$$=\begin{cases}\text{false} & \text{if } d=d_1 \vee d=d_2\\ \text{are-shared}(u,d_2,d_2) & \text{else}\end{cases}$$

are−share(create(u,d)d1,d2)

$$=\begin{cases}\text{false} & \text{if } d=d_1 \vee d=d_2\\ \text{are}-\text{share}(u,d_1,d_2) & \text{else}\end{cases}$$

(2)get−Value 规程

get−value(create(u,d),e)

$$=\begin{cases}\text{get}-\text{value}(u,e) & \text{if } d\neq e\\ \text{error} & \text{else}\end{cases}$$

get−value(set−type(u,d,t),e)

$$=\begin{cases}\text{get}-\text{value}(u,e) & \text{if } d\neq e\\ \text{error} & \text{else}\end{cases}$$

get−value(set−value(u,d,v),e)

$$=\begin{cases}v & \text{if are}-\text{shared}(u,d,e) \vee d=e\\ \text{get}-\text{value}(u,e) & \text{else}\end{cases}$$

get-value(share(u,d1,d2),d)

$$=\begin{cases}\text{get}-\text{value}(u,d_2) & \text{if } d_1=d\\ \text{get}-\text{value}(u,d) & \text{else}\end{cases}$$

(3)get−type 规程

get−type(set−value(u,d,v),e)=get−type(u,e)

get−type(create(u,d),e)

$$=\begin{cases}t_e & \text{if } d=e\\ \text{get}-\text{type}(u,e) & \text{else}\end{cases}$$

get−type(set−type(u,d,t),e)

$$=\begin{cases}t_e & \text{if } d=e\\ \text{get}-\text{type}(u,e) & \text{else}\end{cases}$$

get−type(share(u,d_1,d_2),d)

$$=\begin{cases}\text{get}-\text{type}(u,d_2) & \text{if } d_1=d\\ \text{get}-\text{type}(u,d) & \text{else}\end{cases}$$

(下转第44页)

(上接第85页)

库系统设计时采用的方法，去构造多媒体数据库系统，必须考虑到由于多媒体数据的引入所产生的各种影响。实践表明，分层设计的思想是开发软件系统的有效方法。利用这种方法，我们将庞大而又复杂的多媒体数据库系统分解为若干层次，下层作为上一层实现的虚拟机，把混杂在一起的系统功能的需求细化到每一层。我们可以自下至上逐步实现各层的功能，从而极大地降低了系统实现的复杂性。用这种方式所设计出来的多媒体数据库系统具有可靠性高、便于移植、易于维护等优点，每层的局部改动对整个系统的影响较小。采用这种思想，我们设计开发了多媒体数据库系统 CDB/M[1]，该系统的成功投入使用，也充分显示了这种分层设计方法的有效性。

参 考 文 献

[1] 周龙骧，柴兴无，分布式多媒体数据库系统的分层体系结构，计算机学报，19(7)1996

[2] M. Muhlhauser, et al., Services, Frameworks, and Paradigms for Distributed Multimedia Applications, IEEE Multimedia, Fall 1996

[3] E. Chang, et al, Reducing Initial Latency in Media Servers, IEEE Multimedia, July-September 1997

[4] T. L. Kunii, et al, Issues in Storage and Retrival of Multimedia Data, Multimedia. Systems, 1995,3

[5] J. H. Friedman, et al, An Algorithm for Finding Best Matches in Logarithmic Expected Time, ACM Trans. on Math. Software,1977,3

[6] S Dao, et al, MB*-Tree: An Index Structure for Content-Based Retrieval, In K. C. Nwosu et al, eds., Multimedia Database Systems, Design and Implementation Strategies, Kluwer Academic Publishers, 1996

[7] N. Beckmann, et al, The R* Tree: An Efficient and Robust Access Method for Points and Rectangles, In Proc. ACM SIDMOD Intl. Conf. on the Management of Data Atlantic City, May 1990

[8] D. V. White, et al, Similarity Indexing with the SS-Tree, In Proc. 12th Intl. Conf. on Data Engineering, New Orleans, Louisiana, February 1996

[9] N. Katayama, et al, The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries, In Proc. ACM SIDMODE Intl. Conf. on the Management of Data, AZ, USA,1997

[10] Chiueh Tzi-cker, Content-Based Image Indexing, In:Proc. 20th VLDB Conf. Santiago, Chile, 1994

[11] T. Bozkaya, et al, Distance-Based Indexing for High-Dimensional Metric Spaces, Conf. Same to [9]

[12] ISO, Hypermedia/Time-based Structure Language: HyTime(ISO 10744). ISO,1992

[13] ISO,Multimedia and Hypermedia Information Coding Expert Group. ISO/IEC JTC1/SC29/WG12,MHEG Working Draft"WD. 1. 0,"Version 1. 0(Feb. 1993)

[14] T. D. C. Little, et al, Synchronization and Storage Models for Multimedia Objects. IEEE Journal on Selected Area in Communications, 8,3(April 1990)

[15] G. A. Schloss, et al, Providing Definition and Temporal Structure for Multimedia Data. ACM Multimedia Systems(1995)3

[16] 巩志国，周龙骧，多媒体对象的 Agent 展示集成模型，软件学报已录用。

[17] R. Barber, et al, Query by Content for Largr On-Line Image Collections, Research Report RJ9408 (82660), June 1993, IBM Research Division, New York

[18] S. Hibino, et al, A Visual Multimedia Query Language for Temporal Analysis of Video Data, Same to [6]

[19] S. C. Kau,et al,MQL-A Query Language for Multimedia Database, In Proc. 2nd ACM Intl. Conf. on Multimedia, ACM Press, 1994

计算机科学1998Vol. 25№. 6

多媒体数据库系统的体系结构

An Architecture of Multimedia Database Systems

巩志国 周龙骧

（中国科学院数学研究所 北京100080）

董淑珍

（河北师范大学计算机系东校区 石家庄050016）

摘 要 In this paper, an hierarchical architecture of multimedia database systems is provided, the functions in each level of this architecture are pointed out, and the policies and methods to implement these functions are also analyzed in detail. This hierarchical designing ideas can simplify the systems' implementations, so the portability, maintainability as well as the reliability of multimedia database systems can be guaranteed.

