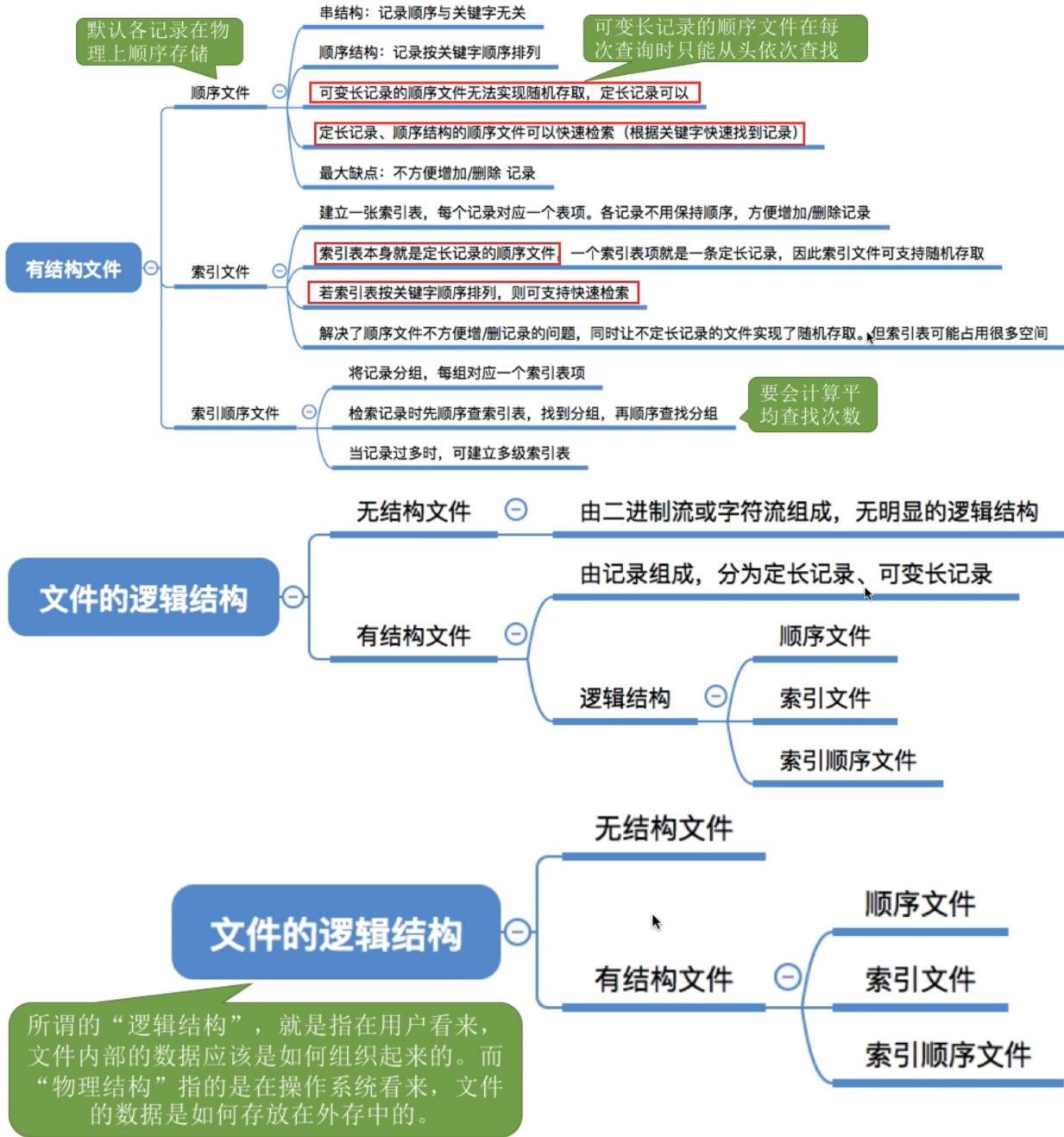


文件的逻辑结构



所谓的逻辑结构，就是在用户看来，文件内部的数据应该是如何组织起来的。而物理结构指的是在操作系统看来，文件的数据是如何存放在外存中的。

类似于数据结构的逻辑结构和物理结构

如线性表就是一种逻辑结构，在用户角度来看，线性表就是一组有先后关系的元素序列，如：a, b, c, d, e...

