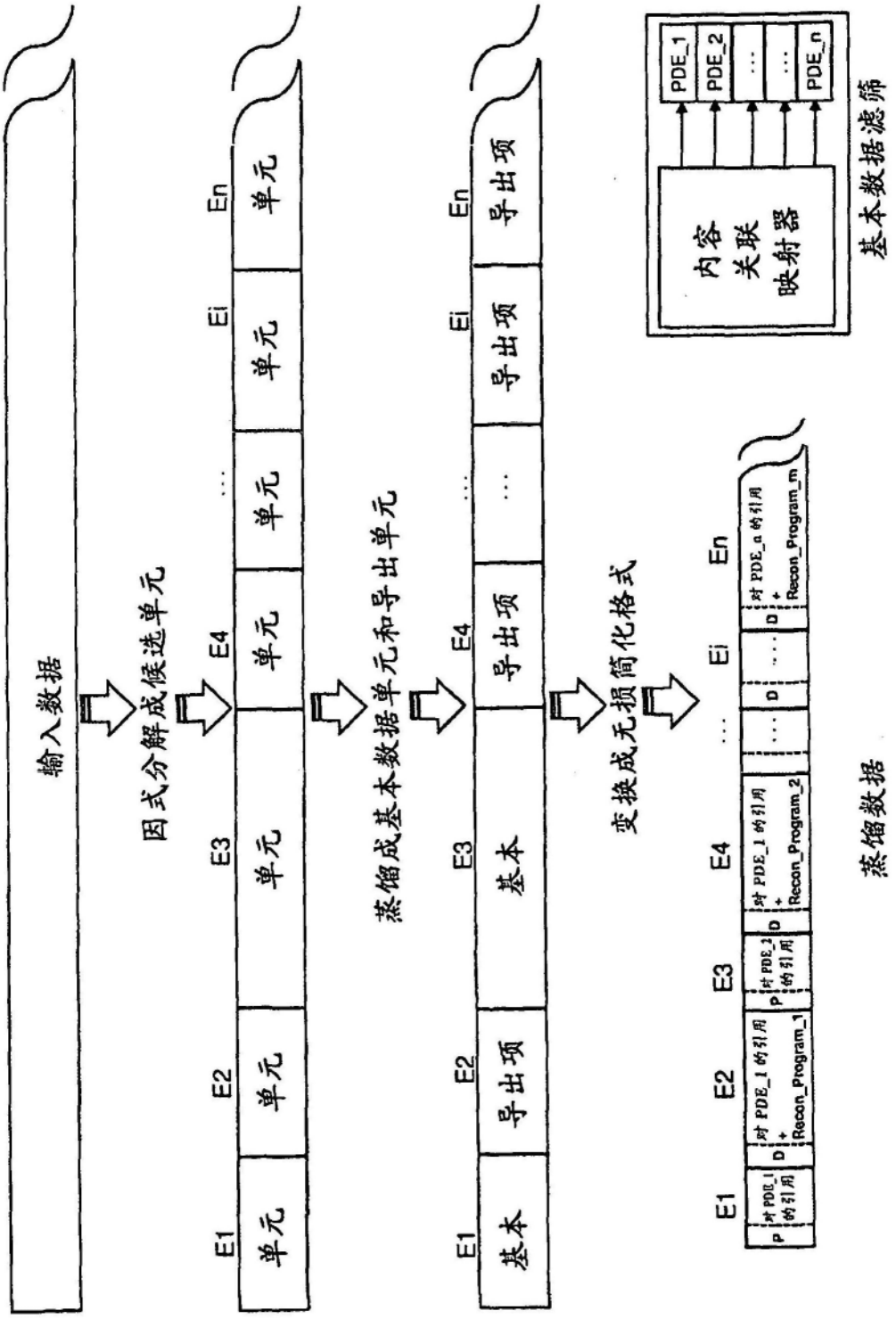

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
基本数据滤筛
内容关联映射器
PDE_1
PDE_2
PDE_n
E1
E2
E3
E4
Ei
En
单元
基本
导出项
对 PDE_1 的引用 + Recon_Program_1
对 PDE_2 的引用
对 PDE_1 的引用 + Recon_Program_2
对 PDE_n 的引用 + Recon_Program_m
P
D

图1I

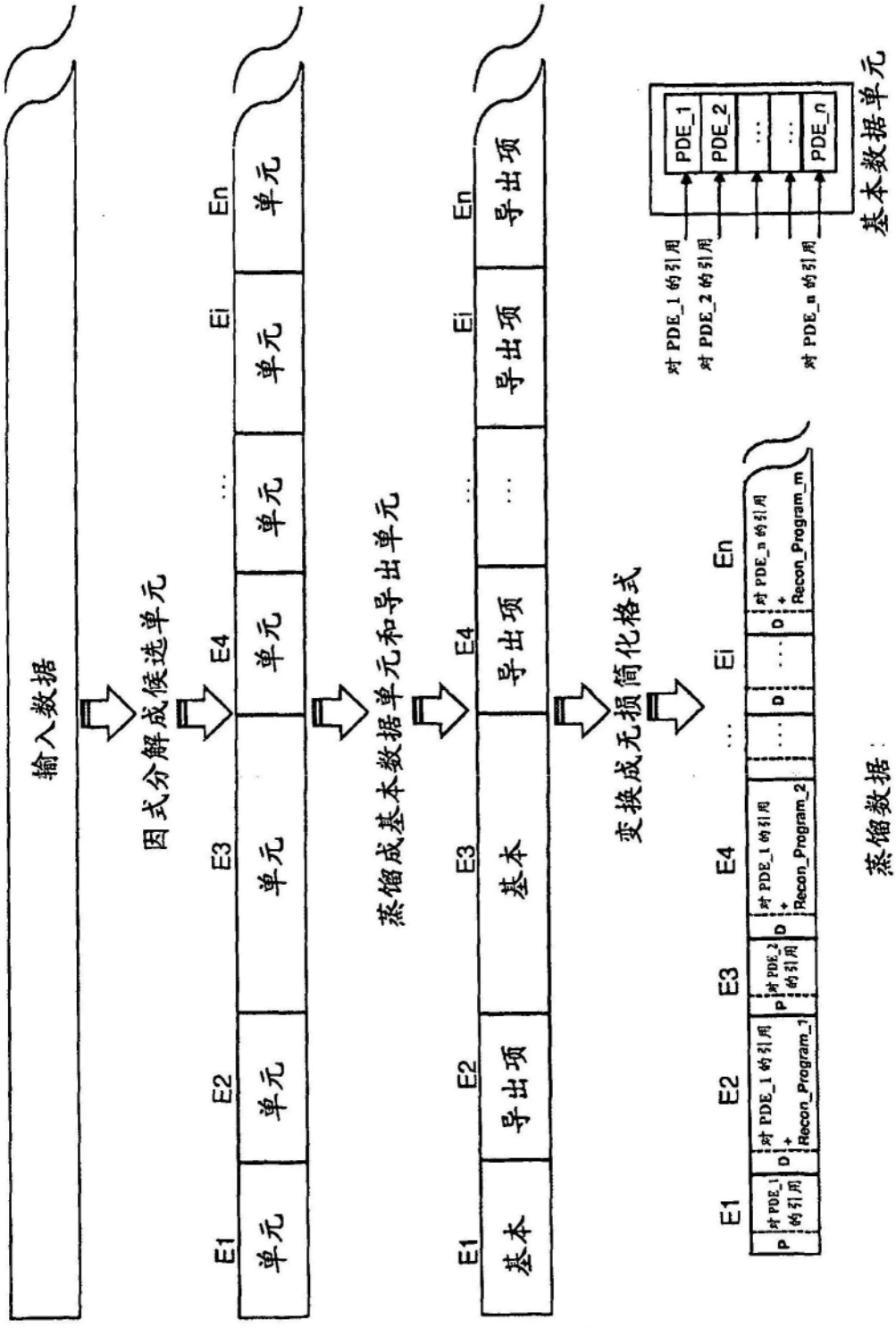

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据：
基本数据单元
E1
E2
E3
E4
Ei
En
单元
基本
导出项
P
D
对 PDE_1 的引用 + Recon_Program_1
对 PDE_2 的引用
对 PDE_1 的引用 + Recon_Program_2
对 PDE_n 的引用 + Recon_Program_m
PDE_1
PDE_2
PDE_n
对 PDE_1 的引用
对 PDE_2 的引用
对 PDE_n 的引用

图1J

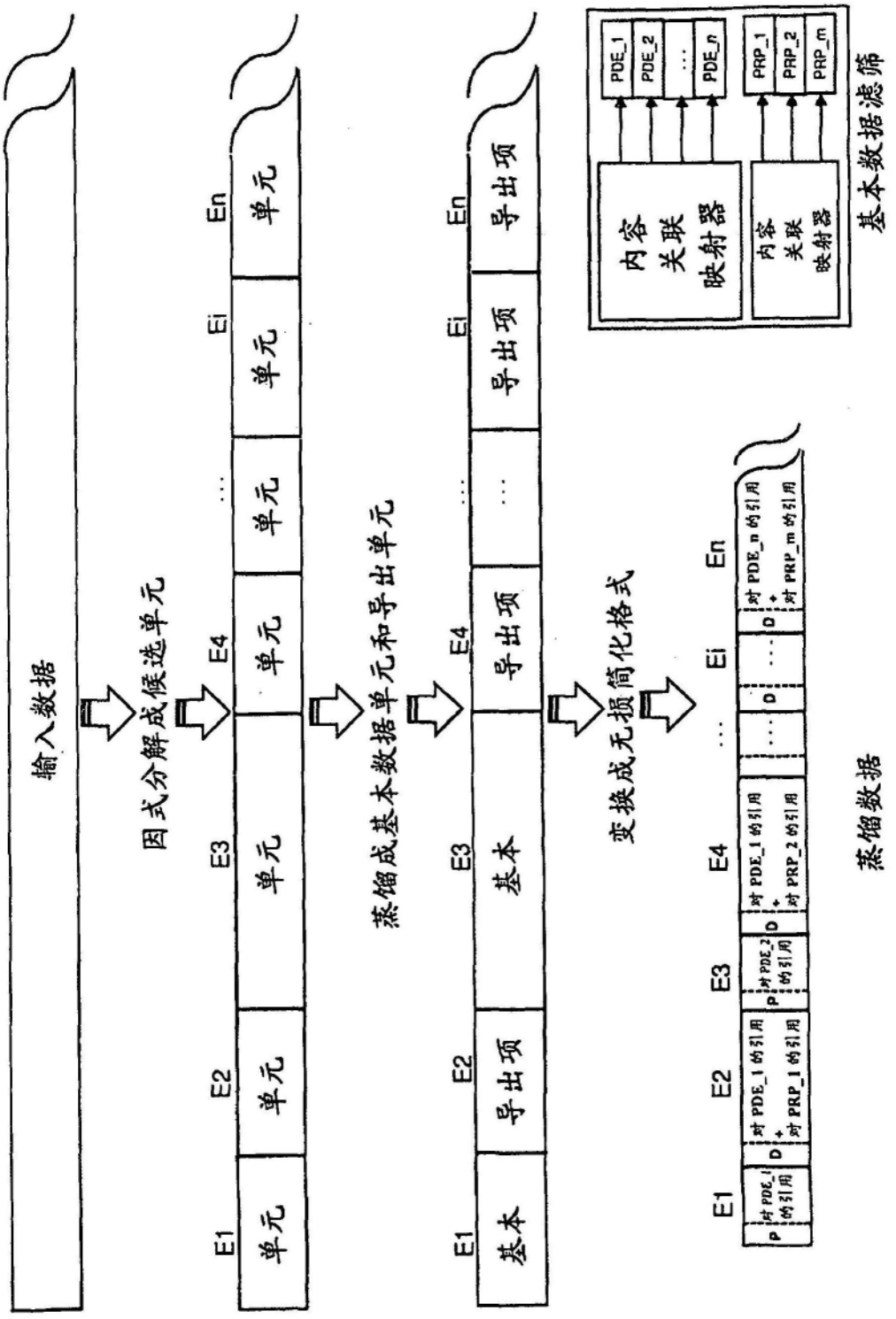

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
基本数据滤筛
E1
E2
E3
E4
Ei
En
单元
基本
导出项
内容关联映射器
PDE_1
PDE_2
PDE_n
PRP_1
PRP_2
PRP_m
P
D
对 PDE_1 的引用
对 PRP_1 的引用
对 PDE_2 的引用
对 PRP_2 的引用
对 PDE_n 的引用
对 PRP_m 的引用

图1K

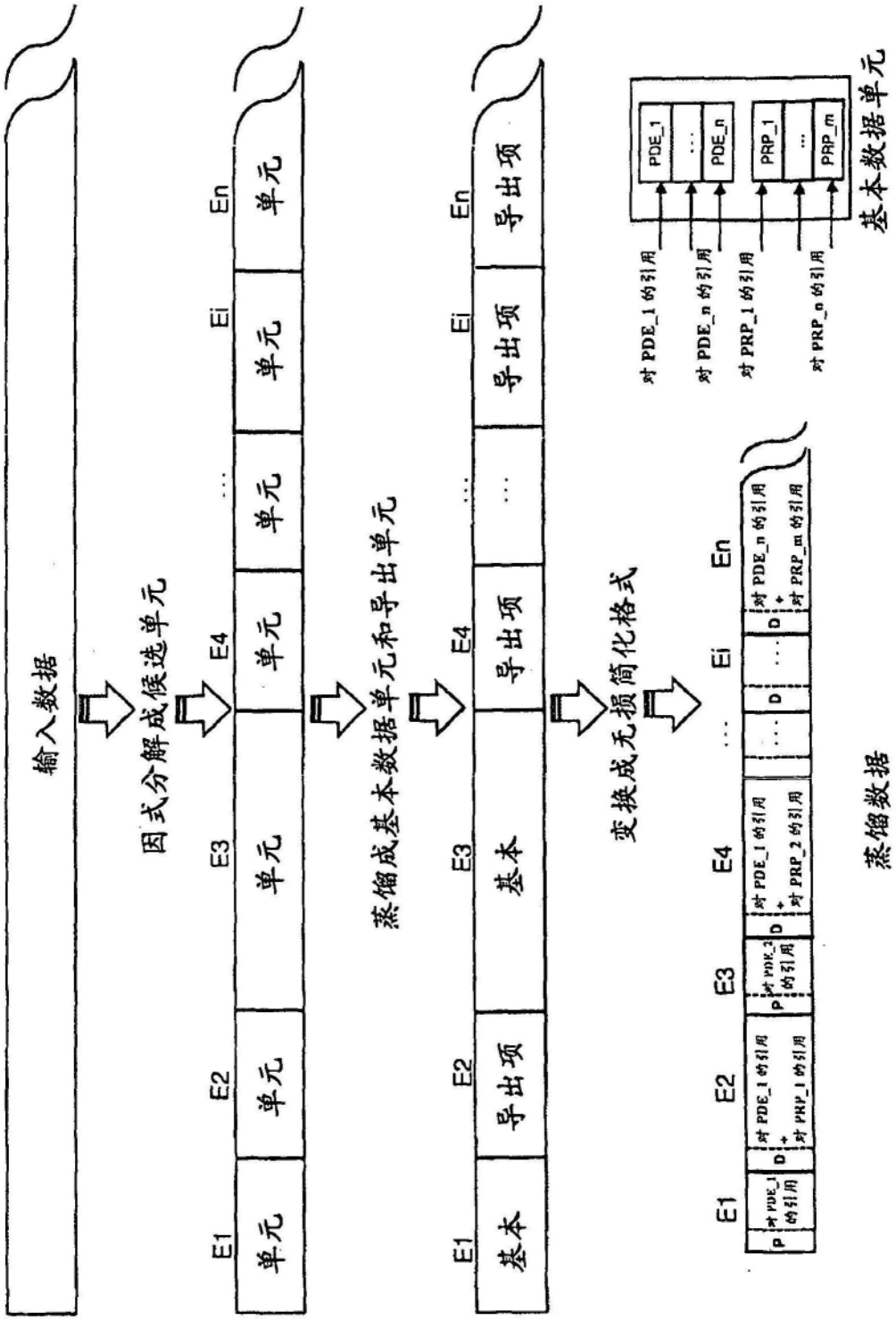

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
基本数据单元
E1
E2
E3
E4
Ei
En
单元
基本
导出项
P
D
对 PDE_1 的引用
对 PRP_1 的引用
对 PDE_3 的引用
对 PDE_1 的引用
对 PRP_2 的引用
对 PDE_n 的引用
对 PRP_m 的引用
PDE_1
PDE_n
PRP_1
PRP_m
对 PDE_1 的引用
对 PDE_n 的引用
对 PRP_1 的引用
对 PRP_n 的引用

图1L

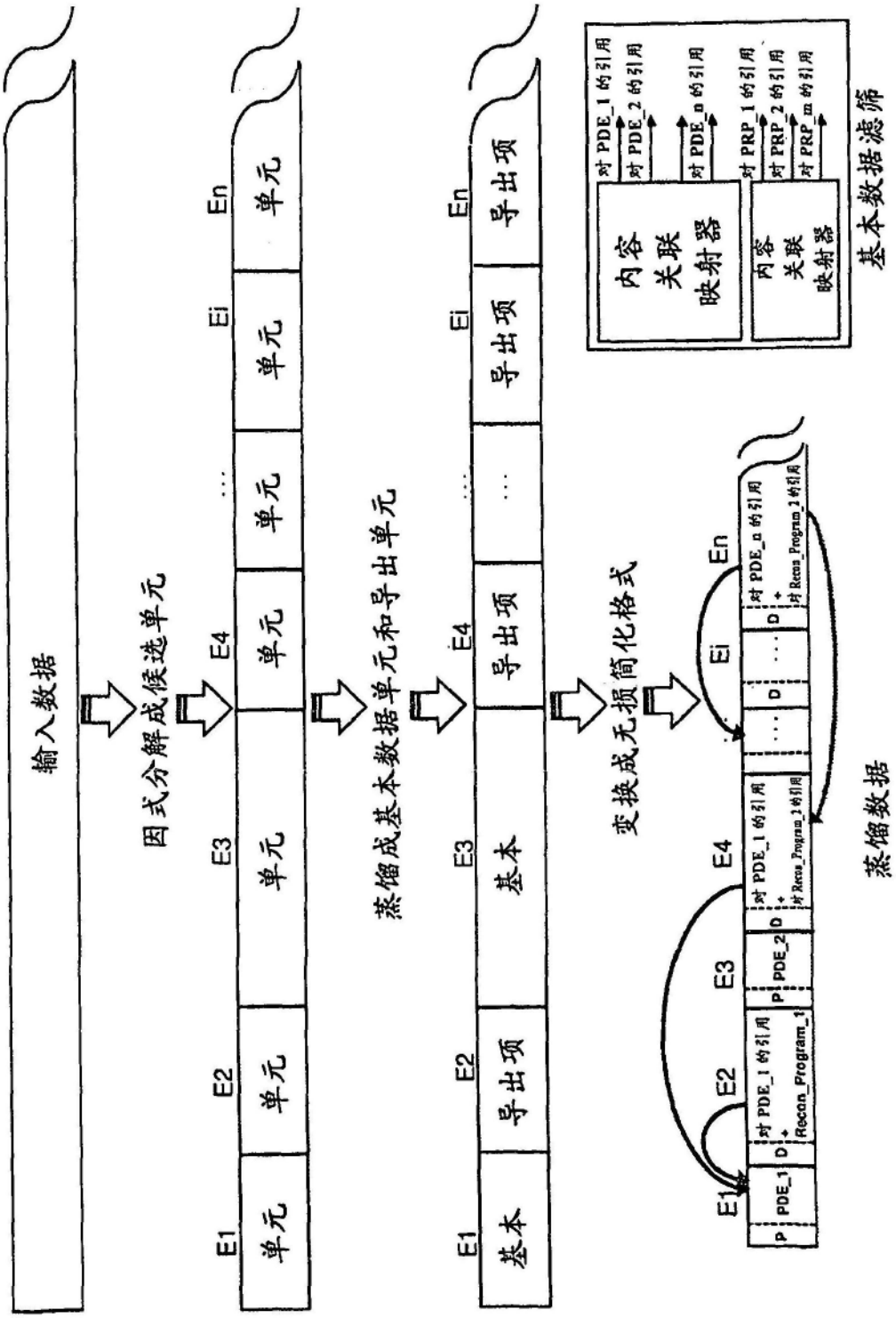

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
基本数据滤筛
E1
E2
E3
E4
Ei
En
单元
基本
导出项
内容关联映射器
对 PDE_1 的引用
对 PDE_2 的引用
对 PDE_n 的引用
对 PRP_1 的引用
对 PRP_2 的引用
对 PRP_m 的引用
PDE_1
PDE_2
Recon_Program_1

图1M

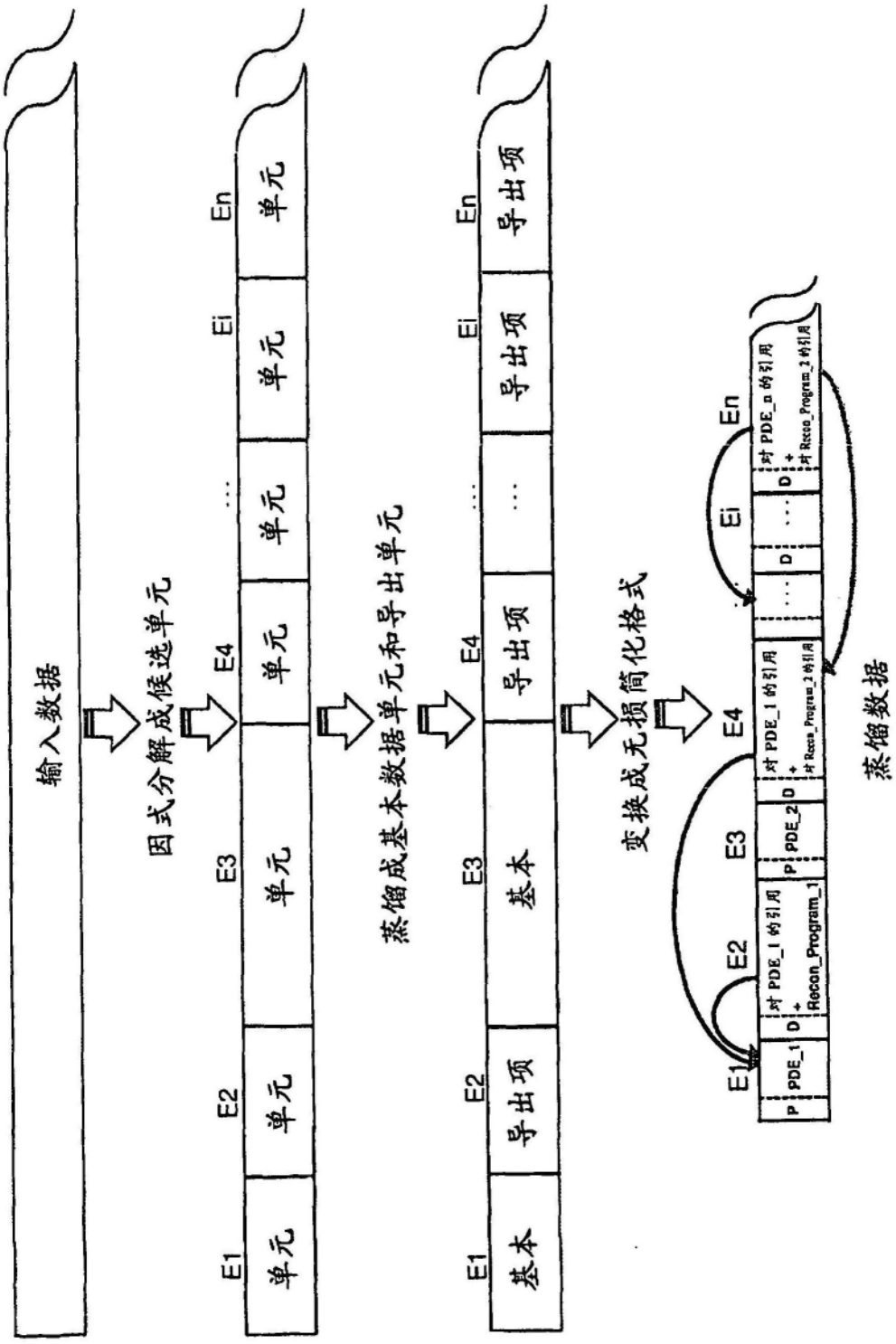

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
E1
E2
E3
E4
Ei
En
单元
基本
导出项
P
PDE_1
D
对 PDE_1 的引用
+
Recon_Program_1
PDE_2
对 PDE_2 的引用
+
Recon_Program_2
对 PDE_n 的引用
+
对 Recon_Program_2 的引用

图1N

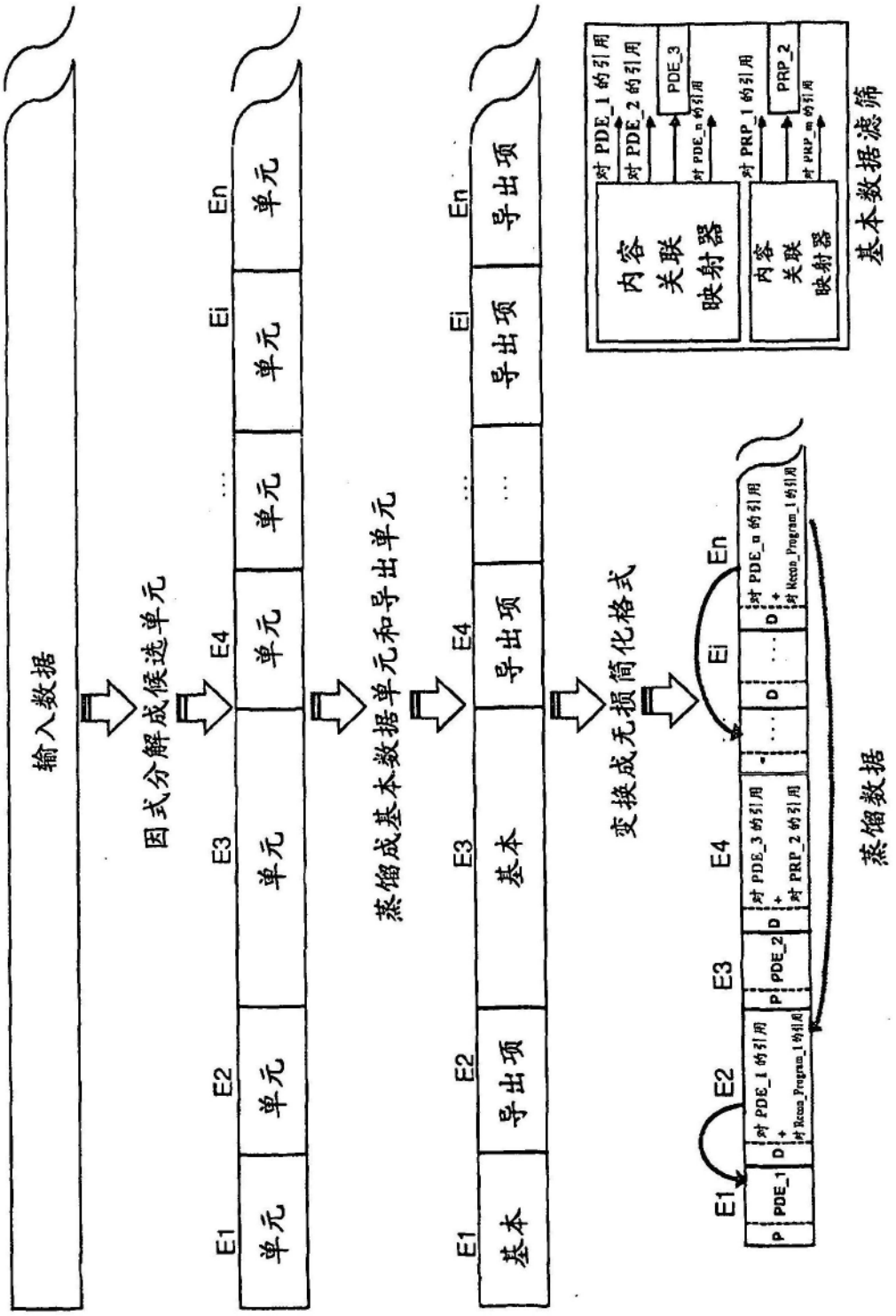

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
基本数据滤筛
E1
E2
E3
E4
Ei
En
单元
基本
导出项
P
D
PDE_1
PDE_2
PDE_3
PRP_2
内容关联映射器
对 PDE_1 的引用
对 PDE_2 的引用
对 PDE_n 的引用
对 PRP_1 的引用
对 PRP_m 的引用
对 Recon_Program_1 的引用

图10

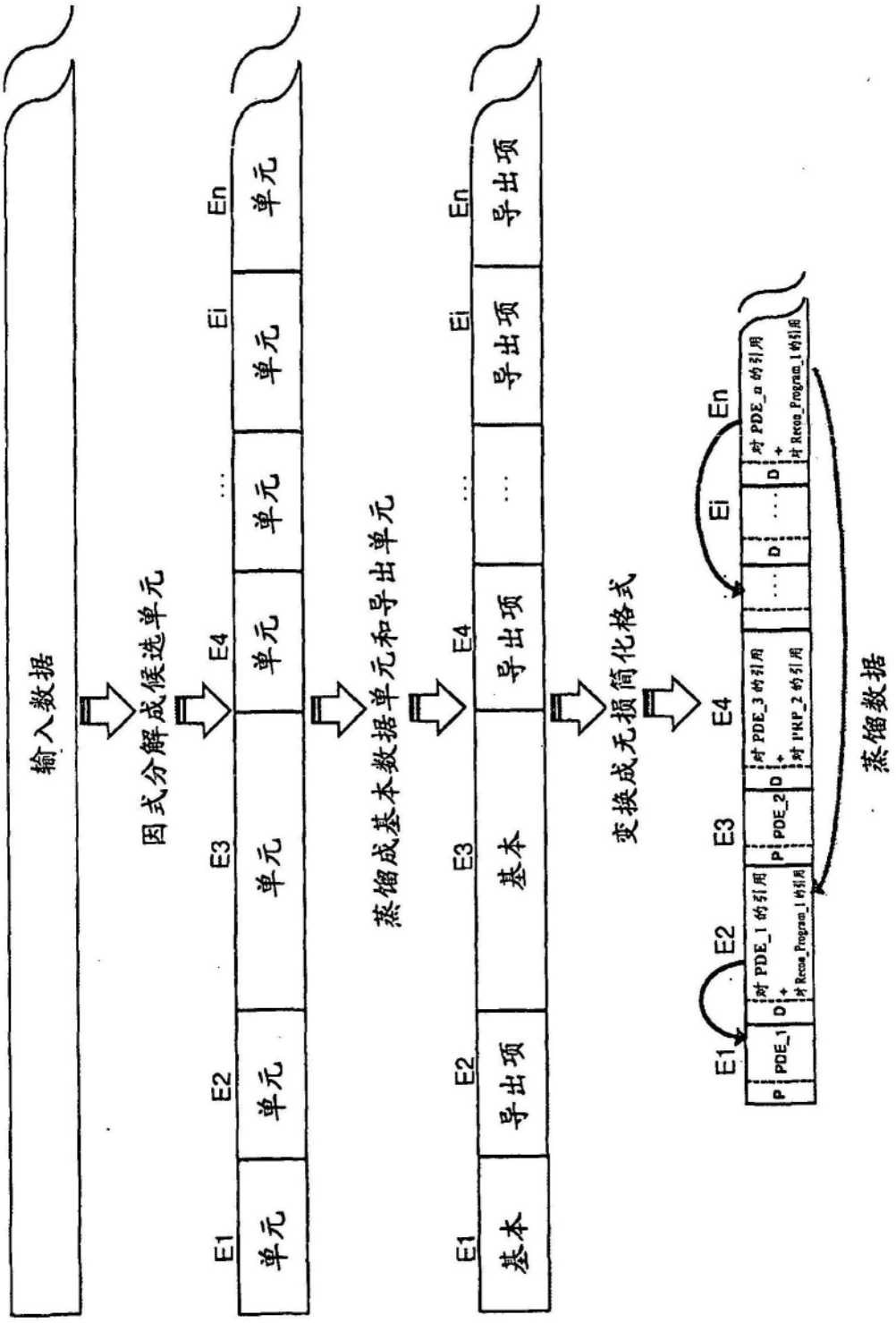

输入数据
因式分解成候选单元
蒸馏成基本数据单元和导出单元
变换成无损简化格式
蒸馏数据
E1
E2
E3
E4
Ei
En
单元
基本
导出项
P
D
PDE_1
PDE_2
对 PDE_1 的引用
对 Recon_Program_1 的引用
对 PDE_3 的引用
对 INP_2 的引用
对 PDE_n 的引用

图1P

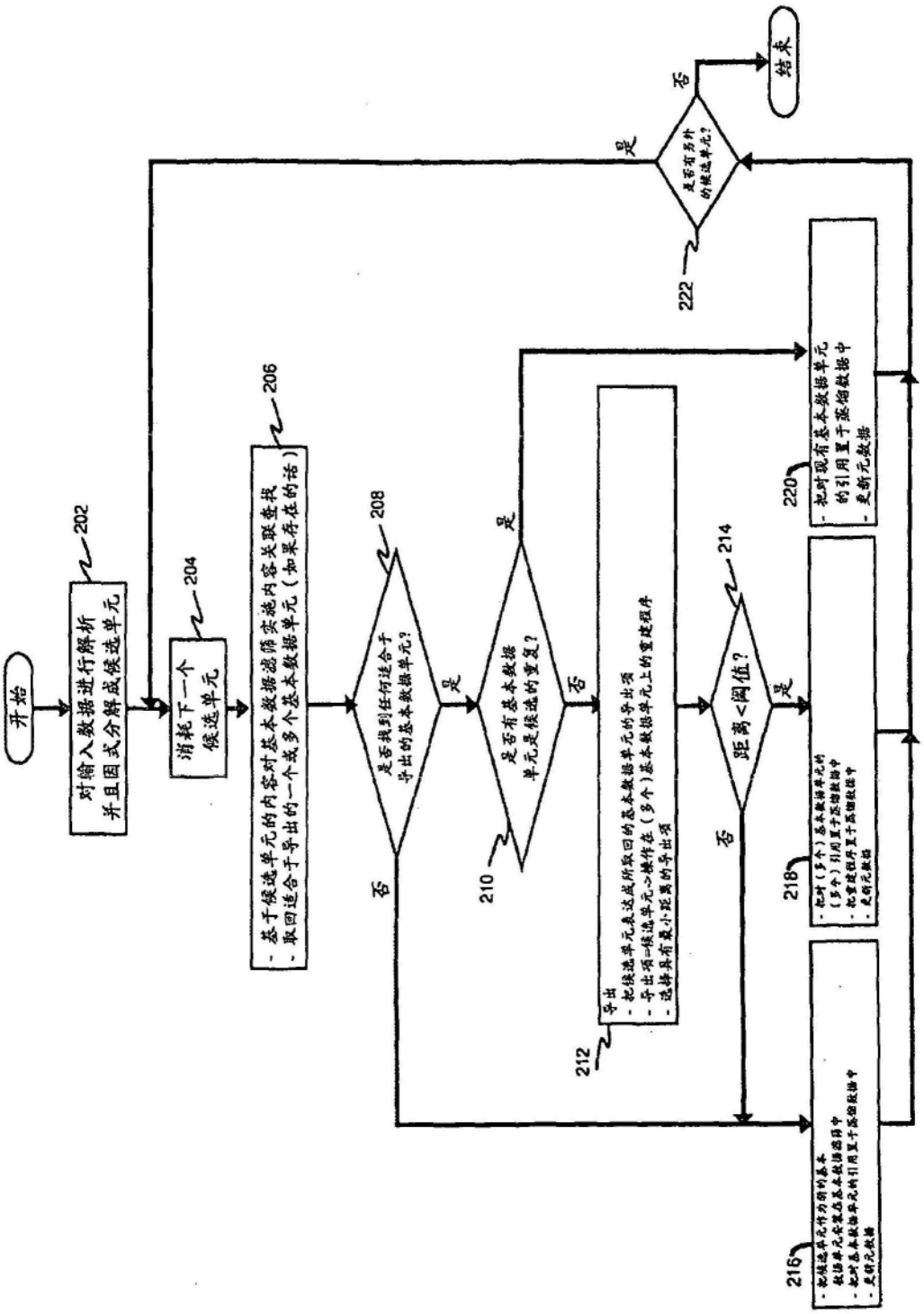

开始
202
对输入数据进行解析并且固式分解成候选单元
204
消耗下一个候选单元
206
- 基于候选单元的内容对基本数据源执行内容关联查找
- 取回适合于导出的一个或多个基本数据单元（如果存在的话）
208
是否找到任何适合于导出的基本数据单元?
否
是
210
是否有基本数据单元是候选的重复?
否
是
212
导出
- 把候选单元表达成所取回的基本数据单元的导出项
- 导出项=候选单元->操作在（多个）基本数据单元上的重建程序
- 选择具有最小距离的导出项
214
距离<阈值?
否
是
216
- 把候选单元作为所附的基本数据单元或者基本数据源中
- 把对基本数据单元的引用置于压缩数据中
- 更新元数据
218
- 把对（多个）基本数据单元的（多个）引用置于压缩数据中
- 把导出程序置于压缩数据中
- 更新元数据
220
- 把对现有基本数据单元的引用置于压缩数据中
- 更新元数据
222
是否有另外的候选单元?
是
否
结束