关键词 Multimedia database systems, Hierarchical architecture, Storage schema level, Access schema level, Conceptual schema level, Integration schema level, Presentation schema level

近几年，计算机的计算能力、数据存储设备、数字化输入设备以及多媒体技术的发展异常迅猛，使计算机对数据的处理能力有了极大提高。多媒体数据已成为用户所需信息资源的重要组成部分。特别是Internet/Intranet技术的发展为全球化多媒体信息的交换与共享提供了有力的手段。基于网络环境的多媒体应用不断涌现，如电子商场(Online Shopping)、电子图书馆(Digital Library)、新闻点播(News On Demand)、多媒体教学(Multimedia Training)等。这些系统以五光十色的多媒体信息形式呈现给用户，使信息系统的发展进入了一个新的高潮。但是作为多媒体应用系统支撑环境的多媒体数据库技术的发展却显得缓慢，困难较多，所研究与开发的原型系统也都普遍带有偏向某一具体应用的倾向。开发一种高效且适合各种应用的多媒体数据库管理系统迫在眉睫。多媒体数据库管理系统(MDBMS)的设计已不能完全沿用传统数据库的设计思想，这主要是因为多媒体数据有其固有的特征：

1)其类型繁多，新的类型还不断涌现，而传统数据库管理系统的数据模式(Schema)一般为静态的，其基本的数据类型固定不变，显然不能够满足其需要。

2)它具有表现特征，即可视特征是其本质，属于它本身密不可分的属性，但传统数据更强调的是数据表现与数据表示的严格分离。

3)它具有多义性，不同行业的人可能会有不同的理解。而传统数据库系统中，数据表示与数据的含义是一一对应的。

4)用户对它的查询描述可能会带有主观性，用户可能根据自己的印象去描述查询需求。而传统数据库的查询一般只是基于严格的字符、数值的匹配。

5)它的展示具有实时性，系统必须按照一定的实时限制来展示它。

6)它的集成不但包括内容之间的集成，而且还要考虑它们之间的空间上的集成与时序上的集成。传统数据的集成一般只是内容上的合成。

7)由于网络带宽和系统处理能力的局限性，为了尽量多地满足不同用户的服务需求，系统不一定必须按照它的存贮属性提交给用户，即使其展示的尺寸、分辨率、播放速率有所降低，一般也不会影响用户对其理解。这又是与传统数据的重要区别。

8)其尺寸庞大，对传统的存贮管理方法也提出了挑战。

由此看出，为了有效地管理好多媒体数据，以支持各种不同的多媒体应用，我们必须认真考虑由多媒体数据新的特点所产生的各种影响，以便设计出高效而又实用的多媒体数据库管理系统。CDB/M是我们经过五年多的时间，设计开发出的一种新型

多媒体数据库管理系统[1]，现在已成功地投入了运行，本文基于我们实现 CDB/M 时的设计思想和开发方法，给出了多媒体数据库管理系统的一种分层体系结构，分析了每个层次实现时所遇到的问题，并给出了相应的解决方法和策略。

1 分层体系结构

七十年代迅速发展起来的系统设计方法学强调分层设计系统。将整个系统划分为功能相互独立的层次，层次之间界面清晰，衔接得当，每个系统层次构成一个虚拟机（抽象数据类型），它定义的数据结构和运算构成一个基本机器提供给它的上一系统层，而它自己又以它的下一层作为基本机器，以下一层提供的数据结构和运算作为基础来实现。这一分层抽象的构想如图1所示。利用这种方法所设计出的软件系统具有容易维护、便于移植、结构优良等特点。

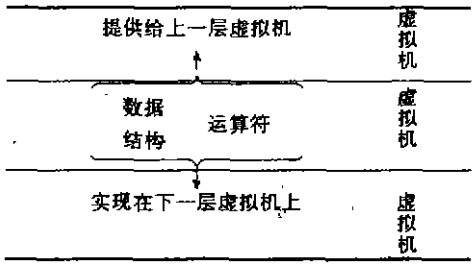

图1 分层抽象结构

根据这种思想，我们将多媒体数据库系统设计为如图2所示的分层体系结构。它的最底层为多媒体硬件层（Multimedia Hardware）MH，该层是支撑整个系统环境所需的各种硬件设备的集合，包括主计算机和外部设备。一般要求主计算机处理多媒体数据的功能要强，内存的尺寸要足够大。外部设备主要包括四类：(1)多媒体输入设备，包括录音机、话筒、录象机、摄象机、扫描仪和数字照相机等，以及相应的适配卡。(2)输出设备，包括大屏幕显示器、音箱、电视机、声频输出卡、视频输出卡等。(3)存储设备，如大容量磁盘阵列 RAID、光盘驱动器等。(4)通讯设备，如 Modem、Fax、LAN、WAN 等。

