

## 10 存储和文件结构

Author: 中山大学 17数据科学与计算机学院 YSY

<https://github.com/ysyisyourbrother>

### 10.1 物理存储介质概述

- 高速缓冲存储器 (cache) : 贵
- 主存储器 (main memory): 断电容易丢
- 快闪存储器 (flash memory): u盘 非易失性
- 磁盘存储器 (磁盘): 长期存储数据
- 光学存储器 (光盘)
- 磁带存储器

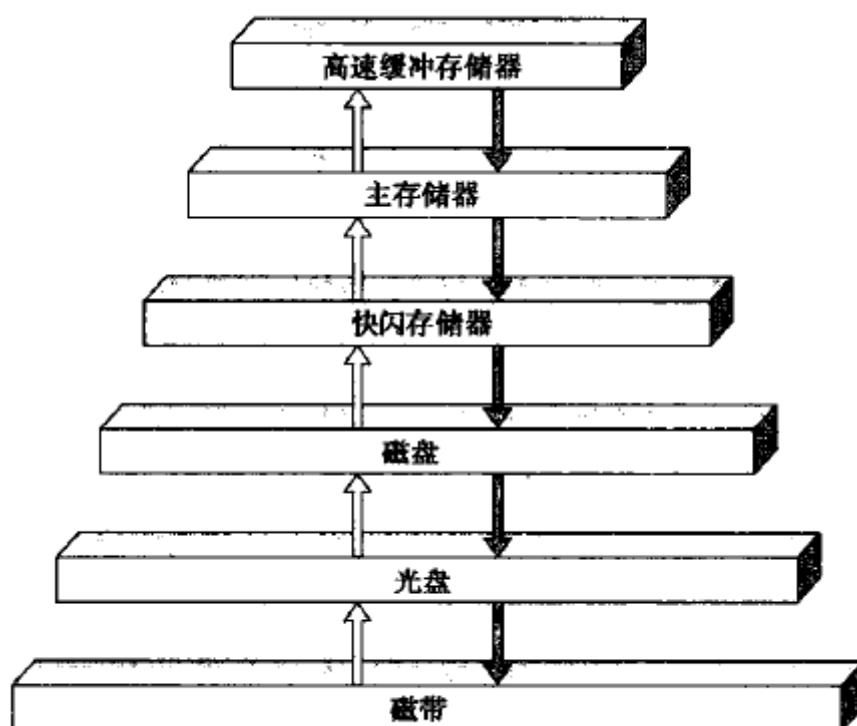
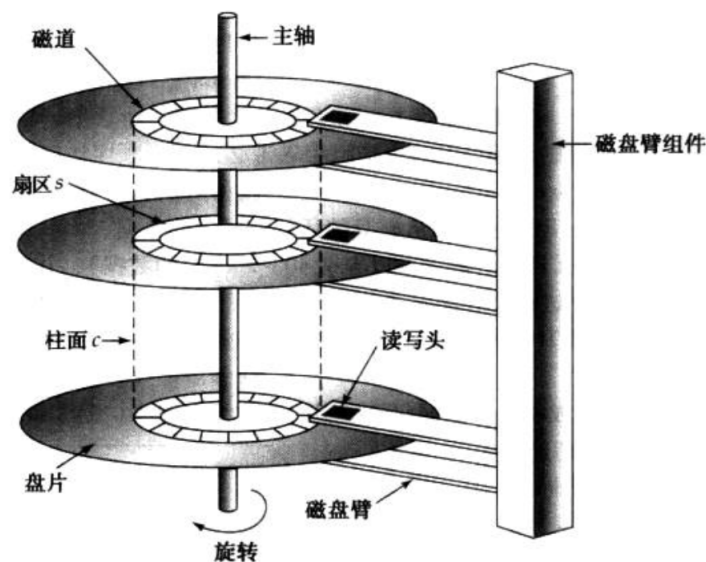


图 10-1 存储设备层次结构

### 10.2 磁盘和快闪存储器



### 10.2.2 磁盘性能的度量

- 访问时间：发出**读写请求**到**数据开始传输**之间的时间，包括：
  1. 磁盘臂必须移动定位到正确的磁道（沿着半径）
  2. 等待磁盘旋转，知道指定扇区出现在下方
  3. 磁盘臂定位时间称为**寻道时间**
  4. **旋转等待时间**：等待访问的扇区出现在读写头下所花费的时间
- 数据传输率：磁盘获取数据或者向磁盘存储数据的速率
- 平均故障时间：磁盘可靠性的度量标准