图2

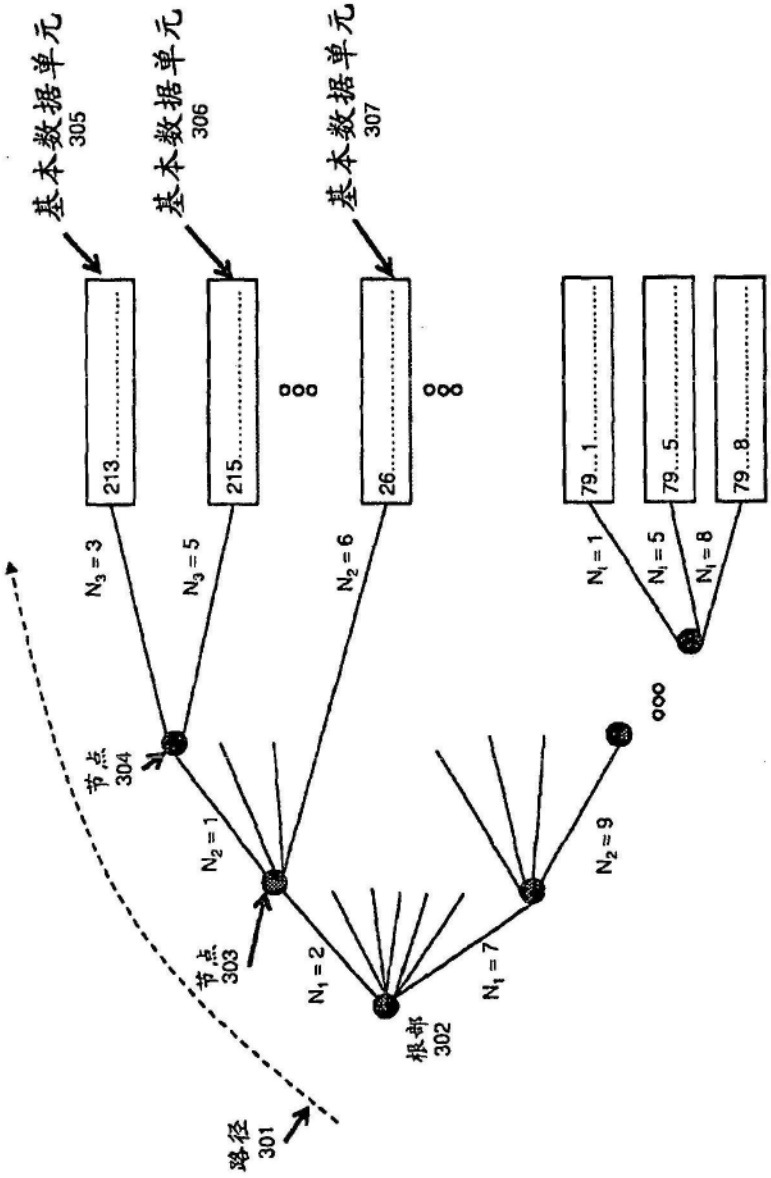

基本数据单元
305
基本数据单元
306
基本数据单元
307
213
215
26
79..1
79..5
79..8
N3 = 3
N3 = 5
N2 = 6
N1 = 1
N1 = 5
N1 = 8
节点
304
N2 = 1
节点
303
N1 = 2
根部
302
N1 = 7
N2 = 9
路径
301

图3A

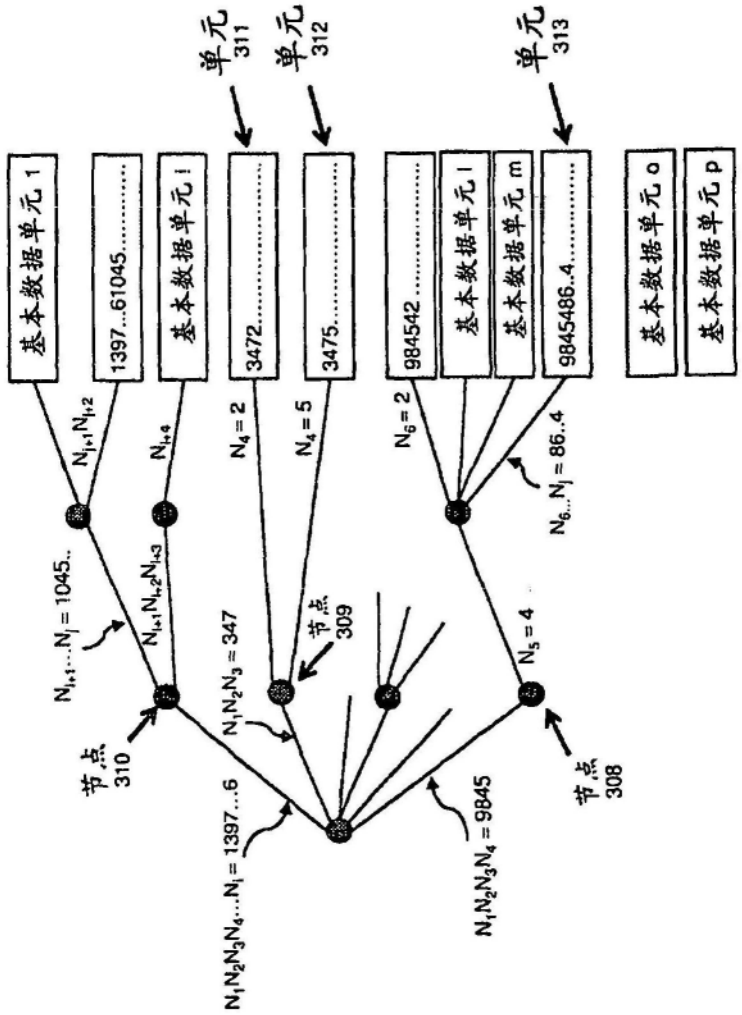

单元 311
单元 312
单元 313
基本数据单元 1
1397...61045.....
基本数据单元 i
3472.....
3475.....
984542.....
基本数据单元 i
基本数据单元 m
9845486..4.....
基本数据单元 o
基本数据单元 p
$N_{i+1}N_{i+2}$
N_{i+4}
$N_4 = 2$
$N_4 = 5$
$N_6 = 2$
$N_6..N_j = 86..4$
$N_{i+1}...N_j = 1045..$
$N_{i+1}N_{i+2}N_{i+3}$
$N_1N_2N_3 = 347$
节点 309
$N_5 = 4$
节点 310
节点 308
$N_1N_2N_3N_4...N_i = 1397...6$
$N_1N_2N_3N_4 = 9845$

图3B

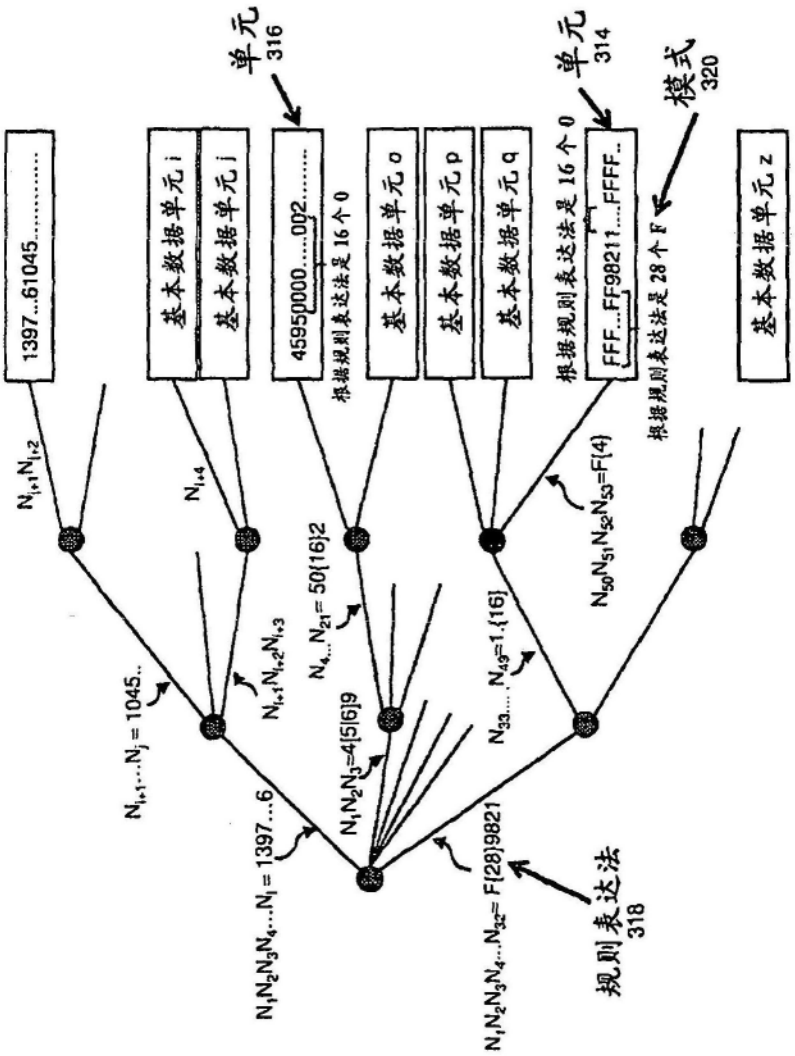
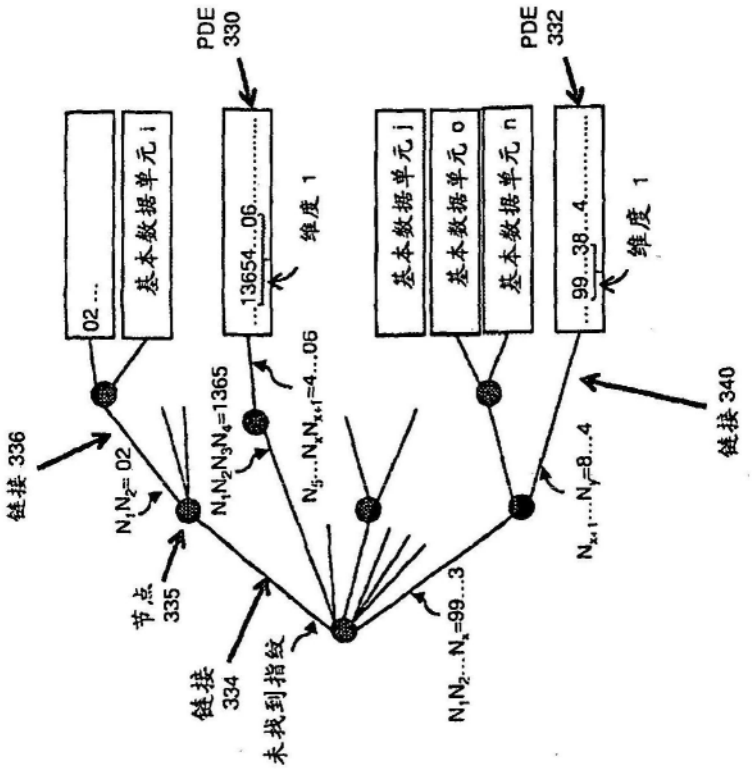

1397...61045.....
基本数据单元 i
基本数据单元 j
单元 316
45950000.....002.....
根据规则表达法是 16 个 0
基本数据单元 o
基本数据单元 p
基本数据单元 q
根据规则表达法是 16 个 0
FFF....FF98211....FFFF..
单元 314
根据规则表达法是 28 个 F
模式 320
基本数据单元 z
$N_{i+1}N_{i+2}$
N_{i+4}
$N_{i+1}...N_j = 1045..$
$N_{i+1}N_{i+2}N_{i+3}$
$N_4...N_{21} = 50\{16\}2$
$N_1N_2N_3 = 4|5|6|9$
$N_{33}....N_{49} = 1.\{16\}$
$N_{50}N_{51}N_{52}N_{53} = F\{4\}$
$N_1N_2N_3N_4...N_I = 1397....6$
$N_1N_2N_3N_4...N_{32} = F\{28\}9821$
规则表达法 318

图3C


PDE 330
PDE 332
02 ...
基本数据单元 i
...13654...06
维度 1
基本数据单元 j
基本数据单元 o
基本数据单元 n
...99...38...4
维度 1
链接 336
N_1N_2= 02
$N_1N_2N_3N_4$=1365
$N_5...N_xN_{x+1}$=4...06
$N_{x+1}...N_y$=8...4
链接 340
节点 335
链接 334
未找到指纹
$N_1N_2...N_x$=99...3

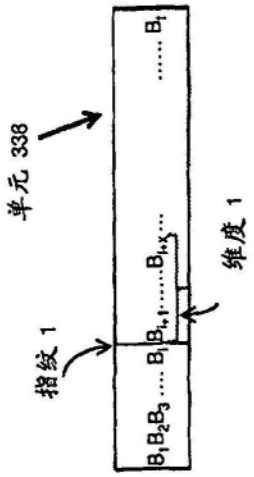

单元 338
B_1
指纹 1
$B_1B_2B_3$ .... B_i B_{i+1}.....B_{i+x}....
维度 1

图3D

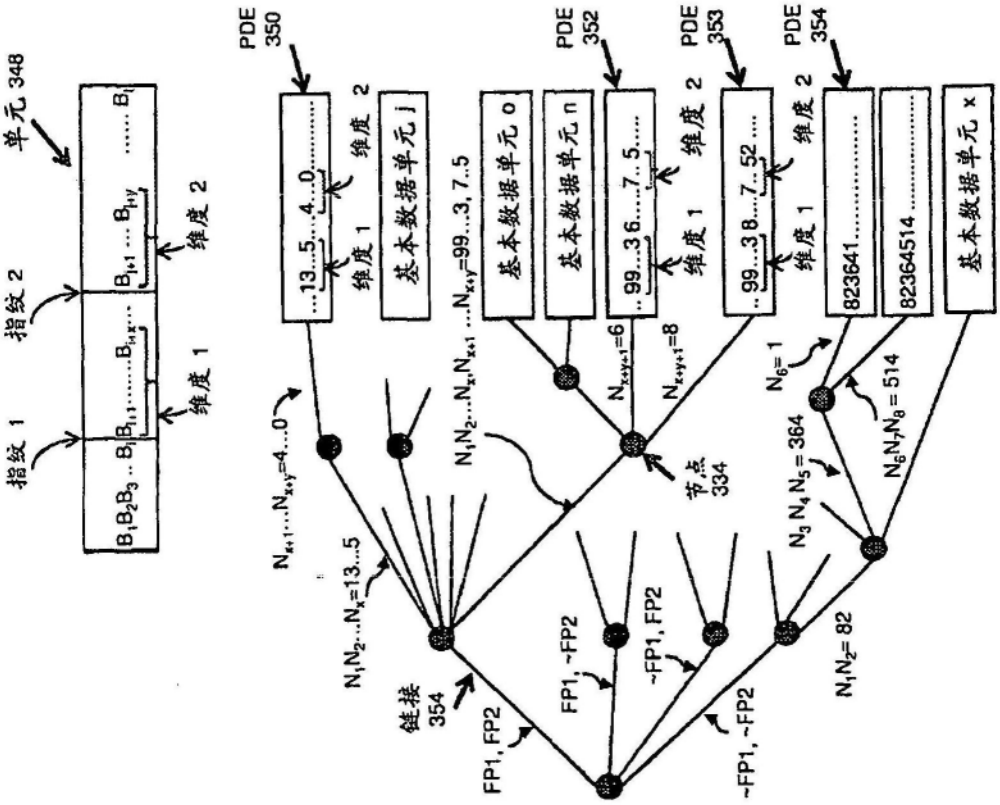
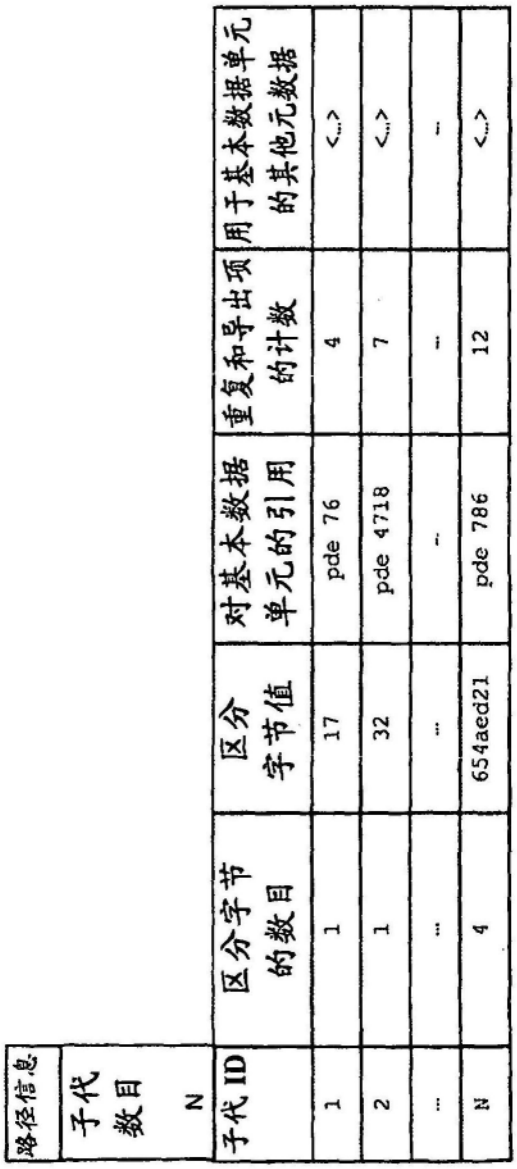


图3E

路径信息					
子代数目 N	子代 ID	区分字节的数目	区分字节值	对子代节点的引用
	1	6	abef12d6743a	节点 27
	2	2	dcfa	节点 16
	...	...	...	...
	N	4	f3231929	节点 4198

图3F



路径信息						
子代数目 N	区分字节的数目	区分字节值	对基本数据单元的引用	重复和导出项的计数	用于基本数据单元的其他元数据
子代 ID					
1	1	17	pde 76	4	<...>
2	1	32	pde 4718	7	<...>
...	...	...	...	...	...
N	4	654aed21	pde 786	12	<...>

图3G

路径信息								
子代数目 N	子代 ID	区分字节的数目	区分字节值	对基本数据单元的引用	导航前瞻字节	重复和导出项的计数	用于基本数据单元的其他元数据
	1	1	17	pde 76	13476231	4	<...>
	2	1	32	pde 4718	00337650	7	<...>
	...	...	...	...	...	...	...
	N	4	654aed21	pde 786	ed189721	12	<...>

图3H

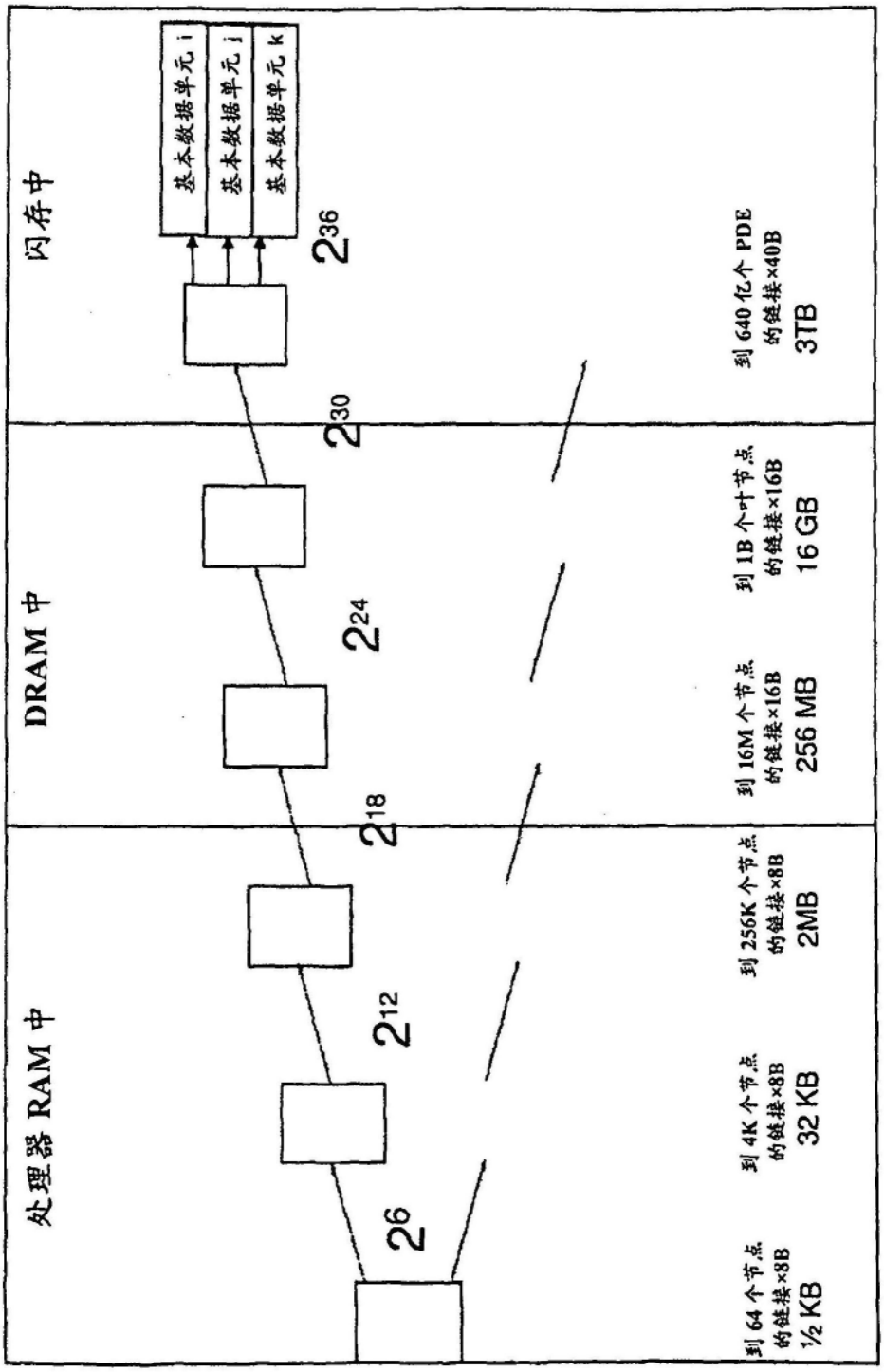

处理器 RAM 中
DRAM 中
闪存中
基本数据单元 i
基本数据单元 j
基本数据单元 k
2^6
2^{12}
2^{18}
2^{24}
2^{30}
2^{36}
到 64 个节点的链接×8B
½ KB
到 4K 个节点的链接×8B
32 KB
到 256K 个节点的链接×8B
2MB
到 16M 个节点的链接×16B
256 MB
到 1B 个叶节点的链接×16B
16 GB
到 640 亿个 PDE 的链接×40B
3TB

图4

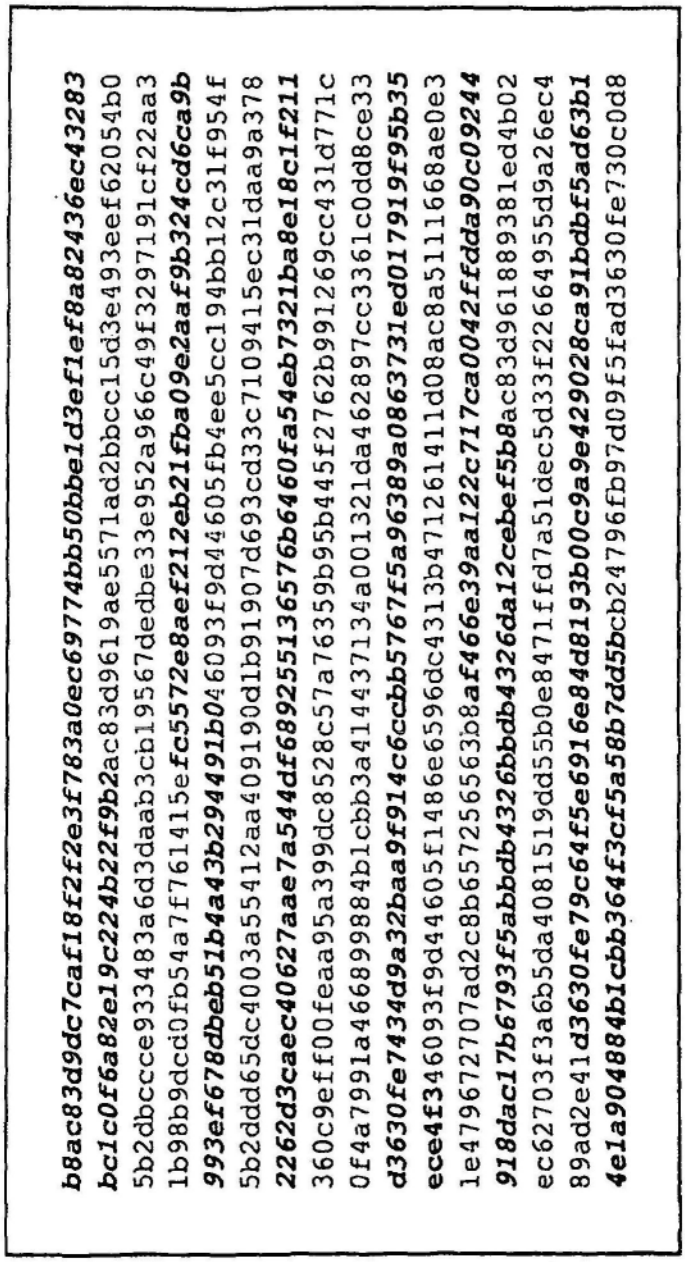

b8ac83d9dc7caf18f2f2e3f783a0ec69774bb50bbe1d3ef1ef8a82436ec43283
bc1c0f6a82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0
5b2dbccce933483a6d3daab3cb19567dedbe33e952a966c49f3297191cf22aa3
1b98b9dcd0fb54a7f7614l5efc5572e8aef212eb21fba09e2aaf9b324cd6ca9b
993ef678dbeb51b4a43b294491b046093f9d44605fb4ee5cc194bb12c31f954f
5b2ddd65dc4003a55412aa409190d1b91907d693cd33c7109415ec31daa9a378
2262d3caec40627aae7a544df6892551365766b6460fa54eb7321ba8e18c1f211
360c9eff00feaa95a399dc8528c57a76359b95b445f2762b991269cc431d771c
0f4a7991a466899884b1cbb3a41443713 4a001321da462897cc3361c0dd8ce33
d3630fe7434d9a32baa9f914c6ccbb5767f5a96389a0863731ed017919f95b35
ece4f346093f9d44605f1486e6596dc4313b471261411d08ac8a5111668ae0e3
1e479672707ad2c8b657256563b8af466e39aa122c717ca0042ffdda90c09244
918dac17b6793f5abbdb4326bbdb4326da12cebef5b8ac83d961889381ed4b02
ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f22664955d9a26ec4
89ad2e41d3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b1
4e1a904884b1cbb364f3cf5a58b7dd5bcb24796fb97d09f5fad3630fe730c0d8

图5A

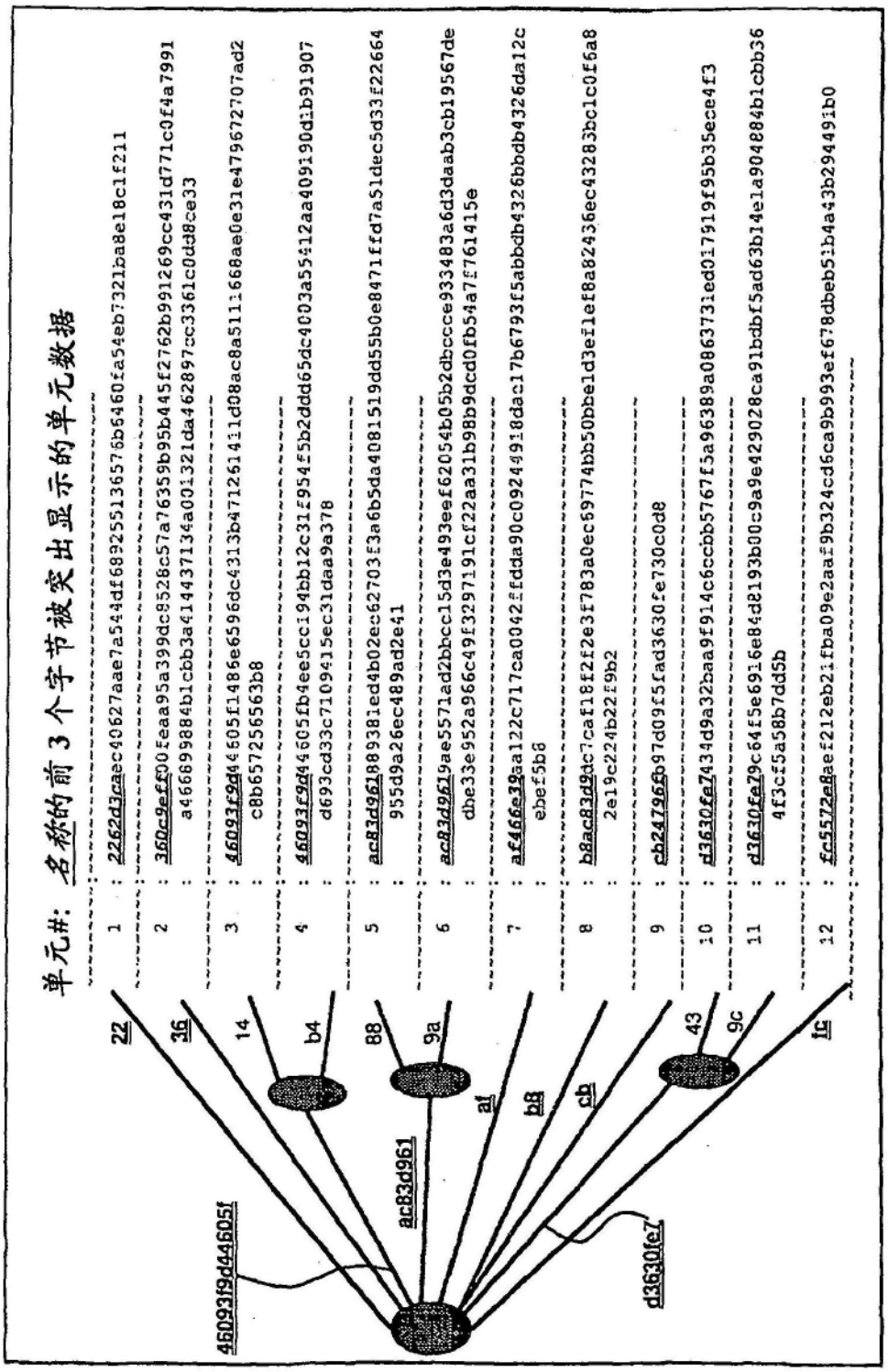

单元#: 名称的前 3 个字节被突出显示的单元数据
1 : 2262d3caec40627aae7a544df6892551365765b6460fa54eb7321ba8e18c1f211
2 : 360c9eff005eaa95a399dc8526c57a76359b95b445f2762b991269cc431d771c0f4a7991
a46669884b1cbb3a614437134a001321da462897cc3361c0dd8ce33
3 : 46093f9d446605f1486e6596dc4313b4712614110d08ac8a5111668ae6e31e479672707ad2
c8b657256563b8
4 : 46093f9d446605fb4ee5cc194bb12c31f954f5b2ddd65dc4003a55412aa409190d1b91907
d693cd33c7109415ec31daa9a378
5 : ac83d961809381ed4b02ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f22664
955d9a26ec489ad2e41
6 : ac83d9619ae5571ad2bbcc15d3e493eef62054b05b2dbcccce933483a6d3daab3cb19567de
dbe33e952a966c49f3297191cf22aa31b98b9dcd0fb54a7f761415e
7 : af466e19aa122c717ca0042ffdda90c09244918dac17b6793f5abbdb4326bbdb4326da12c
ebef5b8
8 : b8ac83d9dc7caf18f2f2e3f783a0ec59774bb50bbe1d3ef1ef8a824436ec43283bc1c0f6a8
2e19c224b22f9b2
9 : cb247966b97d095fad3630fe730c0d8
10 : d3630fa7434d9a32baa9f914c6ccbb5767f5a96389a0863731ed017919f95b35ece4f3
11 : d3630fa79c64f5e6916e84d8193b00c9a9e4290028ca91bdbf5ad63b14e1a904884b1cbb36
4f3cf5a58b7dd5b
12 : fc5572e8aef212eb21fba09e2aaf9b324cd6ca9b993ef678dbeb51b4a43b29449ib0
22
36
14
b4
88
9a
af
b8
cb
43
9c
1c
460931f9d44605f
ac83d961
d3630fe7

图5B

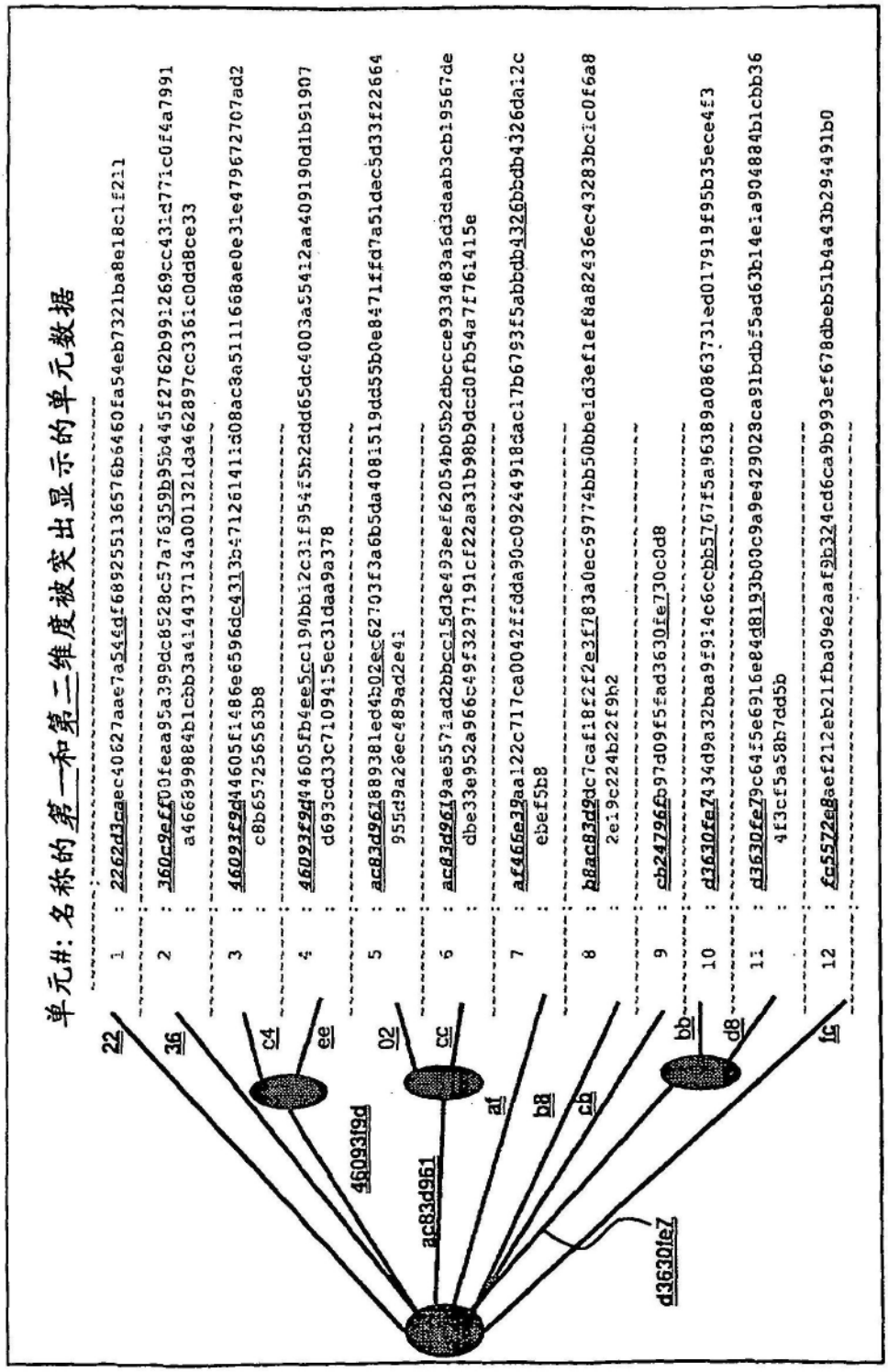

单元#: 名称的第一和第二维度被突出显示的单元数据
1 : 2262d3caec40627aae7a544af6892551365 76b6460fa54eb7321ba8e18c1f211
2 : 360c9aff00feaa95a399dc8528c57a76359b95b445f2762b991269cc431d771c0f4a7991
a466399884b1cbb3a14437134a001321da462897cc3361c0dd8ce33
3 : 46093f9d44605f1486e6596dc4313b2712614116d8ac3a5111668ae0e31e479672707ad2
c8b6572565 63b8
4 : 46093f9d44605fb4ee5cc190bbi2c31f954f5b2dd65dc4003a55412aa409190d1b91907
d693cd33c7109415ec31daa9a378
5 : ac83d9618893816ed4b02ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f22664
955d9a26ec489ad2e41
6 : ac83d9619ae5571ad2bbcc15d3e493eef62054b05b2dbcccce933483a6d3daab3cb19567de
dbe33e952a966c49f3297191cf22aa31b98b9dcd0fb54a7f7614 15e
7 : af466a19aa122c717ca0042ffdda90c09244918dac17b6793f5abbdb4326bbdb4326da12c
ebef5b8
8 : b8ac83d9dc7caf18f2f2e3f783a0ec59774bb50bbe1d3ef1ef8a82436ec43283bc1c0f6a8
2e19c224b22f9b2
9 : cb247966b97d09f5fad3630fe730c0d8
10 : d13630fa7434d9a32baa9f914c6ccbb5767f5a96389a0863731ed0179 19f95b35ece4f3
11 : d3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b14e1a904884b1cbb36
4f3cf5a58b7dd5b
12 : fc5572a8aef212eb21fba09e2aaf9b324cd6ca9b993ef678dbeb51b4a43b294491b0
22
36
c4
ee
02
cc
af
b8
cb
bb
d8
fc
46093f9d
ac83d961
d3630fe7