线性表这种逻辑结构可以用不同的物理结构实现，如：顺序表/链表。顺序表的各个元素在逻辑上相邻，在物理上也相邻；而链表的各个元素在物理上可以是不相邻的。因此，顺序表可以实现“随机访问”，而“链表”无法实现随机访问。

可见，算法的具体实现与逻辑结构、物理结构都有关（文件也一样，文件操作的具体实现与文件的逻辑结构、物理结构都有关）

有结构文件

按文件是否有结构分类，可以分为无结构文件、有结构文件两种。

无结构文件：文件内部的数据就是一系列二进制流或字符流组成。又称“流式文件”。如：

Windows操作系统中的.txt文件。

有结构文件：由一组相似的记录组成，又称“记录式文件”。每条记录有若干个数据项组成。如：数据库表文件。一般来说，每条记录有一个数据项可作为关键字（作为识别不同记录的ID）。根据各记录的长度（占用的存储空间）是否相等，又可分为定长记录和可变长记录两种。

学号	姓名	性别	专业
1120112100	张三	男	挖掘机
1120112101	李四	女	挖掘机
1120112102	王五	男	数据挖掘
1120112103	赵六	男	挖掘机
1120112104	钱七	女	挖掘机
1120112105	狗剩	男	数据挖掘
1120112106	铁柱	女	数据挖掘
1120112107	如花	女	数据挖掘
1120112108	二狗	男	数据挖掘
1120112109	傻根儿	男	数据挖掘
1120112110	旺财	女	数据挖掘

在本例中，“学号”即可作为各个记录的关键字

这是一张数据库表，记录了各个学生的信息

每个学生对应一条记录，每条记录由若干个数据项组成

学号	姓名	性别	专业
1120112100	张三	男	挖掘机
1120112101	李四	女	挖掘机
1120112102	王五	男	数据挖掘
1120112103	赵六	男	挖掘机
1120112104	钱七	女	挖掘机
1120112105	狗剩	男	数据挖掘
1120112106	铁柱	女	数据挖掘
1120112107	如花	女	数据挖掘
1120112108	二狗	男	数据挖掘
1120112109	傻根儿	男	数据挖掘
1120112110	旺财	女	数据挖掘

32 B 学号 32 B 姓名 4 B 性别 60 B 专业

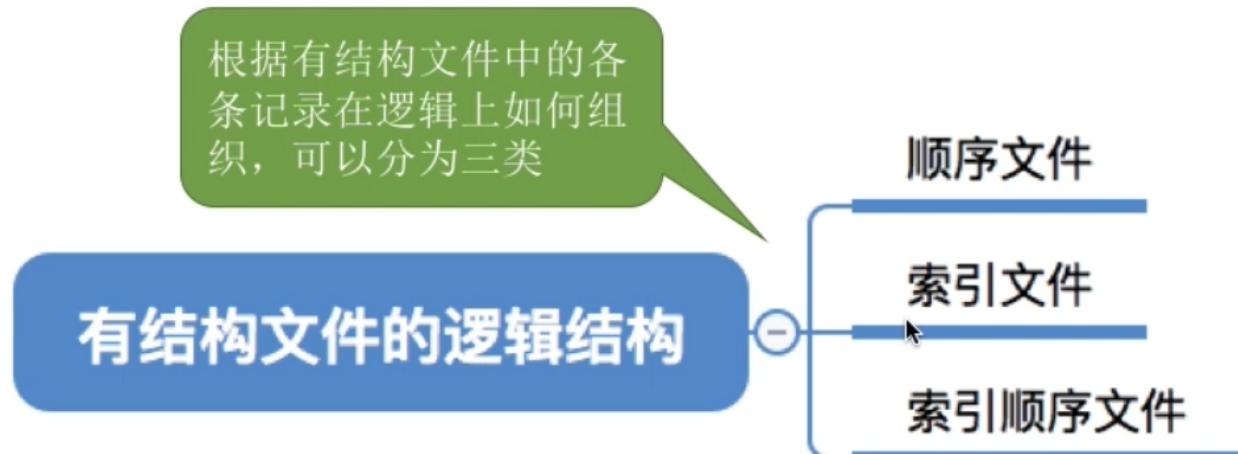
这个有结构文件由定长记录组成，每条记录的长度都相同（共128B）。各数据项都处在记录中相同的位置，具有相同的顺序和长度（前32B一定是学号，之后32B一定是姓名……）