在 MH 层之上是多媒体操作系统层（Multimedia Operating System）MOS。该层主要屏蔽掉硬件设备操作的具体细节，使不同厂家所提供的硬件设备的操作接口统一。由于多媒体数据尺寸大，有实时性要求，传统操作系统对文件的管理以及任务调度的策略已不适合对多媒体数据的处理与展示。传统数据的展示与处理要求有较严格的正确性，所以要求操作系统进行必要的存取校验。多媒体数据对于正确性的要求不高，但对实时性却有一定的要求。另外，传统数据的存取具有随机性，而具有时序属性的多媒体数据的存取一般是连续单调的重复进行（一帧接一帧）。Muhlhauser 和 Gecsei[2]详细讨论了多媒体操作系统的基本特点，在此我们不再赘述。

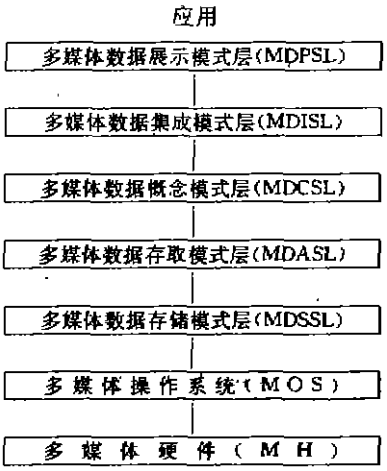

图2 多媒体数据库系统的分层体系结构

在 MOS 层之上为多媒体数据存储模式层（Multimedia Data Storage Schema Level）MDSSL，多媒体数据库管理系统的设计一般从该层开始，其设计任务的大小由操作系统对多媒体数据管理支持的程度而决定。该层主要完成多媒体数据在外存上的存贮分配，屏蔽掉物理存储设备调用的具体细节，并负责三级存储到二级存储的数据迁移，向上提供与物理位置无关的逻辑页面（Page），系统缓冲区由这些页面组成。另外，该层还负责内外存的交换。MDSSL 层之上为多媒体数据存取模式层（Multimedia Data Access Schema Level）MDASL，它以下层提供的系统缓冲区为舞台，将结构化的多媒体数据映射到缓冲区的逻辑页面上，因此 MDASL 层要负责多媒体数据在页面上的编码与编址。由于该层对结构化的多媒体数据到逻辑页面的对应关系最清楚，它还要提供高效的存取路径，以支持按数据结构、内容、时序关系和空间关系等要素对多媒体数据的快速存取。在多用户甚至多个客户（Client/Server 体系结构下）的环境中共享多媒体数据时，需要解决并发控制问题。由于多媒体数据应用中的事务格式（如电视会议）可能和传统的商业应用中很不相同，故 MDASL 层中的事务管理要解决许多与传统

DBMS 中的事务管理很不相同的新问题。与事务管理密切相关并对数据安全十分重要的恢复管理也是MDASL 层的重要任务，在 MDASL 之上是多媒体数据概念模式层（Multimedia Data Conceptual Schema Level）MDCSL，该层应当向它的上一层的各种应用提供中性的即不倾向于某一专门应用的逻辑视图，它对系统表示的客观世界进行中性的概念描述。它除了提供多媒体概念数据模型之外，还要进行数据库存取语言的编译，进行授权和存取检查，做完整性检查和存取优化。它还要对上层提供各种专用视图（即外模式或子模式）的支持。在 MDCSL 之上是多媒体数据集成模式层（Multimedia Data Integration Schema Level）MDISL。时序属性和空间属性是多媒体数据的本质特征，一组多媒体数据在提供给用户展示时，必须考虑它们之间的时序关系和空间关系，否则可能会影响用户对整个展示的理解，这就是集成层的任务。多媒体数据库系统的最上层为多媒体数据展示模式层（Multimedia Data Presentation Schema Level）MDPSL，该层的主要任务是提供多媒体数据库的使用接口。应当允许用户根据自己不同的喜好，对系统的熟悉程度，专业领域的特点等，来构造风格不一的使用界面。

2 策略和方法

用分层设计的方法将多媒体数据库系统复杂的功能细化到每一层，可使我们能够更加清楚地了解设计中所面对的各式各样的问题，并能适应软硬件环境的各种快速的发展变化。我们已经清楚了每层的主要功能，用什么样的策略和方法来完成这些功能是需要进一步考虑的问题。

2.1 存储模式层

2.1.1 存储分配 存储模式层的主要功能是管理系统缓冲区，管理外存和负责内外存交换。动态数据类型与静态数据类型在管理上区别非常大。动态数据具有时序特性，其尺寸庞大，一般要跨越若干个页面来存储。用户经常需要连续存取数据文件所在的页面，而系统一般是不可能将某个文件所在的所有页面同时缓冲在内存的。静态数据的尺寸相对要小，没有时序属性，一般随机存取其页面的可能性较大。由此可见，这两种数据类型的内外存交换策略以及页面大小的定义有不同的需求。传统的系统缓冲区管理方法对于动态数据类型的支持不足。在设计动态数据存储分配时应注意以下因素：(1)动态数据的输入一般为多通道异步输入或多通道同步输入两种方式。前者要考虑媒体流怎样与已经存储的媒体流交错存放，而后者需要考虑同时存放多个媒体流的交错方式。(2)动态数据一般需要随即存取和实时连续存取并用，分配算法应保证这两种存取要求，从底层将媒体流的抖动（Jitter）与延迟（Latency）限制在所容许的范围内。(3)对于媒体流的修改与编辑，不应产生较严重的硬盘重整。因为媒体流非常大，如果只考虑它的实时要求而采用连续分配的方案，当对媒体流的一部分进行修改时，就会产生严重的硬盘重整。这种情况应避免。(4)要充分支持对同一媒体流的不同部分并发存取的用户数。由于媒体流所包含的语义内容非常多，不同用户可能对媒体流的不同部分感兴趣。