图5C

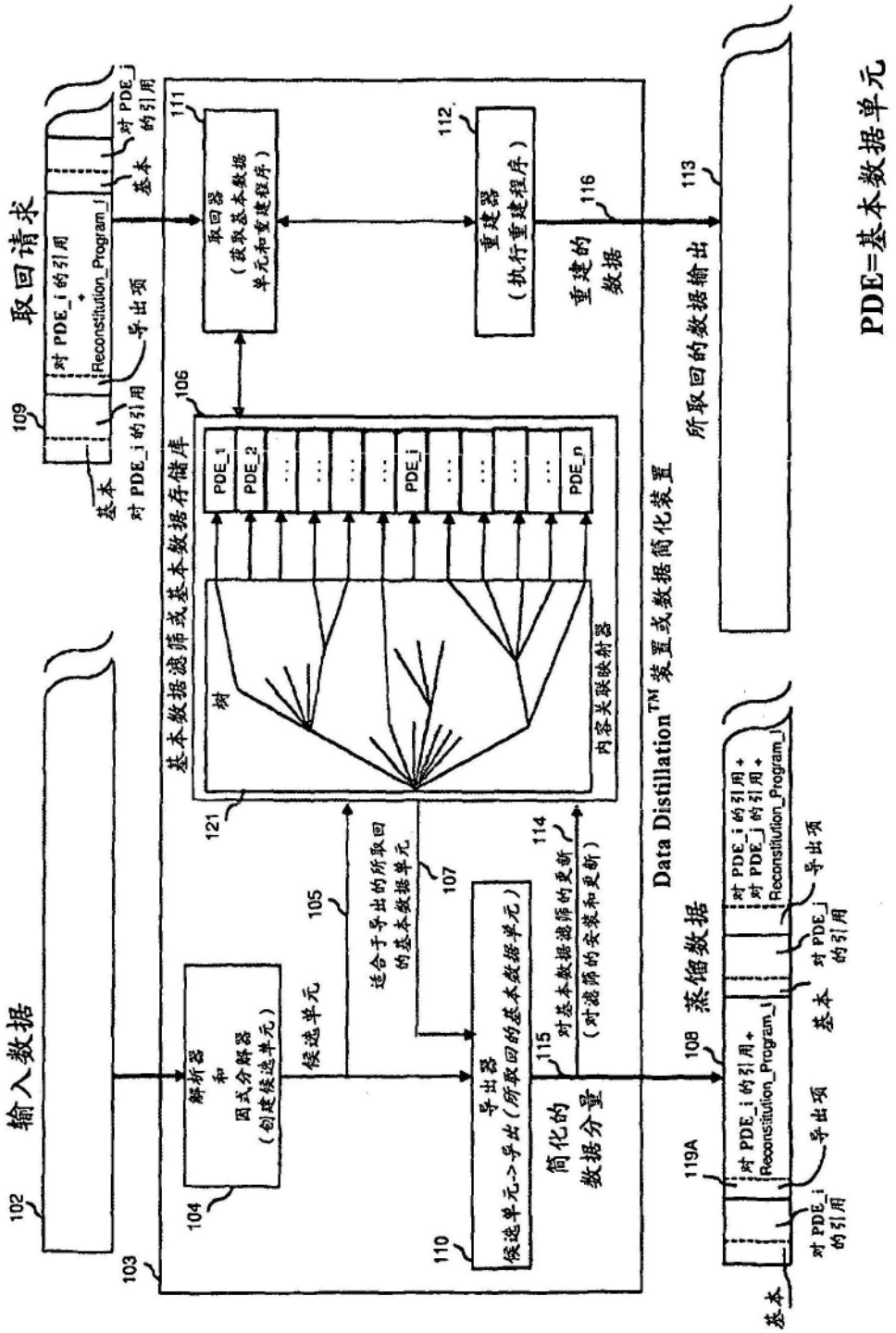

输入数据
102
103
解析器
和
因式分解器
（创建候选单元）
104
候选单元
105
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的
数据分量
115
对基本数据滤筛的更新
（对滤筛的安装和更新）
114
适合于导出的所取回
的基本数据单元
107
基本数据滤筛或基本数据存储库
106
121
树
PDE_1
PDE_2
PDE_i
PDE_n
内容关联映射器
Data Distillation™ 装置或数据简化装置
取回请求
109
对 PDE_i 的引用
+
Reconstitution_Program_i
导出项
基本
对 PDE_j 的引用
111
取回器
（获取基本数据
单元和重建程序）
112
重建器
（执行重建程序）
116
重建的
数据
113
所取回的数据输出
蒸馏数据
108
对 PDE_i 的引用+
Reconstitution_Program_i
基本
导出项
对 PDE_j
的引用
对 PDE_i 的引用+
对 PDE_j 的引用+
Reconstitution_Program_i
119A
对 PDE_i
的引用
基本
PDE=基本数据单元

图6A

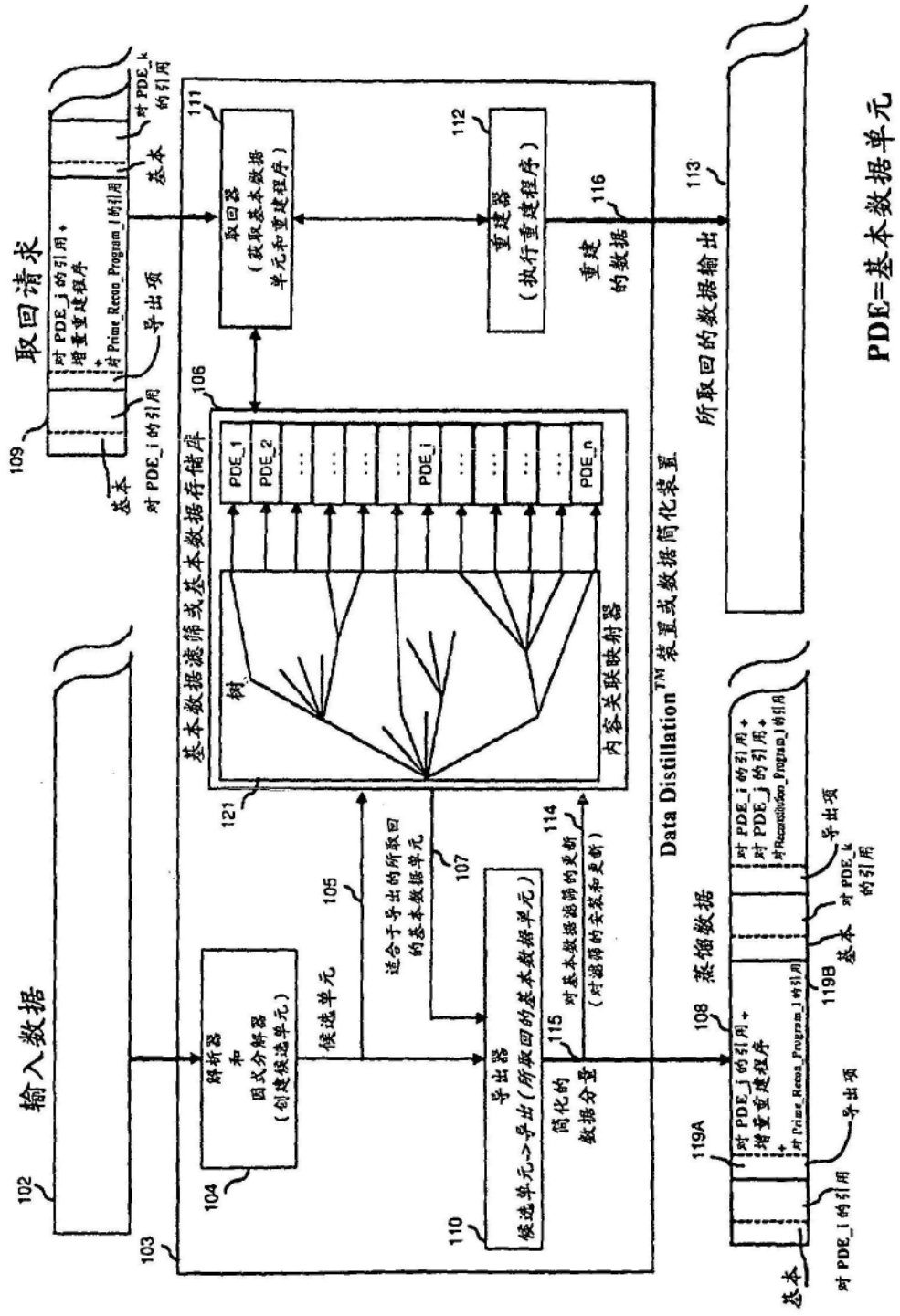

取回请求
对 PDE_j 的引用 +
增量重建程序
+
对 Prime_Recon_Program_1 的引用
基本
对 PDE_k 的引用
导出项
对 PDE_i 的引用
109
基本
对 PDE_i 的引用
106
基本数据筛选或基本数据存储库
PDE_1
PDE_2
PDE_j
PDE_n
树
内容关联映射器
121
111
取回器
（获取基本数据单元和重建程序）
112
重建器
（执行重建程序）
116
重建的数据
113
所取回的数据输出
PDE=基本数据单元
输入数据
102
103
解析器和因式分解器（创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据筛选的更新（对选择的实发和更新）
114
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
对 PDE_j 的引用 +
增量重建程序
+
对 Prime_Recon_Program_1 的引用
119A
基本
对 PDE_i 的引用
导出项
对 PDE_i 的引用 +
对 PDE_j 的引用 +
对 Reconstruction_Program_1 的引用
导出项
对 PDE_k 的引用
基本
119B

图6B

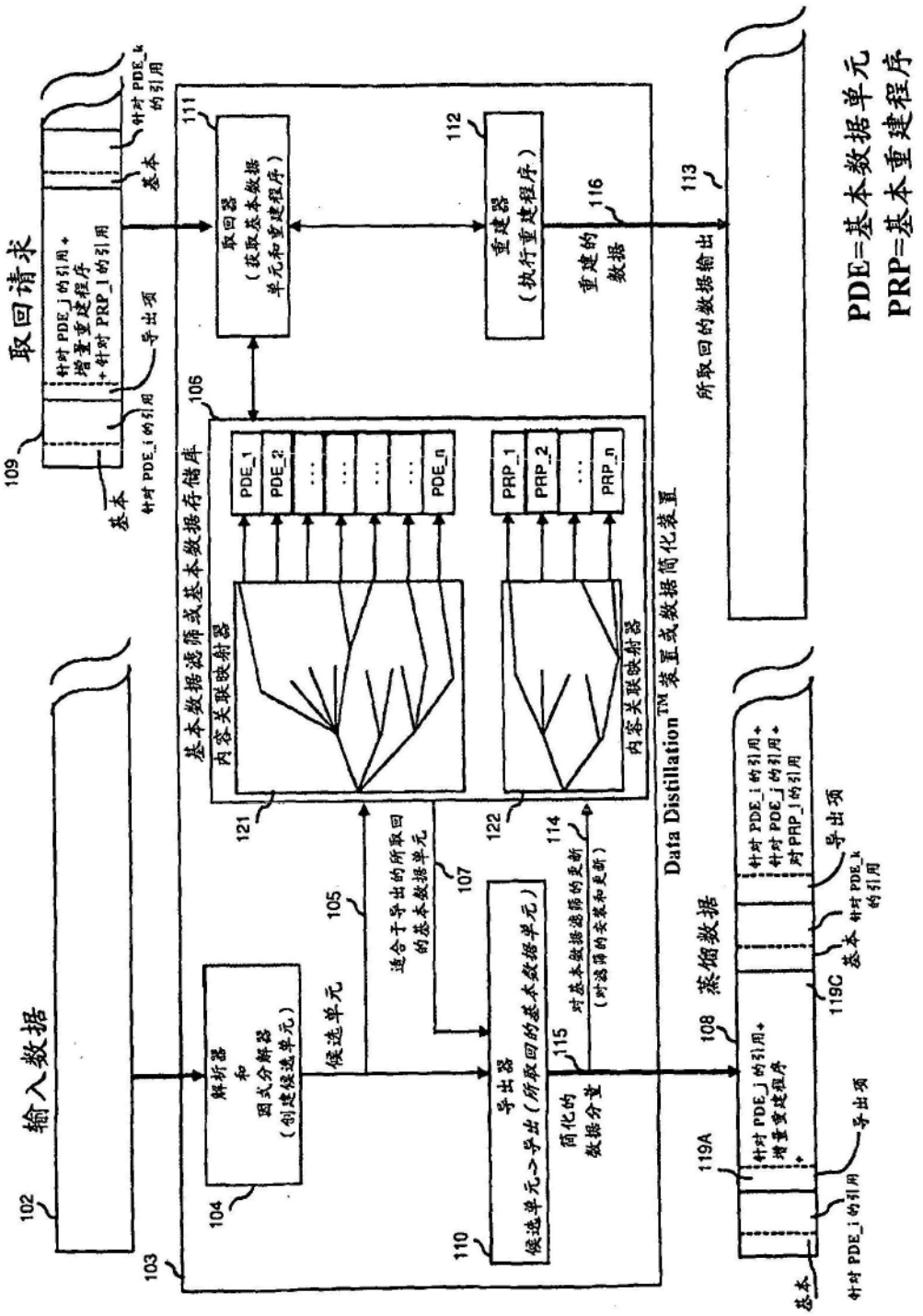

取回请求
针对 PDE_j 的引用+
增量重建程序
+针对 PRP_j 的引用
导出项
针对 PDE_k 的引用
基本
针对 PDE_i 的引用
109
输入数据
102
103
解析器
和
因式分解器
（创建候选单元）
104
候选单元
105
基本数据选择或基本数据存储库
106
内容关联映射器
PDE_1
PDE_2
PDE_n
PRP_1
PRP_2
PRP_n
121
122
适合于导出的所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
110
简化的数据分量
115
对基本数据选择的更新
（对选择的实施和更新）
114
取回器
（获取基本数据单元和重建程序）
111
重建器
（执行重建程序）
112
重建的数据
116
所取回的数据输出
113
Data Distillation™ 装置或数据简化装置
蒸馏数据
108
针对 PDE_i 的引用+
增量重建程序
119A
针对 PDE_i 的引用+
针对 PDE_j 的引用+
对 PRP_j 的引用
119C
PDE=基本数据单元
PRP=基本重建程序

图6C

图 7A 由重建程序对于操作所使用的指令编码的实例

操作	操作码	操作数
插入	0	原始偏移量，插入长度，插入数据（“长度”字节）
删除	1	原始偏移量，长度
替换	2	原始偏移量，替换数据（1字节）
批量替换	3	原始偏移量，长度，替换数据（“长度”字节）
附加	4	长度，附加数据（“长度”字节）
乘法	5	原始偏移量，乘积长度，乘数长度，乘数值
加法	6	原始偏移量，和长度，加数长度，加数值
扩展操作码	7	参见扩展操作表

扩展操作	操作码	范围	操作和操作数
算术	0	单一 PDE	操作，原始偏移量，操作长度，运算符长度，运算符
多单元替换 （用来自第2单元的数据替换）	1	多个 PDE	第2单元句柄，第2单元偏移量，原始偏移量，替换长度
多单元插入（插入来自第2单元的数据）	2	多个 PDE	第2单元句柄，第2单元偏移量，原始偏移量，插入长度
调用和执行内联基本重建程序 （从规定地址获取重建程序，并且 在单一基本数据单元上执行）	3	单一 PDE	对所调用的基本重建程序的引用
获取、修改和执行基本重建程序 （从规定地址获取重建程序，在后续 窗口中应用操作以便导出新的 重建程序，并且随后执行）	4	重建程序	对基本重建程序的引用，窗口
保留	5-7	保留	保留

字段类型	字段大小
操作码或操作	3比特
偏移量	5比特
长度	5比特
引用	24比特
句柄	2比特

注：PDE= 基本数据单元

图7A

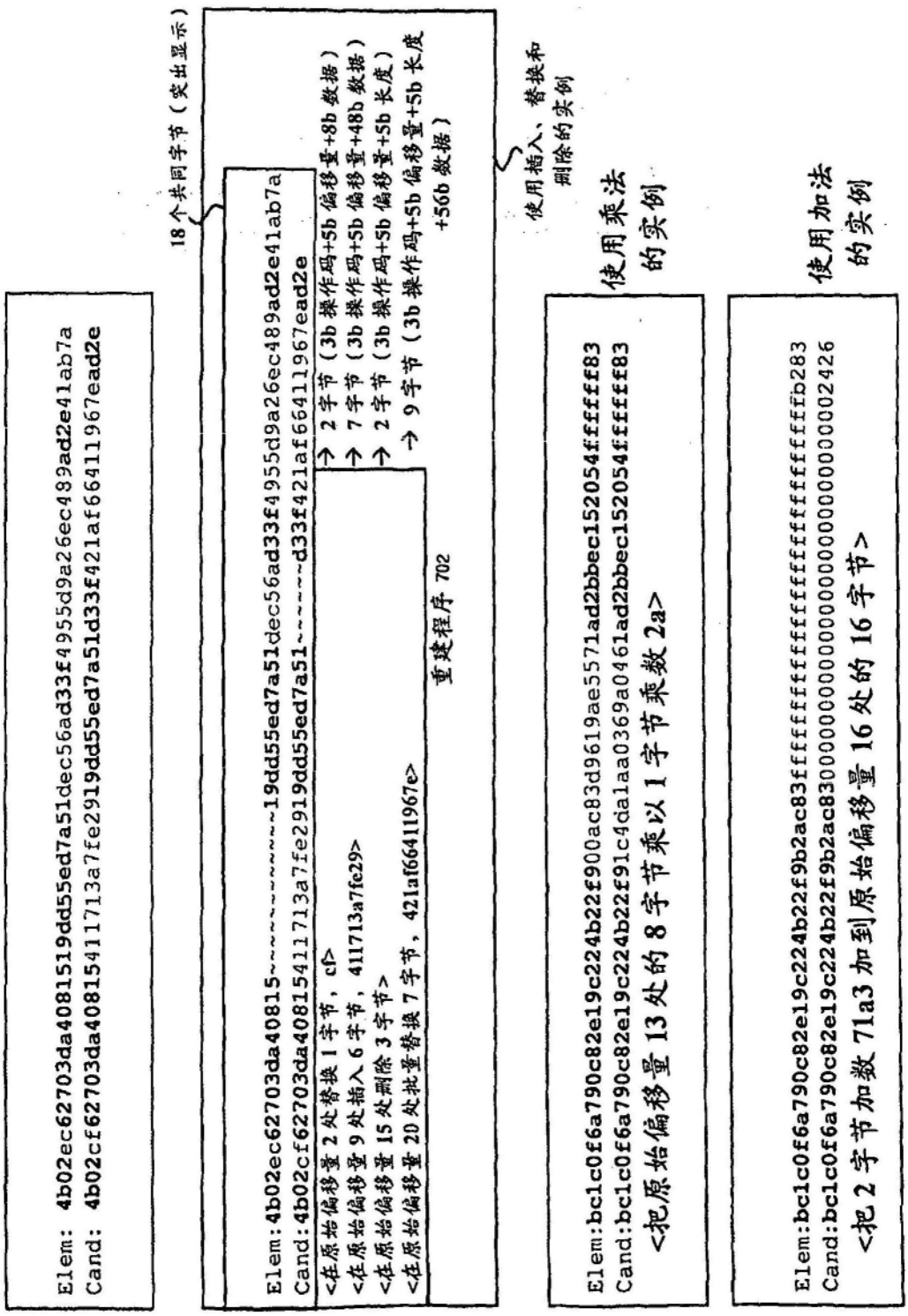

Elem: 4b02ec62703da4081519dd55ed7a51dec56ad33f4955d9a26ec489ad2e41ab7a
Cand: 4b02cf62703da40815411713a7fe2919dd55ed7a51d33f421af66411967ead2e
18个共同字节（突出显示）
Elem:4b02ec62703da40815~~~~~~~~~~19dd55ed7a51dec56ad33f4955d9a26ec489ad2e41ab7a
Cand:4b02cf62703da4081541171 3a7fe2919dd55ed7a51~~~~~d33f421af66411967ead2e
<在原始偏移量2处替换1字节，cf>
<在原始偏移量9处插入6字节，411713a7fe29>
<在原始偏移量15处删除3字节>
<在原始偏移量20处批量替换7字节，421af66411967e>
→ 2字节（3b操作码+5b偏移量+8b数据）
→ 7字节（3b操作码+5b偏移量+48b数据）
→ 2字节（3b操作码+5b偏移量+5b长度）
→ 9字节（3b操作码+5b偏移量+5b长度+56b数据）
重建程序 702
使用插入、替换和删除的实例
Elem:bc1c0f6a790c82e19c224b22f900ac83d9619ae5571ad2bbec152054ffffff83
Cand:bc1c0f6a790c82e19c224b22f91c4da1aa0369a0461ad2bbec152054ffffff83
<把原始偏移量13处的8字节乘以1字节乘数2a>
使用乘法的实例
Elem:bc1c0f6a790c82e19c224b22f9b2ac83ffffffffffffffffffffffffffffb283
Cand:bc1c0f6a790c82e19c224b22f9b2ac830000000000000000000000000002426
<把2字节加数71a3加到原始偏移量16处的16字节>
使用加法的实例

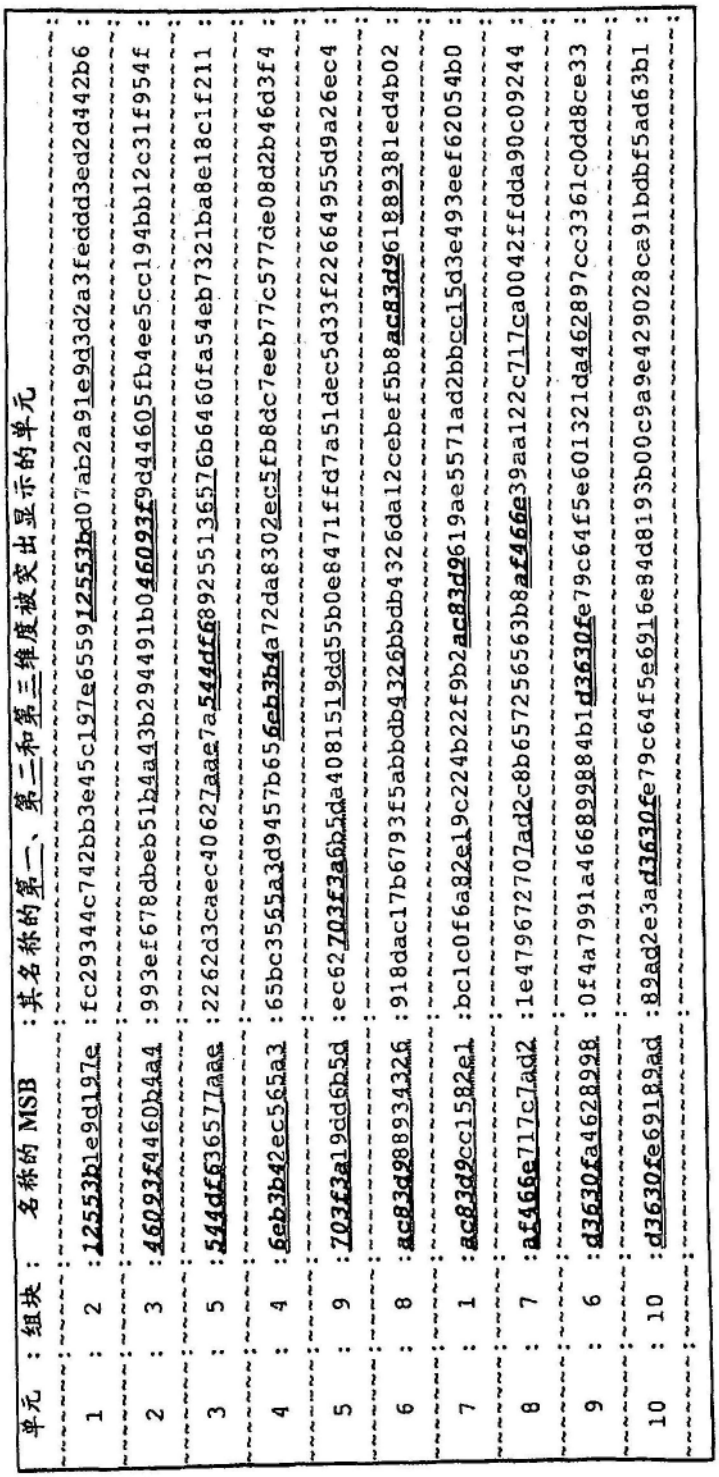
图7B

bc1c0f6a82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0 : 组块 1
fc29344c742bb3e45c197e655912553bd07ab2a91e9d3d2a3feddd3ed2d442b6 : 组块 2
993ef678dbeb51b4a43b294491b046093f9d44605fb4ee5cc194bb12c31f954f : 组块 3
65bc3565a3d9457b656eb3b4a72da8302ec5fb8dc7eeb77c577de08d2b46d3f4 : 组块 4
2262d3caec40627aae7a544df689255136576b6460fa54eb7321ba8e18c1f211 : 组块 5
0f4a7991a466899884b1d3630fe79c64f5e601321da462897cc3361c0dd8ce33 : 组块 6
1e479672707ad2c8b657256563b8af466e39aa122c717ca0042ffdda90c09244 : 组块 7
918dac17b6793f5abbdb4326bbdb4326da12cebef5b8ac83d961889381ed4b02 : 组块 8
ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f22664955d9a26ec4 : 组块 9
89ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b1 : 组块 10

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ 中间有 4200 万个组块 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f22664955d9a26ec4 : 组块 42,000,011
bc1c0f6a82790ce19c224b22f9b2ac83d9619ae5571ad2bbec152054b0aeb712 : 组块 42,000,012
67b289ad2e41d3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad : 组块 42,000,013
0f4a7991a466899884b1d3630fe79c64f5e6916e84d8193b00c9a9e429028ca9 : 组块 42,000,014

图8A
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



单元	组块	名称的 MSB	共名称的第一、第二和第三维度被突出显示的单元
1	2	12553b1e9d197e	fc29344c742bb3e45c197e6559125553bd07ab2a91e9d3d2a3feddd3ed2d442b6
2	3	46093f4460b4a4	993ef678dbeb51b4a43b294491b046093f9d44605fb4ee5cc194bb12c31f954f
3	5	544df63657aaa	2262d3caec40627aae7a544df6892551365 76b6460fa54eb7321ba8e18c1f211
4	4	6eb3b42ec565a3	65bc3565a3d9457b656eb3b4a72da8302ec5fb8dc7eeb77c577de08d2b46d3f4
5	9	703f3a19dd6b5d	ec62703f3a6b5da408151 9dd55b0e8471ffd7a51dec5d33f22664955d9a26ec4
6	8	ac83d2888934326	918dac17b6793f5abbdb4326bbdb4326da12cebef5b8ac83d96188938 1ed4b02
7	1	ac83d2cc1582a1	bc1c0f6a82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0
8	7	af466e717c7ad2	1e479672707ad2c8b657256563b8af466e39aa122c717ca0042ffdda90c09244
9	6	d3630fa4628928	0f4a7991a466899884b1d3630fe79c64f5e601321da462897cc3361c0dd8ce33
10	10	d3630fe691189ad	89ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b1

图8B

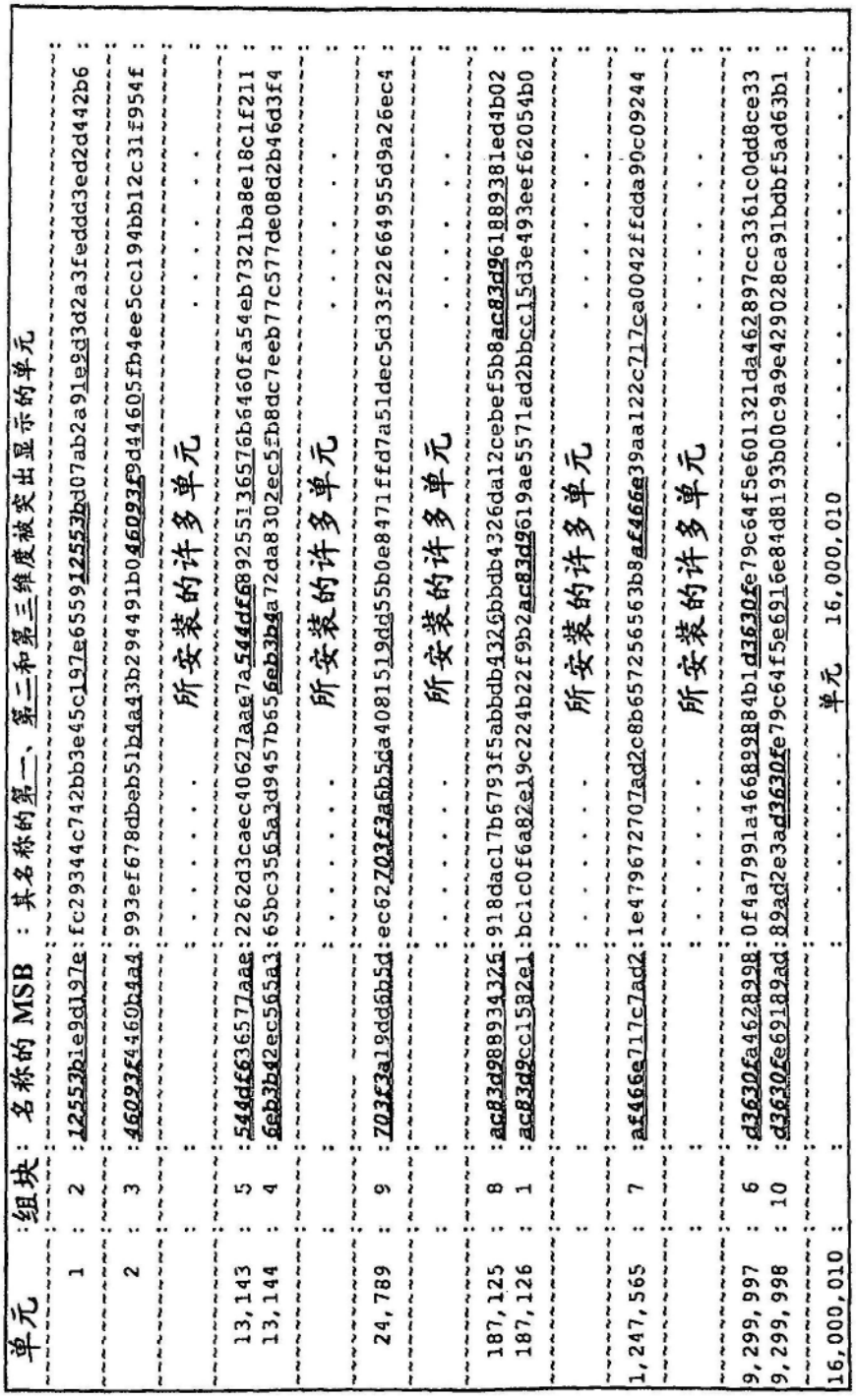

单元 :组块: 名称的 MSB : 其名称的第一、第二和第三维度被突出显示的单元
1 : 2 : 12553b1e9d197e:fc29344c742bb3e45c197e655912553bd07ab2a91e9d3d2a3feddd3ed2d442b6 :
2 : 3 : 46093f4460b4a4:993ef678dbeb51b4a43b294491b046093f9d44605fb4ee5cc194bb12c31f954f :
: : . . . . . 所安装的许多单元 . . . . . :
13,143 : 5 : 544df63657aae:2262d3caec40627aae7a544df68925513657 6b6460fa54eb7321ba8e18c1f211 :
13,144 : 4 : 6eb3b42ec565a3:65bc3565a3d9457b656eb3b4a72da8302ec5fb8dc7eeb77c577de08d2b46d3f4 :
: : . . . . . 所安装的许多单元 . . . . . :
24,789 : 9 : 703f3a19dd6b5d:ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f22664955d9a26ec4 :
: : . . . . . 所安装的许多单元 . . . . . :
187,125 : 8 : ac83d988934325:918dac17b6793f5abbdb4326bbdb4326da12cebef5b8ac83d96188938 1ed4b02 :
187,126 : 1 : ac83d9cc1582e1:bc1c0f6a82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0 :
: : . . . . . 所安装的许多单元 . . . . . :
1,247,565 : 7 : af466e717c7ad2:1e479672707ad2c8b657256563b8af466e39aa122c717ca0042ffdda90c09244 :
: : . . . . . 所安装的许多单元 . . . . . :
9,299,997 : 6 : d3630fa462829 8:0f4a7991a4668929884b1d3630fe79c64f5e601321da462897cc3361c0dd8ce33 :
9,299,998 : 10 : d3630fe691 82ad:82ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b1 :
16,000,010 : : . . . . . 单元 16,000,010 . . . . . :

图8C

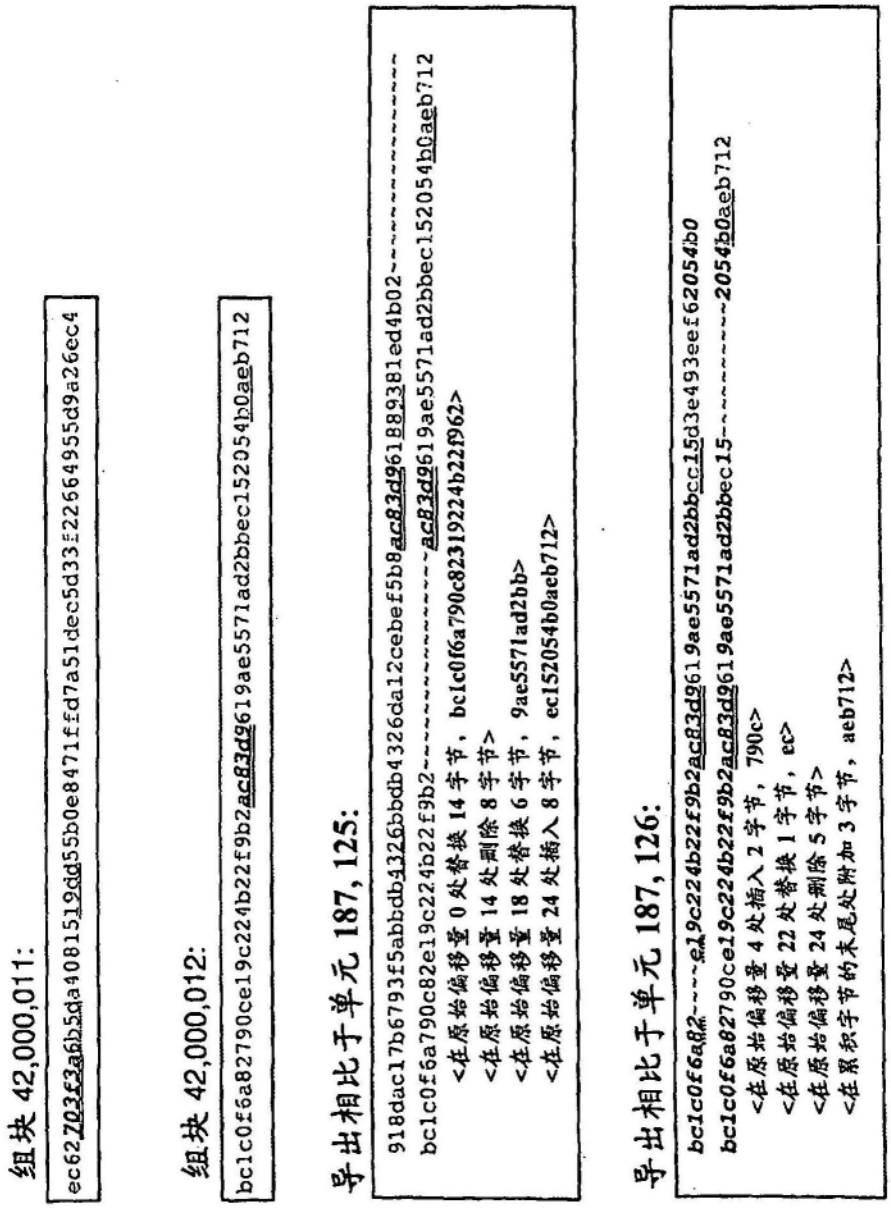

组块 42,000,011:
ec62703f3a6b5da408151 9dd55b0e8471ffd7a51dec5d33f22664955d9a26ec4
组块 42,000,012:
bc1c0f6a82790ce19c224b22f9b2ac83d9619ae5571ad2bbec152054b0aeb712
导出相比于单元 187,125:
918dac17b6793f5abbdb4326bbdb4326da12cebef5b8ac83d96188938 1ed4b02~~~~~~~~~~~~~~~~~~
bc1c0f6a790c82e19c224b22f9b2~~~~~~~~~~~~~~~~~~ac83d9619ae5571ad2bbec152054b0aeb712
<在原始偏移量 0 处替换 14 字节，bc1c0f6a790c823 19224b22f962>
<在原始偏移量 14 处删除 8 字节>
<在原始偏移量 18 处替换 6 字节，9ae5571ad2bb>
<在原始偏移量 24 处插入 8 字节，ec152054b0aeb712>
导出相比于单元 187,126:
bc1c0f6a82~~~~e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0
bc1c0f6a82790ce19c224b22f9b2ac83d9619ae5571ad2bbec15~~~~~~~~~~2054b0aeb712
<在原始偏移量 4 处插入 2 字节，790c>
<在原始偏移量 22 处替换 1 字节，ec>
<在原始偏移量 24 处删除 5 字节>
<在累积字节的末尾处附加 3 字节，aeb712>

