第 5 章

存 储 管 理

数据库是大量的有结构的持久数据集合。如何将这样一个庞大的数据集合以合适的形式组织起来存放在外存上？访问数据时需要先将数据读入内存，如何组织内存缓冲区？内外存交换的策略是什么？这些都是存储管理的关键问题。为回答这些问题，本章将从物理存储介质的特点入手，重点介绍磁盘数据库的逻辑与物理组织方式，包括用户数据记录的表示、记录如何在块中组织存储以及如何组织关系表的存放；介绍元数据的组织方法，讨论缓冲区管理与页面置换策略。

5.1 物理存储系统

5.1.1 存储介质概述

计算机系统中往往存在多种数据存储介质，代表性的存储介质包括高速缓存(cache)、主存储器(main memory)、磁盘(magnetic-disk storage)、固态硬盘(Solid-State Drive，SSD)、光盘(optical storage)和磁带(tape storage)等。

高速缓存用于缓存 CPU 的指令和其操作的数据，由计算机硬件管理它的使用。在 CPU 是瓶颈的系统中，高速缓存的使用方式显得尤其重要。DBMS 一般不管理高速缓存，但在设计查询处理的数据结构和算法时需要考虑高速缓存的影响。

主存储器就是通常所说的内存，它是 CPU 可以直接寻址的存储介质。主存储器具有易失(volatile)性，即在发生电源故障或系统崩溃时，其中的内容会丢失。

磁盘是最常用的持久存储介质，不会因电源故障或系统崩溃而丢失数据。CPU 不能直接寻址磁盘，为了访问磁盘上的数据，系统必须先将数据从磁盘读到内存，执行完操作后，必须把修改过的数据写回磁盘。

固态硬盘是一种新型的持久存储介质，使用闪存存储数据，提供与磁盘类似的访问接口。闪存的成本比内存低，比磁盘高，读写速度也是比

内存低，比磁盘高。

光盘和磁带主要用于备份数据和归档数据，它们通常容量很大，成本相对较低。

根据不同存储介质的速度和成本，可以把它们按层次结构组织起来，如图 5.1 所示。存储介质层次越高，价格越贵，但速度越快。高速缓存和主存储器称为一级存储，磁盘和固态硬盘通常用于联机存储数据，称为二级存储。光盘和磁带用于脱机存储，称为三级存储。虚线以上是易失性存储介质，虚线以下是非易失（持久）性存储介质。空心箭头和实心箭头均代表数据传输。

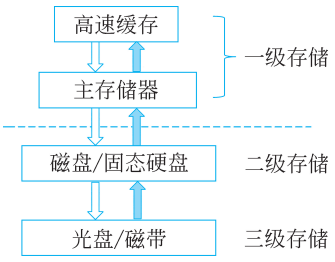


图 5.1　三级存储架构

磁盘和固态硬盘通过存储器接口连接到计算机系统，通常支持串行 ATA（Serial ATA，SATA）接口或串行连接的 SCSI（Serial Attached SCSI，SAS）接口。SAS 接口一般在服务器中使用，SAS 接口比 SATA 接口的数据传输速率高。非易失性存储器标准（Non-Volatile Memory Express，NVMe）接口是为了更好地支持 SSD 开发的逻辑接口标准，通常与 PCIe 接口一起使用。

5.1.2　常用存储介质

本节重点介绍几类常用的存储介质，包括磁盘、固态硬盘和独立磁盘冗余阵列（Redundant Array of Independent Disk，RAID）。

1. 磁盘

磁盘目前是计算机系统最主要的持久存储介质，对磁盘的访问效率直接影响系统的性能。

磁盘通过其盘片表面的磁性材料记录数据信息，盘片的表面在逻辑上可以划分为多个磁道（track），磁道又可以划分为扇区（sector），扇区是磁盘 I/O 的最小单位，扇区大小通常为 512 字节。对于磁盘来说，能保证一次读写正确的单位是一个扇区。