当前从硬件上讲，已有了磁盘阵列技术标准 RAID0—RAID7，其中 RAID0—RAID5 已商业化，从而成倍地增加了磁盘的 I/O 和容量。从软件上讲，专家们也提出了各式各样的数据分配方案。受限块分配策略受到广泛重视[1]，这种方法是针对单硬盘的情况。

设有连续媒体 X_i，根据受限块分配的思想，将 X_i 分成 n 块：$X_{i,1}X_{i,2}\cdots X_{i,n}$。其中每一块被分配到硬盘的一段连续的物理空间。由于磁头从一个物理块移动到另一物理块需要一定的寻址（seek）时间，所以，为了保证流媒体展示的连续性，两个连续块 $X_{i,j}$、$X_{i,j+1}$之间的距离（磁道数）应该限定在某一固定的范围 d 内。即

$$|\mathrm{Track}(X_{i,j+1})-\mathrm{Track}(X_{i,j})|\leqslant d \qquad (1)$$

上式仅考虑了只有一个请求流的情况，如果同时有多个流需要展示，就会出现问题。设同时有两个流 X_i、X_{i+1}需展示，每个流都满足限定条件(1)。为了同时服务这两个请求流，系统需要在 X_i、X_{i+1}之间切换读取，例如在读完 X_i 的某一块 $X_{i,j}$后，磁头要移动到 X_{i+1}的某一块 $X_{i+1,k}$，很有可能 $|\mathrm{Tractk}(X_{i,j})-\mathrm{Tractk}(X_{i+1,k})|>d$。所以当磁头再回到 X_i 去读它的下一块 $X_{i,j+1}$时，所用时间早已超出了限定条件。为了克服以上的不足，人们提出了联合分配模式。该模式将整个磁盘分为大小等同的 R 个区，把每个流的连续块，按折返的循环方式依次分配到相应的区（如图3所示）。

这样，由于区域 R_1、R_2、R_3、R_4是平均划分的，所以每个流的两个连续块之间的距离是受限的。例如，设每个区域大小均为 H，则

$$|\mathrm{Track}(X_{i,j+1})-\mathrm{Track}(X_{i,j})|\leqslant 2H$$

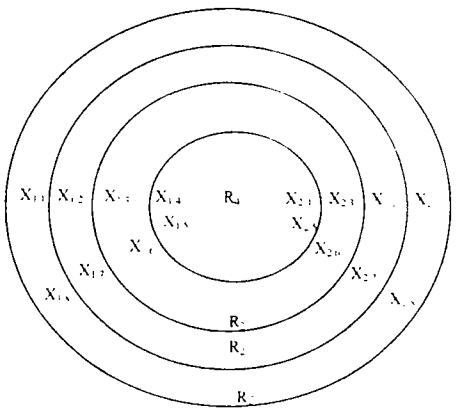

图3 联合分配模式

另外，在同一区域内，每个流的分配块是不受限的，可以在该区域的任何位置。在这种方式下每个流是折返循环存放的，例如流 X_1 是将 $X_{1,1}$、$X_{1,2}$、$X_{1,3}$、$X_{1,4}$ 依次分配到区域 R_1、R_2、R_3、R_4，然后再将 $X_{1,5}$、$X_{1,6}$、$X_{1,7}$，$X_{1,8}$ 按反向依次分配到 R_4、R_3、R_2、R_1 等等，这种折返方式称为双向模式。当有对 X_1 的请求时，磁头的运动方式同电梯的运动方式相似，依次扫过区域 R_1、R_2、R_3、R_4，然后再反向折回。在每一区域，都要读取任何流的请求块。例如，当存在同时对 X_1、X_2 的两个请求时，磁头在 R_1 中同时读取 $X_{1,1}$、$X_{2,1}$，在 R_2 中同时读取 $X_{1,2}$、$X_{2,2}$，…。对 X_1、X_2 的两个请求不一定同时发生，例如当系统正服务于 X_1 时，对 X_2 的请求可能发生在磁头恰好移动到 R_2 区域时。在这种情况下，系统对 X_2 请求的响应必须等到磁头再次返回到区域 R_1 时再开始。对于多数的应用来讲，这段初始等待时间是可以接受的。但对于有严格瞬时响应要求的应用，还必须对上述方法进行某些调整。Edward Chang 和 Hector Garcia-Molina 讨论了三种调整策略：单向模式、顺序模式以及成组扫描模式。这三种方式是对上述双向模式的适度改进，其细节可参见[3]。

由于磁盘阵列具有容量大、吞吐率高等优点，被广泛用于对流媒体的存储。若把一流媒体整个存放到其中的一个单磁盘中，当有多个用户同时访问该流媒体的不同部分时，该盘的 I/O 就成了多个请求的瓶颈。所以常用的方法是将整个流媒体分块，并将连续块循环分配到不同的单盘（图4中的左边四个盘），此方法称为条形分配方案（Disk Striping）[4]。假设一个媒体流含有 N 个块，有 d 个盘，则每个盘上被分配有 N/d 个不连续的块。这样极大地提高了对同一多媒体对象的并发存取数。但是这种分配方案对可靠性产生了影响，一个硬盘的故障会影响对多个流的访问，因为每个流都要跨越磁盘阵列的每个单盘。有效的解决方法是利用其中的一个单盘作为校验盘（如图4中右边的盘），而把流分配到其它盘上。校验可以用硬件完成，也可以用软件在内存中完成，一旦某一磁盘出现故障，可以利用同一条上其它各块以及校验块共同恢复故障盘上的数据块。

