目　　录

第二章　存贮结构

2.1　引　言

由上一章数据库系统典型结构图可以看到，数据库的外部模型和概念模型是用户看到的数据的形式，是数据的逻辑结构。而内部模型是数据的逻辑结构在物理存贮介质（如磁盘）上的组织形式，称存贮结构，它是系统看到的数据形式，故存贮结构又称物理数据库组织。

2.1.1　文件的基本概念

（1）什么是数据项、记录、文件：

数据项：是指数据在物理数据库中存贮的最小有名数据单位，也叫　场（*field*）。如姓名、年令等。这些数据可以是字母、数字或字母数字。在程序中描述其类型和位数。如用*COBOL*描述：

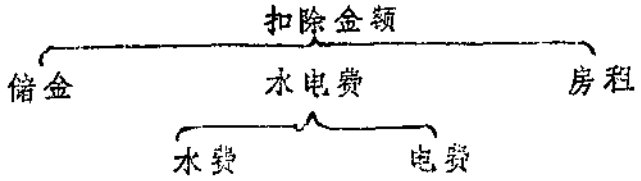
NAME　PICTURE A(20)

AGE PICTURE　9(2)

ADDRESS PICTURE X(20)

*A*表示字母，9表示数字，*X*表示字母数字。

数据集：是记录中数据项的已命名的集合。数据集有两种类型：向量和重复组。向量是一维的数据项的有序集合，且这些数据项具有相同的特征。重复组是在一个记录具体值中出现任意多次的数据的集合。这些数据可以是数据项、向量和重复组。如关于工资的数据中“扣除金额”这个数据项可分解如下：



记录：关于一个实体的数据的总和。记录有类型和具体值之别。一种类型的记录往往有许多记录的具体值组成。

文件：具有相同性质的记录的有名集合。一个数据库往往由若干文件组成。

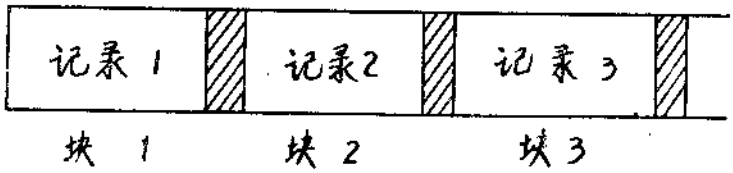
（2）逻辑记录与物理记录

一般说来，一个文件所含有的记录是同格式而且等长的，这种方式处理方便。主存和输入输出设备或外存贮器之间进行交换的数据单位叫做一个块（*BLOCK*），块与物理记录是同义的。因此相应的把以前所说的记录—逻辑上有连系的在存贮器上占有一组邻接单元的数据单位—叫做一个数据记录或逻辑记录，通常简称为记录。

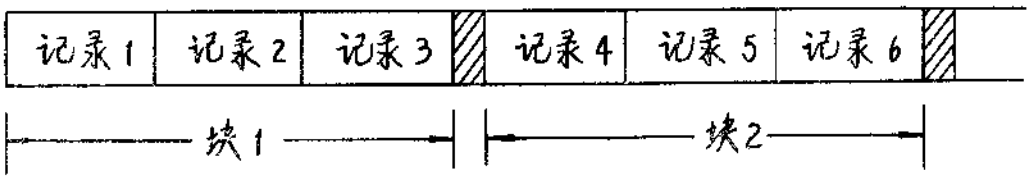
块的大小可以是固定的，如读卡时，如果卡片为80列，那末块的大小为80个字节。磁带和磁盘上的块则是变动的。块与块之间有间隙。

逻辑记录与物理记录之间的对应关系有如下几种方式：

① 一个逻辑记录一个块，称为不组块记录，如：



② 一个块内含有若干个逻辑记录，称为组块记录，如：



③ 一个逻辑记录占有几个块，称为跨块记录。

使用那一种方式，一个块应该多大等受各方面因素—输入输出效率，计算机应用的特征，存贮空间的节约等等的影响。如用组块记录方式，每一块中所含记录越多，则留的间隙相对减少，可节约外存空间，但另一方面，块越大，则内存中占用的缓冲区亦越大。因此，如果机器内存较大可以采用大的块，反之采用小的块。又如若应用系统主要做顺序处理，那末可以采用比较大的块，一次调进较多记录进行处理，反之若应用系统主要作随机处理，则要采用较小的块。块的长度有256、512、1024、2048、4096个字节等数种。