磁盘质量的主要指标是容量、访问时间、数据传输率和可靠性。

磁盘上数据的读写是通过安装在磁盘臂上的读写头完成的，读写头通过在盘片上的移动访问不同的磁道。为了读写磁盘上的数据，磁盘臂必须先移动到正确磁道的上方，然后等待磁盘旋转到它下方指定的扇区。磁盘控制器（disk controller）接收高层次的读写扇区的命令，然后开始操作，将磁盘臂移到正确的磁道，并对数据进行读写。

磁盘臂定位的时间称为寻道时间（seek time），其依赖目标磁道与磁盘臂初始位置之间的距离，典型的寻道时间为 2～20ms。平均寻道时间是在一个随机请求的序列上测量出来的寻道时间的平均值，其取决于磁盘的型号。读写头到达所需的磁道，等待磁盘旋转的时间称为旋转延迟时间（rotational latency time），平均延迟时间应该是磁盘旋转一周时间的一半。需要访问的扇区旋转到读写头下方后，就可以开始数据传输了。因此，访问时间是寻道时间、旋转延迟时间和数据传输时间的总和。

磁盘 I/O 请求通常由文件系统发出，直接操作裸设备上数据的数据库系统也可以发出磁盘 I/O 请求。每个请求指定要访问的磁盘地址，该地址以磁盘块号的形式提供。磁盘块（disk block）是存储分配和检索的逻辑单位，它包含固定数目的连续扇区，例如 Linux 操作系统默认的磁盘块大小是 4KB，数据在磁盘和内存之间以磁盘块

为单位传输。

可以看出数据库 I/O 请求数据块是由数据库系统发给操作系统，再由操作系统发给磁盘控制器，如图 5.2 所示，通常数据库系统的数据块大小是操作系统的数据块的倍数，操作系统的数据块大小是磁盘扇区大小的倍数。

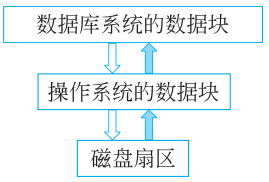


图 5.2 I/O 请求的层次

磁盘的访问分为顺序读写和随机读写两种模式。顺序读写模式针对连续编号的磁盘块，只需要对第一个磁盘块寻道，因此顺序读写模式寻道时间短。随机读写模式针对随机分布在磁盘上的块，对每个请求都进行一次寻道。显然，随机读写模式下数据访问速度明显低于顺序读写模式。通常使用每秒 I/O 操作的次数(I/O Operations Per Second，IOPS)表示磁盘的访问效率。

最后一个经常使用的磁盘度量指标是平均故障时间(mean time to failure)，即可以期望的无任何故障连续运行的时间，这是磁盘可靠性的度量指标。

2. 固态硬盘

闪存有 NOR 和 NAND 两种类型，其中 NAND 闪存主要用于数据存储。使用 NAND 闪存构建的固态硬盘提供与磁盘相同的面向块的接口。其闪存芯片主要由物理块组成，每个物理块分为一定数量的物理页。块是擦写操作的基本单元，而页是读写操作的基本单元。与磁盘相比，SSD 有更好的随机读写速度，并且功耗也明显低于磁盘。

SSD 的写操作比较复杂，必须先擦除再重写。擦除操作必须在擦除块(erase block)上执行，擦除块通常为 256KB～1MB，因此，SSD 通常存在“写放大”的问题。另外，闪存页的可擦除次数是有限的，通常为 100 000～1 000 000 次，一旦超出这个限制，就可能出现错误。

闪存系统通过将来自文件系统的逻辑页号映射到 SSD 的物理页号来降低块擦除速度慢以及更新次数限制的影响。当一个逻辑页被更新时，可以把它重新映射到已经被擦除的任何物理页中，从而避免写放大的问题。每个物理块都有一个小的存储区域保存它的逻辑地址，如果逻辑地址被重新映射到一个不同的物理块，则原来的物理块标记为已删除，包含多个删除页的擦除块要定期擦除。