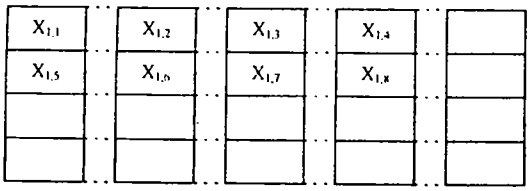

图4 条形分配方案

2.1.2 内外存交换 由于动态媒体其数据量大，而内存又不可能缓冲过多的数据页。传统的 FIFO 及 LRU 内外存交换算法均不适用。大量的多媒体应用具有很高的交互性，多媒体数据存储管理系统应了解这些交互模式以及当前应用中调用这些交互命令的频度，以便将交互的响应时间降到最小。内存管理中两个最主要的问题为预装入（Preloading）和替换（Replacement）策略，由数据请求命令引发的装入策略总不能很好地保证数据展示的连续性，有可能导致展示的抖动。预装入数据块的个数由内存大小、数据读取的时间以及数据的展示时间所决定。交互命令确定了哪些数据块需预装入，即数据块的方向和位置。应用中交互命令的频度确定了替换的策略。例如，Play 和 Fastplay 要求预装入的方向均为向前。但 Play 要求连续装入，而 Fastplay 则可间隔装入。在 VOD 应用中 Play 命令的频度较高，所以替换策略可选用“USE&TOSS”（用过则丢失）方案。但在多媒体编辑应用中，对于 Play 和 Backward 命令的调用频度大致相同，因此选用“USE&TOSS”策略就不合适。所以动态媒体的内外存交换一定要对交互命令模式及频度给予充分考虑，以保证实时性的要求。

2.2 存取模式层

2.2.1 编码与编址 多媒体数据的尺寸庞大，如果不对其进行压缩处理，计算机系统几乎无法对它进行存取和交换。而且整体上数据的冗余度也很大，在允许有一定限度失真的前提下，可以对图象数据进行很大程度的压缩。

一般讲，有两种类型的数据压缩方式：无损压缩和有损压缩。无损压缩能够完全恢复原始数据，但这种方法压缩比较低。典型方法有 Huffman 编码、Fano-Shannon 编码、Lempel-Zev 编码等。有损压缩，在解压恢复时不会达到原来的展示精度，但对于许多应用来讲并不受影响。这类方法如 JPEG、ADPCM、MPEG 等。其中 MPEG 为动态数据的压缩方法，该方法不但在空间上进行冗余压缩，而且还在时间上也进行了基于运动补偿的冗余压缩。

存取系统首先要完成对多媒体数据的压缩编码，然后再把压缩后的数据映射到存储系统所提供的逻辑页面上。静态数据的映射方式可采用关系数据库中记录的编址方式，如 TDD 法、分配表间接地址法等，但此时要注意压缩后的静态多媒体数据其尺寸大小不一的特点，当定义它们在页内的索引或指针时要充分考虑。对于动态媒体，我们一般不可能将整个媒体都放在同一个逻辑页面上，或将逻辑页面设计得足够大使其能够容纳下整个流媒体。从内存缓冲区的设置以及内外存数据交换的方式上讲，这是不可能的。因此，应该将整个流成组分配到每个页面上，每个组含有若干帧（或样值），连续组之间的时序关系要和页面间的顺序关系相对应，以便支持连续存取的需求。另外，由于时间上的冗余压缩，可能使帧的位置有局部性的变动（局部地不满足展示时的顺序关系，如 MPEG 编码），由于解码效率的原因，帧间的局部时序变动只能在同一页内，系统也必须能够根据每一页（而不是整个流）解码该页内的每一帧以及帧间的时序关系，以便支持对某一帧的随机存取。

2.2.2 存取路径　对于具有时序关系和空间关系的多媒体数据来讲，其存取路径结构要比字符数值型元组的存取路径复杂得多。对于时序数据，应该有按时间点和时间段存取数据流某一部分的路径结构，而对于空间数据，系统应提供按空间位置存取部分数据的路径。多媒体数据基于内容的存取路径其构造尤为困难，但却非常关键。称其关键是因为诸多的多媒体应用都需要提供这种支持；称其困难，是因为对多媒体语义内容的描述非常困难，且有不确定性。

相似查询是多媒体数据库基于内容查询的本质需求，能否有效支持这一特性是衡量该系统查询功能强弱的重要标志。一般讲，多媒体对象的内容由一组特征描述，如颜色直方图、物体形状、纹理等。由此可见，多媒体对象的内容可以表示为一个多维向量 $F=(f_1,f_2,\cdots,f_n)$。多媒体对象的相似就是指它们的特征向量之间的距离足够小。假设另有一特征向量 $Q=(q_1,q_2,\cdots,q_n)$，F 和 Q 之间的距离定义为：

$$d(F,Q)=(F-Q)A(F-Q)^T$$

其中 A 为一个 $n\times n$ 对称矩阵。一般讲，A 具有下列性质：

（1） A 为单位矩阵。在这种情况下，d(F,Q)与欧氏空间的距离等价。特征向量的各分量其重要程度相同，彼此之间互不相关。

（2） A 为对角矩阵。在这种情况下，特征向量的各分量有其不同的重要程度。各分量仍然互不相关。

（3） A 为对称矩阵（而非对角线矩阵）。此时不但各分量的重要程度不同，而且有一定的相关性。

第一种情况过于简单，不足以描述多媒体相似的需求。而第三种情况又过于复杂，所以应用上一般采用第二种方式。由此可见，多媒体对象的特征向量构成了 n 维空间的一个有限集合。为了支持相似查询，系统应该提供多维空间上的索引机制。传统的索引结构，如 B 树、倒排表、Hash 变换，已不能够满足多维空间的索引要求，从而人们设计了 K-d 树、MB* 树、R* 树、SS 树、SR 树、以及 VP 树和 MVP 树等索引结构。不失一般性，为了简单起见，我们以二维空间为例，对它们进行分析研究，并假设整个特征向量集合为 S。