图8D

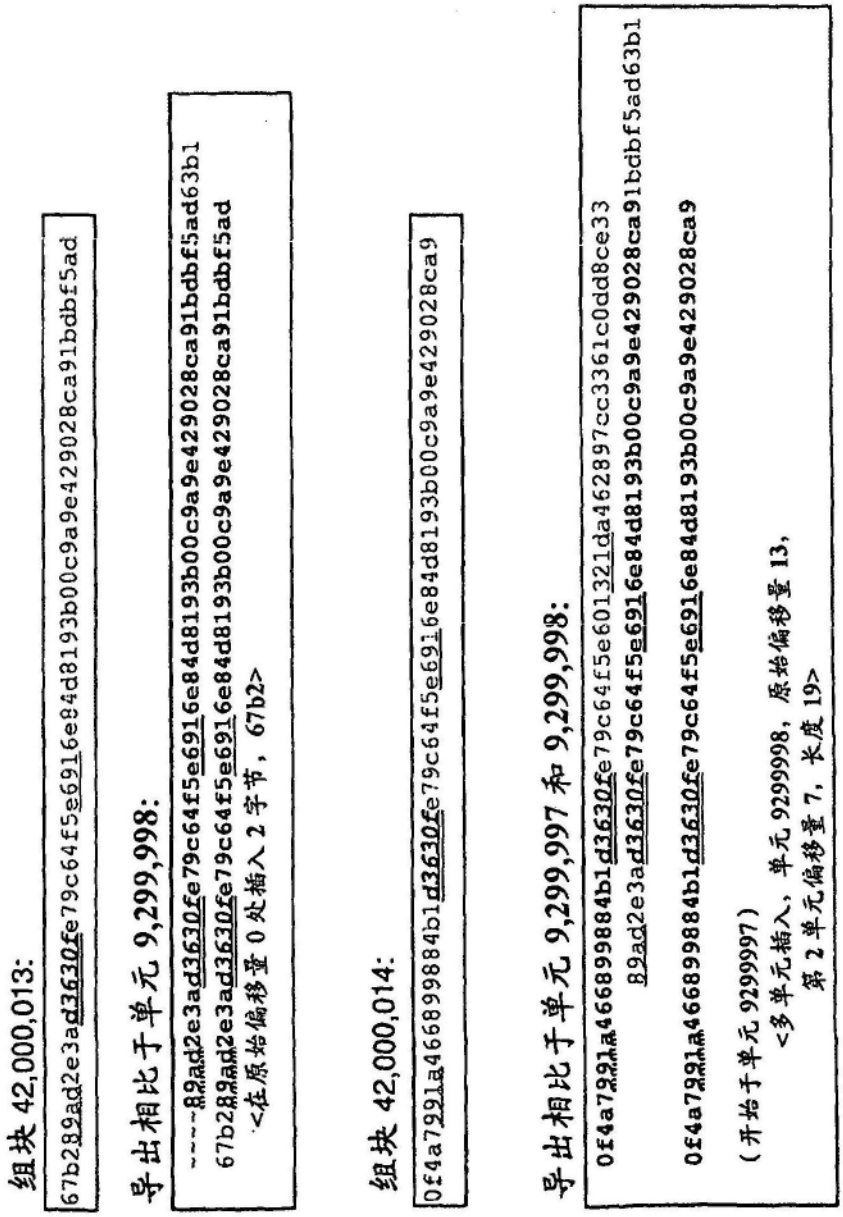

组块 42,000,013:
67b289ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad
导出相比于单元 9,299,998:
----89ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b1
67b289ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad
<在原始偏移量 0 处插入 2 字节，67b2>
组块 42,000,014:
0f4a7291a466899884b1d3630fe79c64f5e6916e84d8193b00c9a9e429028ca9
导出相比于单元 9,299,997 和 9,299,998:
0f4a7291a466899884b1d3630fe79c64f5e601321da462897cc3361c0dd8ce33
89ad2e3ad3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b1
0f4a7291a466899884b1d3630fe79c64f5e6916e84d8193b00c9a9e429028ca9
(开始于单元 9299997)
<多单元插入，单元 9299998，原始偏移量 13，
第 2 单元偏移量 7，长度 19>
(开始于单元 9299997)

图8E

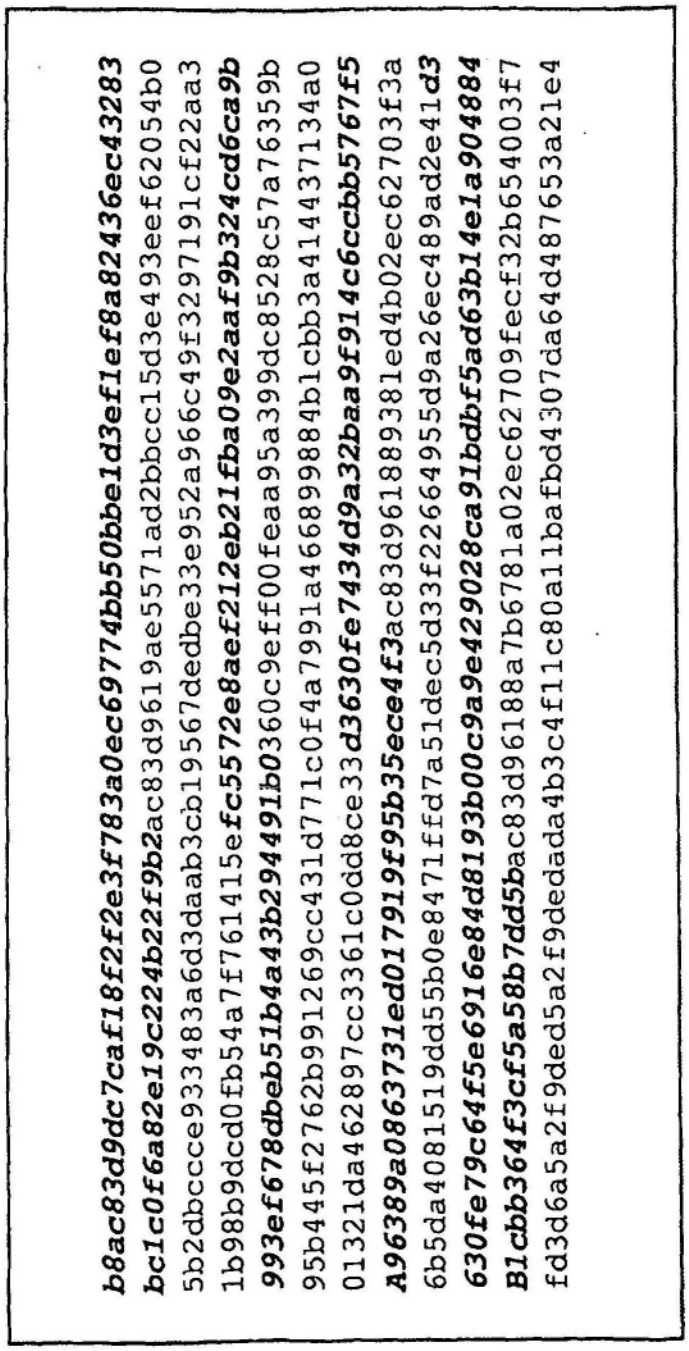

b8ac83d9dc7caf18f2f2e3f783a0ec69774bb50bbe1d3ef1ef8a82436ec43283
bc1c0f6a82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0
5b2dbccce933483a6d3daab3cb19567dedbe33e952a966c49f3297191cf22aa3
1b98b9dcd0fb54a7f761415efc5572e8aef212eb21fba09e2aaf9b324cd6ca9b
993ef678dbeb51b4a43b2944491b0360c9eff00feaa95a399dc8528c57a76359b
95b445f2762b991269cc431d771c0f4a7991a46689988 4b1cbb3a41443713 4a0
01321da462897cc3361c0dd8ce33d3630fe7434d9a32baa9f914c6ccbb5767f5
A96389a0863731ed017919f95b35ece4f3ac83d961889381ed4b02ec62703f3a
6b5da4081519dd55b0e8471ffd7a51dec5d33f22664955d9a26ec489ad2e41d3
630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b14e1a904884
B1cbb364f3cf5a58b7dd5bac83d96188a7b6781a02ec62709fecf32b654003f7
fd3d6a5a2f9ded5a2f9dedada4b3c4f11c80a11bafbd4307da64d487653a21e4

图9A

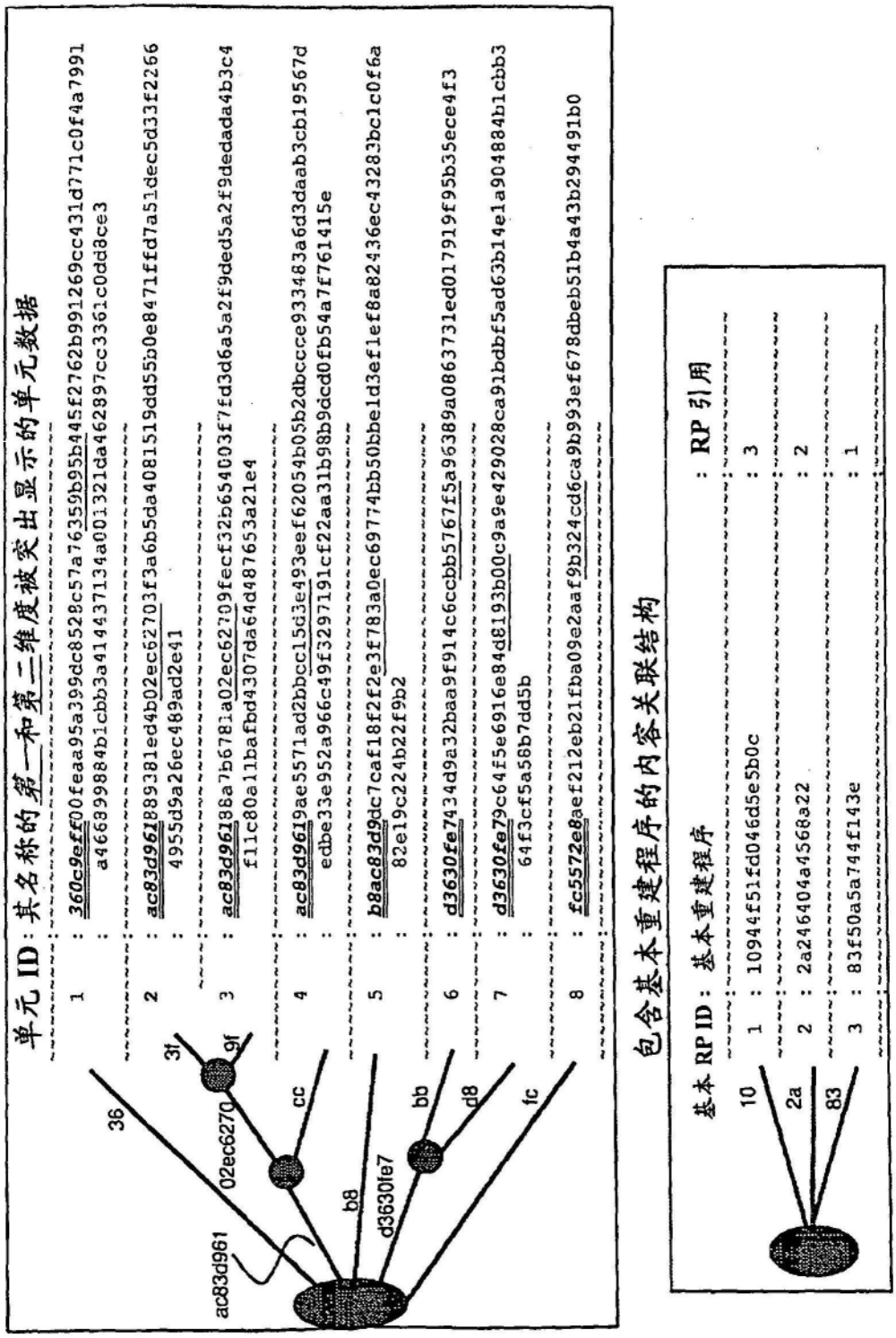

滤筛
单元 ID：共名称的第一和第二维度被突出显示的单元数据
1 : 360c9eff00feaa95a399dc8528c57a76359b95b445f2762b991269cc431d771c0f4a7991
: a4663999884b1cbb3a41443713 4a001321da462897cc3361c0dd8ce3
2 : ac83d9618899381ed4b02ec62703f3a6b5da4081519dd55b0e8471ffd7a51dec5d33f2266
: 4955d9a26ec489ad2e41
3 : ac83d9618 6a7b6781a02ec62709fecf32b654003f7fd3d6a5a2f9ded5a2f9dedada4b3c4
: f11c80a11bafbd4307da6f4d487653a21e4
4 : ac83d9619ae5571ad2bbcc15d3e493eef62054b05b2dbcccc933483a6d3daab3cb19567d
: edbe33e952a966c49f3297191cf22aa31b98b9dcd0fb54a7f761415e
5 : b8ac83d9dc7caf18f2f2e3f783a0ec69774bb50bbe1d3ef1ef8a82436ec43283bc1c0f6a
: 82e19c224b22f9b2
6 : d3630fe7434d9a32baa9f914c6ccbb5767f5a96389a0863731ed017919f95b35ece4f3
7 : d3630fe79c64f5e6916e84d8193b00c9a9e429028ca91bdbf5ad63b14e1a904884b1cbb3
: 64f3cf5a58b7dd5b
8 : fc5572e8aef212eb21fba09e2aaf9b324cd6ca9b993ef678dbeb51b4a43b294491b0
包含基本重建程序的内容关联结构
基本 RP ID： 基本重建程序 : RP 引用
1 : 10944f51fd046d5e5b0c : 3
2 : 2a246404a45568a22 : 2
3 : 83f50a5a744f143e : 1
36
3f
9f
02ec6270
cc
b8
ac83d961
d3630fe7
bb
d8
fc
10
2a
83

图9B

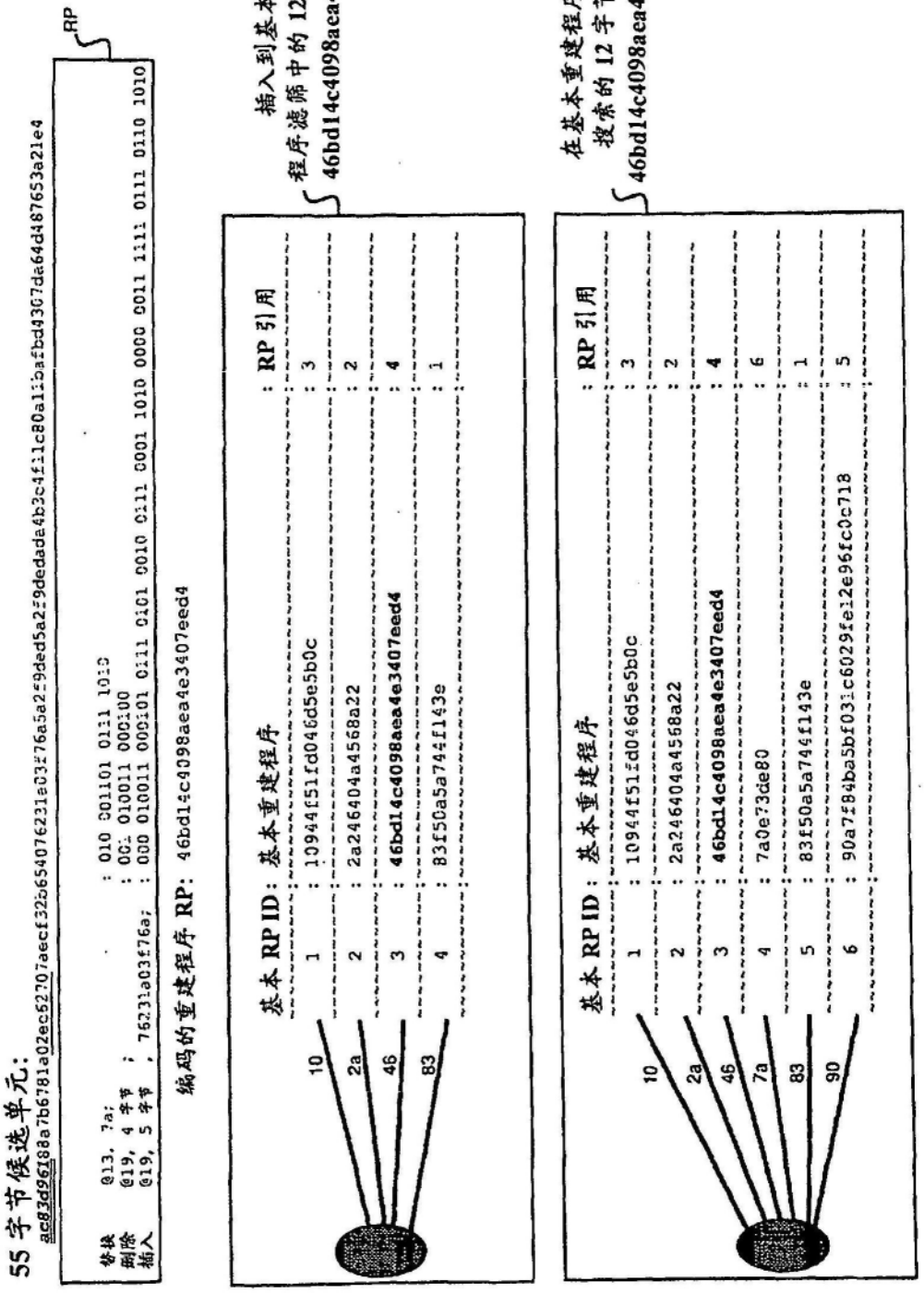

55字节候选单元:
ac83d96188a7b6781a02ec52707aecf32b654076231e03f76a5a2f9ded5a2f9dedade4b3c4f11c80a1ibafbd4367da64d487653a21e4
RP
替换 013, 7a;
删除 019, 4 字节 ;
插入 019, 5 字节 , 76231a03f76a;
010 001101 0111 1013
002 010011 000100
000 010011 000101 0111 0101 0010 0111 0001 1010 0000 0011 1111 0111 0110 1010
编码的重建程序 RP: 46bd14c4098aea4e3407eed4
基本 RP ID: 基本重建程序 : RP 引用
1 : 10944f51fd046d5e5b0c : 3
2 : 2a246404a4568a22 : 2
3 : 46bd14c4098aea4e3407eed4 : 4
4 : 83f50a5a744f143e : 1
10
2a
46
83
插入到基本重建程序滤筛中的 12 字节 RP,
46bd14c4098aea4e3407eed4
基本 RP ID: 基本重建程序 : RP 引用
1 : 10944f51fd046d5e5b0c : 3
2 : 2a246404a4568a22 : 2
3 : 46bd14c4098aea4e3407eed4 : 4
4 : 7a0e73de80 : 6
5 : 83f50a5a744f143e : 1
6 : 90a7f84ba5bf03ic6029fe12e96fc0c718 : 5
10
2a
46
7a
83
90
在基本重建程序滤筛中搜索的 12 字节 RP,
46bd14c4098aea4e3407eed4

图9C

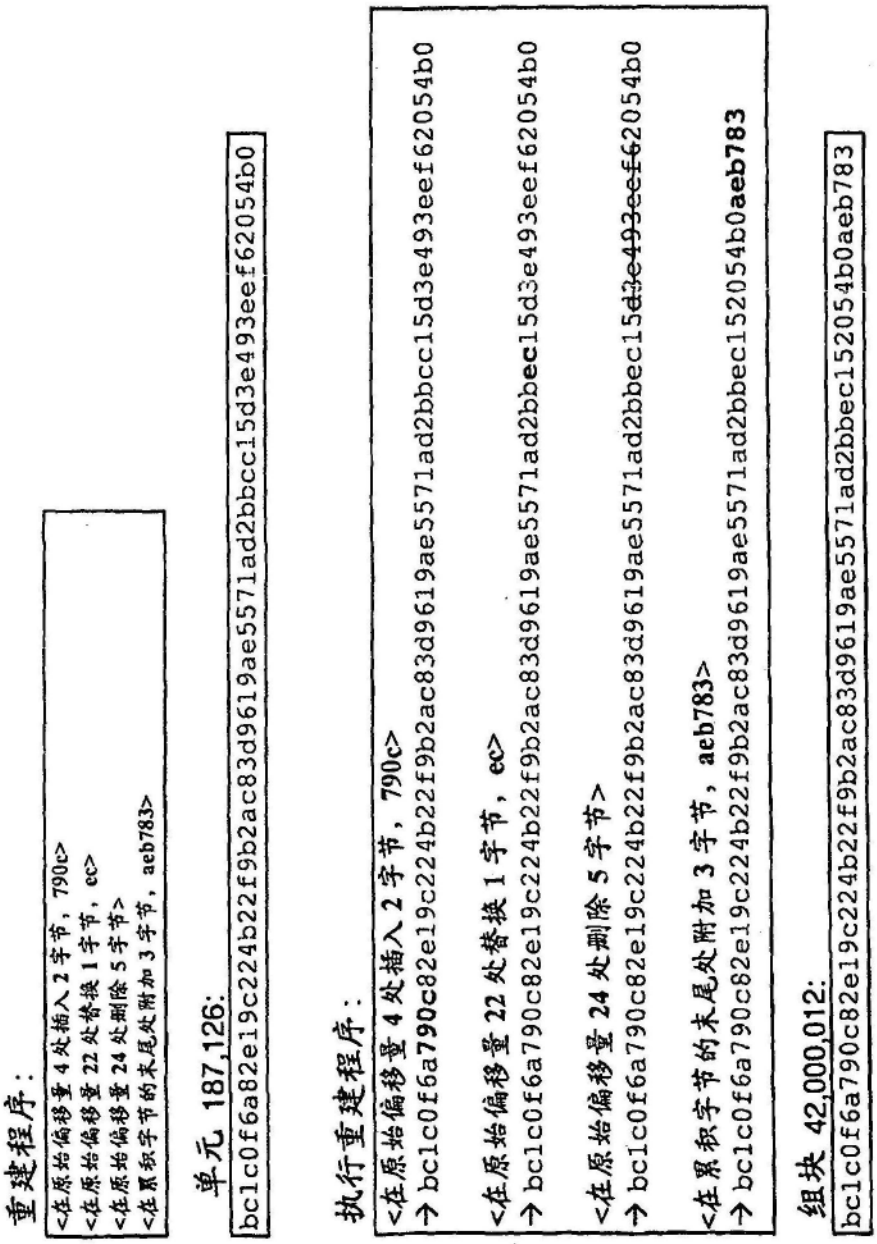

重建程序：
<在原始偏移量 4 处插入 2 字节，790c>
<在原始偏移量 22 处替换 1 字节，ec>
<在原始偏移量 24 处删除 5 字节>
<在累积字节的末尾处附加 3 字节，aeb783>
单元 187,126:
bc1c0f6a82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0
执行重建程序：
<在原始偏移量 4 处插入 2 字节，790c>
→ bc1c0f6a790c82e19c224b22f9b2ac83d9619ae5571ad2bbcc15d3e493eef62054b0
<在原始偏移量 22 处替换 1 字节，ec>
→ bc1c0f6a790c82e19c224b22f9b2ac83d9619ae5571ad2bbec15d3e493eef62054b0
<在原始偏移量 24 处删除 5 字节>
→ bc1c0f6a790c82e19c224b22f9b2ac83d9619ae5571ad2bbec152054b0
<在累积字节的末尾处附加 3 字节，aeb783>
→ bc1c0f6a790c82e19c224b22f9b2ac83d9619ae5571ad2bbec152054b0aeb783
组块 42,000,012:
bc1c0f6a790c82e19c224b22f9b2ac83d9619ae5571ad2bbec152054b0aeb783

图10A

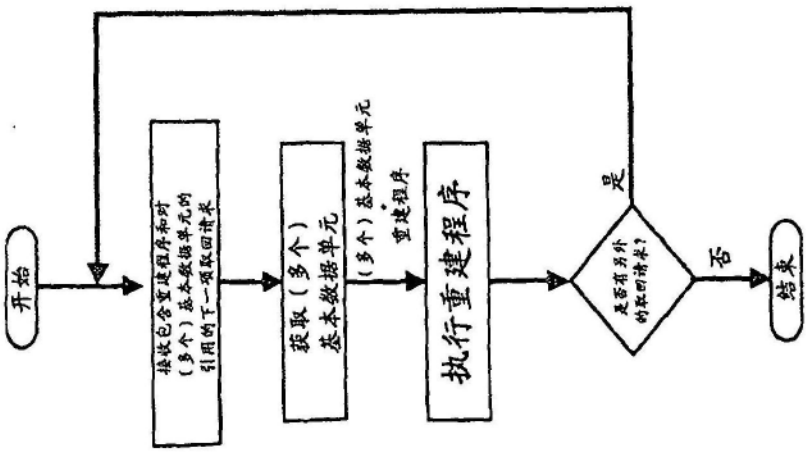

开始
接收包含对重建程序和对（多个）基本数据单元的引用的下一项取回请求
获取（多个）基本数据单元
（多个）基本数据单元
重建程序
执行重建程序
是否有另外的取回请求?
是
否
结束

图10B

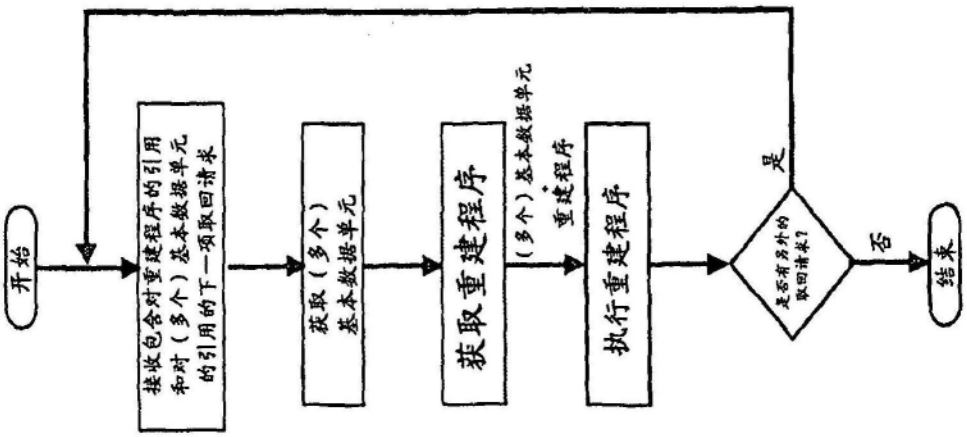

开始
接收包含对重建程序的引用和对（多个）基本数据单元的引用的下一项取回请求
获取（多个）基本数据单元
获取重建程序
（多个）基本数据单元
重建程序
执行重建程序
是否有另外的取回请求?
是
否
结束

图10C

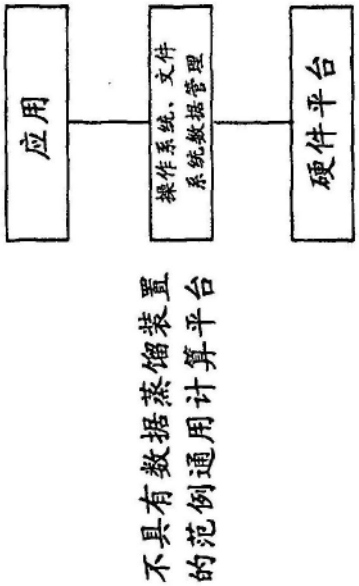
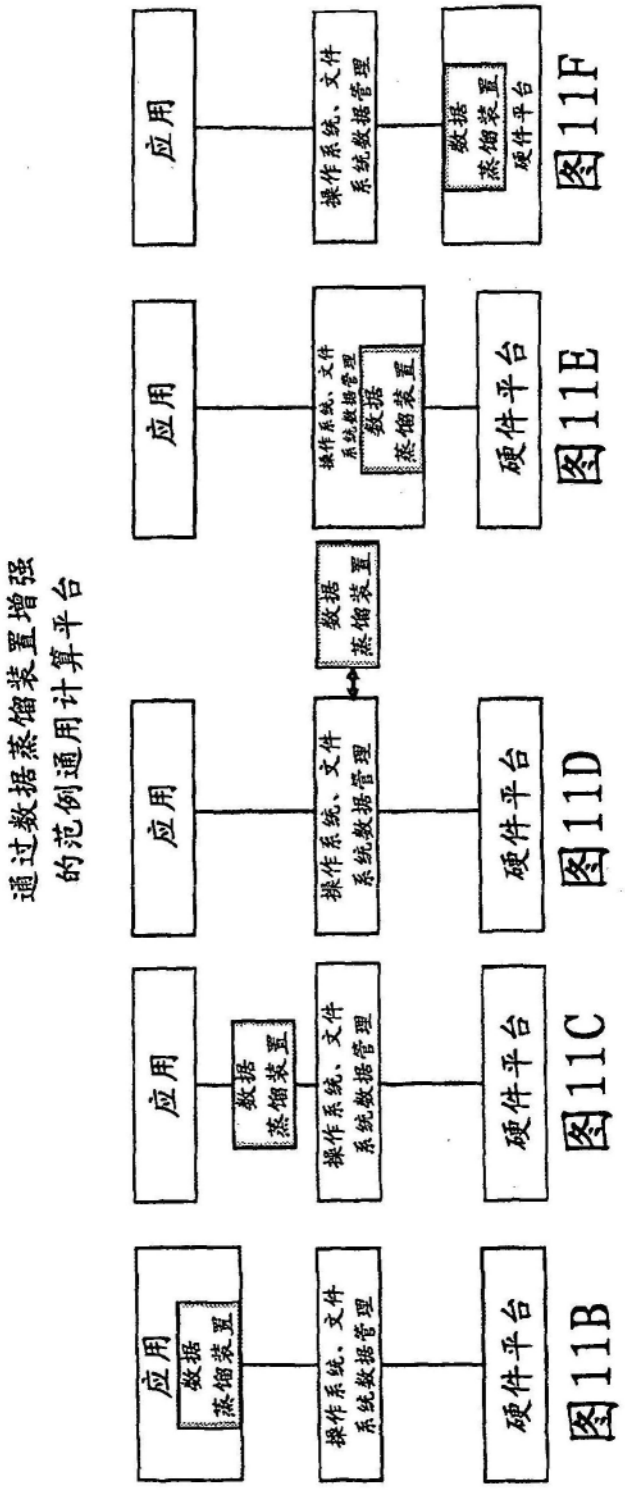


图11A


通过数据蒸馏装置增强
的范例通用计算平台
应用
数据
蒸馏装置
操作系统、文件
系统数据管理
硬件平台
图11B
图11C
图11D
图11E
图11F

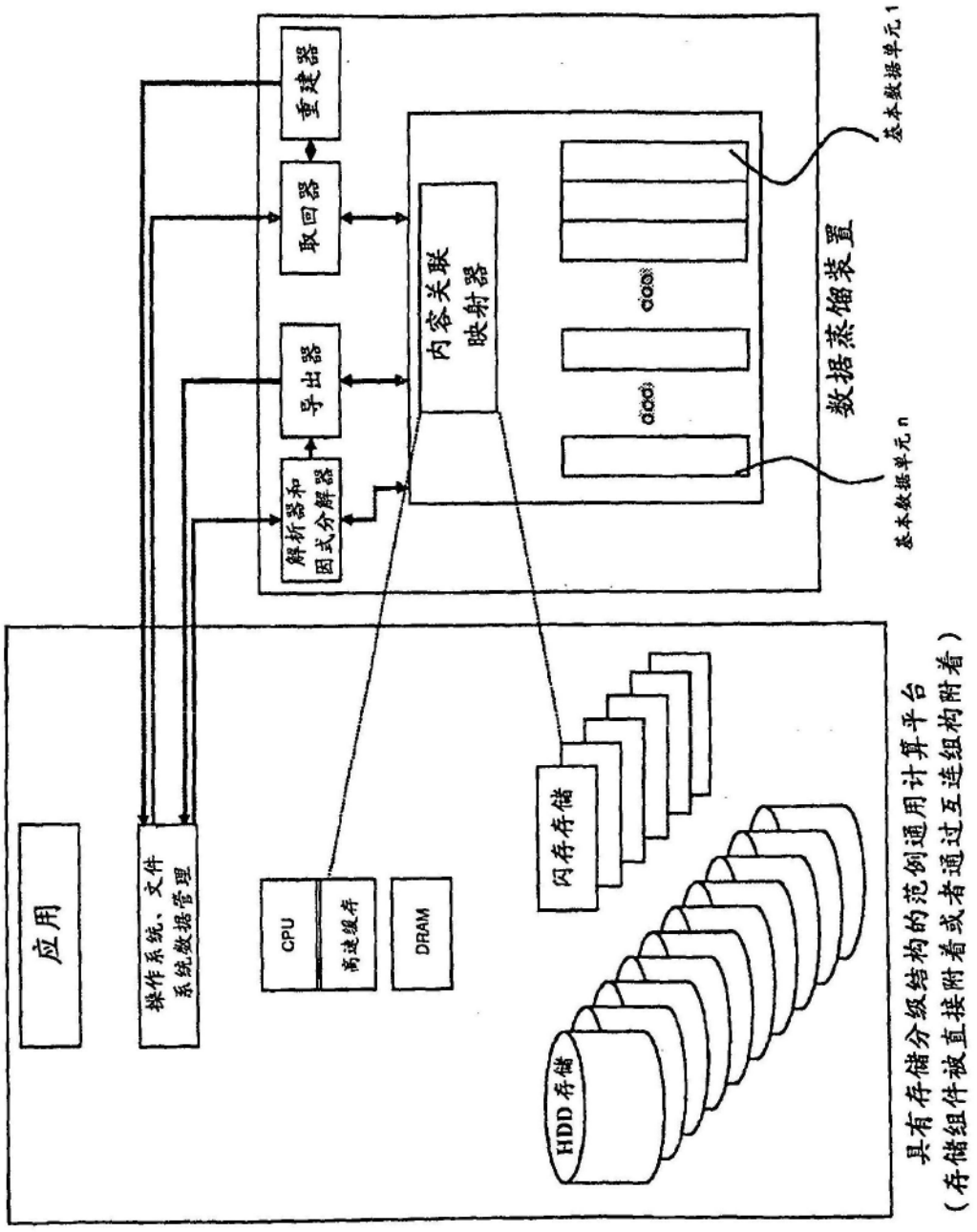

重建器
取回器
导出器
解析器和
因式分解器
内容关联
映射器
数据蒸馏装置
基本数据单元 1
基本数据单元 n
应用
操作系统、文件
系统数据管理
CPU
高速缓存
DRAM
闪存存储
HDD 存储
具有存储分级结构的范例通用计算平台
（存储组件被直接附着或者通过互连组构附着）