现代的存储系统都支持磁盘和 SSD 的组合使用，频繁访问但很少更新的数据适合放在 SSD 中。

3. 独立磁盘冗余阵列

现在很多应用的数据对存储需求量非常大，需要多个磁盘存储这些数据。为了提高磁盘的性能和可靠性，独立磁盘冗余阵列(RAID)的磁盘组织技术应运而生。RAID 系统通过冗余技术提高了磁盘的可靠性，通过并行操作提高了磁盘的读写性能。RAID 系统具备的高可靠性、高性能以及易于管理和操作的特性，使得它在数据库系统中得到广泛的应用。

1) 数据冗余技术

冗余是解决可靠性问题的重要手段。对于磁盘来说，最简单的冗余方法就是复制每个磁盘的内容，这种技术称为**镜像**(mirroring)，其中的一个磁盘称为数据盘，另一个磁盘称为冗余盘。这样，一个逻辑磁盘由两个物理磁盘组成，每次写操作都必须在

两个磁盘上执行。如果其中一个磁盘出现故障，则可以从第二个磁盘中读取数据，从而提高数据的可靠性。

镜像技术逻辑简单，但成本昂贵，下面介绍另一种称为“奇偶校验块”的数据冗余方法。对于一个给定的块集合，其奇偶校验块的第 i 位是集合中所有块的第 i 位的“异或”(XOR)。即如果 a、b 两个值不相同，则 a XOR $b=1$；如果 a、b 两个值相同，则 a XOR $b=0$。如果一个集合中任何一块的内容由于故障而丢失，则可以通过计算集合中剩余块的位异或与奇偶校验块恢复出该块的内容。

例如，假设有 3 个数据盘，1 个冗余盘，每个数据块由 8 位组成，奇偶校验块存储在冗余盘中。如果在数据盘中的第一块有如下数据序列：

盘 1：11110000

盘 2：10101010

盘 3：00111000

则冗余盘的第一块的奇偶校验位如下所示：

盘 4(冗余盘)：01100010

简单来说，所有位 XOR 操作的结果就是：如果所有位中有偶数个 1，则校验位为 0；如果有奇数个 1，则校验位为 1。例如，盘 1、2、3 的第一位取值为 1、1、0，有 2 个 1，则第一位的校验位为 0。可以看出，所有位与它的奇偶位的集合中 1 的个数总和一定是偶数。

冗余盘的存在，对于读操作没有影响，但是对于写操作，在修改数据盘的同时，必须同时更新冗余盘。

如果其中一个盘的数据出现故障，则可以通过其他盘的数据和冗余盘上的奇偶校验位推断出故障盘中该位的数值。但是如果两个或两个以上盘的数据出现故障，则通过奇偶校验位就无法推断出故障数据的数值。这种方法在一定程度上解决了数据可靠性问题，成本相对来说比镜像方法低很多。

2) 数据条带技术

多个磁盘的存在为数据的并行处理提供了可能。把数据拆分到多个磁盘上，对数据的读写就可以在多个磁盘上并行执行，从而提高读写性能。这种把数据拆分到多个磁盘上的技术称为数据条带(striping data)技术。

最常用的数据拆分方法是按块拆分(block level striping)，将数据块拆分到多个磁盘上。这种方法是把磁盘阵列逻辑上看作一个大磁盘，并对块进行逻辑编号。假设逻辑块号从 0 开始，则对于 n 个磁盘的阵列，逻辑号为 i 的块在物理上分派到第 (i mod n)+1 磁盘上。在进行大型文件读写时，可以并行地从 n 个磁盘上同时读取 n 块。

3) RAID 级别

从前面的介绍可以看出，数据条带化能提高磁盘数据的读写性能，冗余则能提高磁盘的可靠性，但是不同的方法其成本和可靠性程度是不同的，在成本、可靠性、性能之间权衡，可形成多种 RAID 方案，称为 RAID 级别。

RAID 共有 7 个级别(0～6)，其中 2、3、4 目前不再使用。

RAID0 是指具有块级拆分但没有任何冗余的磁盘阵列，如图 5.3(a)所示，数据的条带分布在两个磁盘中。

RAID1 是指具有块级拆分的磁盘镜像，如图 5.3(b)所示，数据拆分成条带，两个磁盘中的数据一模一样。有些厂商使用 RAID10 表示带拆分的镜像，不带拆分的镜像称为 RAID1。