K-d 树[5]是将数据集合 S 向每个坐标轴上作投影，选取最长投影值的中值作为切割点，将整个数据集合分割为两个（图5a），然后对每个子集进行同样的操作（图5b），依次进行下去直至每个集合充分小。这种分割方式简单整齐，对严格的匹配查询可快速查出所需内容。多媒体对象的相似查询是搜索出与目标点距离小于某一数值的所有对象，如果目标点在分割的边界上，系统就不得不对与该点相邻的各区域进行搜索，效率较低。MB* 树[6]的建立是通过先将 S 用垂直线分割（不一定等分），然后再对每一子区域用平行线独立进行分割（不一定等分），如图6所示。虽然这种分割方式比 K-d 树灵活，可以均衡各子区域所包含 S 中点的个数，使 MB* 树比较平衡。当进行相似查询时，仍然存在与 K-d 树相同的问题。

R* 树[7]是 R 树的变种，是为了快速查询矩形块而设计的多维索引结构，R* 树的每个结点对应一个矩形，该矩形是其所有的子结点所对应矩形的最小闭包（如图7）。页结点的矩形是该结点所包含 S 中点

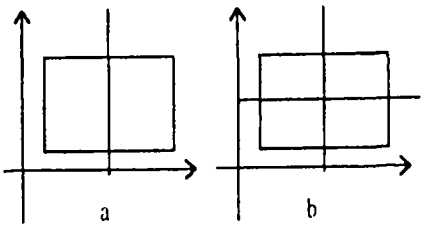

图5 K-d 树索引的建立

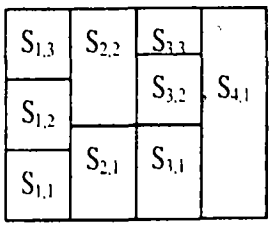

图6 MB*树索引的建立

的最小矩形闭包。由此可见，根结点对应于整个集合S的最小闭包矩形。R*树同K-d树相比有如下不同：(1)R*树结点是由其各子结点矩形的闭包来构造出，自底向上；K-d树则是利用沿各个轴平均分割的办法构造，自顶向下。(2)R*树每一层的结点可能相交。K-d树在同一层上的各结点互不相交。当查询一个单点时，K-d树只需在每一层搜索一个结点，R*树每层有可能搜索几个结点(因为同层结点的区域有可能相交)。但是，K-d树不平衡，可能含有许多空结点，这使得索引树庞大，降低了搜索速度。R*树平衡较好，使结点数减少，从而弥补了结点相交所造成的不足。经过大量实验和应用表明，R*树一般要优于K-d树。对于相似查询，系统要检索出的数据集合为：

$$ResultSet=\{X|d(X,Q)\leqslant d, X\in S\},$$

其中Q为目标点。在树的每一层，必须搜寻所有与ResultSet相交的结点。当Q位于结点的相交区域时，对查询速度会有很大的影响。随着空间维数的增加，查询效率非线性下降。

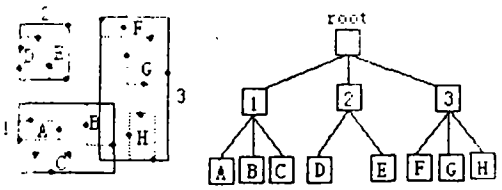

图7 R*树索引的建立

SS树[8]是对R*树的一种改进。R*树的每个结点都要记录该结点所对应的矩形，包括相应的顶点坐标和边的长度。随着维数的增加，使每个结点的存储量急剧膨胀，从而极大地降低了搜索效率。SS树利用最小闭包球面作为每个结点所对应的区域(如图8)，用圆心坐标和半径就可以将球面表示出来，使结点的存储尺寸明显减少。另外，SS树其结点的插入与分裂策略也进行了适当的改进，使整体性能较R*树有了一定的提高。但对于单点查询和相似查询存在与R*树相同的问题。SR树[9]的思想是基于对SS树和R*树所建立方法的综合，其中将每个结点同时用最小球面闭包和最小矩形闭包共同描述，显然增加了每个结点的存储开销，使SR树的创建较为困难。但是，通过用两种方式刻画结点区域，SR树改进了其查询算法，并通过实验说明了SR树比SS树和R*树的查询效率都要高，其细节请参见[9]。