学号	姓名	性别	特长
1120112100	张三	男	腿特长
1120112101	李四	女	腿毛特长
1120112102	王五	男	
1120112103	赵六	男	
1120112104	钱七	女	
1120112105	狗剩	男	
1120112106	铁柱	女	
1120112107	如花	女	
1120112108	二狗	男	熟读唐诗三百首，琴棋书画样样精通，上得了厅堂下得了厨房，精通Java、C++、Python和任意一种脚本语言……(后面还有1万字……)
1120112109	傻根儿	男	
1120112110	旺财	女	

32 B 学号 32 B 姓名 4 B 性别 (长度不确定) 特长

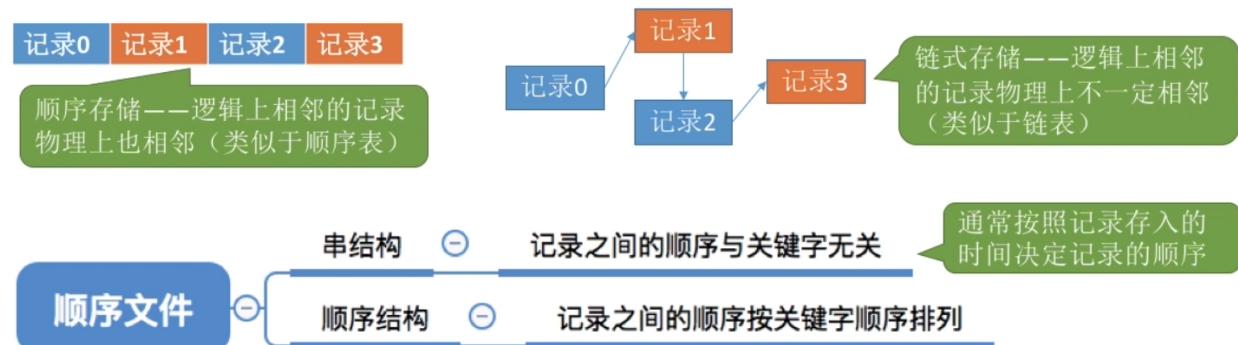
这个有结构文件由可变长记录组成，由于各个学生的特长存在很大区别，因此“特长”这个数据项的长度不确定，这就导致了各条记录的长度也不确定。当然，没有特长的学生甚至可以去掉“特长”数据项。

这个有结构文件由可变长记录组成，由于各个学生的特长存在很大区别，因此特长这个数据项的长度不确定，这就导致了各条记录的长度也不确定。当然，没有特长的学生甚至可以去掉特长数据项。