RAID5 指奇偶校验块交叉分布的磁盘阵列。在前面的例子中，把所有的奇偶校验块都放到一个盘上(冗余盘)，其带来的问题是所有数据盘的修改都会修改冗余盘，如果有 n 个数据盘，则对冗余盘的写次数是数据盘平均写次数的 n 倍。RAID5 对这种冗余方式进行了改进，把奇偶校验块分布到所有盘中，这样，每个盘都可以作为某些块的冗余盘使用，如图 5.3(c)所示。

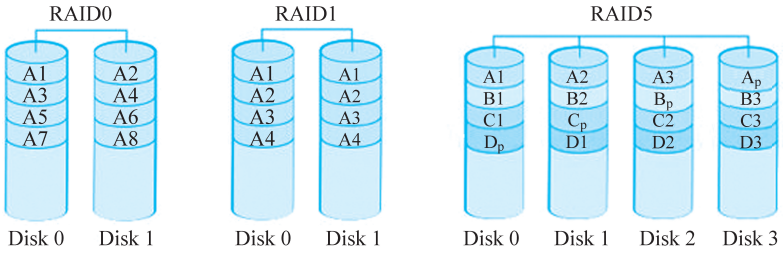


图 5.3 RAID 级别

RAID5 可以保证只有一个磁盘出现故障数据可以恢复，但是多个磁盘出现故障就无能为力了，RAID6 使用纠错码存储冗余，可以应对多个磁盘发生故障的情况，相关知识可以参看相关文献，这里不再展开讲解。

在实际的应用场景中，选择 RAID 级别时应该考虑的因素包括：

(1) 数据冗余带来的成本；

(2) 应用对磁盘 I/O 性能的需求；

(3) 对磁盘数据可靠性的需求。

RAID0 的性能最好，没有数据冗余，用于对性能要求高，而数据可靠性并非至关重要的应用场景。RAID1 用于对数据可靠性要求高，磁盘成本不是重点考虑因素的应用场景，其写性能优于 RAID5。RAID5 有更小的存储开销，但随机写的开销较大，适合读多写少的应用场景。RAID6 保证了更高级别的安全性，存储成本高于 RAID5。

5.1.3 磁盘 I/O 性能的提升策略

在大多数情况下，磁盘 I/O 是影响系统性能的瓶颈，因此如何优化磁盘 I/O 受到极大关注。这里不讨论硬件的升级，只讨论如何从软件角度缓解 I/O 瓶颈。通常的思路包括：①尽可能减少物理 I/O 的次数；②尽可能提升 I/O 的性能；③尽可能并行起来，例如采用异步的方式，而不是等待。

具体地，缓解 I/O 瓶颈的常用措施包括：

(1) 使用缓冲区，把常用的数据块缓存在内存中，减少物理 I/O 的发生。当然使用缓冲区相当于数据冗余，需要考虑数据一致性的问题。

(2) 采用合适的数据组织方式以减少 I/O 的发生，并且还有可能把随机访问模式优化成顺序访问模式，从而提高 I/O 效率。5.2.4 节将介绍数据组织方式，可以针对不同的场景采用特殊的组织方式，例如，数据的顺序存放、数据的聚簇存放和列存储方

式等。

(3) 有针对性地预读或使用系统提供的异步 I/O 能力,把等待 I/O 的时间并行执行其他任务,从而提高系统的整体性能。

5.2　数据组织

数据库是长期存储在计算机内有组织、可共享的大量数据的集合。如何将这样一个庞大的数据集合以最优的形式组织起来存放在外存上是一个非常重要的问题。“优”应包括两方面:一是存储效率高,节省存储空间;二是读取效率高,速度快,代价小。本节主要介绍数据库的逻辑组织方式和物理组织方式。

5.2.1　数据库的逻辑与物理组织方式

数据库数据存放的基础是文件,对数据库的任何操作最终要转化为对文件的操作。所以在数据库的物理组织中,基本问题是如何利用操作系统提供的基本的文件组织设计数据库数据的存放方法,这实际上也就对应了数据存储管理的两种方式。