（3）磁盘简介

文件主要存放在外存贮器上，目前数据库系统用来存贮数据的外存主要是磁盘。磁带只能存贮顺序文件；新型的移位寄存器式存贮器件如电荷耦合器CCD(*Charge Coupled Device*)和磁泡存贮器MBM(*magnetic Bobble memory*)仍不成熟，但有生命力。下面简要地介绍磁盘的一些物理性能。

（1）单片磁盘。单片磁盘机构如图2－1所示，电动机带动磁盘高速旋转，读／写臂来回移动，读／写磁头在控制器的控制下读／写磁盘上的数据信息。

磁盘为一厚约几毫米，表面涂有几微米厚磁性材料层的金属圆盘，用以记录数据。由于磁盘是旋转的，所以盘面上的数据信息呈圆周形式排列，形成多个宽为30～100微米的圆环，称为磁盘（相邻两磁道的中心间距约50～120微米），不同型号磁盘面上的磁道数目不等，小容量的为200多个，中容量的为400多个，大容量约有800多个，一般从外向内的顺序予以编号，最外圈的磁道为0磁道。

每个盘面又划分成物理上大小相等的若干个扇形，不同型号磁盘的扇形数目不一，通常有6、8、10、16、18、24、32等几种规格。每个磁道在每一扇形中的部分均称之为块(*BLOCK*)。由于同心圆的内、外圆周长不等，所以块的物理长度也不一样，最外磁盘的块可能比最内磁道的块长二分之一，但各个块记录的数据量却是相等的。

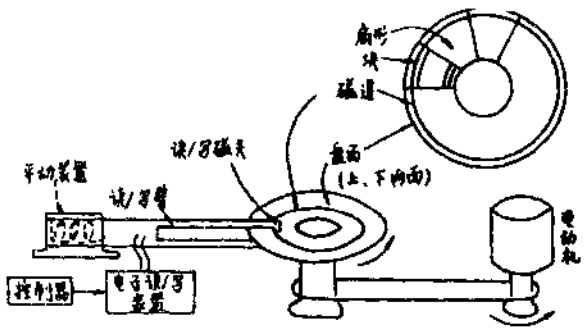


图 2－1　单片磁盘机构

（2）磁盘组。为了加大存贮容量而又不增加机械装置，往往把数个同型号的磁盘置于同一轴上，构成一个磁盘组。如图2－2所示，有11个磁盘装在一根轴上，除顶、底两个盘面不用外，其余的20个盘面记录数据信

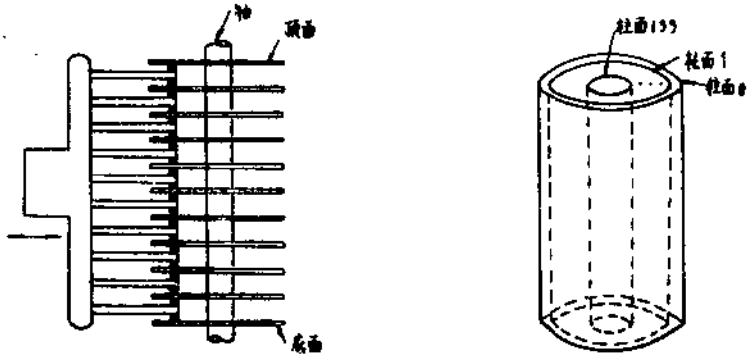


图 2-2　磁盘组与柱面

息。20个读／写磁头装在同一驱动装置上的10个读／写臂上，一起往复作平动。马达则带动轴心使11个磁盘一起高速旋转。

由于磁盘的型号、物理尺寸和磁道的排列位置完全一样，所以离轴心等距离的各个盘面上的磁道（均匀排列的20个圆周）组成一个柱面。故磁盘组的柱面数就是一个盘面的磁道数。一般从外向内编号以标识柱面，从上往下编号以标识不同盘面上的磁道。在同一柱面内任何地方读／写数据，读／写臂的机械位置都是相同的，具有不需寻道时间来依次读／写各磁道上的数据，故数据通常是按柱面而不按盘面存贮。