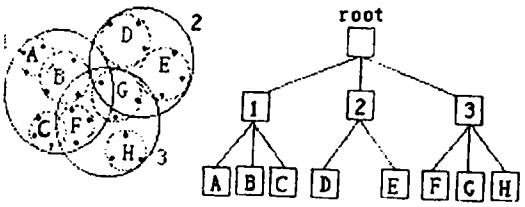

图8 SS树索引的建立

VP树[10]则是真正根据距离建立的多维空间索引结构，以克服以上索引结构对相似查询支持的不足。VP树索引结构的建立方式如下。在数据集合S中，选取一特殊点v(vantage point)，再由系统计算出S中的任何其他点p与v之间的距离d(p,v)($p\in S-\{v\}$)。取μ为所有这些d(p,v)的中值。然后将S按距离μ分成两部分S_1、S_2，如图9所示。最后再对S_1、S_2重复上述过程，递归进行，直到区域足够小。假设用户查询与给定点q的距离小于等于某给定阈值σ的所有点，只有在$\mu-\sigma<d(q,v)\leqslant\mu+\sigma$成立时，才需对$S_1$、$S_2$均搜索，其他情况仅需搜索其中之一。这种方式显著地增加了相似查询的搜索速度，其查询效率不会因维数的增加而明显下降，是一种新型多维空间索引结构。MVP树[11]是VP树的改进方式。首先在S中选取v1作为第一个vantage point。再在S-{v1}中选取v2使d(v2,v1)最大，v2作为第二个vantage point。仿照VP树建立的方式，先利用v1将S分成两个区域S_1和S_2，再用v2分别将S_1和S_2各分

成两个区域 $S_{1,1}$、$S_{1,2}$及 $S_{2,1}$、$S_{2,2}$。重复以上过程直到区域足够小。经实验，MVP 树比 VP 树的查询效率有了一定的提高，详情参见[11]。

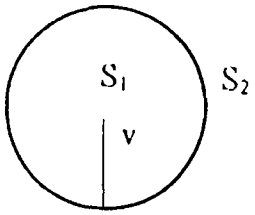

图9 VP 树索引的建立

2.2.3 并发控制 多媒体数据库支持各种类型的多媒体应用。有些应用，如协同设计，需要若干人同时操作于同一多媒体对象，并且修改操作是经常性的，为了保证数据的一致性，必须进行必要的并发控制。并发控制的目标是使事务集合的执行可串行化（事务执行的结果与某一串行执行的结果一致），常用的并发控制分为两类：封锁和乐观方法。封锁是指在某事务操作某一对象时，系统让有冲突操作的其它事务等待以达到可串行化。乐观方法则是允许其它事务继续执行，然后再在某一时间点上检测，将有冲突的事务回退。封锁适合于冲突操作频繁的应用；乐观方法则适合于冲突较少的应用。因此，上面所提到的协同设计应采用封锁机制。有些多媒体应用，如视频点播，很少甚至没有冲突操作，应该采用乐观方法甚至不加任何并发控制。由此可见，必须让事务管理系统了解应用的类型，以便采用不同的控制手段。

2.3 概念模式层

该层首先要解决的问题是，应当选择什么样的数据模型才适合对多媒体数据进行刻画。传统的数据模型经历了从层次模型，到网状模型，再到关系模型的逐步改进和发展。但是多媒体数据其内容丰富、操作复杂，空间关系和时序关系是其本质特征，模式识别的算法与多媒体数据密切相关。由此可见，多媒体数据模型一定要能够表达出多媒体数据之间的复杂语义关系，用户对多媒体数据的操作方式一定要尽量简单。可供选择的模型包括语义联系模型 SAM、面向对象的语义联系模型 OSAM、扩充的关系模型、扩充的 NF^2 模型、以及函数模型，但人们更倾向于采用面向对象模型 OOM。

多媒体数据具有多视性，即不同领域的人对其意义的理解可能会有不同，导致他们对多媒体数据操作各异。所以，系统除了以数据子集的方式提供对视图的支持外，还应该对每一领域维护该领域的一个知识库，以支持多媒体数据的多视性。语义网、框架均可选作领域知识的表示模型。另外，多媒体数据的领域知识可能是个不断完善的过程，因此理想的知识库系统还应具有学习和推理的功能。这样，用户不但可以利用显式的知识，而且还可以根据隐含的知识操作多媒体对象。

概念模式层要负责模式的演进、多媒体对象的定义、对象定义的维护、对象方法与数据属性之间的映射等。多媒体对象的数据来源不同于传统数据库。传统数据库中数据一般是由用户手工输入的，输入方式较为简单、一致。对于多媒体数据，其来源可能为一个已经存在的文件或来自扫描仪、摄影机或网络的输入。

存取优化对于数据库查询语言执行效率的提高至关重要，由系统自行优化存取操作可以使数据库复杂的存取策略对用户透明。存取优化的目的就是使中间结果最小化。优化方法分为两类，即代数优化和非代数优化。多媒体数据存取优化的难度在于其特征描述的不完全性。对于多媒体对象的查询应分为严格匹配和相似匹配。相似匹配的查询计算复杂、难度大，但对于多媒体数据库非常重要。相似匹配的基本依据为相似测度，它是多维空间的测度（例如，图象的相似可以用其颜色直方图、物体形状、大小、重心、方向等特征综合描述），查询所输出的结果一般是一组按相似程度排列的多媒体对象。无论是相似匹配的计算量还是其输出结果均比严格匹配大得多。因此在匹配筛选中，应该首先执行严格条件匹配，然后再进行相似筛选，这样可减少中间结果的大小和计算时间。多媒体对象的查询处理过程应该有领域知识库的支持，以便对查询语言中的模糊概念、相似程度等参数进行不同的解释。

2.4 集成模式层

多媒体集成是多媒体信息系统的重要组成部分，因为多媒体应用系统的开发，不可避免地要解决各种单媒体对象的集成问题。多媒体对象的集成是一种多维集成，包括空间、时序及内容三个方面。因此，该层要维护媒体多维集成的模型。