第一种方式是每一个数据库对象(例如每一个基本表、索引等)都对应一个或多个操作系统的文件,并独占这些文件。数据文件的空间管理由操作系统支持,需要空间时,扩展数据文件即可;回收空间时,把空闲空间归还给操作系统。这种数据组织方式本质上是将存储管理交由操作系统去完成,例如,PostgreSQL、KingbaseES 等 DBMS 采用此种存储管理方式。

在这种存储管理方式中,数据文件中的数据是以块(页)为单位进行组织的。由于元组的删除或更新,这些元组占用的空间可以再回收利用,不会立即归还给操作系统,因此数据文件内的空闲空间仍由 DBMS 管理。

第二种方式是整个数据库对应一个或若干文件,由 DBMS 进行存储管理,这种存储方式也称为段页式存储管理方式。Oracle、SQL Server 等 DBMS 采用这种存储管理方式。

在段页式存储管理方式中,DBMS 会首先向操作系统申请一个大文件,当数据量不断增加,文件空间不够时,DBMS 会向操作系统申请追加新的文件。为了方便管理,DBMS 通常会对数据库空间进行逻辑划分,以增加灵活性。不同 DBMS 从**逻辑上**划分数据库空间的方式并不完全一样,一种常见的划分方式是将数据库组织成“**表空间—段—分区—数据块**”的形式,如图 5.4 所示。一个表空间(tablespace)对应磁盘上一个或多个物理文件,但一个物理文件只能属于一个表空间。一个数据库可以有多个表空间,从逻辑上组织数据库中的数据存储,例如系统表空间、联机表空间、临时表空间等。一个表空间逻辑上可以由多个段(segment)组成,每个段逻辑上可以组织不同类型的数据,例如数据段、索引段、临时段等。一个段逻辑上可以由多个分区(extent)组成,每个分区由一组连续的数据块(block)组成。块是数据库的磁盘存取单元,其大小为操作系统块的整倍数。

从逻辑上划分数据库空间是为了方便数据的管理。从物理上看,数据库中的数据最终以文件的形式存储在磁盘上。每个文件物理上分成定长的存储单元,即操作系统的物理块。物理块是存储分配和 I/O 处理的基本单位。一个物理块可以存放表中的

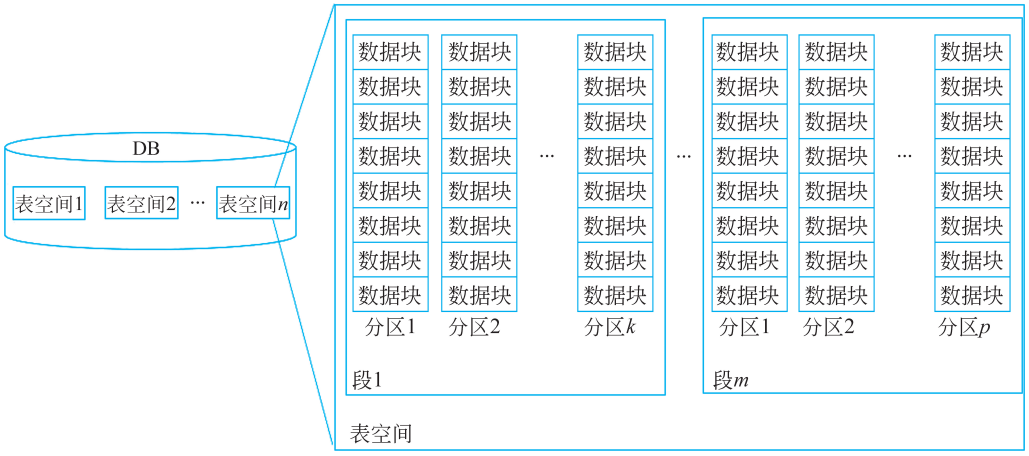


图 5.4 一种数据库的逻辑组织方式

多个元组(记录),一个表通常会占用多个块。因此数据库的物理组织形式是“**文件—块—记录**”。图 5.5 是逻辑组织方式与物理组织方式之间的对应关系,其中的箭头表示了对应关系。接下来,5.2.2 节将具体介绍如何组织单条记录,5.2.3 节将介绍如何在一个块中存放一组记录,5.2.4 节将介绍如何组织一个关系表在块中的存储。