### 10.2.3 磁盘块访问的优化

一个**块**是一个逻辑单元，它包含**固定数目的连续扇区**

## 10.3 RAID

为提高性能和可靠性，**独立磁盘冗余阵列(RAID)**技术被引入

### 10.3.1 通过冗余提高可靠性

引入**冗余**可以提高可靠性，即存储正常情况下不需要的额外信息

**镜像**：复制每一张磁盘，这样一张逻辑磁盘由两张物理磁盘组成。这种方法很昂贵，而且每一个数据修改的时候要在两边同时修改。

### 10.3.2 通过并行提高性能

通过在多张磁盘上进行**数据拆分**来提高传输速率

**比特级拆分**：将数据每个字节按比特拆开，存储到多个磁盘上

**块级拆分**：整个磁盘阵列看作一个大磁盘。当读一个大文件时，可以并行的从n个磁盘上读取n个块。

### 10.3.3 RAID级别

- RAID0 块级拆分但没有任何冗余
- RAID1 块级拆分的磁盘镜像
- RAID5 块交叉的分布奇偶校验位的组织结构



图 10-3 RAID 级别

## 10.5 文件组织

一个**文件**在逻辑上组织成为记录的一个序列

每个文件分成定长的存储单元，称为**块**

### 文件组织

**文件**：一个文件在逻辑上组织称记录的一个序列，这些记录映射到磁盘块上

**块**：每个文件分成定长的储存单元。块是储存分配和数据传输的基本单元

**文件头**：在文件的开始处，分配一定数量的字节作为文件头，包含文件各种信息

**空闲列表**：被删除的记录形成的列表

### 10.5.1 定长记录

#### 定长记录

假如我们为每个属性分配可以容纳的最大字节数，假如一个记录最大可能占有的空间为53个字节。一个简单的方法是使用53个字节存放每个记录。这种方法存在问题：

1. 除非块的大小恰好是53的倍数，否则记录可能会跨两个块

2. 从这个结构中删除很困难，需要别的记录填充，或者标记为删除。

解决办法：

对第一个问题：只保存能完整容纳的最大记录数

对第二个问题：当一条被删除后，把所有记录进行移动。或者在文件头中存储内容被删除的第一个记录的地址，第一个记录再指向下一个，形成一个链表。

在文件开始处，分配一定数量的字节作为**文件头**，包含有关文件的各种信息

被删除的记录形成链表，被称为**空闲链表**

头文件				
记录0	10101	Srinivasan	Comp. Sci.	65000
记录1				
记录2	15151	Mozart	Music	40000
记录3	22222	Einstein	Physics	95000
记录4				
记录5	33456	Gold	Physics	87000
记录6				
记录7	58583	Califeri	History	62000
记录8	76543	Singh	Finance	80000
记录9	76766	Crick	Biology	72000
记录10	83821	Brandt	Comp. Sci.	92000
记录11	98345	Kim	Elec. Eng.	80000

图 10-7 删除了第 1、4 和 6 条记录的图 10-4 中的文件

## 10.5.2 变长记录

- 多种记录类型在一个文件中存储
- 允许一个或多个字段是变长的记录类型
- 允许可重复字段的记录类型，例如数组或多重集合

存在的技术问题：

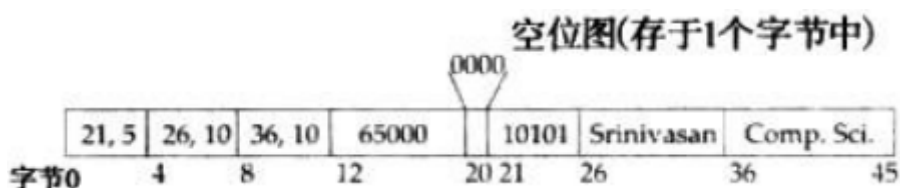
- 如何描述一条记录，使得块中记录可以被拿出
- 在块中如何储存变长的记录，使得块中的记录可以轻松抽取

变长记录包括两个部分：

- 初始部分是定长属性，分配所需字节数，如数字，日期
- 后面是变长属性，在记录初始部分表示为一个对（偏移量，长度）值，偏移量表示在记录中该属性开始的位置，长度表示变长属性的字节长度。

无论是定长还是变长属性，记录初始部分存储有关每个属性的固定长度的信息。

图 10-8 描述这样一个记录表示的例子。该图显示了一个 *instructor* 记录，它的前三个属性 *ID*、*name* 和 *dept\_name* 是变长字符串，其第 4 个属性 *salary* 是一个大小固定的数值。我们假设偏移量和长度值存储在两个字节中，即每个属性占 4 个字节。*salary* 属性假设用 8 个字节存储，并且每个字符串占用和其拥有字符数一样多的字节数。