2．1．2　数据库操作速度的估计

典型的数据库操作有四种：寻找一记录、插入一记录、删除和修改一记录。无论那种操作至少须在磁盘上找到该记录所在地址，由文件系统计算出记录第一个字节所在的绝对地址（所在块加上记录起始位置在块内相对位移的字节数），而磁盘上块的绝对地址通常以磁盘组号、柱面号、磁道号、块号来表示。找到记录所在地址后，磁盘与内存进行交换信息，交换次数越多，越费时间，每次访盘存取时间包括：

（1）选磁盘组，用电子线路实现，可以很快。

（2）选定磁盘组后，要确定数据块所在的柱面，这需移动磁头，机械动作较慢，称“寻找时间”，其数量级约为50毫秒。

（3）确定数据在同一柱面上的那个磁道，用电子线路实现较快。

（4）选定磁道后要确定准确的位置，因磁盘一直在旋转，最好的情

况是正好要选的位置刚转过来，那么立即可读／写，而最坏的情况是要选的位置刚转过去，那么要等待转一圈的时间才能读／写，故平均需转半圈的时间，称旋转延时，一般约 10～20ms。

故一般说来，数据库操作速度主要决定于内存与磁盘进行块交换的次数，块交换所需时间主要决定于找块和块传送的时间。至于*DBMS* 在内存的 操作还是较快的。对于一种物理数据组织好坏的重要标志之一，就是在完成上述四种数据基本操作所需时间，即响应速度如何。

2．1．3 指示器（*pointer*）

在文件系统中，经常用到大量的指示器，例如图 2－3 中的“城市”场，因重复较多，可另开辟一个 *city file*，而在原来 *city* 场的具体值 中放一指向该城市的指示器即可，如图 2－3 所示，指向物理块的指示器可以是该块的绝对地址，而指向一记录的指示器也可以是该记录所在块的绝对地址，这时为了在块内找到该记录，可以利用记录的其它标识，例如按记录的关键字去找等等，这种办法的一个好处是当记录在块内挪动时，不必改变指示器。