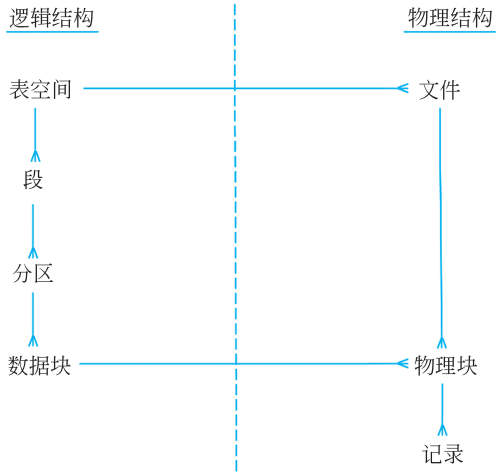


图 5.5 数据库逻辑组织方式与物理组织方式的对应关系

5.2.2 记录表示

关系表的元组可以以定长记录和变长记录两种形式存储。

1. 定长记录

以定长记录形式存储数据,指的是关系表中的每个元组将占据相同大小的空间。即使表中有变长字段,也可以以定长记录形式存储,这时会为变长字段预留最大长度的空间。

例如,对于第 3 章例 3.5 创建的 Products(PID, PName, Price, Category, SID)表,采用定长记录形式,其一条记录的存储形式如图 5.6 所示,这里给 VARCHAR 数据类型的属性分配了最大长度的空间,并假设 DECIMAL 数据类型占用 9 字节。

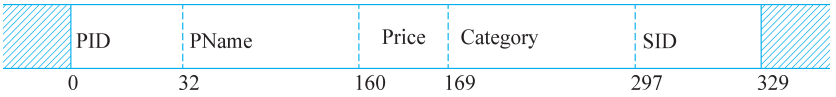


图 5.6　定长记录存储

有些硬件系统对内存中数据的起始地址有要求，比如 4 的倍数或 8 的倍数。如果数据在磁盘上如图 5.6 所示这样紧密存储，在读入内存时就需要做地址转换，为了简化地址转换，并方便系统在不同平台上移植，可以在外存中也保证各字段起始地址是 4 或 8 的倍数，如图 5.7 所示。在图 5.7 中，每个字段都从 8 的倍数的地址起始，虽然 Price 是 9 字节，但实际分配给它 16 字节。

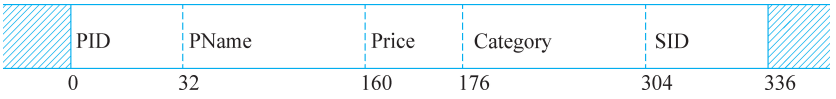


图 5.7　定长记录存储，起始地址为 8 的倍数

以定长方式存储记录的好处是能快速定位到记录及其属性的物理位置，增删改也比较方便和快捷，但显然会浪费一些存储空间。

2. 变长记录

Products 表的 PID 属性、PName 属性和 Category 属性都是变长字符串，所以 Products 表也可以以变长记录形式存放，以节省存储空间。在组织变长记录时，应保证能快速地访问一条记录，并能快速地访问其中的所有属性，包括定长和变长属性。通常有 3 种存放变长记录的方式。

第一种方式是在每条记录的头部记录该条记录的长度，并在该条记录的每个变长字段前记录该字段的长度，如图 5.8 所示。在这个例子中，假设中文字符采用 GBK 编码，每个汉字占 2 字节。

图 5.8　加长度标记的变长记录存储方式

第二种方式是先存放定长字段，后存放变长字段，第一个变长字段紧随定长字段之后，从第二个变长字段开始，在记录首部用指针（偏移量）指向变长字段，如图 5.9 所示。

第三种方式是将定长字段与变长字段分开存储在不同的块中，图 5.10 示意了这种存储方式，但实际上这种方式更多的是用于 BLOB（Binary Large OBject，二进制大对象）等类型数据的存储。这种存储方式的好处是，如果经常访问的是定长字段，则可以减少数据存取时的 I/O 次数。