图11G

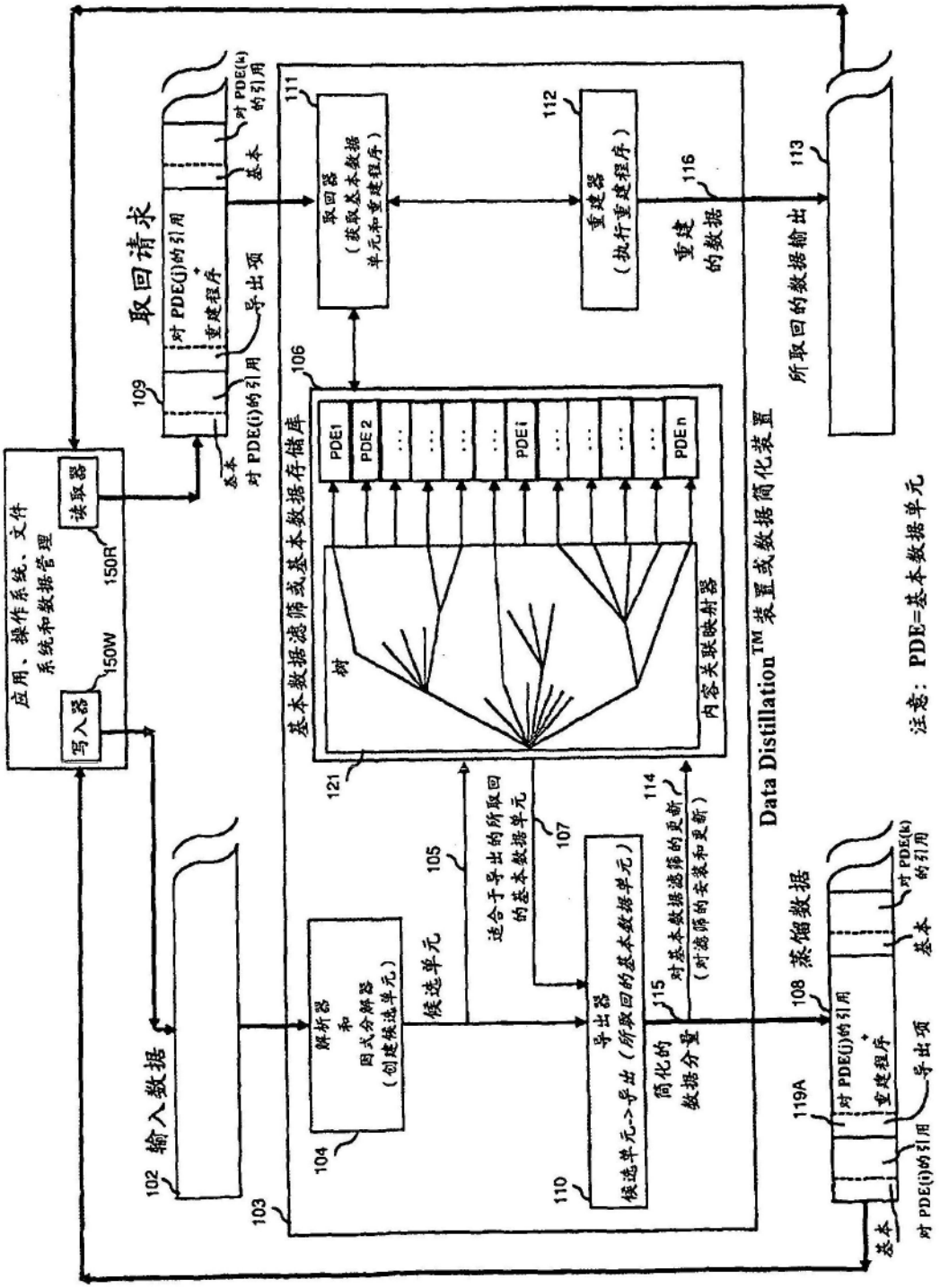
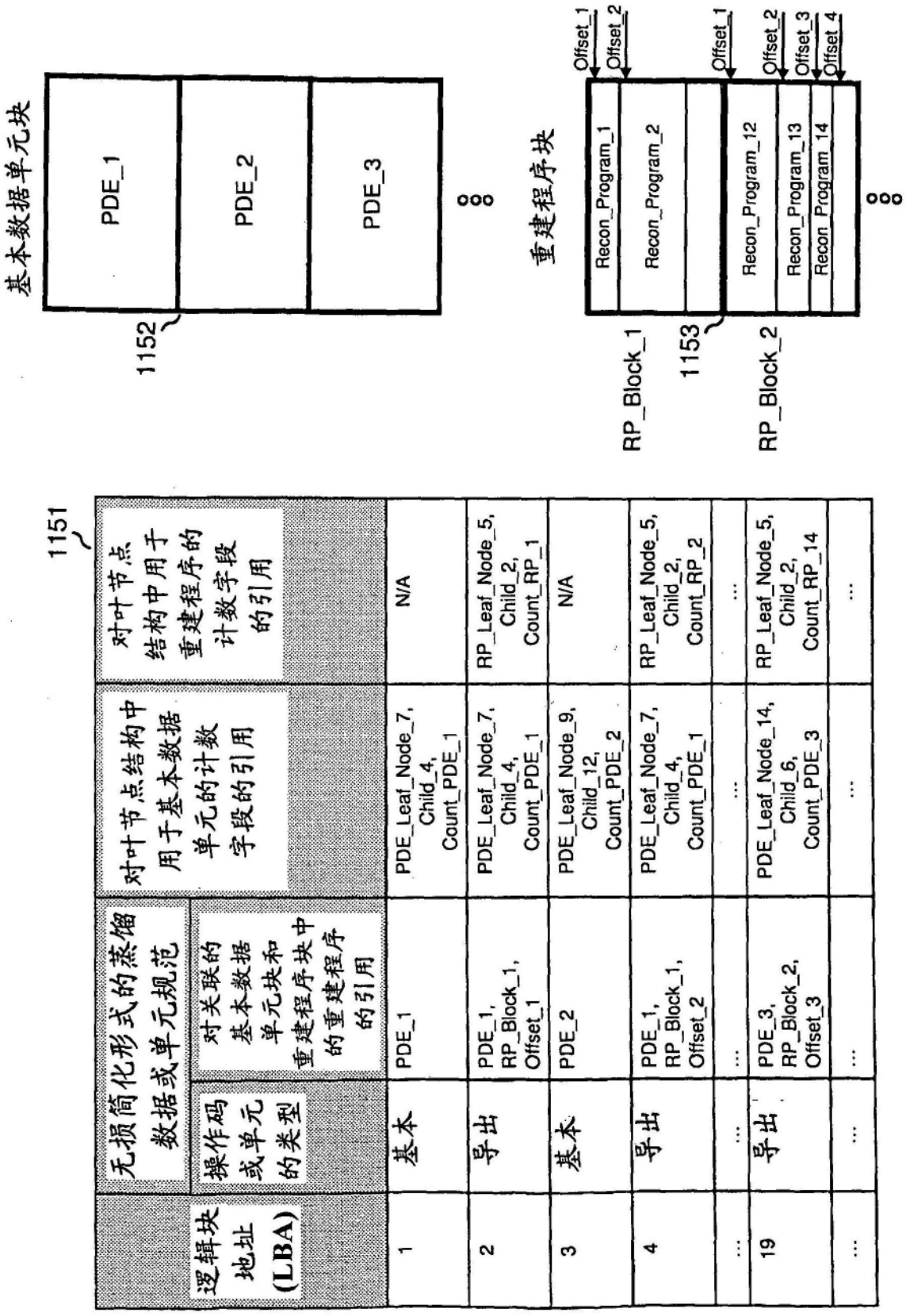

应用、操作系统、文件
系统和数据管理
写入器
150W
读取器
150R
取回请求
对 PDE(j)的引用
重建程序
导出项
对 PDE(i)的引用
基本
对 PDE(k)的引用
109
111
取回器
（获取基本数据单元和重建程序）
112
重建器
（执行重建程序）
116
重建的数据
113
所取回的数据输出
基本数据滤筛或基本数据存储库
106
PDE1
PDE 2
PDEi
PDEn
树
内容关联映射器
Data Distillation™ 装置或数据简化装置
121
输入数据
102
103
解析器
和
因式分解器
（创建候选单元）
104
候选单元
105
适合于导出的所取回的基本数据单元
107
导出器
候选单元->导出（所取回的基本数据单元）
简化的数据分量
110
115
对基本数据滤筛的更新
（对滤筛的安装和更新）
114
蒸馏数据
108
119A
对 PDE(j)的引用
重建程序
导出项
基本
对 PDE(k)的引用
对 PDE(i)的引用
基本
注意：PDE=基本数据单元

图11H



逻辑块地址 (LBA)	无损简化形式的蒸馏数据或单元规范		对叶节点结构中用于基本数据单元的计数字段的引用	对叶节点结构中用于重建程序的计数字段的引用
	操作码或单元的类型	对关联的基本数据单元块和重建程序块中的重建程序的引用		
1	基本	PDE_1	PDE_Leaf_Node_7, Child_4, Count_PDE_1	N/A
2	导出	PDE_1, RP_Block_1, Offset_1	PDE_Leaf_Node_7, Child_4, Count_PDE_1	RP_Leaf_Node_5, Child_2, Count_RP_1
3	基本	PDE_2	PDE_Leaf_Node_9, Child_12, Count_PDE_2	N/A
4	导出	PDE_1, RP_Block_1, Offset_2	PDE_Leaf_Node_7, Child_4, Count_PDE_1	RP_Leaf_Node_5, Child_2, Count_RP_2
...	...	...	...	...
19	导出	PDE_3, RP_Block_2, Offset_3	PDE_Leaf_Node_14, Child_6, Count_PDE_3	RP_Leaf_Node_5, Child_2, Count_RP_14
...	...	...	...	...

图11I

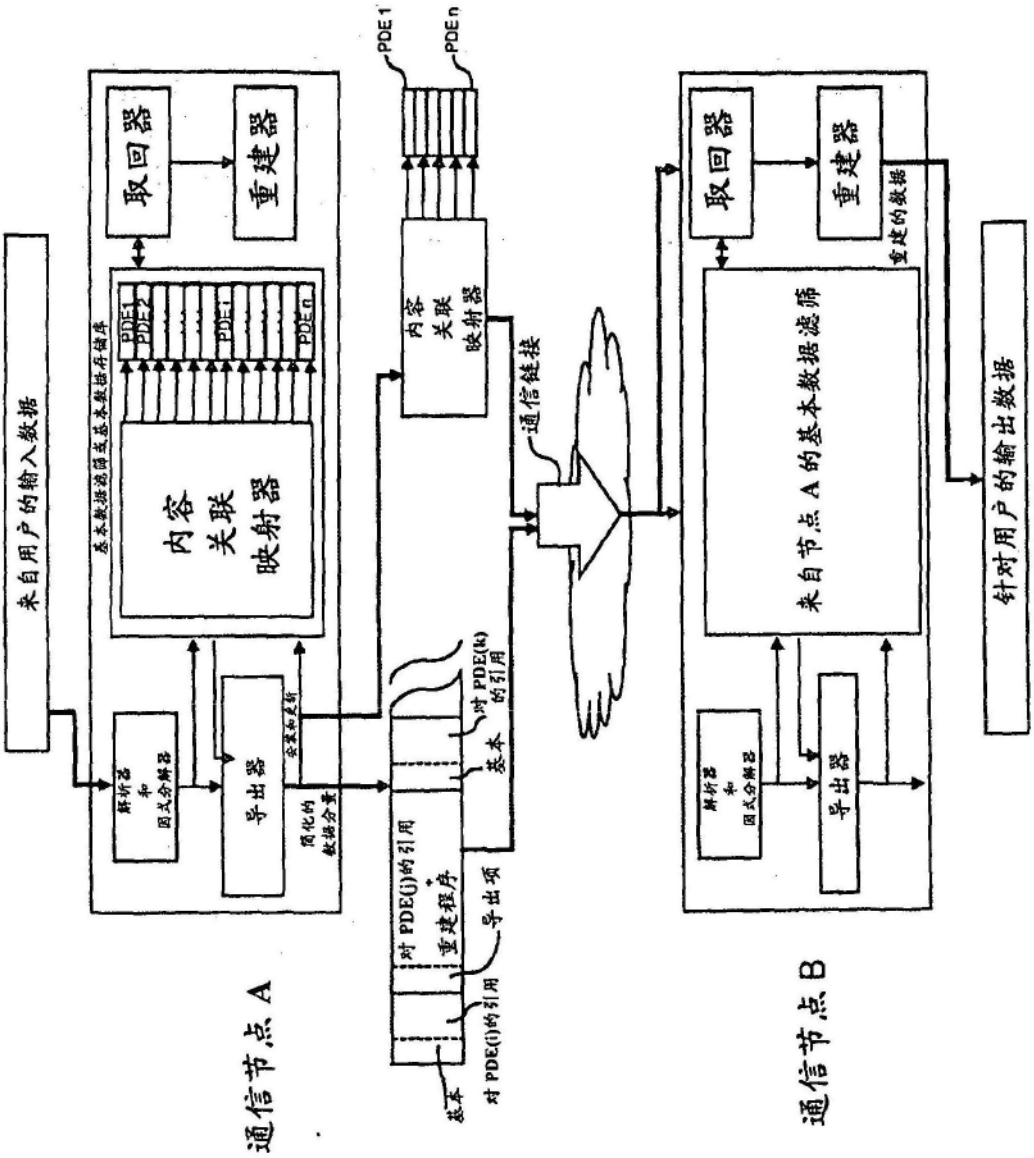

来自用户的输入数据
取回器
重建器
PDE1
PDE2
PDE1
PDEn
内容关联映射器
解析器和因式分解器
导出器
PDE 1
PDE n
内容关联映射器
通信链接
对 PDE(j)的引用
重建程序
导出项
对 PDE(i)的引用
基本
对 PDE(k)的引用
通信节点 A
取回器
重建器
来自节点 A 的基本数据滤筛
解析器和因式分解器
导出器
针对用户的输出数据
通信节点 B

图12A

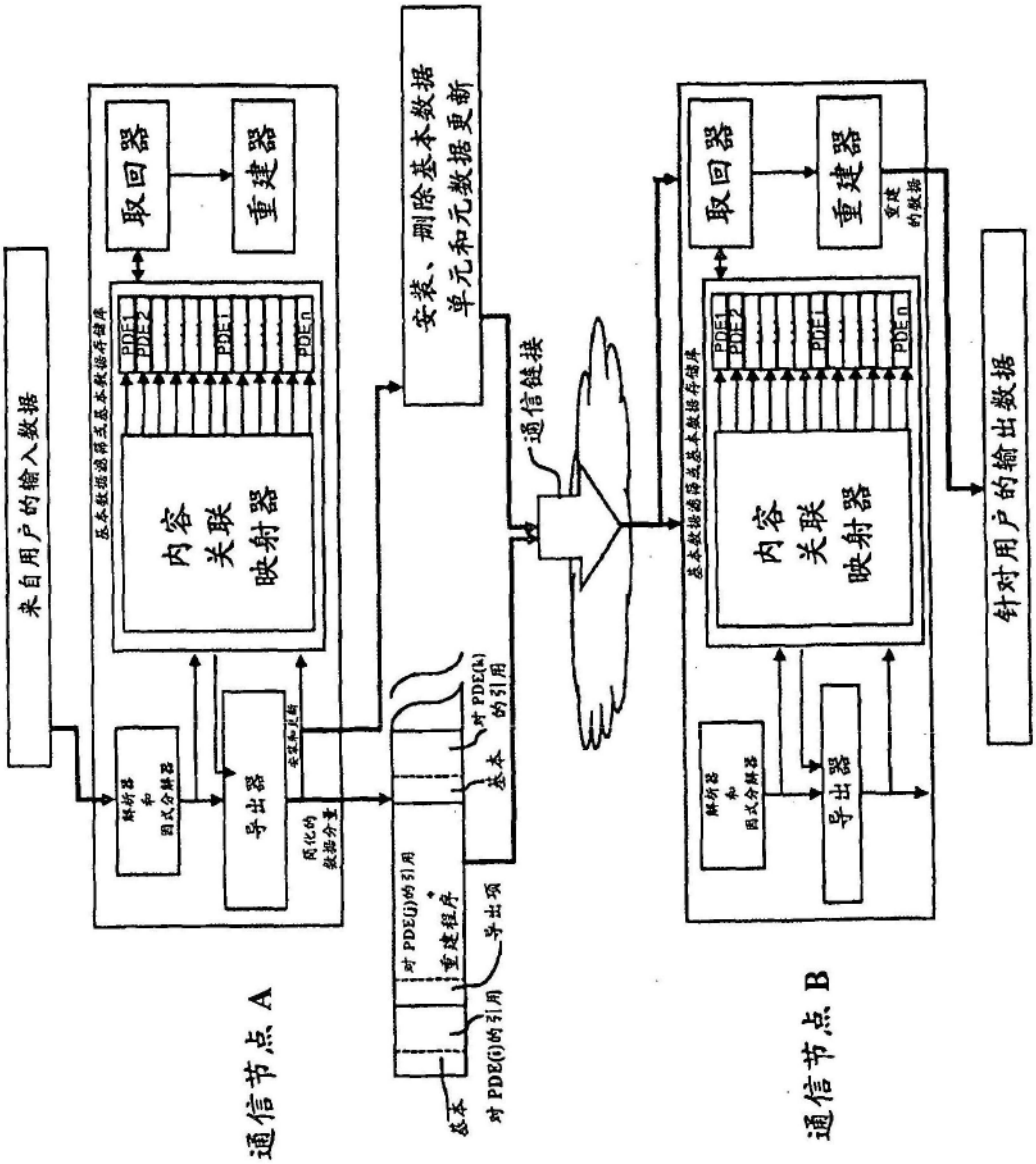

来自用户的输入数据
取回器
重建器
内容关联映射器
PDE1
PDE2
PDE1
PDEn
解析器和固式合辅器
导出器
安装、删除基本数据单元和元数据更新
通信链接
对PDE(i)的引用
重建程序
导出项
基本
对PDE(i)的引用
对PDE(k)的引用
通信节点A
通信节点B
针对用户的输出数据

图12B

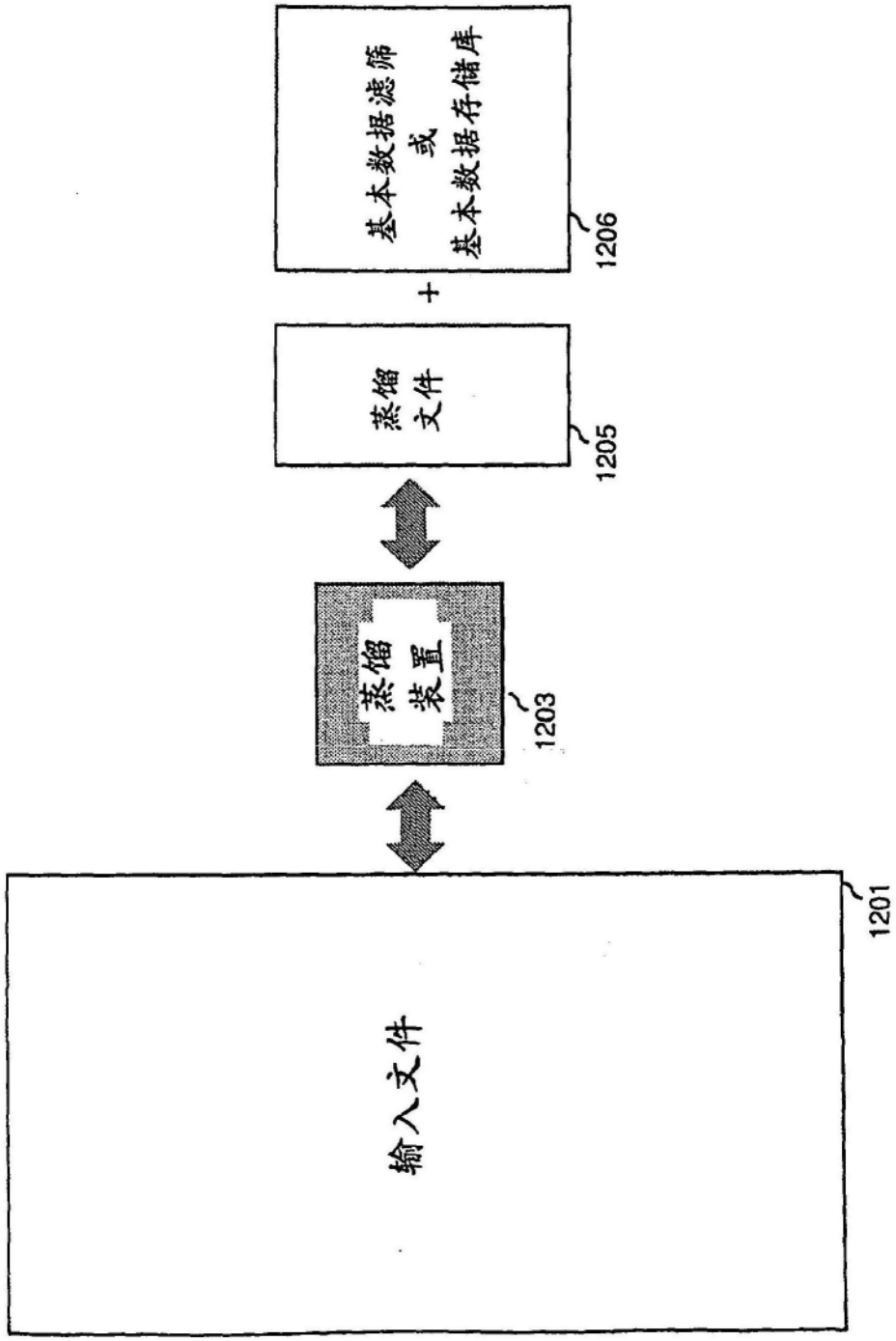


图12C

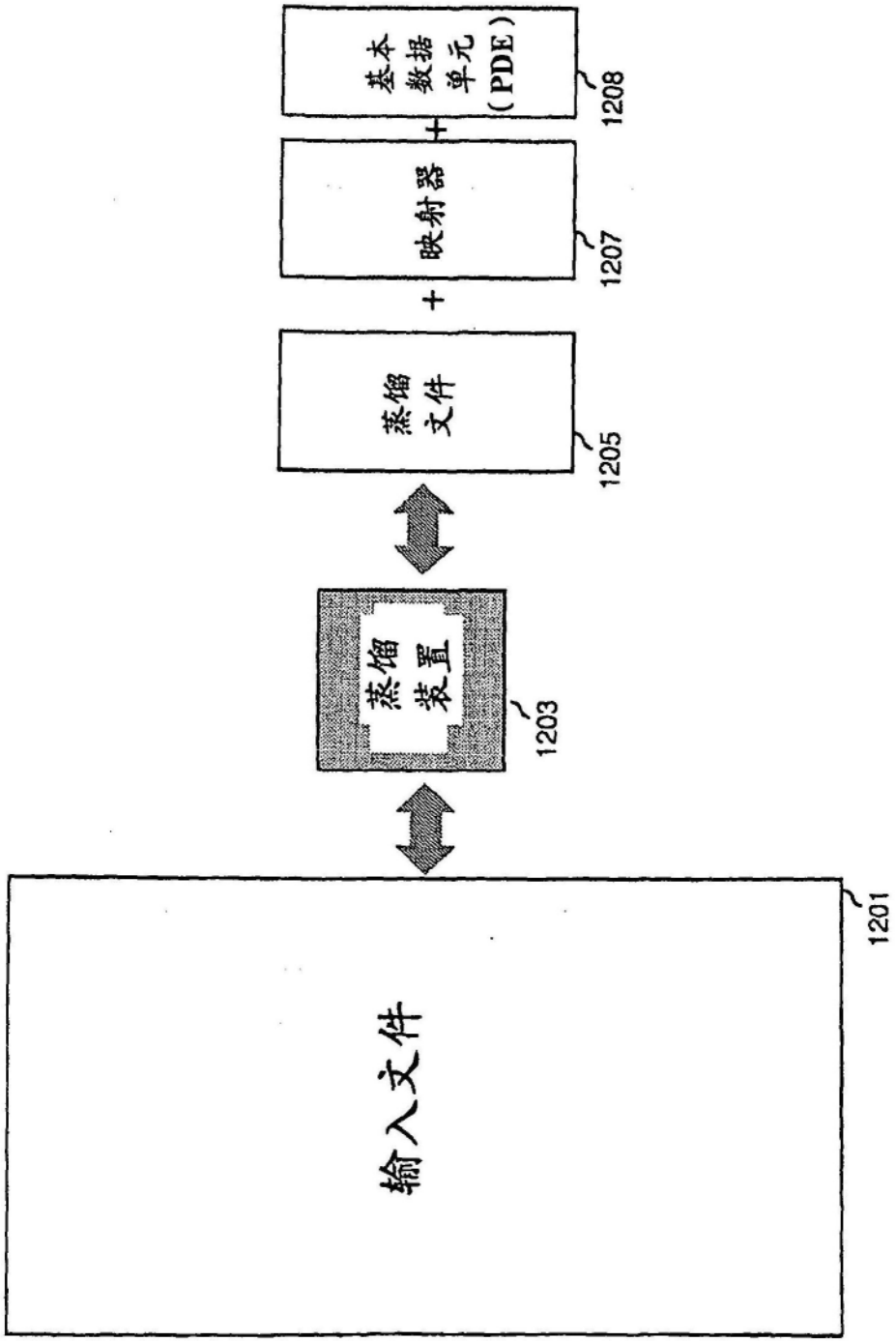


图12D

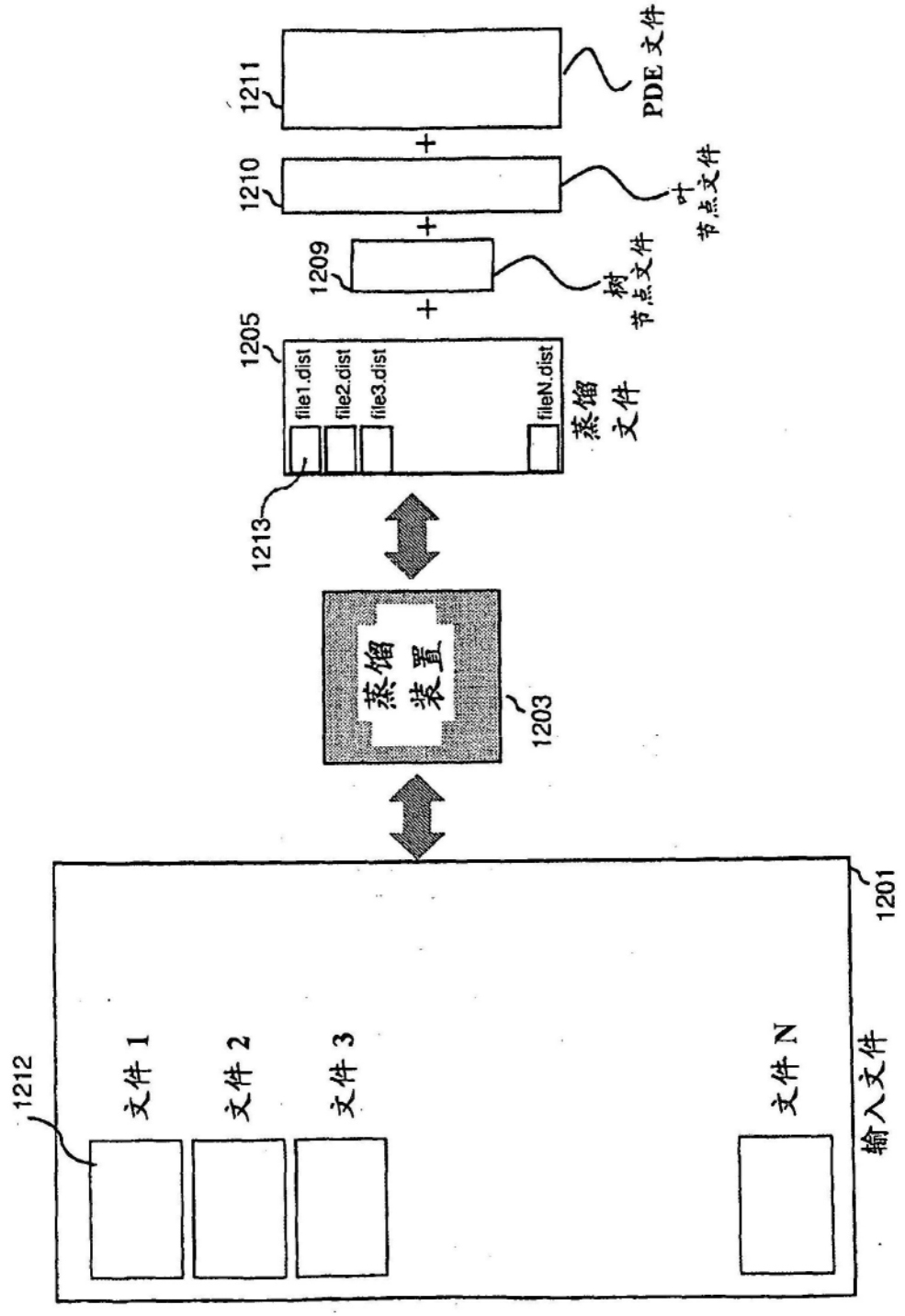

1211
PDE 文件
1210
叶节点文件
1209
树节点文件
1205
file1.dist
file2.dist
file3.dist
fileN.dist
蒸馏文件
1213
蒸馏装置
1203
1201
1212
文件 1
文件 2
文件 3
文件 N
输入文件

图12E

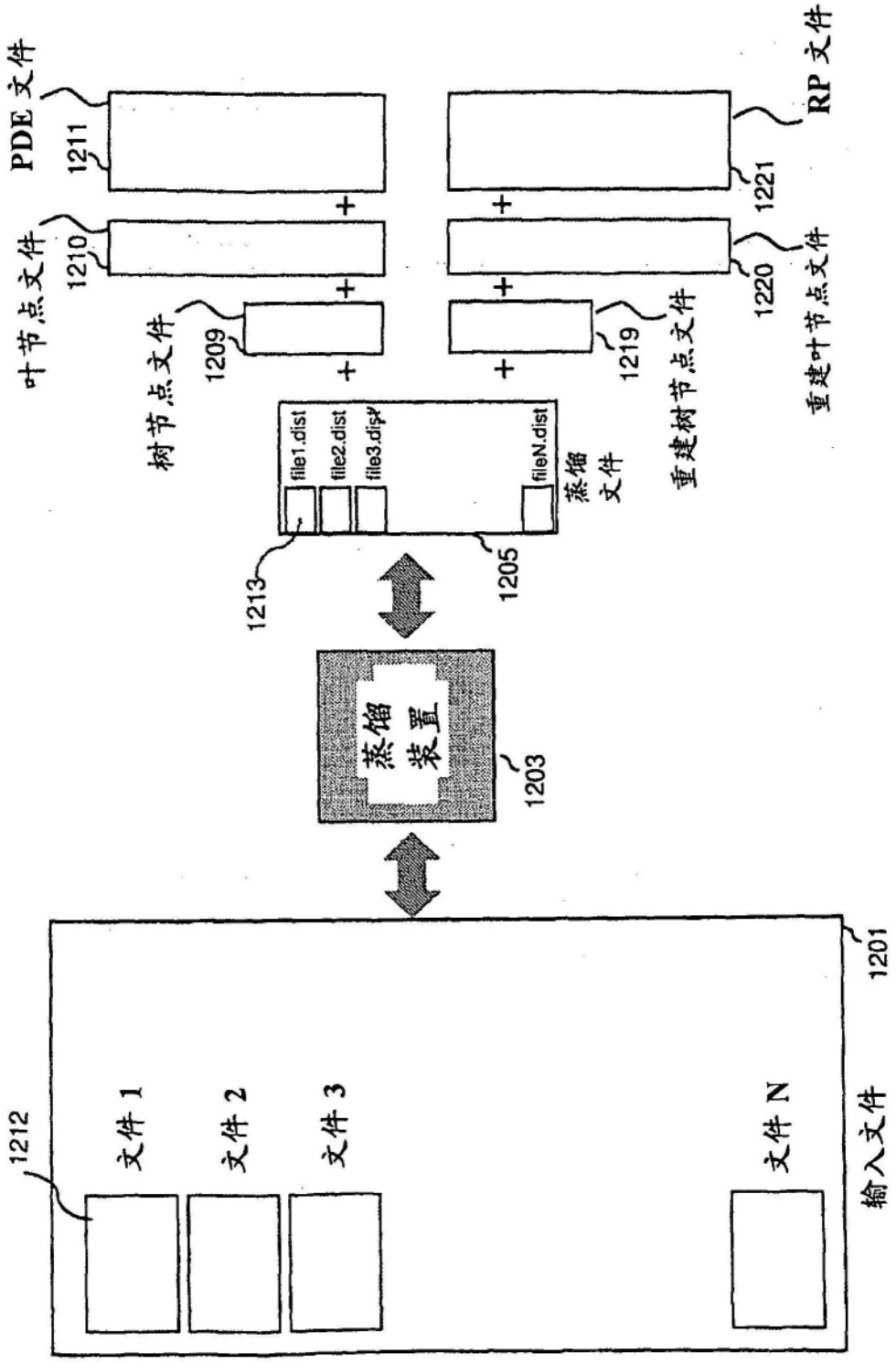

PDE 文件
1211
叶节点文件
1210
树节点文件
1209
file1.dist
file2.dist
file3.dist
fileN.dist
蒸馏文件
1213
1205
蒸馏装置
1203
1212
文件 1
文件 2
文件 3
文件 N
输入文件
1201
1219
重建树节点文件
1220
重建叶节点文件
1221
RP 文件

图12F

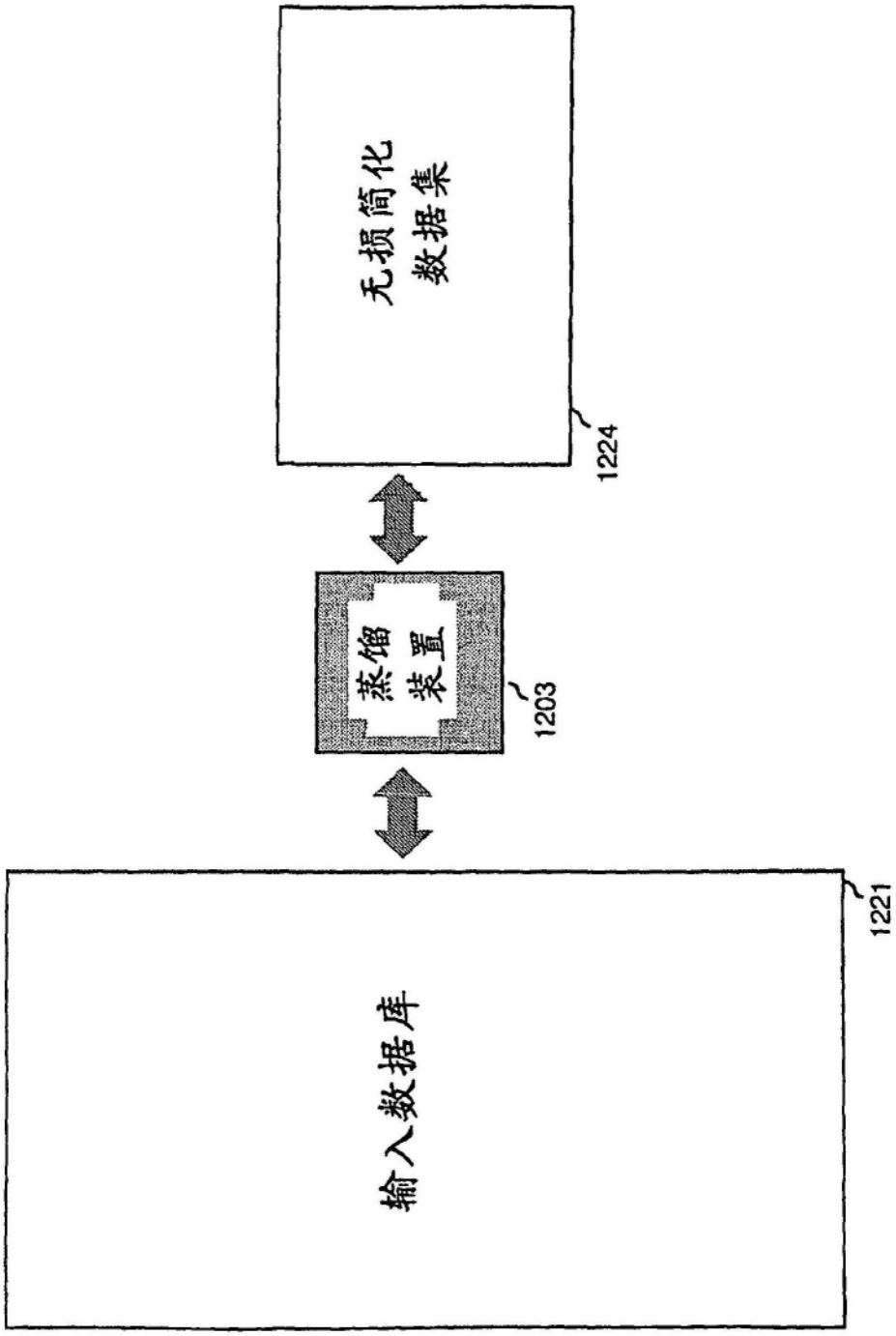


图12G

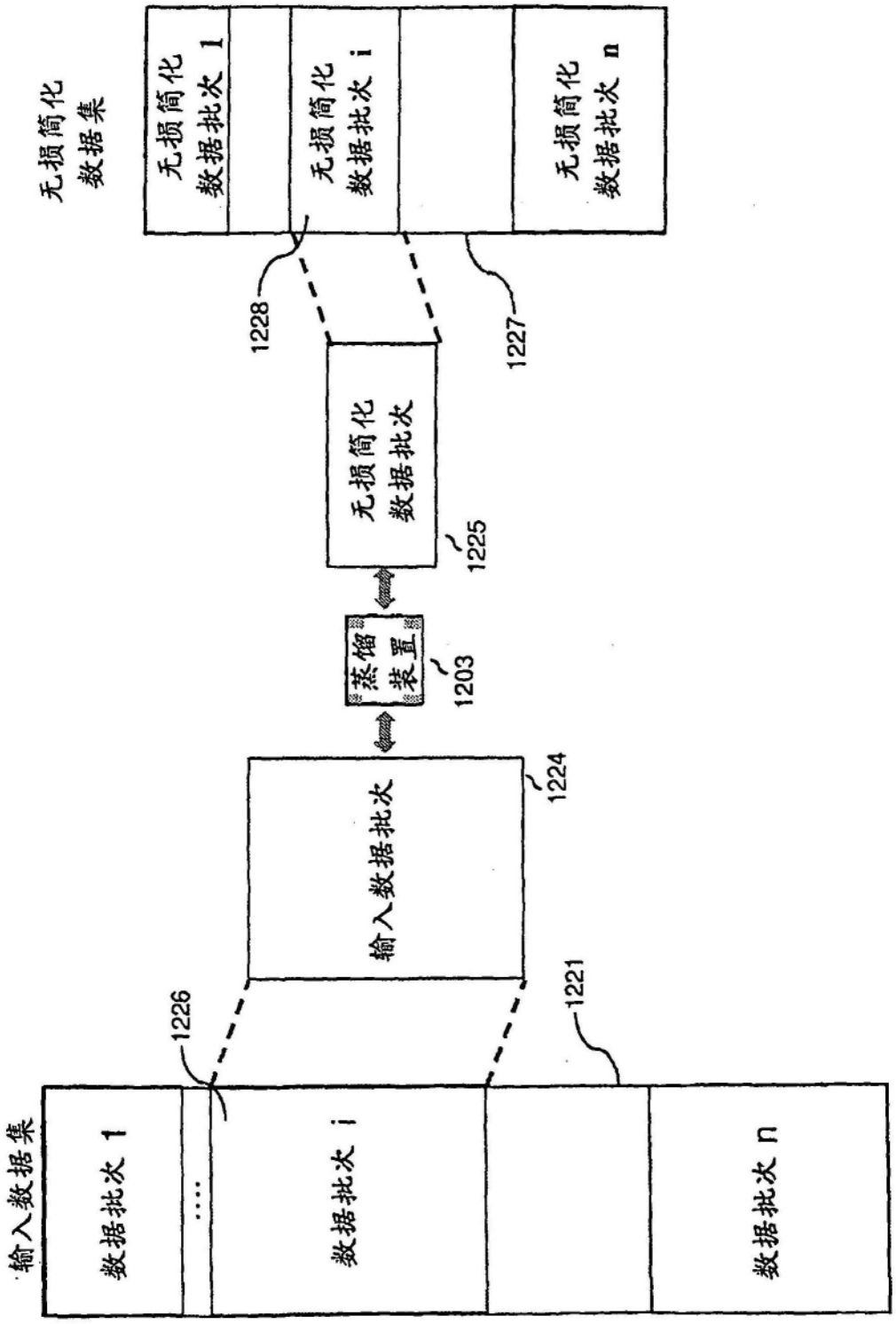

无损简化
数据集
无损简化
数据批次 1
无损简化
数据批次 i
无损简化
数据批次 n
1228
1227
无损简化
数据批次
1225
蒸馏
装置
1203
输入数据批次
1224
1226
1221
输入数据集
数据批次 1
数据批次 i
数据批次 n