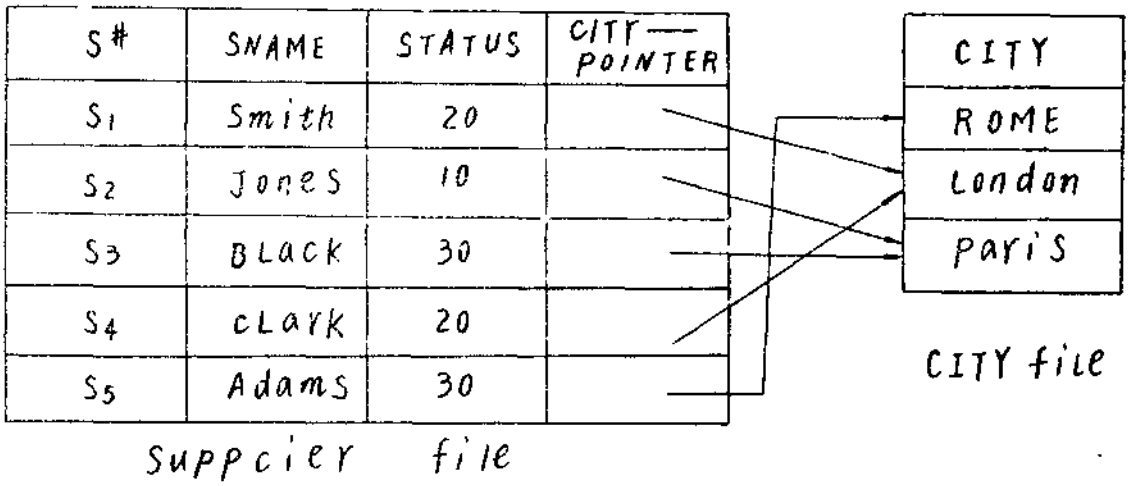


图 2—3 城市数据用指示器代替

2．1．4 关键字（*Keys*）

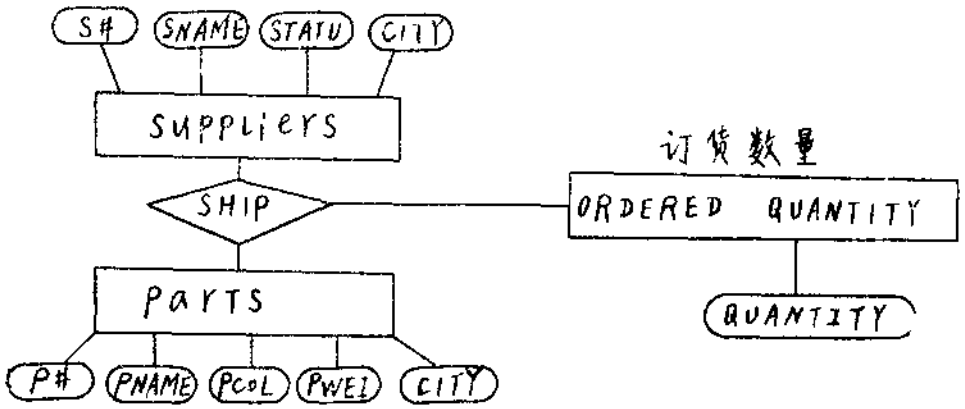
我们可以回忆一下上章所讨论的实体—联系模型，讲到文件可用来存贮两种类型的信息：

（1） 一类文件可表示实体集，这时每个记录代表一个实体，而场就代表实体的属性，属性或若干属性汇集能唯一标识该实体的称键，同样一个场或

若干场的但能唯一标识该记录的叫关键字。关键字可能不止一个，这时我们选择其中一个叫关键字（*Primary Key*），其余的叫辅助关键字（*Condidate Key*）。

（2）另一类文件可表示联系，例如一个两实体集的联系，其每一记录可由一对指示器（P_1，P_2）组成，P_1指向代表第一实体集中一实体的记录，P_2指向与此有联系的第二实体集中一实体的记录，这类文件的关键字往往就是所有场的集合，而没有其它关键字。

如：供应者*S*(*Suppliers*)和零件(*Parts*)分别为两个实体集，而*SP*为二者联系。

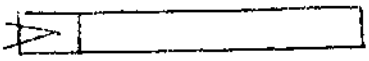


供应者的关键字为$S^{\#}$，零件的关键字为$P^{\#}$，而二者联系的关键字为$S^{\#}P_i^{\#}$，关键字类型有两类：

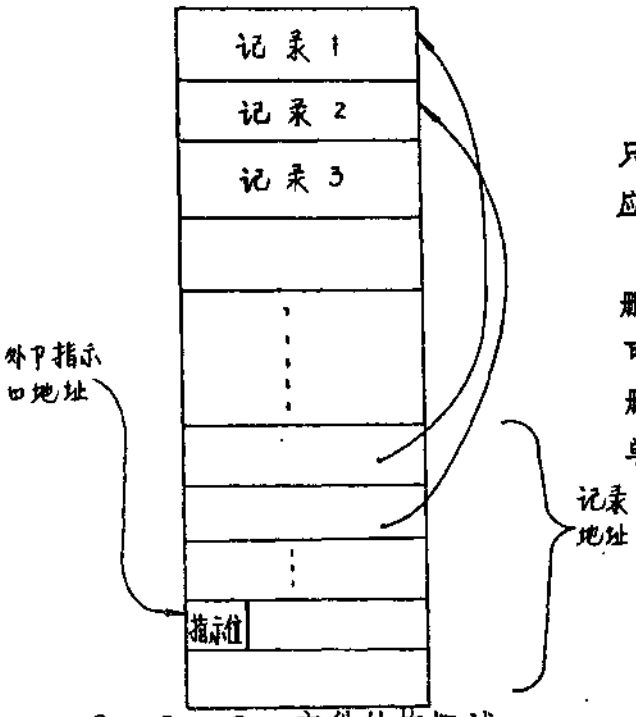
① 由实体集来的，如*S*文件中$S^{\#}$。

② *SP*中的键由原来的联系而变来的，这类文件的关键字往往是各相联系实体其键的集合。

2．1．5 钉定（*Pinned*）和未钉定的记录

在物理库中，记录钉死在某位置上，且被许多指示器所指向，该记录自建立以后，一般不许移动（因为由指示器找记录容易，而修改指示器不易），这样即使某记录删除后其物理空间也不宜再使用，为了防止产生错误，在记录位置设置删除位  ，0示未删，1示已删。这样删除后空间利用很差，介决办法：

1．指示器指向该记录所在块的地址，块内记录再用其它办法安排，即记录可以在块内不钉死。



2。采用间接地址

图中如记录2地址可以单动，只修改记录地址。而外部指示器对应一指定记录。外指示器不变化。

指示位：0未删，1已删。当删除的记录位置需重新存放记录时，可以让很多删除指示都指向用来作删除指示的单元，使这种指示节省单元。

2。1。6 文件结构概述

文件的结构方式一般可分为四种，即顺序结构、随机结构、链结构和树状结构。

（1）顺序结构

顺序文件结构在数据处理历史上是最早使用的。其中的数据记录是根据记录中某一共同的属性来排列，其次序一般是由记录中的某一基本数据项值的大小顺序而排列，这时这一基本项就是关键字。