5.2.3　块的组织

定长记录和变长记录如何在块中组织存储呢？

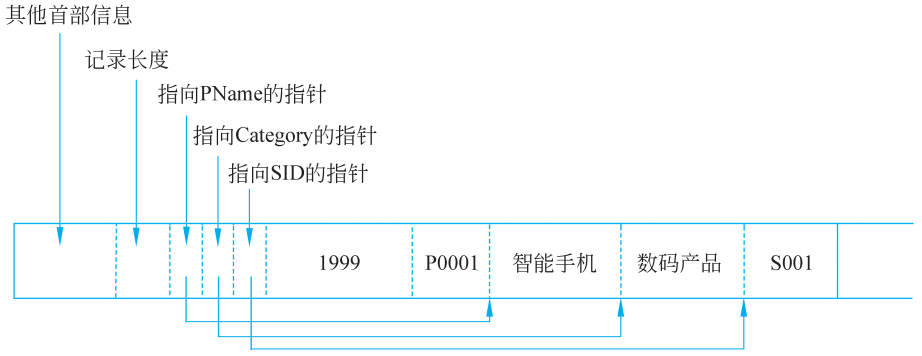


图 5.9 先定长字段后变长字段的变长记录存储方式

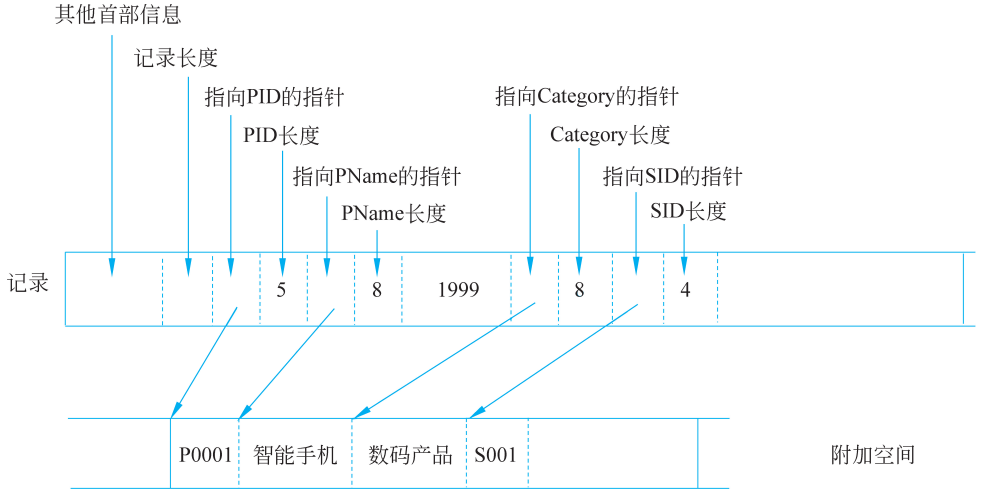


图 5.10 定长字段与变长字段分开存储的变长记录存储方式

1. 定长记录的块组织

定长记录的存储方式在实现上相对比较简单，可以把一个表中的元组依次存放在块中，图 5.11 是 Products 表采用定长存储方式时存放在一个物理块中的示意图。每个块的首部可以有一个块头信息，记录该块的 ID、最后一次修改和访问该块的时间戳、空闲空间的头指针等。