图12H

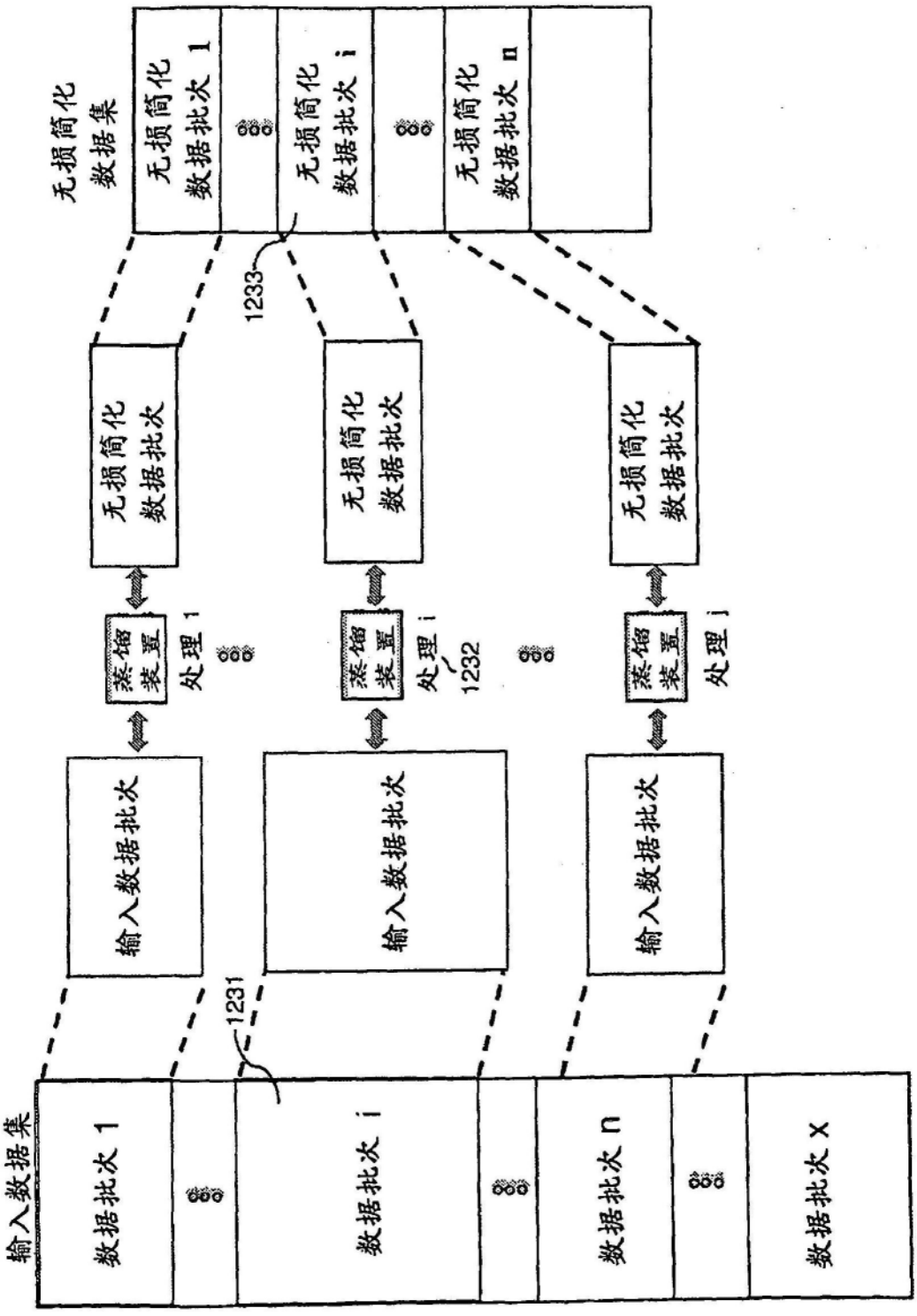

输入数据集
数据批次 1
数据批次 i
数据批次 n
数据批次 X
1231
输入数据批次
输入数据批次
输入数据批次
蒸馏装置
处理 1
蒸馏装置
处理 i
1232
蒸馏装置
处理 j
无损简化数据批次
无损简化数据批次
无损简化数据批次
1233
无损简化数据集
无损简化数据批次 1
无损简化数据批次 i
无损简化数据批次 n

图12I

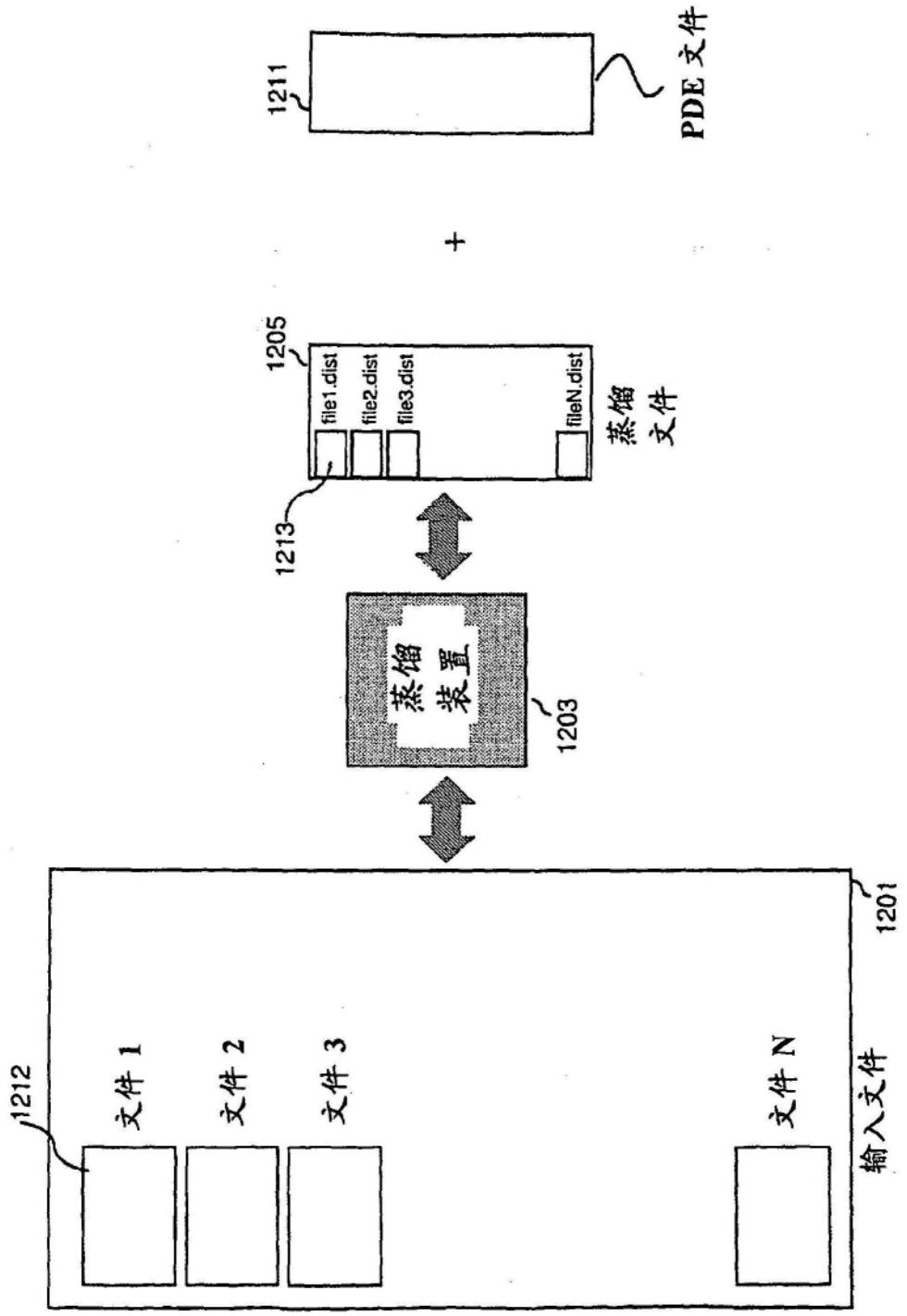

1211
PDE 文件
+
1205
file1.dist
file2.dist
file3.dist
fileN.dist
蒸馏文件
1213
蒸馏装置
1203
1201
1212
文件 1
文件 2
文件 3
文件 N
输入文件

图12J

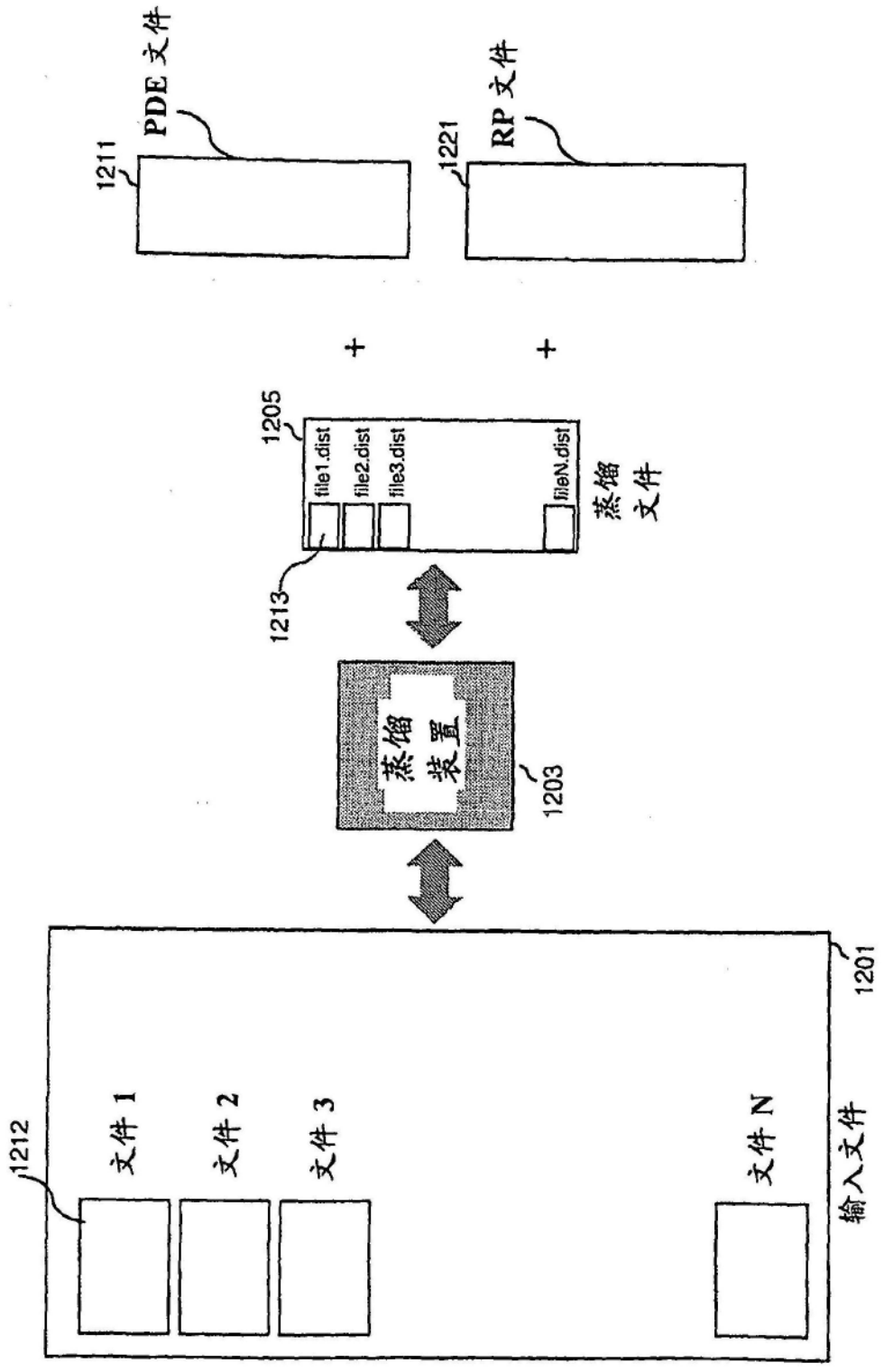

PDE 文件
1211
RP 文件
1221
+
+
1205
file1.dist
file2.dist
file3.dist
fileN.dist
蒸馏文件
1213
蒸馏装置
1203
1201
1212
文件 1
文件 2
文件 3
文件 N
输入文件

图12K

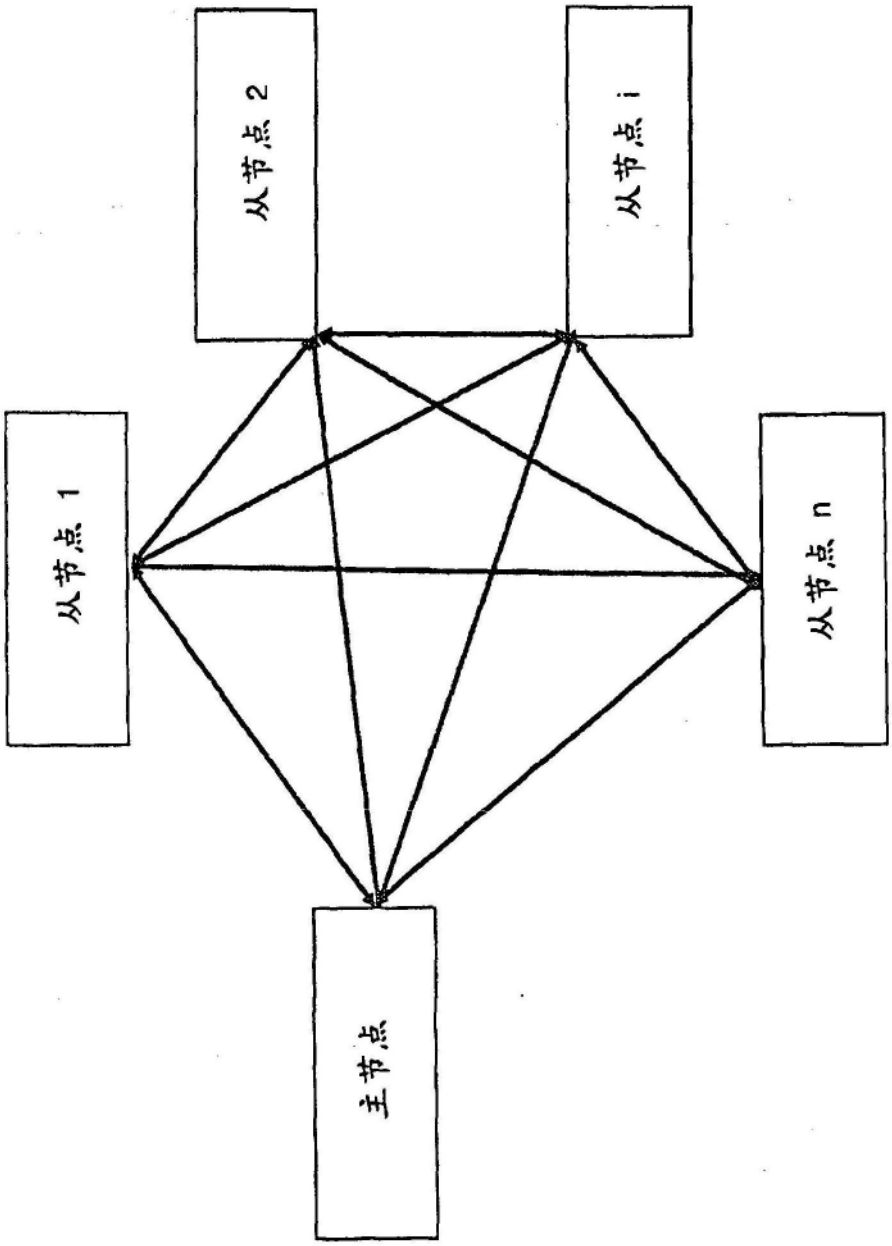


图12L

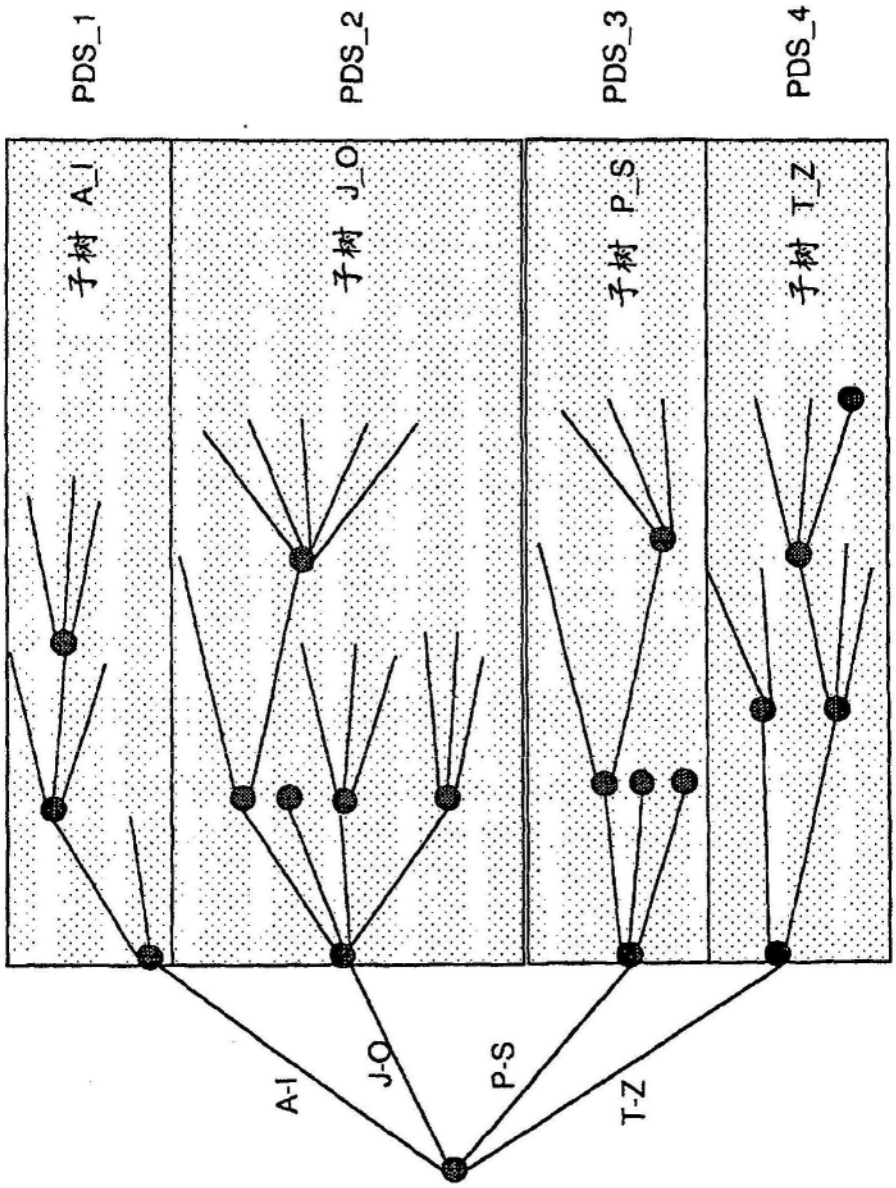


图12M

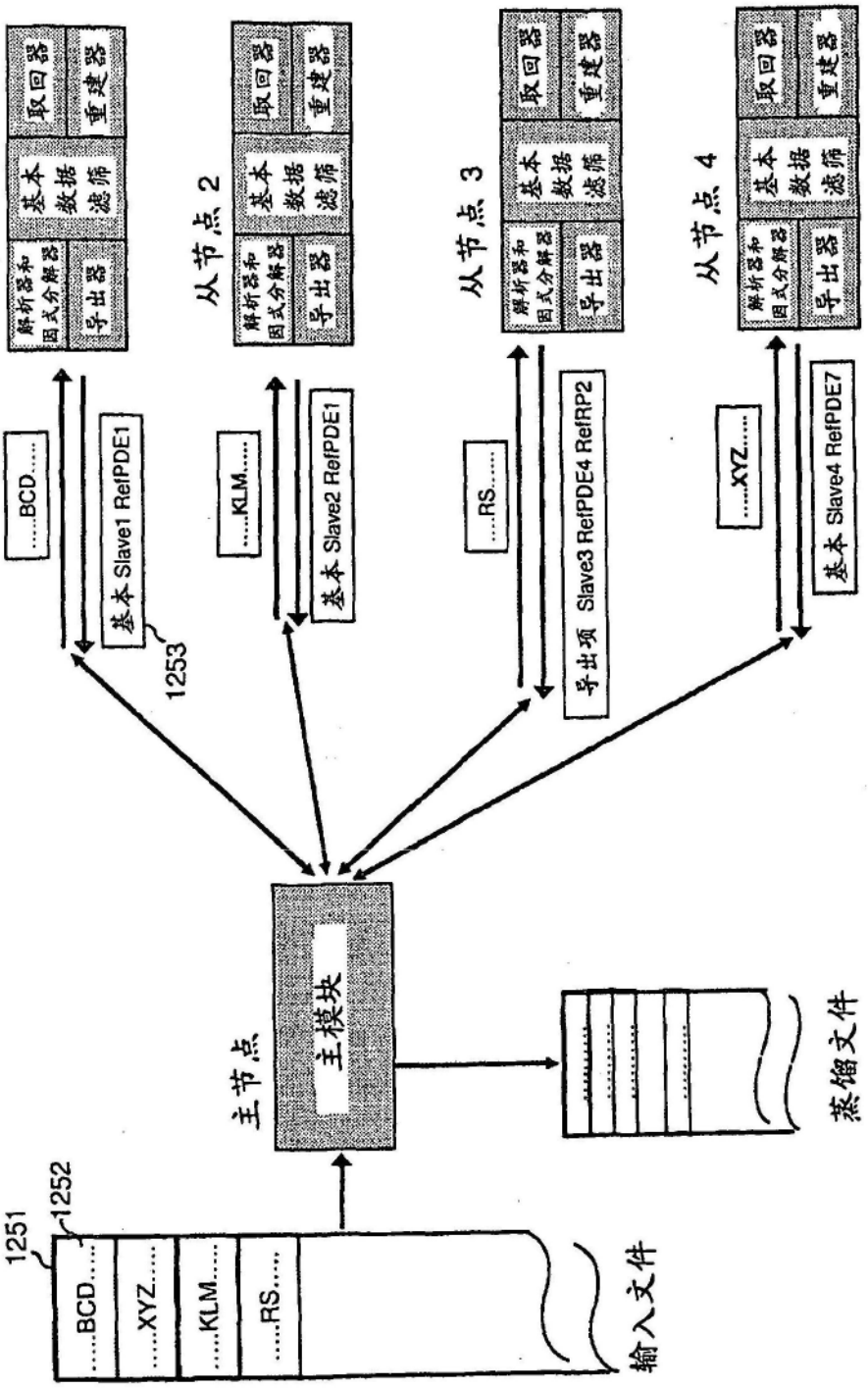

从节点 1
从节点 2
从节点 3
从节点 4
解析器和因式分解器
基本数据滤筛
取回器
重建器
导出器
.....BCD.....
基本 Slave1 RefPDE1
1253
.....KLM.....
基本 Slave2 RefPDE1
.....RS.....
导出项 Slave3 RefPDE4 RefRP2
.....XYZ.....
基本 Slave4 RefPDE7
主节点
主模块
蒸馏文件
1251
1252
.....BCD.....
.....XYZ.....
.....KLM.....
.....RS.....
输入文件

图12N

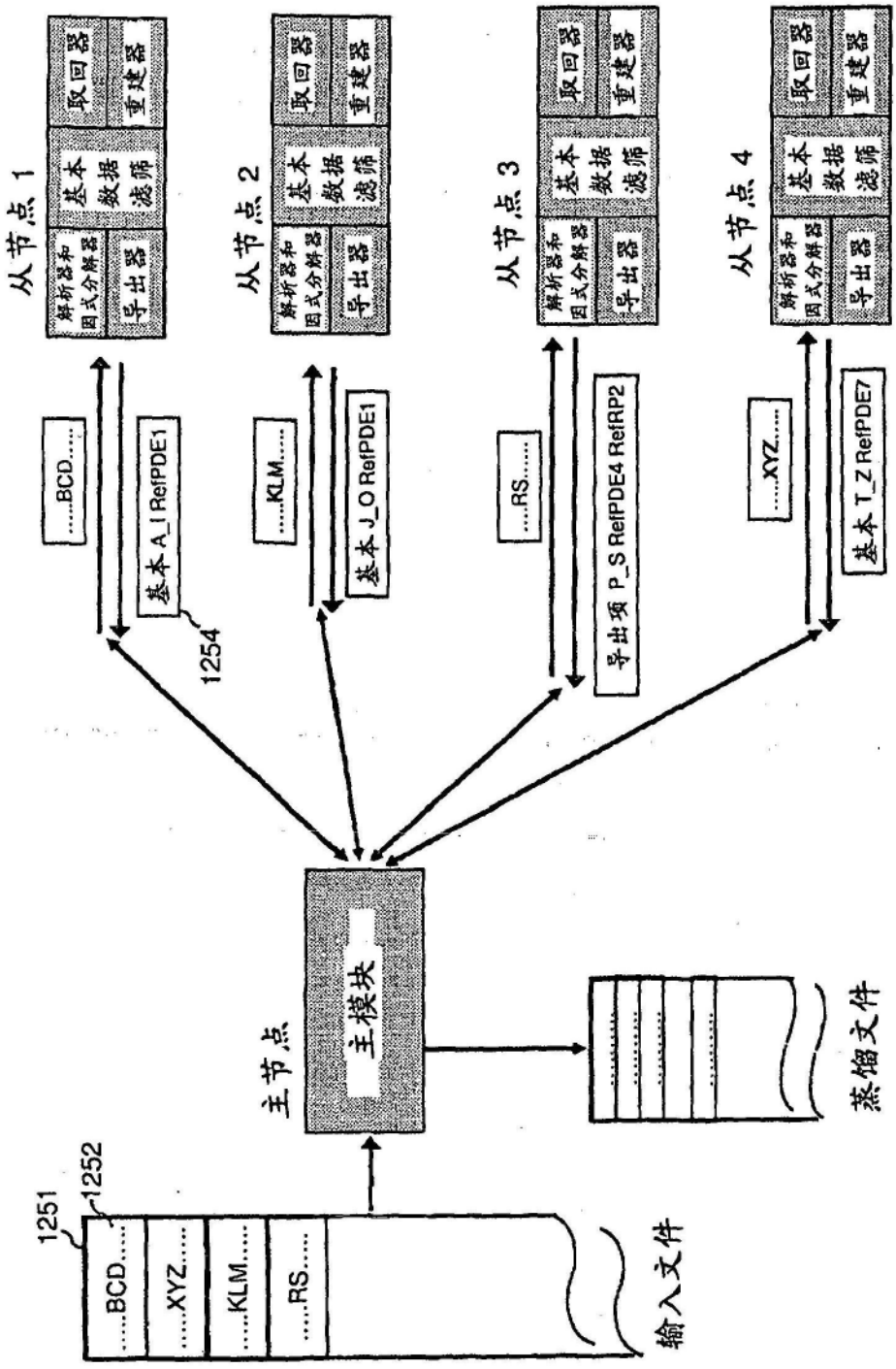

从节点 1
从节点 2
从节点 3
从节点 4
解析器和因式分解器
基本数据滤筛
取回器
重建器
导出器
.....BCD.....
基本 A_I RefPDE1
1254
.....KLM.....
基本 J_O RefPDE1
.....RS.....
导出项 P_S RefPDE4 RefRP2
.....XYZ.....
基本 T_Z RefPDE7
主节点
主模块
蒸馏文件
1251
1252
.....BCD.....
.....XYZ.....
.....KLM.....
.....RS.....
输入文件

图120

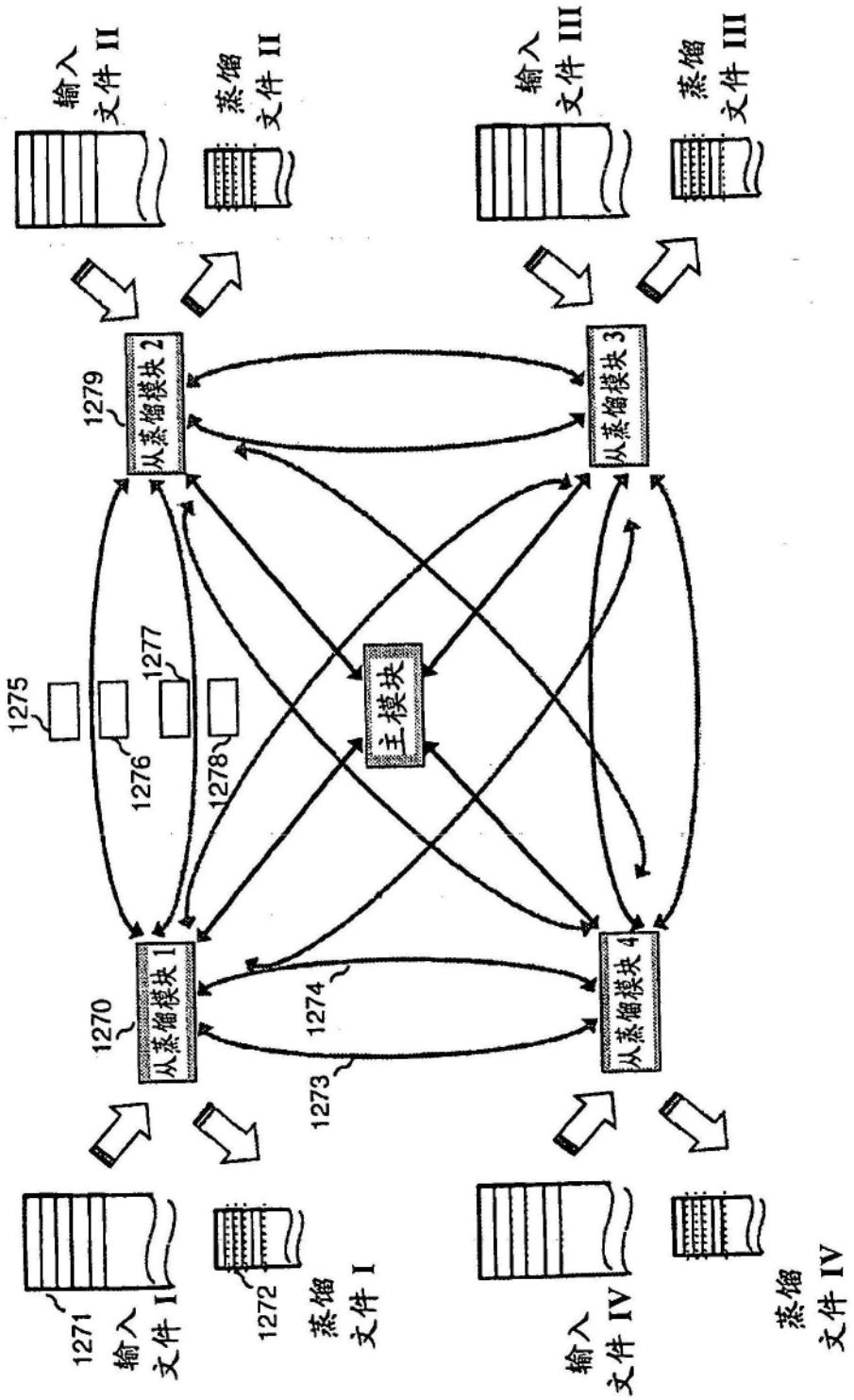

输入文件 II
蒸馏文件 II
输入文件 III
蒸馏文件 III
1279
从蒸馏模块 2
从蒸馏模块 3
1275
1277
1276
1278
主模块
1270
从蒸馏模块 1
从蒸馏模块 4
1274
1273
1271
输入文件 I
1272
蒸馏文件 I
输入文件 IV
蒸馏文件 IV

图12P

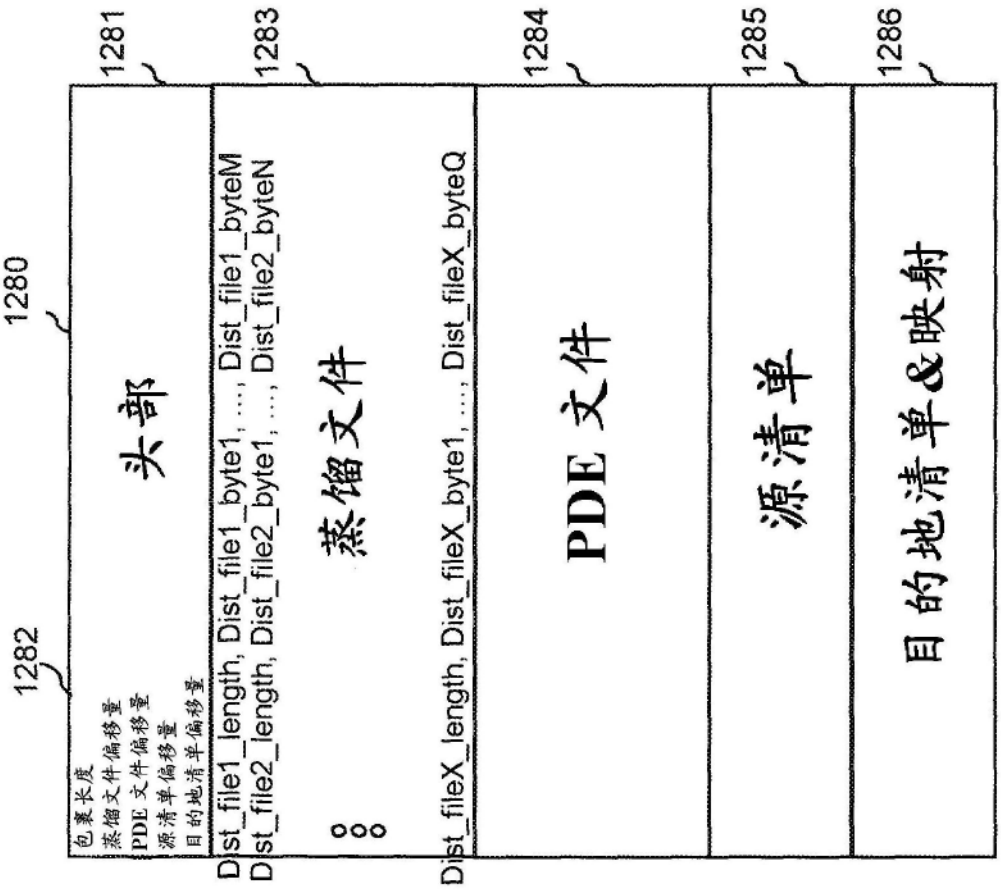

1280
1281
1282
1283
1284
1285
1286
包裹长度
蒸馏文件偏移量
PDE 文件偏移量
源清单偏移量
目的地清单偏移量
头部
Dist_file1_length, Dist_file1_byte1, ..., Dist_file1_byteM
Dist_file2_length, Dist_file2_byte1, ..., Dist_file2_byteN
蒸馏文件
Dist_fileX_length, Dist_fileX_byte1, ..., Dist_fileX_byteQ
PDE 文件
源清单
目的地清单&映射