（2）随机结构

在随机结构中，数据记录存放于直接存取型的存贮设备上。在数据记录的关键字与其地址之间建立了某种关系。随机文件的记录就是按这种关系存放，并利用这种关系进行存取。依连系办法不同又分为：

① 直接地址结构

在某些数据库系统中的“数据库码”(*Database Key*)，就是这种结构。可以直接利用它进行存取。

② 索引结构

这种结构使用一个索引表，其中包括一组关键字和对应的地址。查阅索取表，由记录的关键字可以得出这个记录所在的地址。由此地址则可找到该

记录。当增加或删除记录时，应对索引表及时加以修改。

③ 计算寻址结构

在这种方法中，关键字经过某种计算处理，转换成相应的地址。这种计算式方法就是通常所说的散列（*hash* 也叫杂凑）

（3）表结构（*List*）

表结构的特点是使用了指针（*potnter*）来表达各个记录之间的关联。指针可以表示记录的绝对地址，也可以表示相对于文件中第一个记录的相对地址。

表结构又可分为线性表、倒排表和环形表。

①、线性表（*Linear List*）

按记录之间的相对位置组成的一组记录，称为线性表。线性表按其插入和删除操作的方式，可以有以下三种情况常用到：

（*a*）堆栈（*Stack*）

其中所有的插入和删除只能在其同一端进行，这一端称为“顶端”（*top*），即“后进先出”队（*LIFO—Last in First out*）。

（*b*）队列（*queue*）其插入在“后端”（*back*）进行，而删除在“前端”（*front*）进行，这是一个“先进先出”队（*FIFO—First in first out*）。

（*c*）两用队（*deque*）其插入和删除可在左端进行，也可在右端进行。

②、倒排文件（*inverked file*）

倒排文件由一些倒排表组成。每一个倒排表涉及到记录中某个关键字的一个特定值。同时，有一些指针指向文件中所有具有该关键字的这个值的那些记录。

③、环（*ring*）

一个环就是一个线性表，但其首尾相连。在环中，可以从任意一个单元开始查找，直到查遍整个环。

（4）树状结构（*tree*）

一个树状文件结构相应于一个层次结构。其中的每一层的一个节点，可以存放一个符号，也可以一个目录。因此依其存放的内容，将他们称为符号树或目录树。

下面主要对顺序文件、索引文件、*B*树文件、*Hash* 文件、变长记录、倒排文件进行讨论。

2．2 顺序文件

人们总希望快速访问文件，就要对文件中记录顺序按某种方法进行组织。前曾述及，主关键字是能够唯一标识记录的，那么可以不按记录到达的先后次序，而按主关键字值的顺序组织文件，称为顺序文件。即对每一个记录，按主关键字的值赋予一个序数，序数为 i 的记录，其物理顺序亦为 i，则文件中记录的物理顺序和逻辑顺序是一致的。当找不到合适的数据项作为主关键字，可选取两个以上数据项或给某数据项附加一个人工域组成关键字。

折半查找。当顺序文件全部在内存时，且文件的开始地址为 BA，末尾地址为 EA，记录长度为 L 个字节，则文件中点记录的地址为：

$$MA = \left\lfloor \frac{BA + EA}{2L} \right\rfloor \times L$$

所谓折半查找就是：每次查找文件给定部分的中点记录，根据该记录的主关键字值等于、小于还是大于给定值，来分别决定记录已找到、还是在给定部分的前一半或后一半，而后继续折半查找。上述过程一直进行到找出所需记录，或能确定这样的记录不存在为止。图 2－6 是在 40 个记录的文件中，查找主关键字值为 17 的记录过程。

链结构和块链结构的顺序文件不能使用折半查找的方法，就是在外存的向量结构的顺序文件，由于涉及读块操作和中点记录的地址计算，实现折半查找仍然很耗费时间。

探查（*probing*）。探查是先行一次概略查找后再顺序扫描。有如查词典，先预测被查单词（记录）的大概位置，而后从这个位置向前或向后顺序扫描，直至找到待查的单词（记录）或确定其不存在为止。探查的好坏完全取决于概略查找的准确程度（即预测对真值的偏离程度）。也可以一次概略查找后再接一次概略查找；而后顺序扫描。探查只能在向量结构的文件中使用。