当前对集成模型的讨论比较多。HyTime 被推荐为国际标准[12]，该模型虽然对超链、事件调度、层次性元素都提供了强有力的支持，允许用户根据对象在测度空间的位置描述该对象。但是，由于这种描述方法利用链机制捕捉对象之间的同步，用户不得不花相当多的时间和精力去描述这些同步链接。也

就是说这些同步链是与实例相关的，而非模板式的。因此 HyTime 对同步的描述非常繁琐且易出错。另一个国际标准为 MHEG[13]，该模型是为了交换异构系统中所开发的多媒体和超媒体对象而定义的一个公共标准。这个公共标准是独立的基本的信息单位的规范表示。任何多媒体应用均可利用这个标准进行数据处理和交换。MHEG 利用面向对象的概念，给出这些对象的编码表示，但是对于对象的同步描述的支持非常有限。Little 和 Ghafoor 引入了 OCPN（对象编排 Petri 网）模型[14]。该模型直观方便，可用图形方式建立起媒体对象之间复杂的同步关系。但是模型要求给定对象的展示时间。这对于主动媒体对象（如视频、声频）不甚合适，因为由于系统或通信的原因，流媒体不一定能按预先给定的时间播放完。对用户来讲，有一定的延时和抖动一般也是可以接受的，OCPN 模型不能有效地支持这种特性。使用这种方法的另一个问题就是 OCPN 的图模式向可计算模型的转换问题。Schloss 和 Wynblatt 提出了分层多媒体数据模型 LMDM[15]。该模型自底向上的四个层次依次为多媒体对象、多媒体事件、多媒体展示、多媒体编排。多媒体对象采用面向对象的机制给出了多媒体对象的定义。多媒体事件定义了若干时序算子，利用这些算子描述多媒体的时序合成，但是这些合成算子显然违反了多媒体对象的封装特性。

我们采用 Agent 模型[16]描述多媒体对象的集成，该模型基于事件、条件和动作。该模型不但能描述多媒体对象的内容之间的集成关系，而且还包含对时序关系、空间关系的集成刻画。在该模型中，多媒体对象之间的同步由事件驱动，通过消息的传递实现，这样 Agent 和对象可以分布在不同的物理位置，系统的伸缩性大，成为真正的面向对象模型。由于我们引入了主动对象、被动对象及时钟的概念，使模型对多媒体之间的同步描述非常灵活，是一种动态集成模型。Agent 具有普通对象的结构，因此管理方式也同于一般对象。

2.5 展示模式层

多媒体数据库系统的最外层为展示模式层，它包括一组用户接口设计工具，用于支持不同媒体的展示、复合媒体的布局设计、多媒体展示的 QoS 描述及人机交互等。一般包括查询语言接口、浏览支持和多媒体编辑等功能。传统上，数据库的查询语言应该是概念层要解决的问题。但由于多媒体数据的可视特性，对查询的描述一般也要借助于可视化的手段方可表示清楚，所以我们把它归到展示层。

SQL 语言作为关系数据库的标准查询语言，具有简单易用、面向集合的特点。但这种语言处理多媒体的基于内容的查询并不十分有效，尤其对于多媒体数据的相似（Like This）查询更显无力。QBIC[17]是一种可视化的查询语言。它允许用户利用样本图象、自己建立的草图、颜色、纹理和关键词进行查询。TVQL[18]是一种为查询时序多媒体数据而设计的可视化查询语言，用户可以利用时序图构造查询界面。时序数据的内容由注解表示，其中每个注解是利用开始帧、开始时间、结束帧、结束时间、时间长度和主题内容等属性来描述的。因此，用户可以利用注解之间的时序关系、空间关系和主题概念来查询出所需的部分。MQL[19]是为多媒体数据库的查询而设计的一种语言，其语法结构为：

〈S〉::=SELECT〈A〉〈V〉FROM〈R〉WHERE〈C〉

其中〈A〉为所要显示的范围，〈V〉为多媒体对象的版本号，〈R〉是所要搜索的范围，〈C〉表示查询的约束条件。从形式上看，除增加了版本号外，其语法结构与标准的 SQL 语言相似。但是在它的查询条件中，可以用"CONTAINS"表示相似查询。

理想的多媒体数据查询语言应该允许用户根据数据类型、时序关系、空间关系、模糊概念和相似程度等来表达查询需求。应允许用户对结果的反馈操作以便逐步精化其结果。相似查询的结果应是在某种测度空间里的按相似程度排列的一组多媒体对象。因此，多媒体查询语言既要保留 SQL 语言简单易用、面向集合的特点，又要增加面向对象以及可视化的特点。

接口中对于浏览的支持不可缺少，尤其考虑到当前 WWW 技术的广泛应用，为全球信息共享奠定了基础。我们这里的系统接口通过对 HTML 文档格式的支持实现数据浏览功能。在我们的系统中，HTML 文档以多媒体对象方式存储在数据库中，HTML 文档实时生成。这样可以作为一个 Web 服务器，通过 HTTP 协议，利用 Mosaic、Netscape 或其它 WWW 浏览工具，实现远端访问数据库。接口的另一主要功能为媒体的编辑，包括对图象、文本、视频和音频等媒体的编辑修改功能。总之，用户可以通过接口提供的工具设计自己的多媒体应用系统，如电子购物、多媒体文档管理、电子图书馆等。

结束语 多媒体数据库系统的设计目标是使其能够有效地支持类型各异的多媒体应用。由于多媒体数据的固有特性，使我们不能完全按照传统数据

（下转第34页）