顺序文件

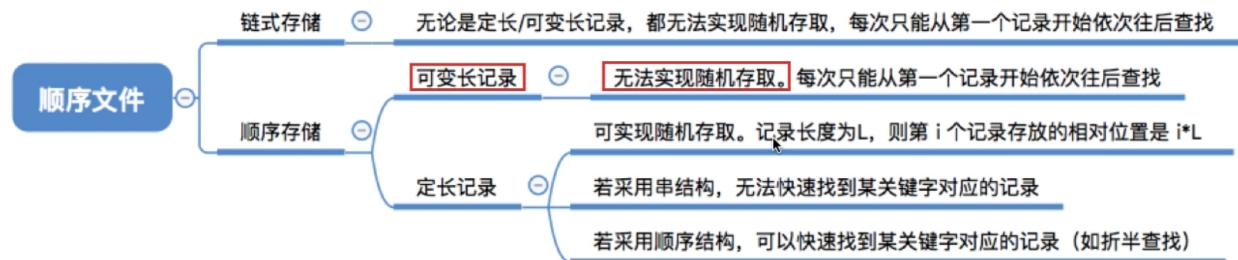
文件中的记录一个接一个地顺序排列（逻辑上），记录可以是定长的或可变长的。各个记录在物理上可以顺序存储或链式存储。



假设：已经知道了文件的起始地址（也就是第一个记录存放的位置）

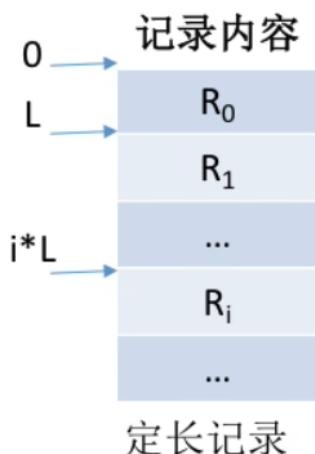
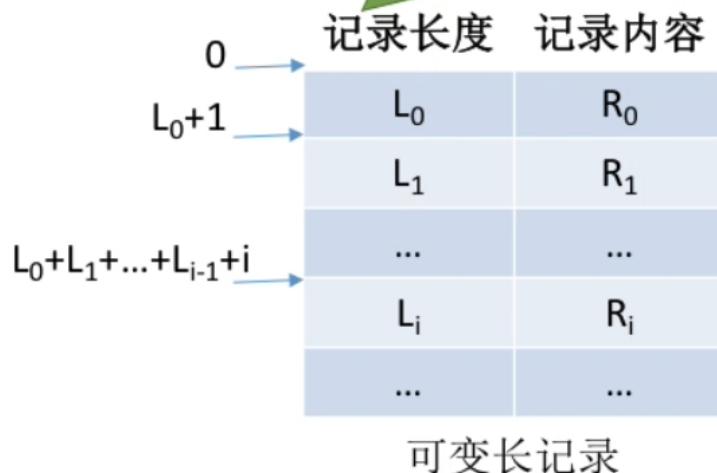
思考1：能否快速找到第*i*个记录对应的地址？（即能否实现随机存取）