由此可知，向量结构的顺序文件查找比较灵活，但对存贮空间的分配要求很苛刻，实际上不易实现。此外，向量结构亦不适应于频繁插入的文件。链结构虽然在空间分配上比较灵活，但长链难于维持，且查找过程费时。块链结构则兼二者之长，是一种较理想的结构。

对于连续处理的情况，顺序文件结构是一种最经济的结构方式。但如果对它不按顺序进行存取，则处理起来很慢。同时，小量的修改或插入是非常不合算的。这是因为任何对顺序文件的修改或插入，都要求把整个文件重新复制一遍。

一切存于磁带上记录都只能是顺序的，而存于磁盘上的记录，即可以是随机的，也可以是顺序的。顺序文件结构的基本优点就是在连续存取时速度较快，即如果文件中的第 n 个记录刚被存取过，而下一个要存取的是第 $n+1$ 个记录，则这个存取将会很快完成。

2．2．1　如何确定关键字值的顺序

关键字的取值无非是整数、实数、字母、字符，或者它们的组合。整数和实数有数值的顺序；字母有词汇编辑（字典式）的顺序。一般来说，关键字值的顺序可用下面的方式确定：设有字符串 $x_1 x_2 \cdots x_k$，$y_1 y_2 \cdots y_m$，当条件

（1）$k<m$，并且 $x_i = y_i$，$i=1,\cdots,k$

（2）存在某一 $j \leqslant m$（$j \leqslant k, m$），使得：$x_1 = y_1, \cdots x_{j-1} = y_{j-1}$，并且 $x_j < y_j$ 之一成立时，确定 $x_1 \cdots x_k < y_1 \cdots y_m$。例如：

$aaaa<aaaaaa$，$CBED<CBEDA$，$nbedxyz<nd$，$HELPED<I$

当然，机器总是按字符编码的二进位串的大小比较顺序的，所以，对数字、字母编码时，要使编码的二进位串顺序与它们原有意义的顺序一致。

如果关键字系由多个数据项组成，则先按第一个数据项的值排序；若第一个数据项的值相同，再按第二个数据项的值排序，直到完全确定顺序为止。如果对关键字中所有数据项的值能够同时进行比较，则可按整个关键字的值一次排序。这里的顺序表示法是字典式顺序的一般化，即用任意数据项值的顺序表代替字母顺序表。

2．2．2　顺序文件的存储组织

顺序文件的记录逻辑上是按主关键字值的顺序排列的，但在物理存储器中，可用三种不同的办法实现。

（1）向量结构。计算机的存储空间是按绝对地址顺序连续排列的，故存储顺序文件时可按绝对地址顺序连续存放记录。这时文件的物理顺序实现文件的逻辑顺序，其逻辑结构与物理结构一致，这就是向量结构的特征（图 2—4a）。

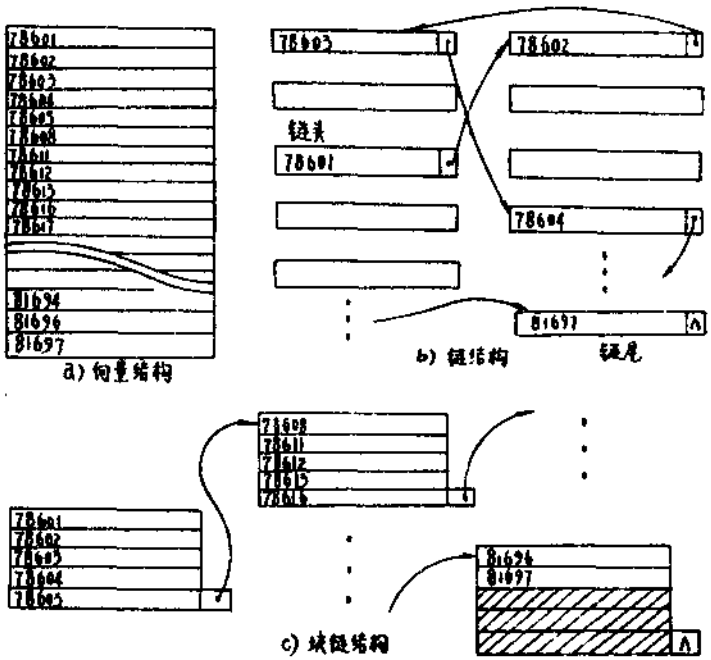


图 2－4　顺序文件的存储结构

（2）链结构。文件的逻辑顺序不用存储空间的绝对地址顺序，而用链（*chain*）实现。一个文件以记录为单位分散在存储空间的不同位置，每一个记录都带有一个指向下一记录的指针。最后一个记录带有链结束标记∧（图 2－4 *b*）。