图12Q

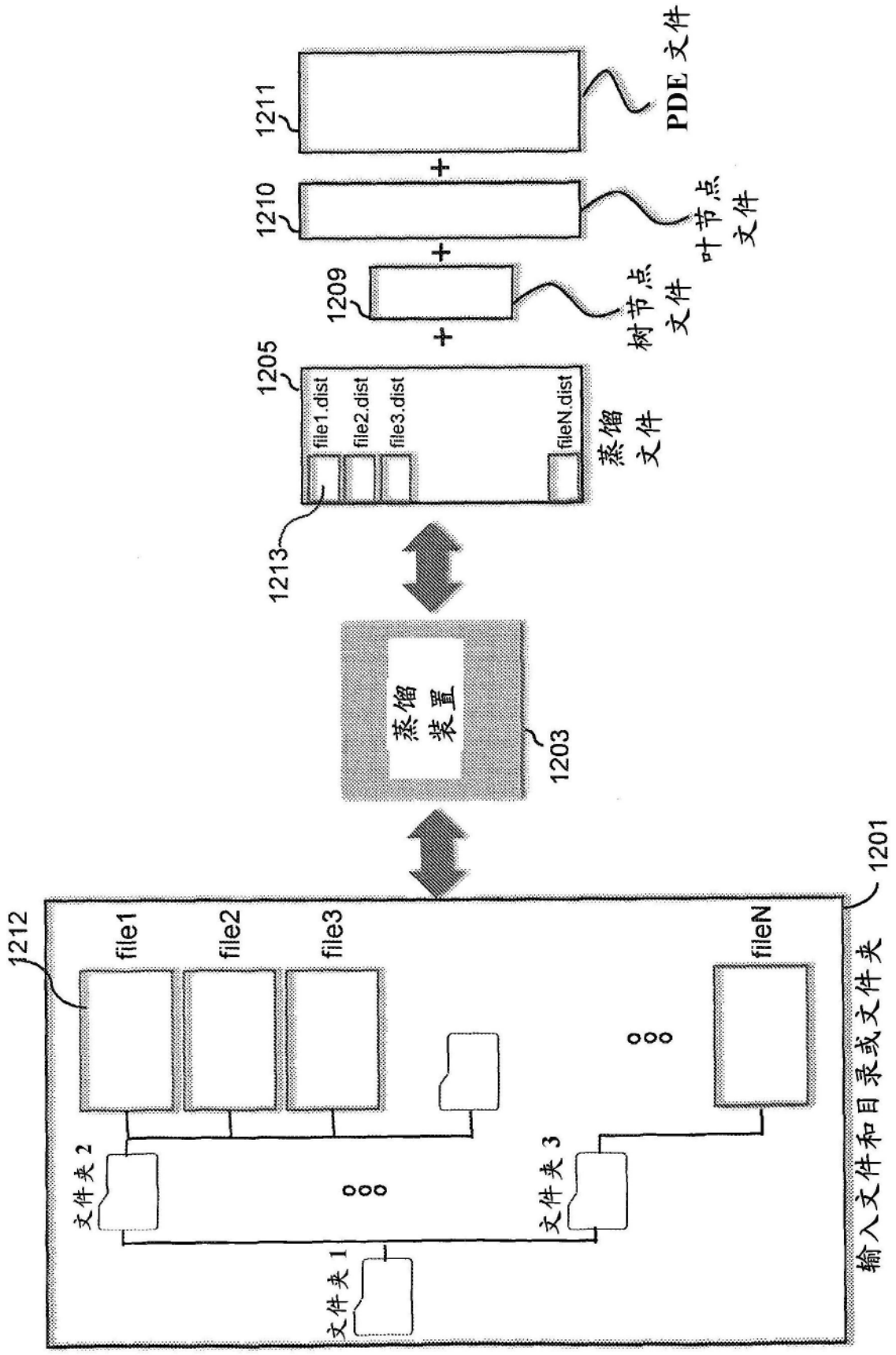
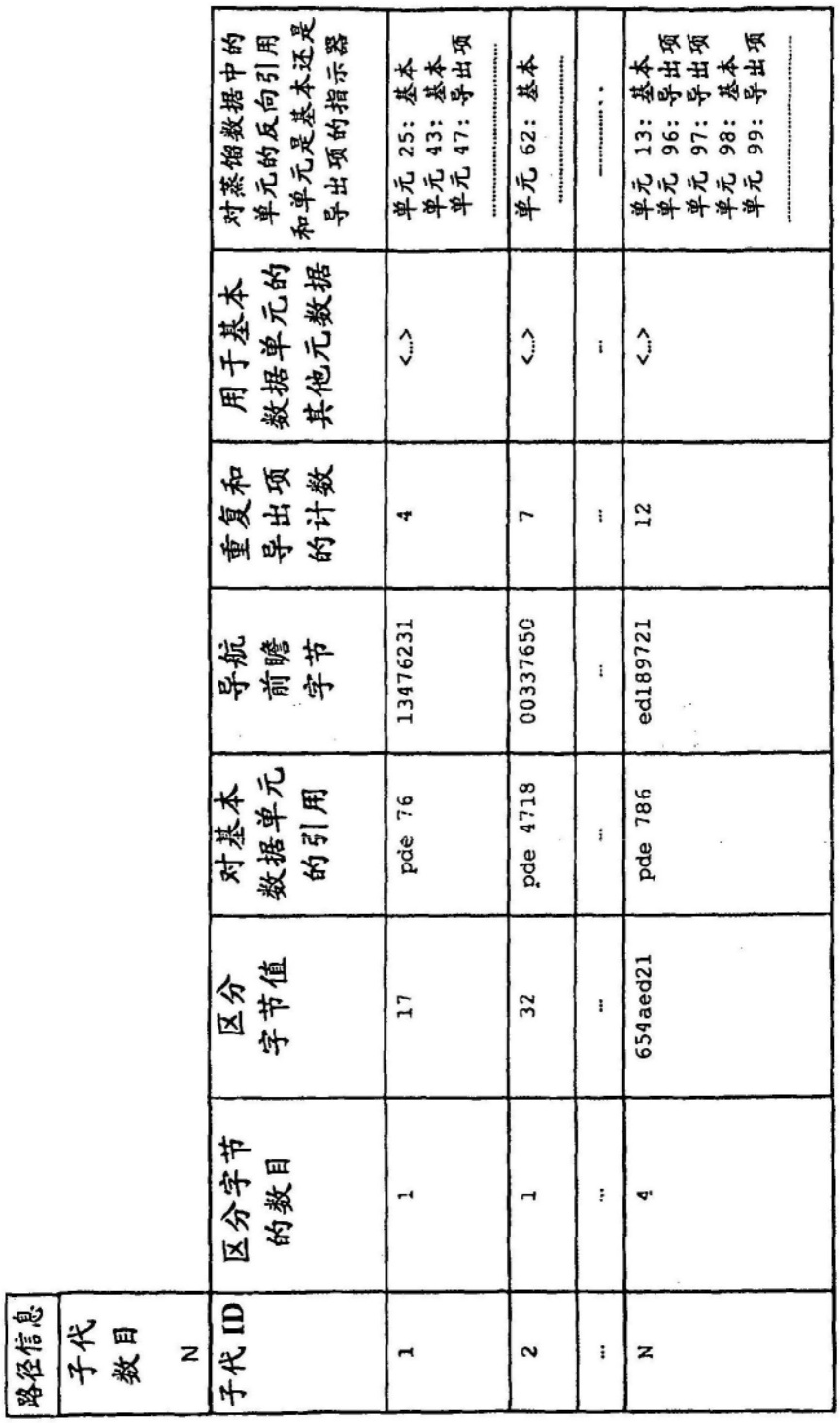

1211
PDE 文件
1210
叶节点文件
1209
树节点文件
1205
file1.dist
file2.dist
file3.dist
fileN.dist
蒸馏文件
1213
蒸馏装置
1203
1212
file1
file2
file3
fileN
文件夹 2
文件夹 3
文件夹 1
输入文件和目录或文件夹
1201

图12R



路径信息 子代数目 N								
子代 ID	区分字节的数目	区分字节值	对基本数据单元的引用	导航前瞻字节	重复和导出项的计数	用于基本数据单元的其他元数据	对蒸馏数据中的单元的反向引用和单元是基本还是导出项的指示器
1	1	17	pde 76	13476231	4	<...>	单元 25：基本 单元 43：基本 单元 47：导出项 .....
2	1	32	pde 4718	00337650	7	<...>	单元 62：基本 .....
...	...	...	...	...	...	...	.....
N	4	654aed21	pde 786	ed189721	12	<...>	单元 13：基本 单元 96：导出项 单元 97：导出项 单元 98：基本 单元 99：导出项 .....

图13

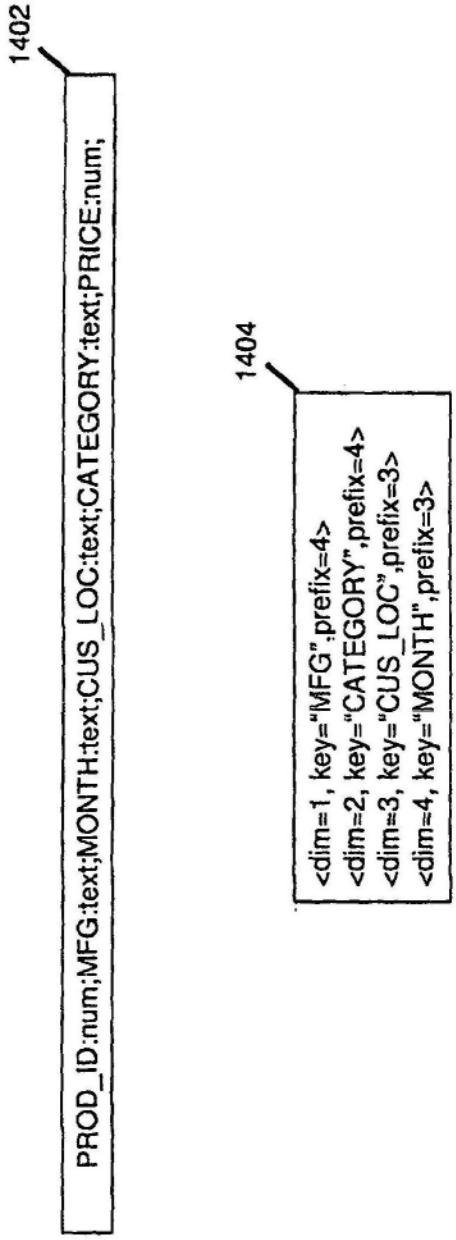

1402
PROD_ID:num;MFG:text;MONTH:text;CUS_LOC:text;CATEGORY:text;PRICE:num;
1404
<dim=1, key="MFG",prefix=4>
<dim=2, key="CATEGORY",prefix=4>
<dim=3, key="CUS_LOC",prefix=3>
<dim=4, key="MONTH",prefix=3>

图14A

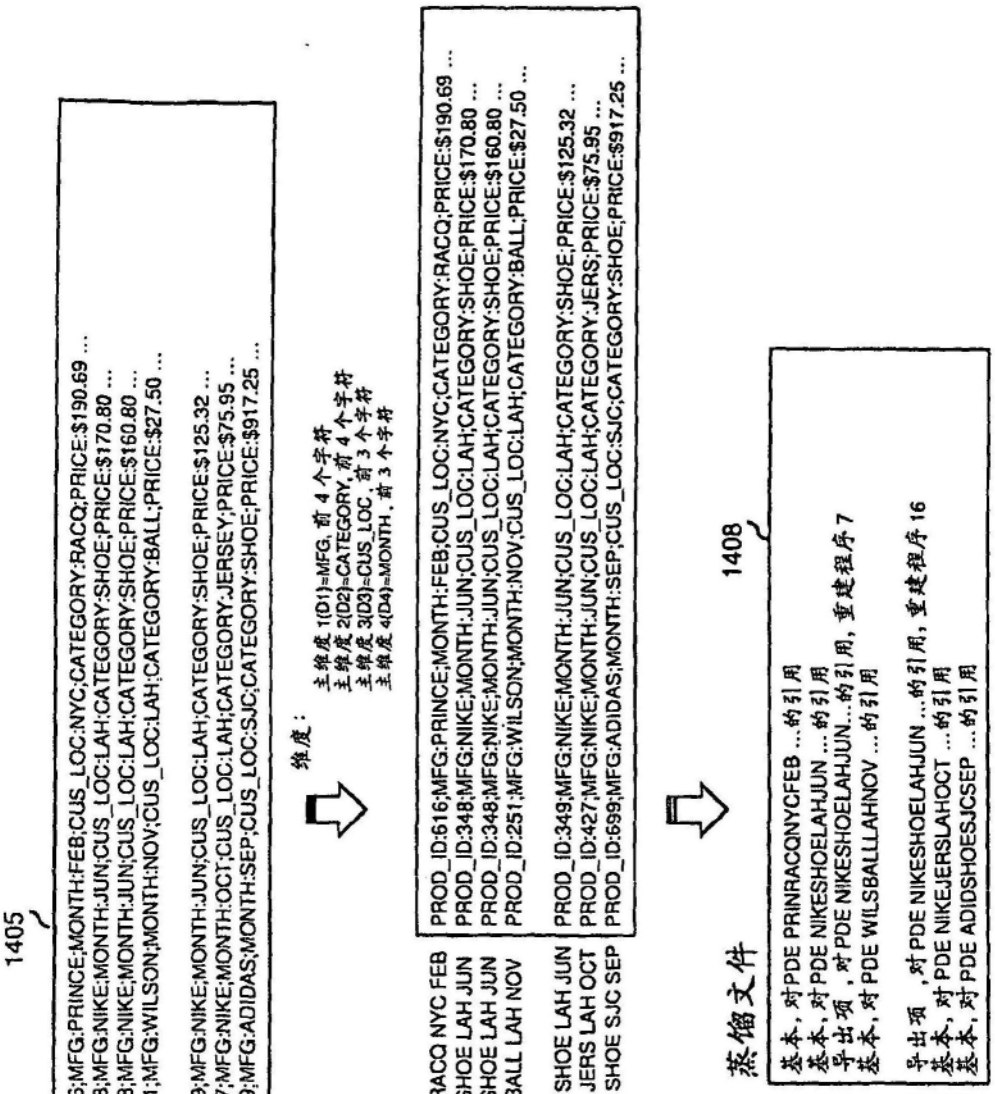

File
1405
交易 1 PROD_ID:616;MFG:PRINCE;MONTH:FEB;CUS_LOC:NYC;CATEGORY:RACQ;PRICE:$190.69 ...
交易 2 PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$170.80 ...
交易 3 PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$160.80 ...
交易 4 PROD_ID:251;MFG:WILSON;MONTH:NOV;CUS_LOC:LAH;CATEGORY:BALL;PRICE:$27.50 ...
交易 58 PROD_ID:349;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$125.32 ...
交易 59 PROD_ID:427;MFG:NIKE;MONTH:OCT;CUS_LOC:LAH;CATEGORY:JERSEY;PRICE:$75.95 ...
交易 60 PROD_ID:699;MFG:ADIDAS;MONTH:SEP;CUS_LOC:SJC;CATEGORY:SHOE;PRICE:$917.25 ...
维度：
主维度 1(D1)=MFG, 前 4 个字符
主维度 2(D2)=CATEGORY, 前 4 个字符
主维度 3(D3)=CUS_LOC, 前 3 个字符
主维度 4(D4)=MONTH, 前 3 个字符
1406
1407
候选单元 1, D1 D2 D3 D4=PRIN RACQ NYC FEB
候选单元 2, D1 D2 D3 D4=NIKE SHOE LAH JUN
候选单元 3, D1 D2 D3 D4=NIKE SHOE LAH JUN
候选单元 4, D1 D2 D3 D4=WILS BALL LAH NOV
候选单元 58, D1 D2 D3 D4=NIKE SHOE LAH JUN
候选单元 59, D1 D2 D3 D4=NIKE JERS LAH OCT
候选单元 60, D1 D2 D3 D4=ADID SHOE SJC SEP
PROD_ID:616;MFG:PRINCE;MONTH:FEB;CUS_LOC:NYC;CATEGORY:RACQ;PRICE:$190.69 ...
PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$170.80 ...
PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$160.80 ...
PROD_ID:251;MFG:WILSON;MONTH:NOV;CUS_LOC:LAH;CATEGORY:BALL;PRICE:$27.50 ...
PROD_ID:349;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$125.32 ...
PROD_ID:427;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:JERS;PRICE:$75.95 ...
PROD_ID:699;MFG:ADIDAS;MONTH:SEP;CUS_LOC:SJC;CATEGORY:SHOE;PRICE:$917.25 ...
蒸馏文件
1408
单元 1 基本，对PDE PRINRACQNYCFEB ...的引用
单元 2 基本，对PDE NIKESHOELAHJUN ...的引用
单元 3 导出项，对PDE NIKESHOELAHJUN...的引用，重建程序 7
单元 4 基本，对PDE WILSBALLLAHNOV ...的引用
单元 58 导出项，对PDE NIKESHOELAHJUN ...的引用，重建程序 16
单元 59 基本，对PDE NIKEJERSLAHOCT ...的引用
单元 60 基本，对PDE ADIDSHOESJCSEP ...的引用

图14B

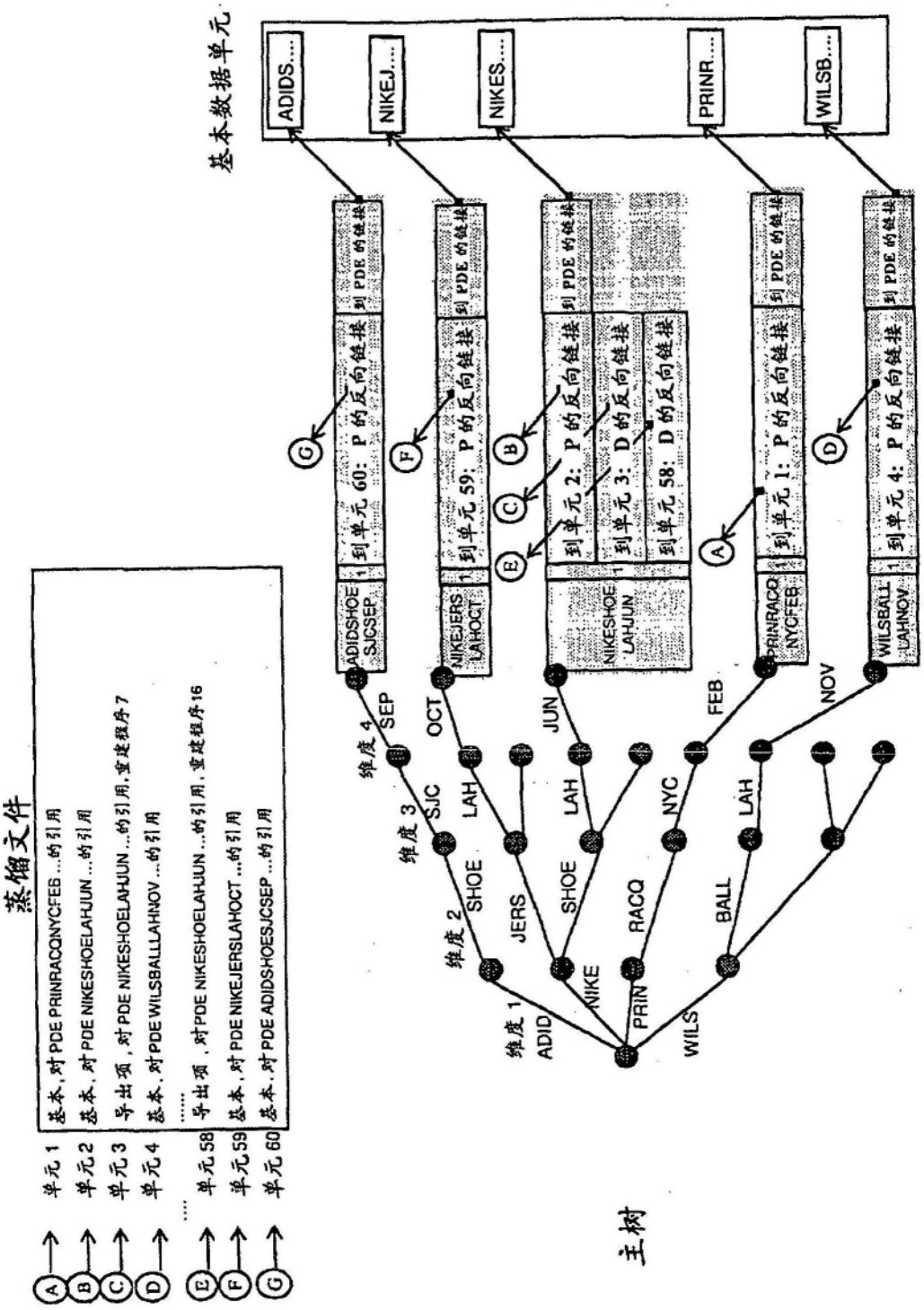

蒸馏文件
单元 1 基本.对PDE PRINRACONYCFEB ...的引用
单元 2 基本.对PDE NIKESHOELAHJUN ...的引用
单元 3 导出项，对PDE NIKESHOELAHJUN ...的引用.重建程序7
单元 4 基本.对PDE WILSBALLLAHNOV ...的引用
.....
单元 58 导出项，对PDE NIKESHOELAHJUN ...的引用.重建程序16
单元 59 基本.对PDE NIKEJERSLAHOCT ...的引用
单元 60 基本.对PDE ADIDSHOESJCSEP ...的引用
A
B
C
D
E
F
G
基本数据单元
ADIDS....
NIKEJ....
NIKES....
PRINR....
WILSB....
ADIDSHOESJCSEP
NIKEJERSLAHOCT
NIKESHOELAHJUN
PRINRACQNYCFEB
WILSBALLLAHNOV
到单元 60: P 的反向链接
到单元 59: P 的反向链接
到单元 2: P 的反向链接
到单元 3: D 的反向链接
到单元 58: D 的反向链接
到单元 1: P 的反向链接
到单元 4: P 的反向链接
到 PDE 的链接
维度 4
SEP
OCT
JUN
FEB
NOV
维度 3
SJC
LAH
NYC
维度 2
SHOE
JERS
RACQ
BALL
维度 1
ADID
NIKE
PRIN
WILS
主树

图14C

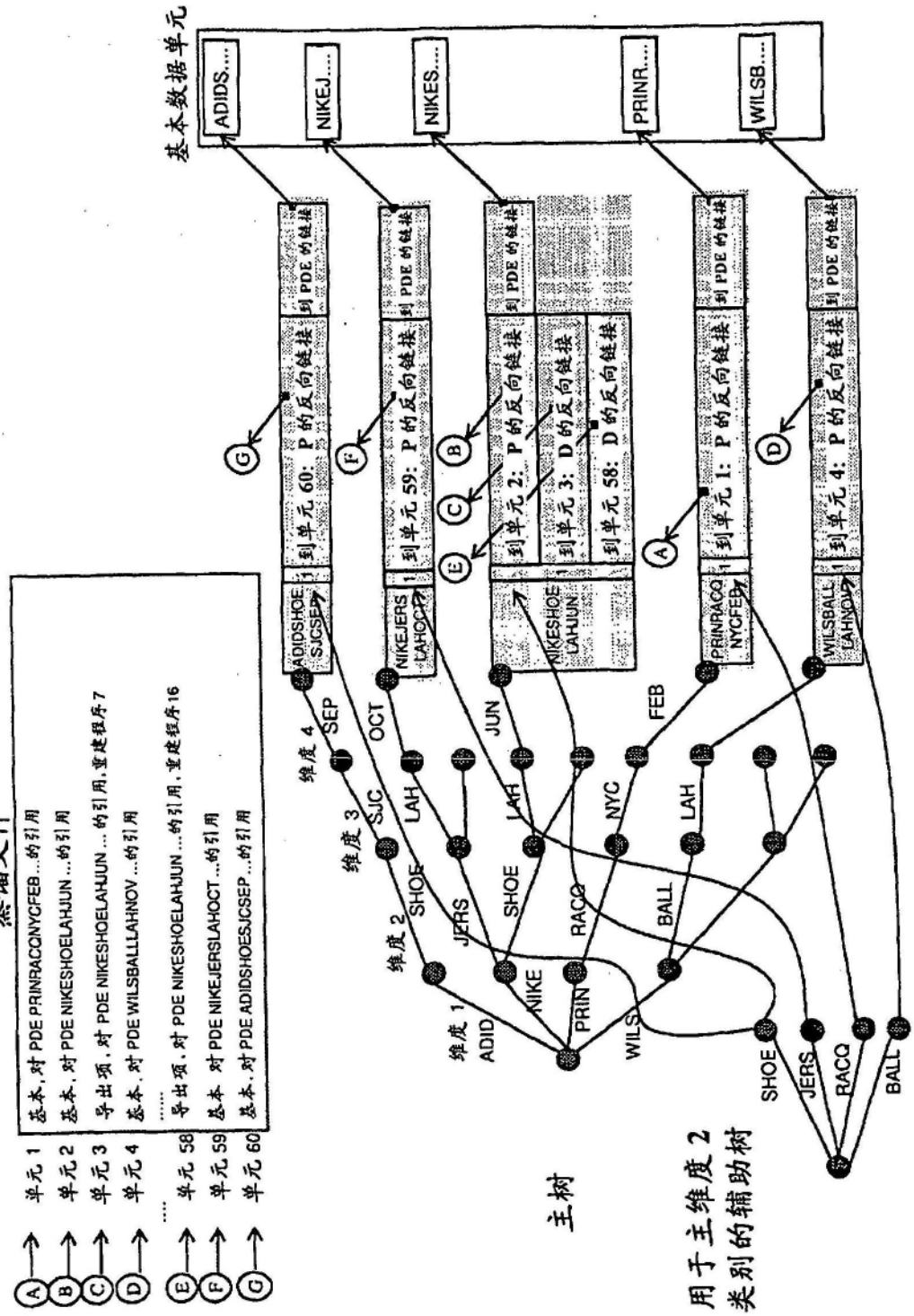
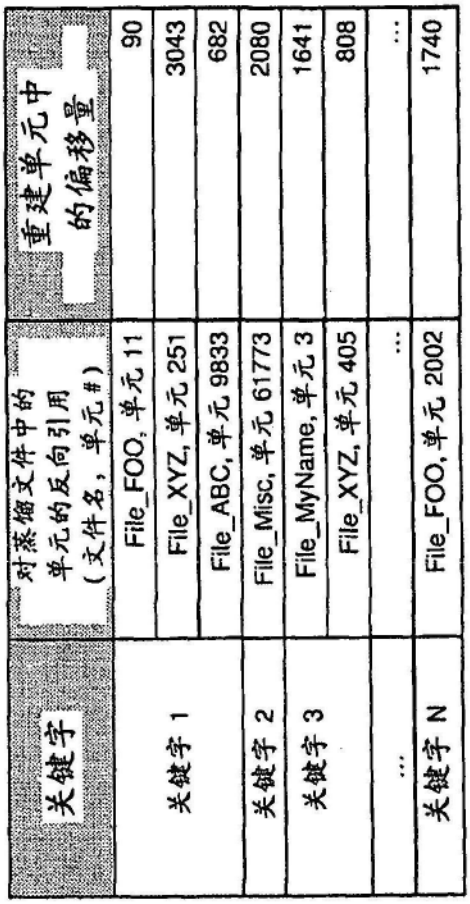

蒸馏文件
单元 1 基本,对 PDE PRINRACONYCFEB ...的引用
单元 2 基本,对 PDE NIKESHOELAHJUN ...的引用
单元 3 导出项,对 PDE NIKESHOELAHJUN ... 的引用,重建程序 7
单元 4 基本,对 PDE WILSBALLLAHNOV ...的引用
.....
单元 58 导出项,对 PDE NIKESHOELAHJUN ...的引用,重建程序 16
单元 59 基本 对 PDE NIKEJERSLAHOCT ...的引用
单元 60 基本,对 PDE ADIDSHOESJCSEP ...的引用
基本数据单元
ADIDS....
NIKEJ....
NIKES....
PRINR....
WILSB....
到单元 60: P 的反向链接
到单元 59: P 的反向链接
到单元 2: P 的反向链接
到单元 3: D 的反向链接
到单元 58: D 的反向链接
到单元 1: P 的反向链接
到单元 4: P 的反向链接
到 PDE 的链接
ADIDSHOE SJCSEP
NIKEJERS LAHOCT
NIKESHOE LAHJUN
PRINRACO NYCFEB
WILSBALL LAHNOV
维度 1
维度 2
维度 3
维度 4
ADID
NIKE
PRIN
WILS
SHOE
JERS
RACQ
BALL
SEP
OCT
JUN
FEB
SJC
LAH
NYC
主树
用于主维度 2 类别的辅助树

图14D



关键字	对落俏文件中的单元的反向引用（文件名，单元#）	重建单元中的偏移量
关键字 1	File_FOO, 单元 11	90
	File_XYZ, 单元 251	3043
	File_ABC, 单元 9833	682
关键字 2	File_Misc, 单元 61773	2080
关键字 3	File_MyName, 单元 3	1641
	File_XYZ, 单元 405	808
...	...	...
关键字 N	File_FOO, 单元 2002	1740

图15

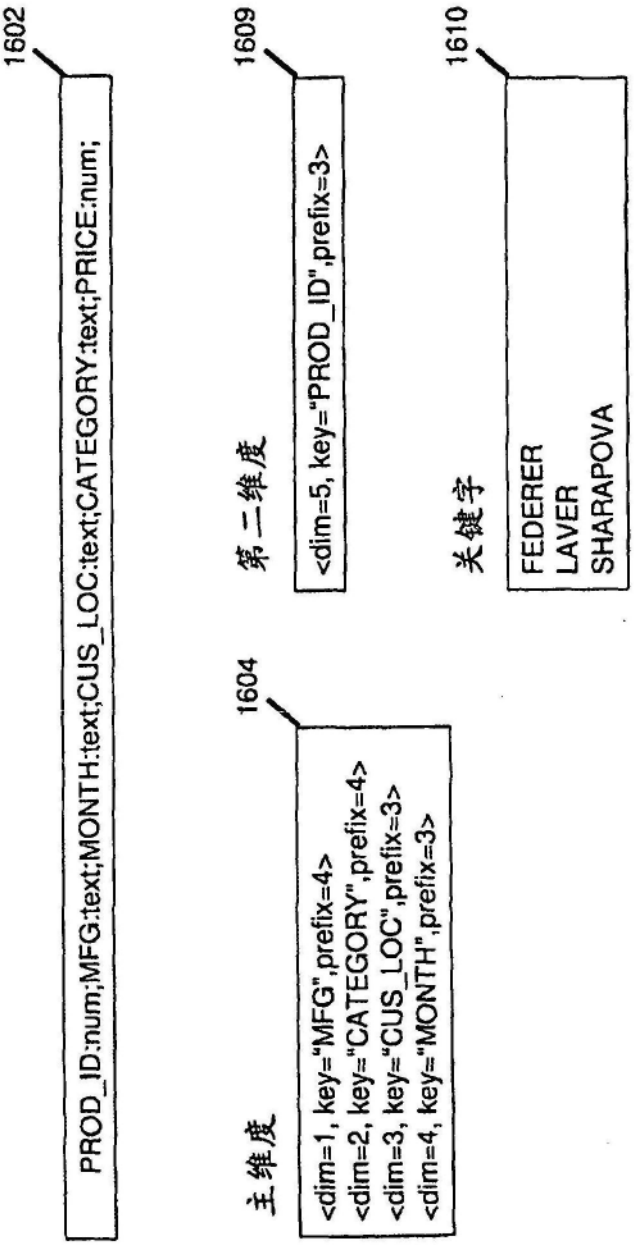

1602
PROD_ID:num;MFG:text;MONTH:text;CUS_LOC:text;CATEGORY:text;PRICE:num;
1609
第二维度
<dim=5, key="PROD_ID",prefix=3>
1610
关键字
FEDERER
LAVER
SHARAPOVA
1604
主维度
<dim=1, key="MFG",prefix=4>
<dim=2, key="CATEGORY",prefix=4>
<dim=3, key="CUS_LOC",prefix=3>
<dim=4, key="MONTH",prefix=3>

图16A

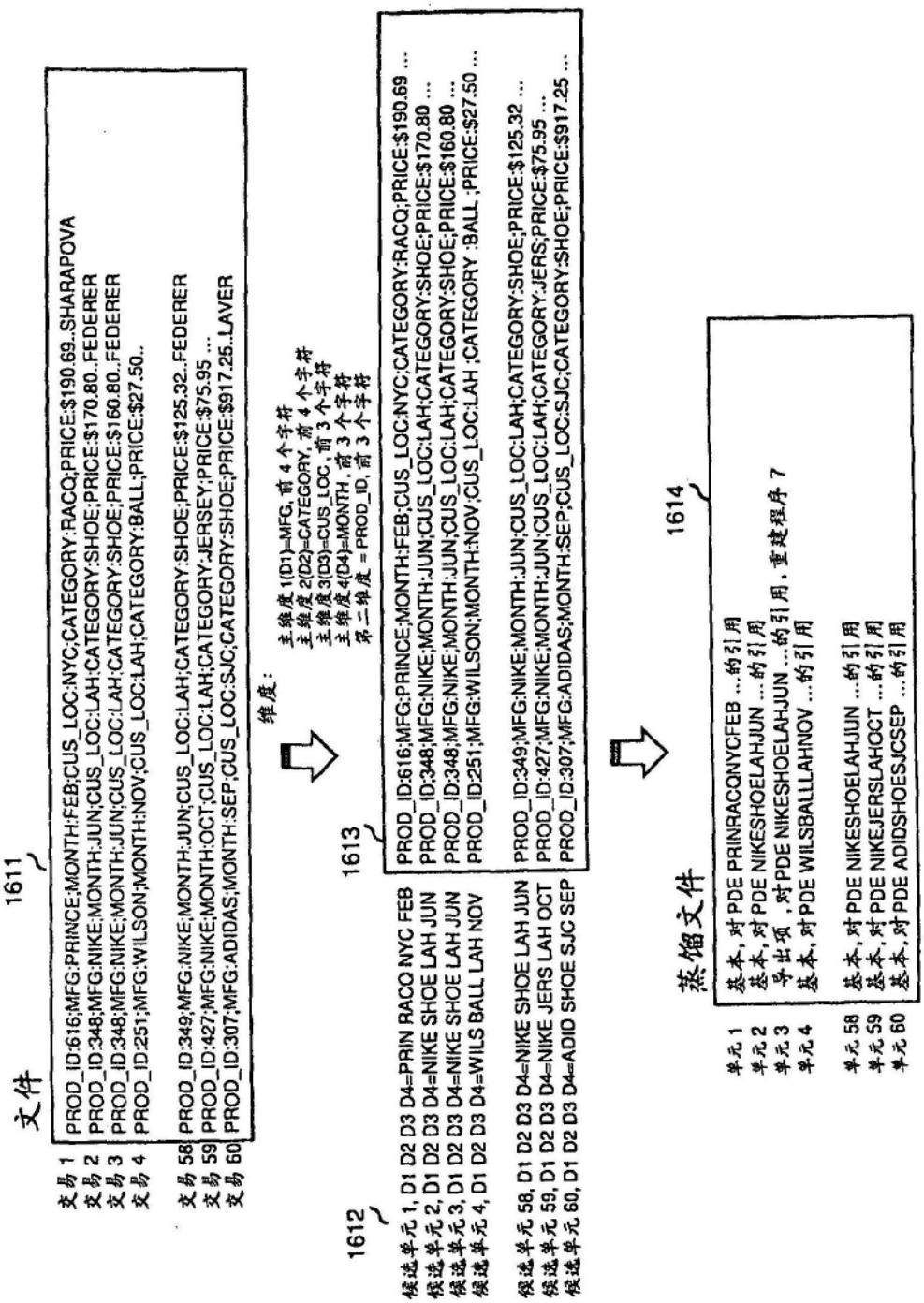

文件
1611
交易 1 PROD_ID:616;MFG:PRINCE;MONTH:FEB;CUS_LOC:NYC;CATEGORY:RACQ;PRICE:$190.69..SHARAPOVA
交易 2 PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$170.80..FEDERER
交易 3 PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$160.80..FEDERER
交易 4 PROD_ID:251;MFG:WILSON;MONTH:NOV;CUS_LOC:LAH;CATEGORY:BALL;PRICE:$27.50..
交易 58 PROD_ID:349;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$125.32..FEDERER
交易 59 PROD_ID:427;MFG:NIKE;MONTH:OCT;CUS_LOC:LAH;CATEGORY:JERSEY;PRICE:$75.95 ...
交易 60 PROD_ID:307;MFG:ADIDAS;MONTH:SEP;CUS_LOC:SJC;CATEGORY:SHOE;PRICE:$917.25..LAVER
维度:
主维度1(D1)=MFG, 前 4 个字符
主维度2(D2)=CATEGORY, 前 4 个字符
主维度3(D3)=CUS_LOC, 前 3 个字符
主维度4(D4)=MONTH, 前 3 个字符
第二维度 = PROD_ID, 前 3 个字符
1612
候选单元 1, D1 D2 D3 D4=PRIN RACO NYC FEB
候选单元 2, D1 D2 D3 D4=NIKE SHOE LAH JUN
候选单元 3, D1 D2 D3 D4=NIKE SHOE LAH JUN
候选单元 4, D1 D2 D3 D4=WILS BALL LAH NOV
候选单元 58, D1 D2 D3 D4=NIKE SHOE LAH JUN
候选单元 59, D1 D2 D3 D4=NIKE JERS LAH OCT
候选单元 60, D1 D2 D3 D4=ADID SHOE SJC SEP
1613
PROD_ID:616;MFG:PRINCE;MONTH:FEB;CUS_LOC:NYC;CATEGORY:RACQ;PRICE:$190.69 ...
PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$170.80 ...
PROD_ID:348;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$160.80 ...
PROD_ID:251;MFG:WILSON;MONTH:NOV;CUS_LOC:LAH ;CATEGORY :BALL ;PRICE:$27.50 ...
PROD_ID:349;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:SHOE;PRICE:$125.32 ...
PROD_ID:427;MFG:NIKE;MONTH:JUN;CUS_LOC:LAH;CATEGORY:JERS;PRICE:$75.95 ...
PROD_ID:307;MFG:ADIDAS;MONTH:SEP;CUS_LOC:SJC;CATEGORY:SHOE;PRICE:$917.25 ...
蒸馏文件
1614
单元 1 基本,对 PDE PRINRACQNYCFEB ...的引用
单元 2 基本,对 PDE NIKESHOELAHJUN ...的引用
单元 3 导出项 ,对PDE NIKESHOELAHJUN ...的引用,重建程序 7
单元 4 基本,对PDE WILSBALLLAHNOV ...的引用
单元 58 基本,对 PDE NIKESHOELAHJUN ...的引用
单元 59 基本,对 PDE NIKEJERSLAHOCT ...的引用
单元 60 基本,对PDE ADIDSHOESJCSEP ...的引用