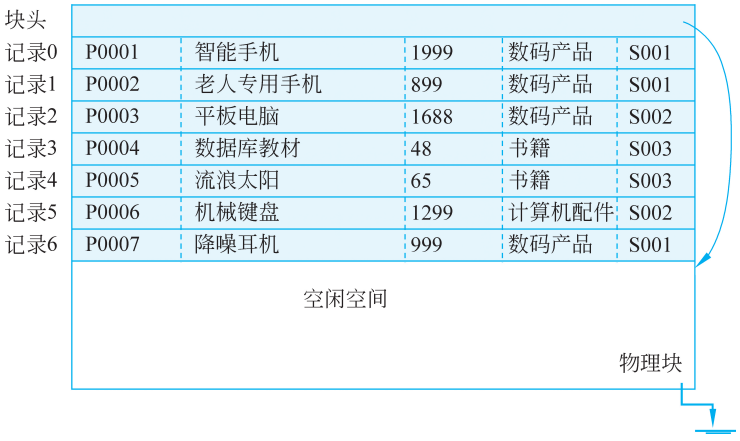


图 5.11 定长记录的块组织

以定长方式存储记录，修改元组时直接在原位置修改即可，不会出现原位置存放不下，需要迁移记录的情形。删除记录时，空间回收也很简单，只需要将空闲空间加入空闲空间链表中即可，不需要移动空闲空间，因为后面插入的新元组，其大小与被删除的元组是一样的，可以直接利用被删除元组的空间。图 5.12 是在图 5.11 所示的 Products 表中删除 P0003 和 P0006 两个元组后的情形。

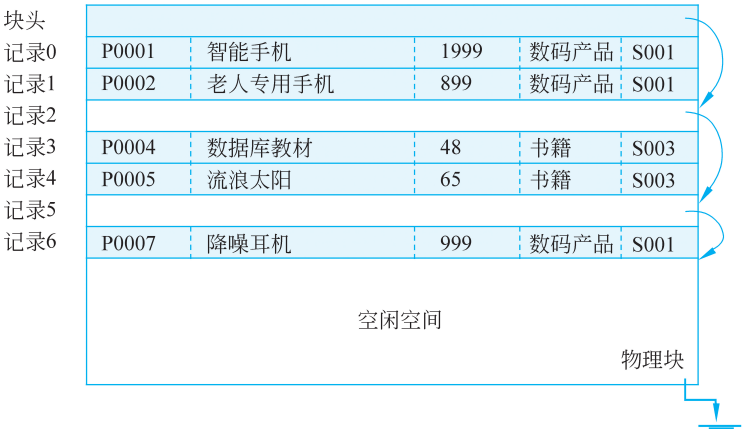


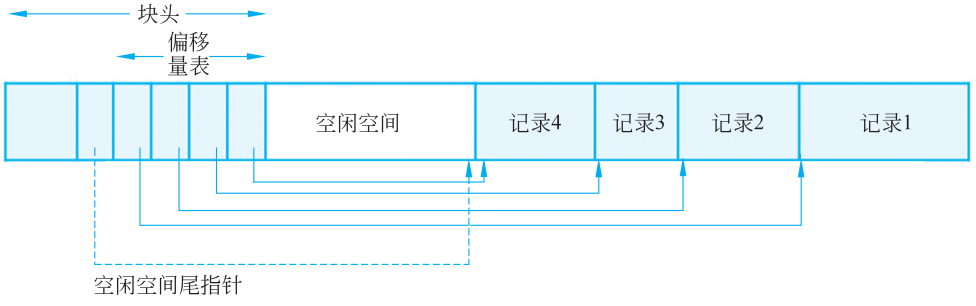
图 5.12　元组删除后定长记录的块组织

2. 变长记录的块组织

对于变长记录，块的组织就要稍复杂一点了，为了能够在块中快速定位到一条变长记录，通常需要在块头存放各条记录的指针（块内偏移量），空闲空间集中存储在块的中间部分，同时在块头有一个空闲空间尾指针（块内偏移量），以便在插入记录时快速找到空闲空间，如图 5.13 所示。

在块中组织变长记录存储时，元组从块的尾部开始连续存放。插入新的元组时，从空闲空间尾部为其分配空间，在偏移量表中记录该元组的起始位置，并调整空闲空间尾指针。

删除一个元组时，会在偏移量表中为该元组指针设置删除标记，释放该元组所占用的空间，并移动物理位置在其前面的元组，以保证空闲空间连续。相应地，被移动元组在偏移量表中的指针以及空闲空间尾指针也需要做相应修改。由于物理块通常都比较小，移动记录的开销并不大。



(a) 变长记录的块组织示意图

图 5.13　变长记录的块组织