（3）块链结构。这种结构是上述两种结构的合成。其基本特征是，在一个物理数据块中的记录按地址顺序连续存放，而块与块之间用指针链接，以维持逻辑顺序（图 2－4 *c*）。

2．2．3　顺序文件的查找

由于文件的物理结构不同，查找办法也不同。现对记录均属同一记录类型的向量结构的顺序文件给出四种按主关键字查找记录的方法：

（1）顺序扫描。从文件的第一个记录开始，按记录的顺序依次往下查，直至找到匹配主关键字给定值的记录为止。故查找一个记录平均要读半个文件，速度太慢。链结构与块链结构也可以顺序扫描查找，不同的是要加进读取指针的操作，以便决定下一记录或一下块的地址。

（2）分块查找。也称跳跃查找，即每查一个记录后，跳过若干个记 录

再查。或者说，把文件分成若干块，每次查一块中的最后一个记录，并判断所要查找的记录是否在本块中，在则顺序查找该块的记录；不在则跳到下一块再重复上述步骤，直至找到所需记录或走到文件末端为止。

例如，有一个学生顺序文件，其主关键字为学号，现要查找一个学号为78625的学生记录，若每5个记录分为一块，查找过程如图2－5所示。

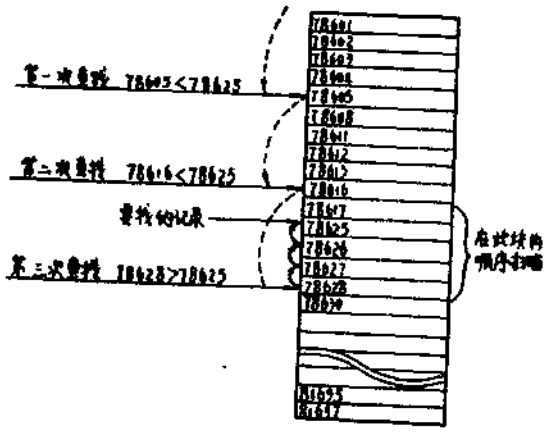


图 2—5 分块查找

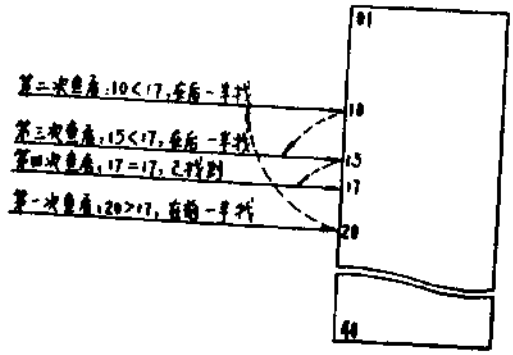


图 2－6 折半查找

2．3 随机结构之一 —散列方法（hashing）

为了克服顺序文件存取的困难，又由于出现可直接访问的大容量的磁盘，使得随机结构的文件组织成为可能。用以标识记录的标识符（一般称之为关键字）与记录地址之间存在一种直接关系，即对关键字变换成记录地址，按此地址到磁盘上存贮或检索记录，这种方法叫散列法。

2．3．1 散列方法的简要回顾

把关键字作为记录存取地址的存取方法是一种最直接了当的散列方法。但关键字的数值范围都大大超过所使用的实际地址范围。如果仍使用此法，就得扩大地址范围，于是所存贮的记录势必以极为松散的状态散布在存贮内。从而造成存贮空间的极大浪费。

（1）散列函数

克服直接散列法的缺点，往往用一种变换，通过变换算法能把较大范围的关键字集合映射到某一特定地址范围。若一关键字的值为K，该记录的存贮地址为A，用一个函数$h(k)$与K联系起来，即$A=h(K)$，其中$h(K)$称为散列函数。例如：学生记录的学号（关键字）按系别班级编排，学 号