图16B

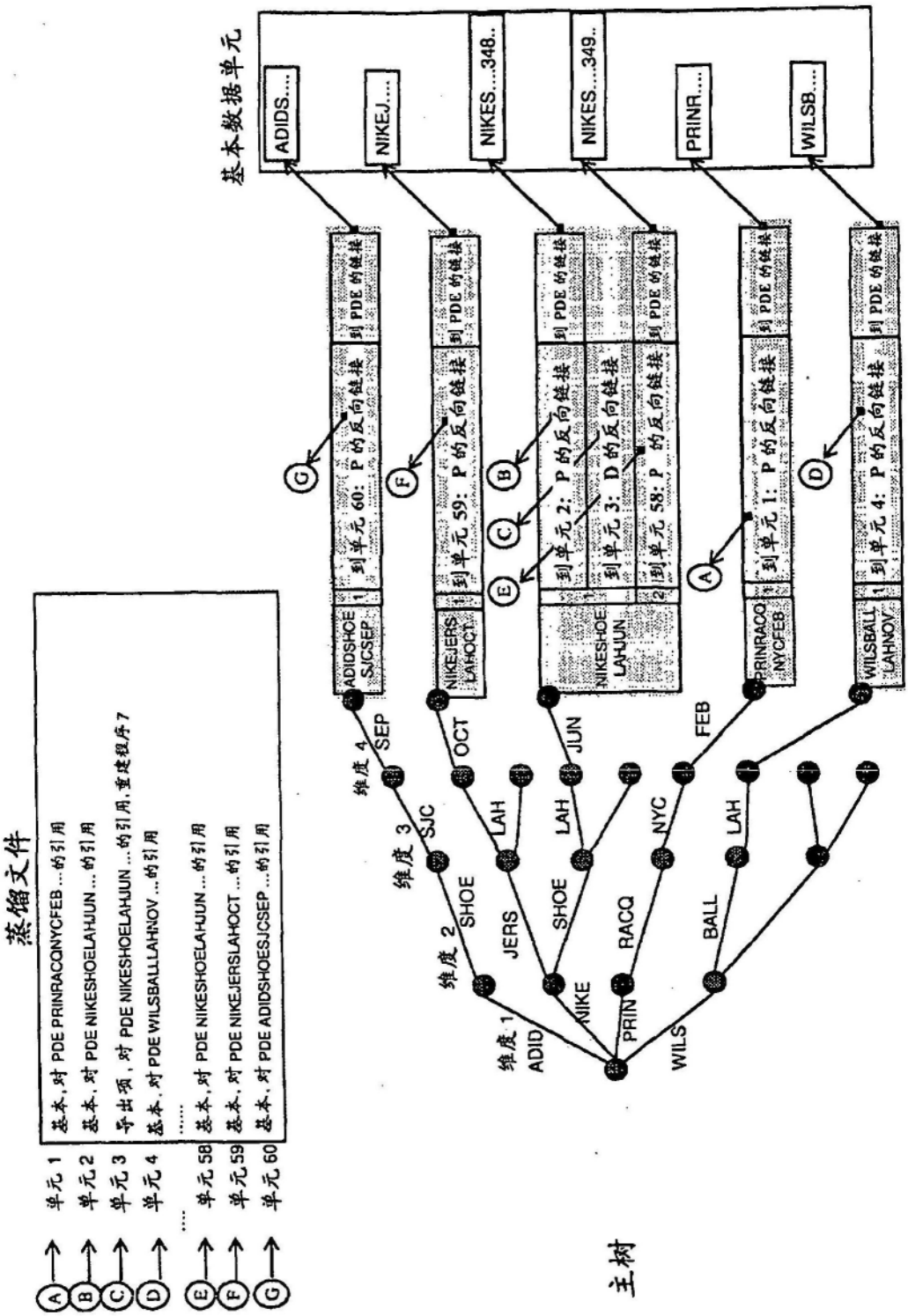

蒸馏文件
单元 1 基本，对 PDE PRINRACQNYCFEB ...的引用
单元 2 基本，对 PDE NIKESHOELAHJUN ...的引用
单元 3 导出项，对 PDE NIKESHOELAHJUN ...的引用，重建程序 7
单元 4 基本，对 PDE WILSBALLLAHNOV ...的引用
单元 58 基本，对 PDE NIKESHOELAHJUN ...的引用
单元 59 基本，对 PDE NIKEJERSLAHOCT ...的引用
单元 60 基本，对 PDE ADIDSHOESJCSEP ...的引用
基本数据单元
ADIDS....
NIKEJ....
NIKES....348..
NIKES....349..
PRINR....
WILSB....
ADIDSHOE SJCSEP
NIKEJERS LAHOCT
NIKESHOE LAHJUN
PRINRACQ NYCFEB
WILSBALL LAHNOV
到单元 60: P 的反向链接
到单元 59: P 的反向链接
到单元 2: P 的反向链接
到单元 3: D 的反向链接
到单元 58: P 的反向链接
到单元 1: P 的反向链接
到单元 4: P 的反向链接
到 PDE 的链接
维度 4 SEP OCT JUN FEB
维度 3 SJC LAH LAH NYC LAH
维度 2 SHOE JERS SHOE RACQ BALL
维度 1 ADID NIKE PRIN WILS
主树

图16C

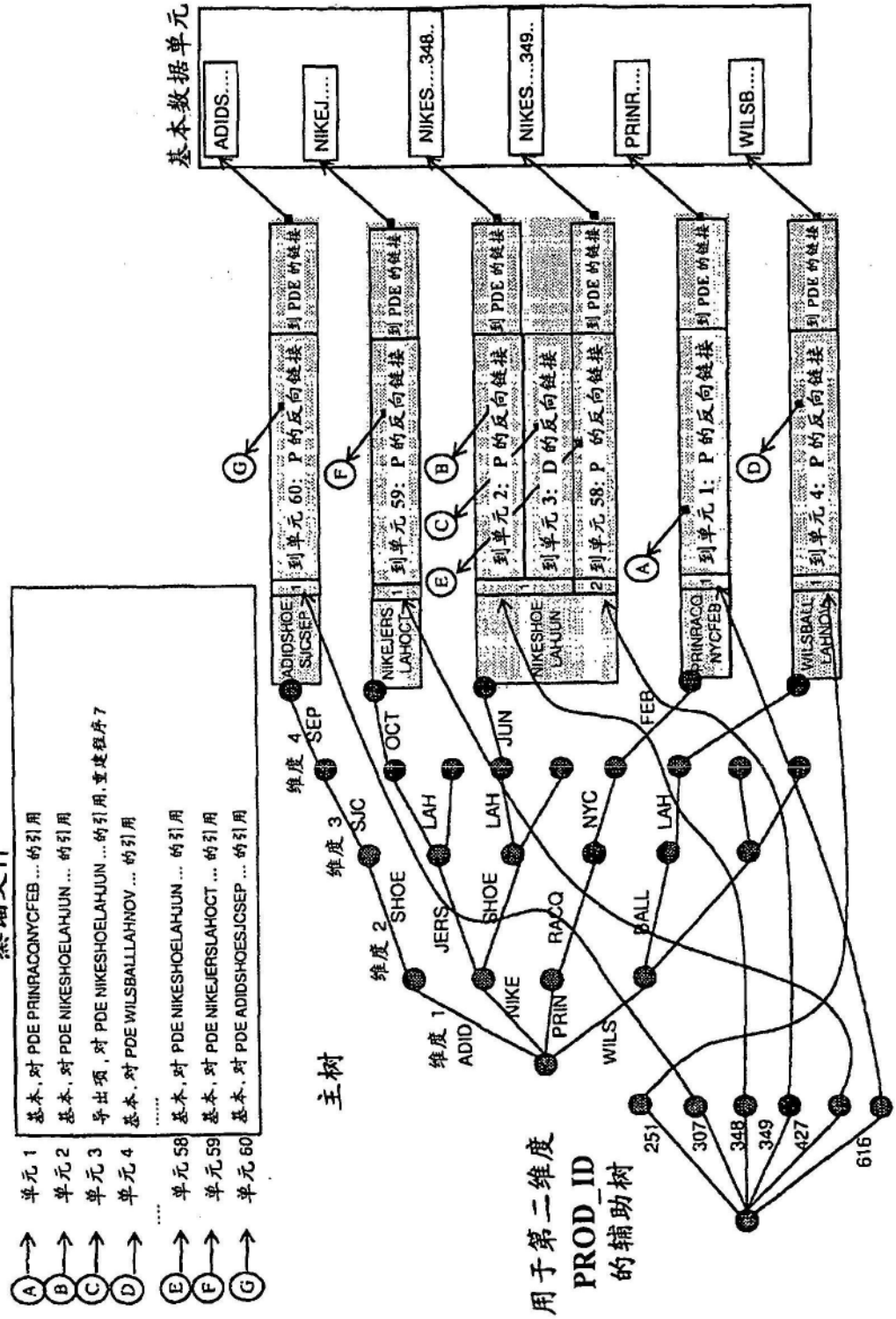

蒸馏文件
单元 1 基本，对 PDE PRINRACQNYCFEB ... 的引用
单元 2 基本，对 PDE NIKESHOELAHJUN ... 的引用
单元 3 导出项，对 PDE NIKESHOELAHJUN ... 的引用，重建程序 7
单元 4 基本，对 PDE WILSBALLLAHNOV ... 的引用
.....
单元 58 基本，对 PDE NIKESHOELAHJUN ... 的引用
单元 59 基本，对 PDE NIKEJERSLAHOCT ... 的引用
单元 60 基本，对 PDE ADIDSHOESJCSEP ... 的引用
基本数据单元
ADIDS....
NIKEJ....
NIKES....348..
NIKES....349..
PRINR....
WILSB....
到单元 60: P 的反向链接
到单元 59: P 的反向链接
到单元 2: P 的反向链接
到单元 3: D 的反向链接
到单元 58: P 的反向链接
到单元 1: P 的反向链接
到单元 4: P 的反向链接
到 PDE 的链接
ADIDSHOE SJCSEP
NIKEJERS LAHOCT
NIKESHOE LAHJUN
PRINRACQ NYCFEB
WILSBALL LAHNOV
主树
维度 1
维度 2
维度 3
维度 4
ADID
NIKE
PRIN
WILS
SHOE
JERS
RACQ
BALL
LAH
NYC
JUN
FEB
SEP
OCT
SJC
用于第二维度 PROD_ID 的辅助树
251
307
348
349
427
616

图16D

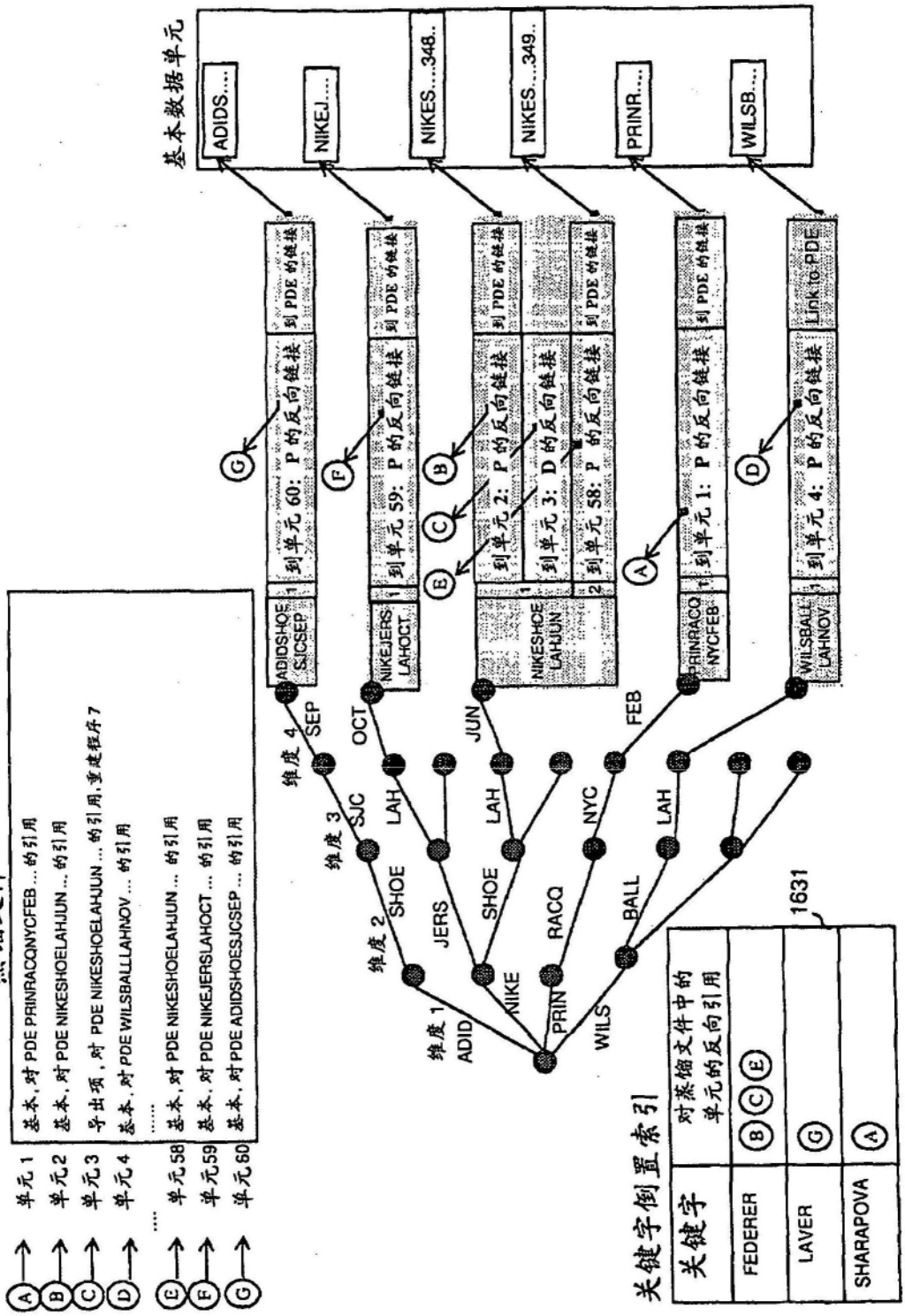

蒸馏文件
单元 1 基本，对 PDE PRINRACQNYCFEB ... 的引用
单元 2 基本，对 PDE NIKESHOELAHJUN ... 的引用
单元 3 导出项，对 PDE NIKESHOELAHJUN ... 的引用，重建程序 7
单元 4 基本，对 PDE WILSBALLLAHNOV ... 的引用
.....
单元 58 基本，对 PDE NIKESHOELAHJUN ... 的引用
单元 59 基本，对 PDE NIKEJERSLAHOCT ... 的引用
单元 60 基本，对 PDE ADIDSHOESJCSEP ... 的引用
基本数据单元
ADIDS....
NIKEJ....
NIKES....348..
NIKES....349..
PRINR....
WILSB....
到单元 60: P 的反向链接
到单元 59: P 的反向链接
到单元 2: P 的反向链接
到单元 3: D 的反向链接
到单元 58: P 的反向链接
到单元 1: P 的反向链接
到单元 4: P 的反向链接
到 PDE 的链接
Link to PDE
ADIDSHOE SJCSEP
NIKEJERS LAHOCT
NIKESHOE LAHJUN
PRINRACQ NYCFEB
WILSBALL LAHNOV
维度 1
维度 2
维度 3
维度 4
ADID
NIKE
PRIN
WILS
SHOE
JERS
RACQ
BALL
SJC
LAH
NYC
SEP
OCT
JUN
FEB
关键字倒置索引
关键字
对蒸馏文件中的单元的反向引用
FEDERER
LAVER
SHARAPOVA
1631

图16E

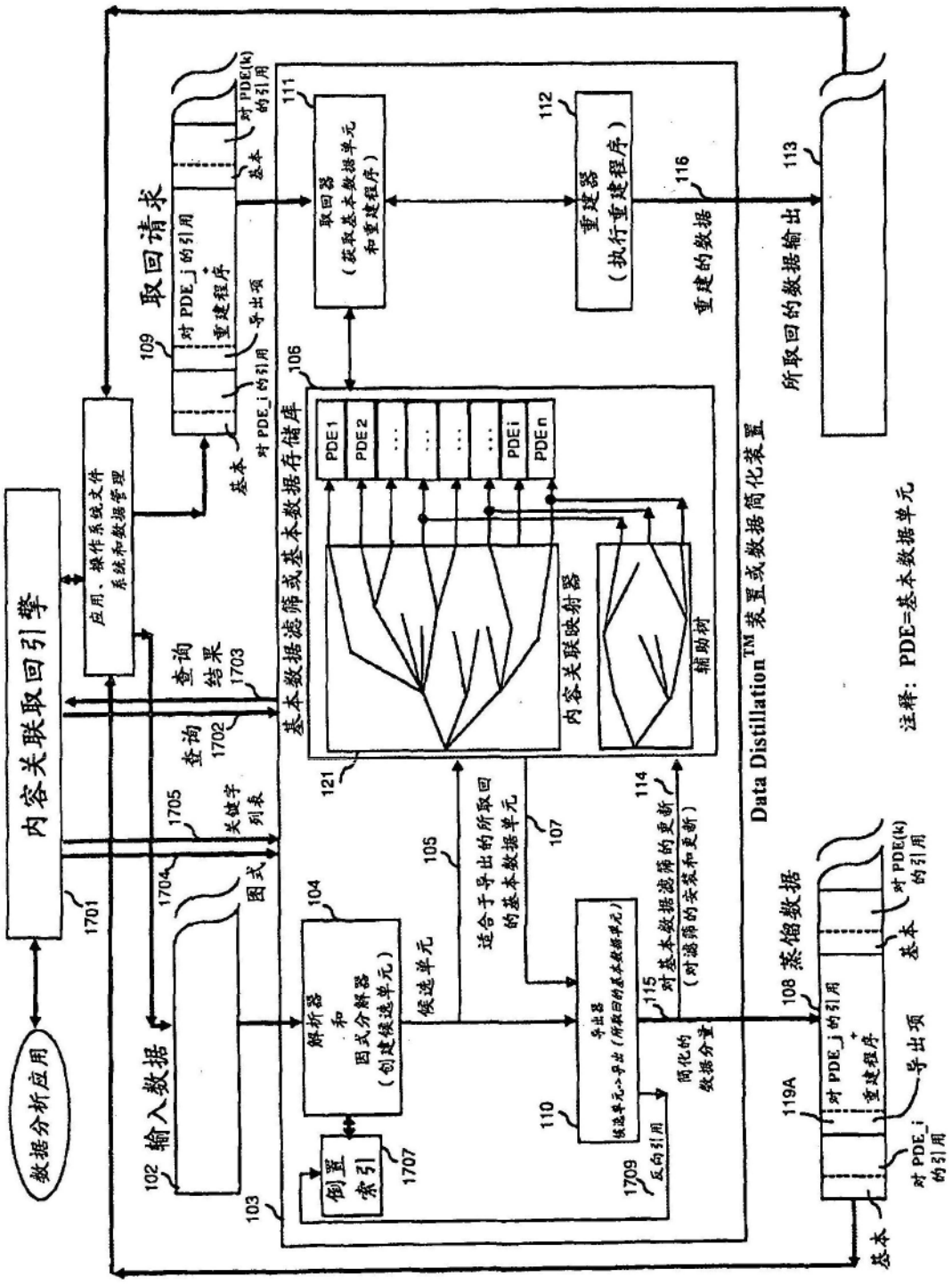

数据分析应用
内容关联取回引擎
输入数据
102
1701
1704
1705
关键字列表
查询结果
1703
查询
1702
应用、操作系统文件系统和数据管理
取回请求
109
对 PDE_j 的引用
重建程序
导出项
对 PDE_i 的引用
基本
对 PDE(k)的引用
基本数据筛选筛或基本数据存储库
106
PDE1
PDE2
PDEi
PDEn
内容关联映射器
辅助树
121
取回器
（获取基本数据单元和重建程序）
111
重建器
（执行重建程序）
112
重建的数据
116
所取回的数据输出
113
Data Distillation™ 装置或数据简化装置
注释：PDE=基本数据单元
解析器和因式分解器（创建候选单元）
104
候选单元
适合于导出的所取回的基本数据单元
105
107
114
对基本数据筛的更新（对筛筛的实现和更新）
导出器
（候选单元→导出（所取回的基本数据单元））
110
115
对基本数据筛的更新
简化的数据分量
108
蒸馏数据
119A
对 PDE_j 的引用
重建程序
导出项
基本
对 PDE(k)的引用
对 PDE_i 的引用
倒置索引
1707
反向引用
1709
103

图17

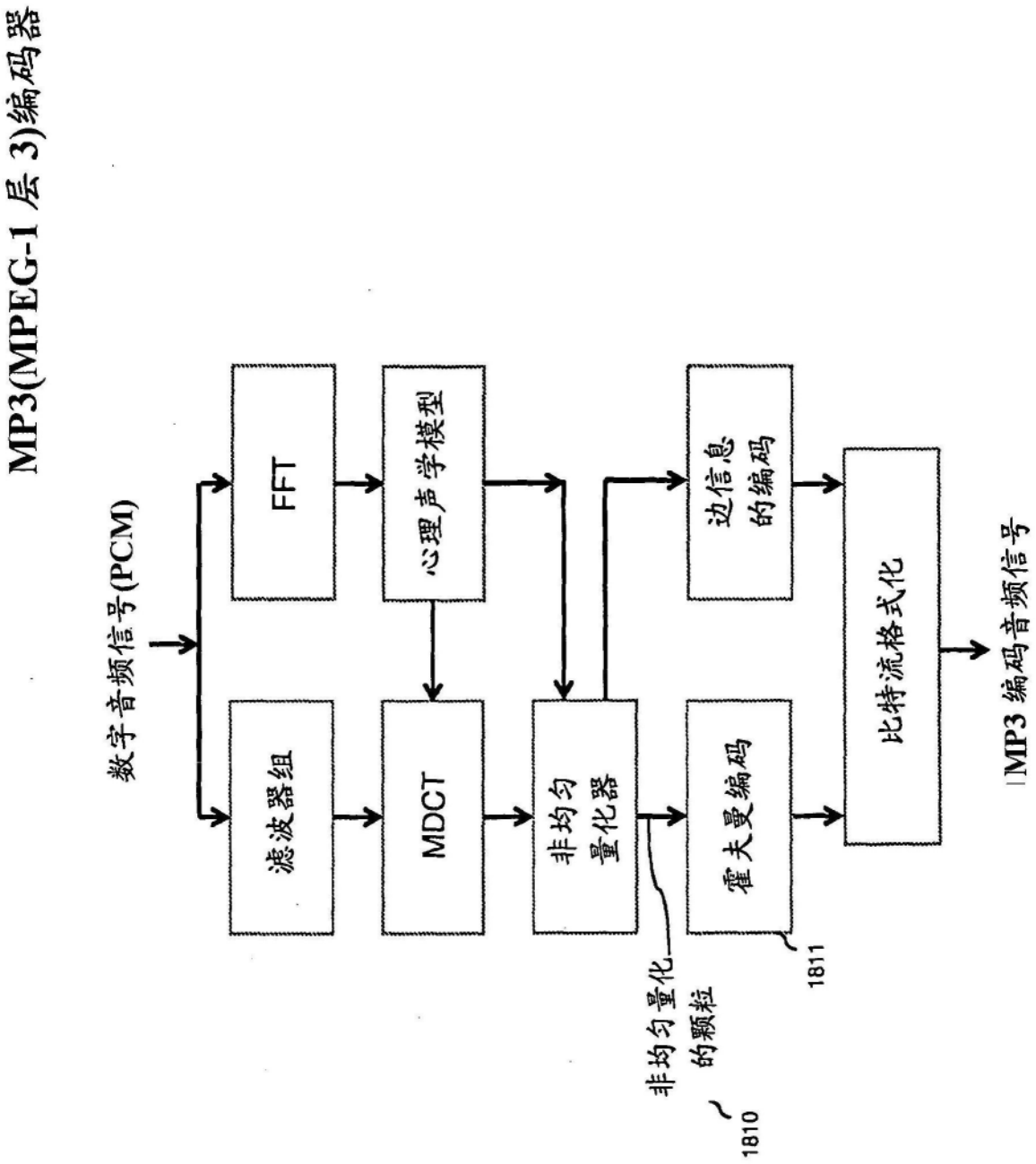

MP3(MPEG-1 层 3)编码器
数字音频信号(PCM)
FFT
心理声学模型
滤波器组
MDCT
非均匀量化器
霍夫曼编码
边信息的编码
比特流格式化
MP3 编码音频信号
非均匀量化的颗粒
1810
1811

图18A

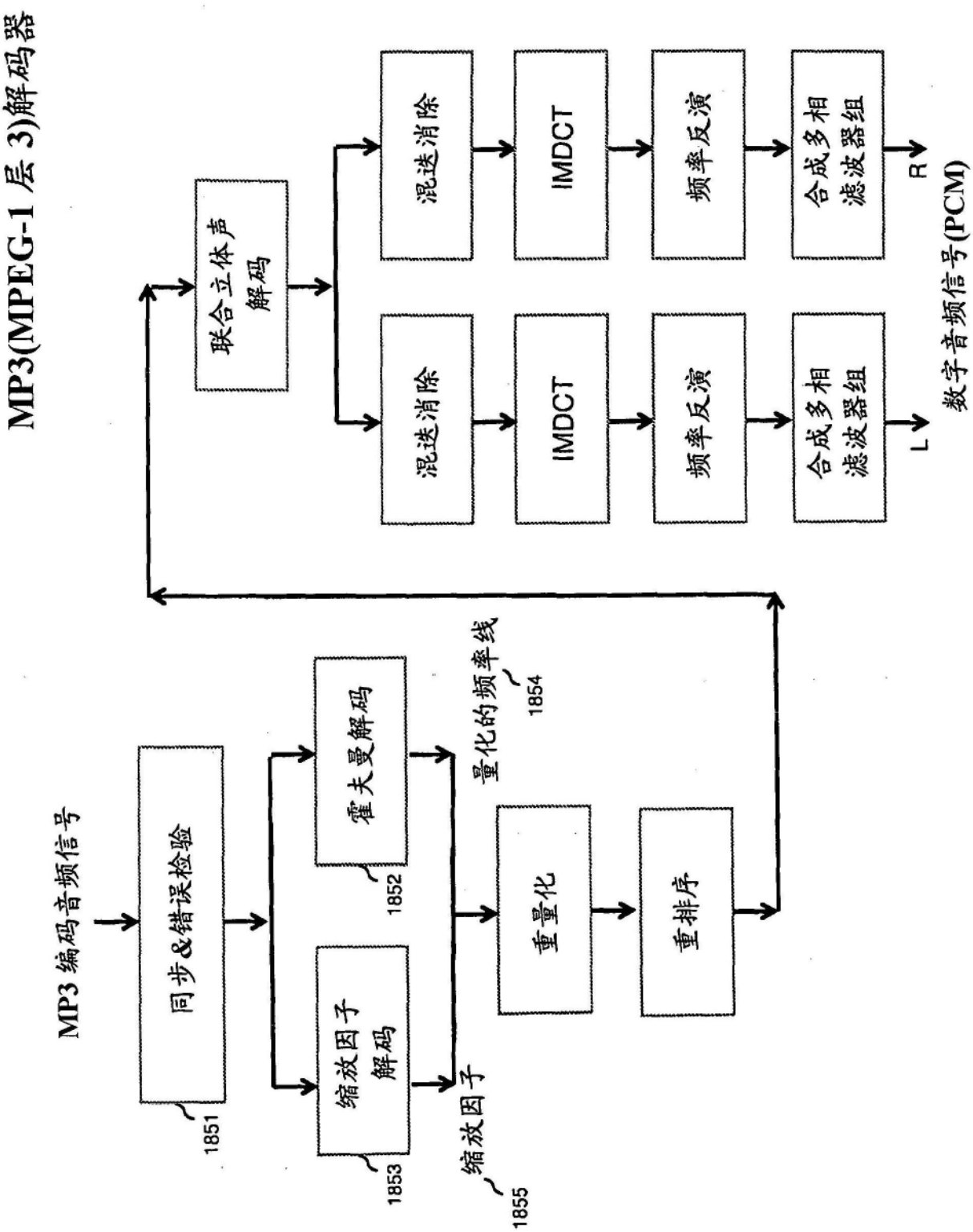

MP3(MPEG-1 层 3)解码器
MP3 编码音频信号
同步&错误检验
1851
霍夫曼解码
1852
缩放因子解码
1853
量化的频率线
1854
缩放因子
1855
重量化
重排序
联合立体声解码
混迭消除
IMDCT
频率反演
合成多相滤波器组
L
R
混迭消除
IMDCT
频率反演
合成多相滤波器组
数字音频信号(PCM)

图18B

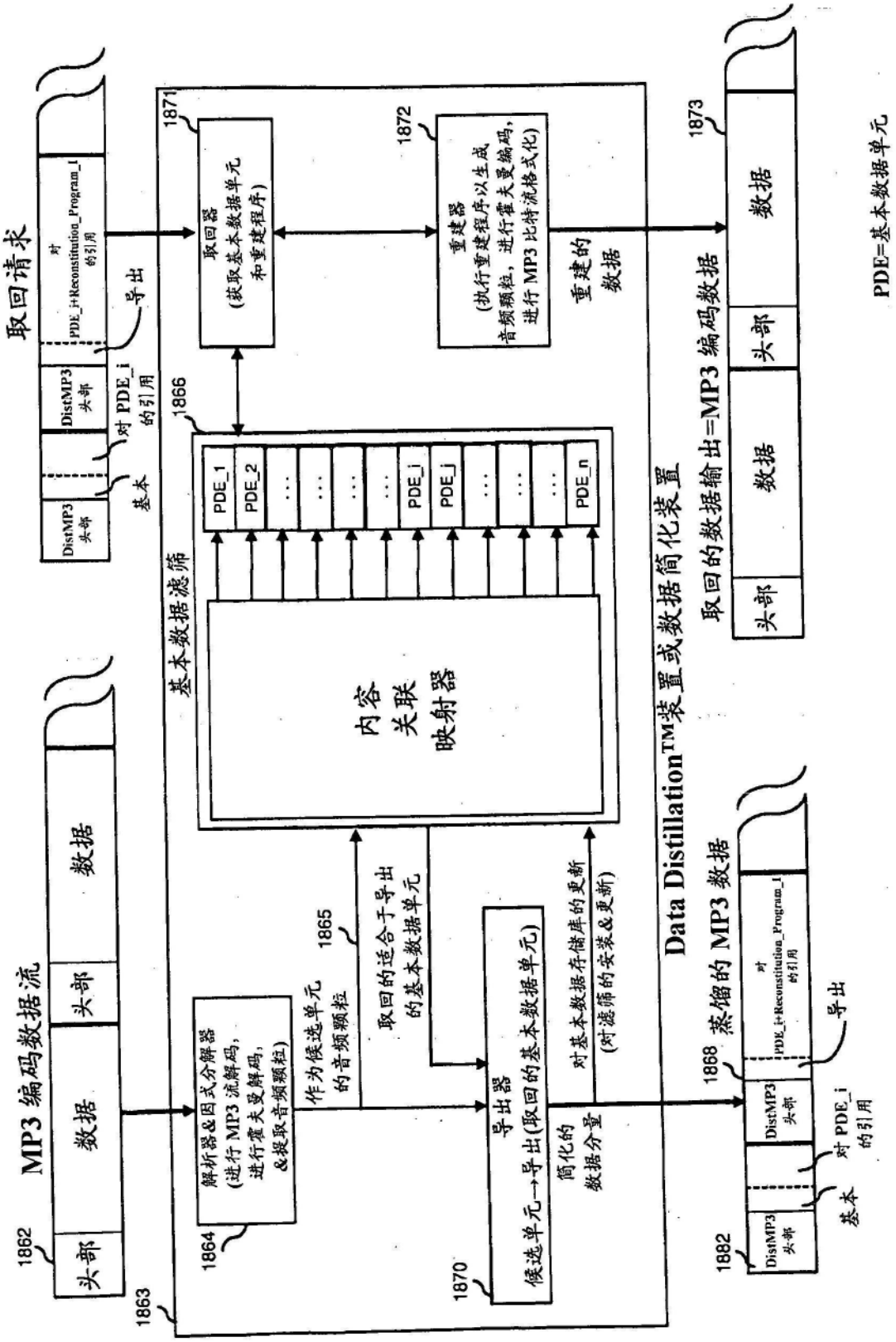

取回请求
对 PDE_i+Reconstitution_Program_1 的引用
DistMP3 头部
对 PDE_i 的引用
基本
导出
1871
取回器
(获取基本数据单元和重建程序)
1872
重建器
(执行重建程序以生成音频颗粒，进行霍夫曼编码，进行 MP3 比特流格式化)
1873
重建的数据
头部
数据
头部
数据
取回的数据输出=MP3 编码数据
PDE=基本数据单元
1866
基本数据滤筛
PDE_1
PDE_2
PDE_j
PDE_j
PDE_n
内容关联映射器
MP3 编码数据流
1862
头部
数据
头部
数据
1863
1864
解析器&因式分解器
(进行 MP3 流解码，进行霍夫曼解码，&提取音频颗粒)
作为候选单元的音频颗粒
1865
取回的适合于导出的基本数据单元
1870
导出器
候选单元→导出(取回的基本数据单元)
简化的数据分量
对基本数据存储库的更新
(对滤筛的安装&更新)
Data Distillation™装置或数据简化装置
1868
蒸馏的 MP3 数据
对 PDE_i+Reconstitution_Program_1 的引用
DistMP3 头部
对 PDE_i 的引用
导出
1882
DistMP3 头部
基本

图18C

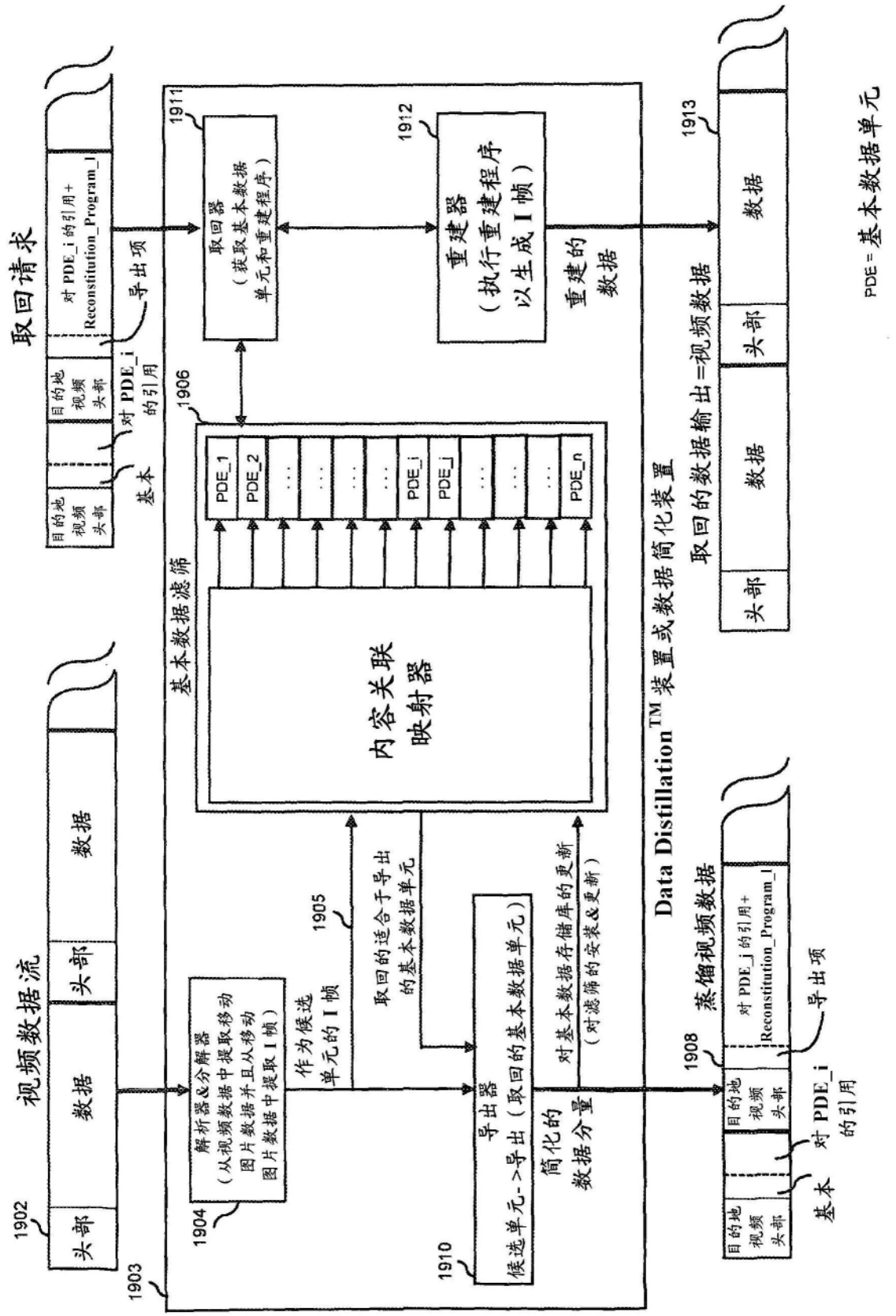

取回请求
视频数据流
头部
数据
头部
数据
1902
1903
1904
解析器&分解器
（从视频数据中提取移动
图片数据并且从移动
图片数据中提取I帧）
作为候选
单元的I帧
1905
取回的适合于导出
的基本数据单元
1910
导出器
候选单元->导出（取回的基本数据单元）
简化的
数据分量
对基本数据存储库的更新
（对滤筛的安装&更新）
基本数据滤筛
内容关联
映射器
PDE_1
PDE_2
PDE_i
PDE_j
PDE_n
1906
1911
取回器
（获取基本数据
单元和重建程序）
1912
重建器
（执行重建程序
以生成I帧）
重建的
数据
1913
数据
头部
数据
头部
取回的数据输出=视频数据
Data Distillation™ 装置或数据简化装置
1908
蒸馏视频数据
目的地
视频
头部
对PDE_j的引用+
Reconstitution_Program_I
导出项
对PDE_i
的引用
基本
目的地
视频
头部
对PDE_i的引用+
Reconstitution_Program_I
导出项
目的地
视频
头部
基本
对PDE_i
的引用
PDE = 基本数据单元

图19