思考2：能否快速找到某个关键字对应的记录存放的位置





需要显式地给出记录长度，假设用1字节表示记录长度

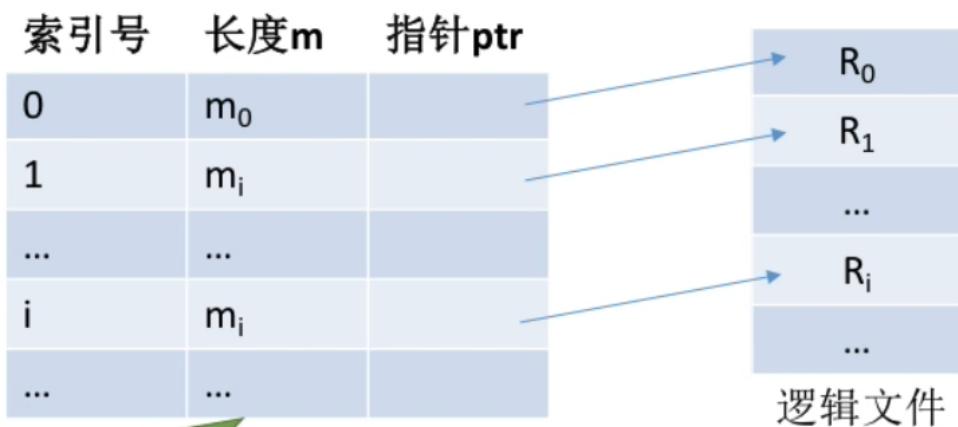


结论：定长记录的顺序文件，若物理上采用顺序存储，则可实现随机存取；若能再保证记录的顺序结构，则可实现快速检索（即根据关键字快速找到对应记录）

注：一般来说，考试题目中所说的顺序文件指的是物理上连续存储的顺序文件。之后的讲解中提到的顺序文件也默认如此。可见，顺序文件的缺点是增加/删除一个记录比较困难（如果是串结构则相对简单）

索引文件

对于可变长记录文件，要找到第 i 个记录，必须先顺序地查找前 $i-1$ 个记录，但是很多应用场景中又必须使用可变长记录。如何解决这个问题？



建立一张索引表以加快文件检索速度。每条记录对应一个索引项。

文件中的这些记录在物理上可以离散地存放。

索引表本身是定长记录的顺序文件。因此可以快速找到第i个记录对应的索引项。

可将关键字作为索引号内容，若按关键字顺序排序，则还可以支持按照关键字折半查找。

每当要增加/删除一个记录时，需要对索引表进行修改。由于索引文件有很快的检索速度，因此主要用于对信息处理的及时性要求比较高的场合。

另外，可以用不同的数据项建立多个索引表。如：学生信息表中，可用关键字学号建立一张索引表。也可用姓名建立一张索引表。这样就可以根据姓名快速地检索文件了。（SQL就支持根据某个数据项建立索引的功能）

索引顺序文件

思考索引文件的缺点：每个记录对应一个索引表项，因此索引表可能会很大。比如：文件的每个记录平均只占8B，而每个索引表项占32个字节，那么索引表都要比文件内容本身大4倍，这样对存储空间的利用率就太低了。

索引顺序文件是索引文件和顺序文件思想的结合。索引顺序文件中，同样会为文件建立一张索引表，但不同的是：并不是每个记录对应一个索引表项，而是一组记录对应一个索引表项。



索引顺序文件的索引项也不需要按关键字顺序排列，这样可以极大地方便新表项的插入

索引顺序文件是索引文件和顺序文件思想的结合。索引顺序文件中，同样会为文件建立一张索引表，但不同的是：并不是每个记录对应一个索引表项，而是一组记录对应一个索引表项。

在本例中，学生记录按照学生姓名的开头字母进行分组。每个分组就是一个顺序文件，分组内的记录不需要按关键字排序



用这种策略确实可以让索引表“瘦身”，但是是否会出现不定长记录的顺序文件检索速度慢的问题呢？

检索效率分析



若一个顺序文件有10000个记录，则根据关键字检索文件，只能从头开始顺序查找（这里指的并不是定长记录、顺序结构的顺序文件），**平均须查找5000个记录**。

若采用索引顺序文件结构，可把10000个记录分为 $\sqrt{10000} = 100$ 组，每组100个记录。则需要先顺序查找索引表找到分组（共100个分组，因此索引表长度为100，平均需要查50次），找到分组后，再在分组中顺序查找记录（每个分组100个记录，因此平均需要查50次）。可见，采用索引顺序文件结构后，**平均查找次数减少为50+50=100次**。

同理，若文件共有 10^6 个记录，则可分为1000个分组，每个分组1000个记录。根据关键字检索一个记录平均需要查找 $500+500=1000$ 次。这个**查找次数依然很多**，如何解决呢？

多级索引顺序文件

为了进一步提高检索效率，可以为顺序文件建立多级索引表。例如，对于一个含 10^6 个记录的文件，可先为该文件建立一张低级索引表，每100个记录为一组，故低级索引表中共有10000个表项（即10000个定长记录），再把这10000个定长记录分组，每组100个，为其建立顶级索引表，故顶级索引表中共有100个表项。