**空位图：**用来表示哪个属性是空值。

如果salary是空值，则该位图的第四位将置1，存储在12到19字节的salary值将被忽略。

**分槽的页结构：**用于在块中组织变长的记录。每个块开始处有一个块头，包含以下信息：

1. 块头中记录条目的个数
2. 块中空闲空间的末尾处
3. 一个由包含记录位置和大小记录组成的数组



图 10-9 分槽的页结构

**插入记录：**实际记录从块尾部开始连续排列。块中空闲空间是连续的，在块头数组的最后一个条目和第一条记录之间。如果插入一条记录，在**空闲空间的尾部**给记录分配空间，并将包含这条记录大小和位置的条目添加到块头中。

**删除记录：**所占用空间被释放，它的条目被设置为删除状态(如大小设置为-1)。并且被删除记录之前的记录被移动(移动代价不会很大，因为块大小有限制)，将空闲空间连在一起，块头的空闲空间末尾指针也要修改。

分槽的页结构**要求没有指针直接指向记录。**

大多数关系数据库限制记录不大于一个块的大小以简化缓冲区管理和空闲空间管理，大对象常常储存到一个特殊文件中而不是与短记录储存在一起。

## 10.6 文件中记录的组织

给定一个记录的集合之后，下一个问题就是如何在文件中组织它们。

1. **堆文件组织：**一条记录可以放在文件中任何地方。记录没有顺序，通常每个关系使用一个单独的文件
2. **顺序文件组织：**记录根据其搜索码的值顺序存储。
3. **散列文件组织：**在每条记录的某些属性上计算一个散列函数。散列函数确定记录应该放在文件的哪个块中。

通常，每个关系的记录用一个单独的文件存储，但是在多表聚簇文件组织中，几个不同关系的记录存储在同一个文件中。


### 顺序文件组织

顺序文件是为了高效处理某个搜索码的顺序排序的记录而设计的。。

**搜索码**是任何一个属性或属性的集合，它不一定是主码。

为了快速按照搜索码的顺序获取记录，我们可以用**指针**把记录链接起来，为了减少顺序文件处理的块访问数，我们在**物理上**按搜索码顺序或者尽可能地接近搜索码顺序存储记录。

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	



在插入和删除记录后维持物理顺序移动代价很高。可以使用指针链表来管理删除。

对于插入操作：

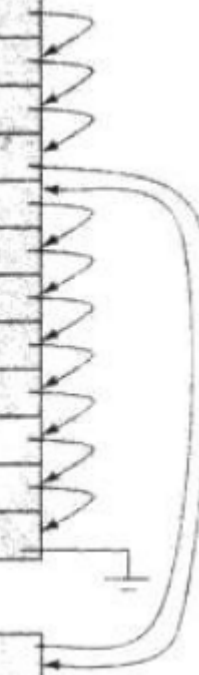
1. 在文件中按搜索码顺序定位
2. 如果当前块又空闲记录，就插入，否则插入到一个溢出块中，然后调整指针。

插入在溢出块会导致访问中可能经常跳到别的块去，导致访问速度变慢。当搜索码顺序和物理顺序之间一致性完全丧失，此时文件需要重组，代价很高。在插入很少的情况下，物理顺序比较容易保持

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

32222	Verdi	Music	48000	
-------	-------	-------	-------	--



很多大型操作系统在文件管理方面并不直接依赖于下层的操作系统，而是让操作系统分配数据库一个大的操作系统文件，数据库系统把所有关系存储在这个文件中，并自己管理这个文件。

多数数据库在一个给定块中只存储一个关系的记录。这可以简化数据管理，但是在有些情况下，在一个块中存储多个关系的记录会很有用：

```
select dept_name, building, budget, ID, name, salary
from department natural join instructor
```

在这个查询计算department关系和instructor关系链接。对department的每个元组，系统必须找到具有相同dept\_name的instructor元组。

dept_name	building	budget
Comp. Sci.	Taylor	100000
Physics	Watson	70000

图 10-12 department 关系

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

图 10-13 instructor 关系

下面是一个高效执行设计department自然连接instructor的查询而设计的文件结构。每个系的instructor元组存储在具有相同dept\_name的deparment元组附近。这种结构将两个关系的元组混合在一起，而允许对连接的高效处理。

Comp. Sci.	Taylor	100000
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

图 10-14 多表聚簇文件结构

这种处理会导致其他类型的查询变慢：

```
select *
from department
```

这个查询需要访问更多的块，因为每个块包含更少的department记录。

为了找到所有department关系的元组，我们可以用指针：

Comp. Sci.	Taylor	100000	
45564	Katz	75000	
10101	Srinivasan	65000	
83821	Brandt	92000	
Physics	Watson	70000	
33456	Gold	87000	

图 10-15 带指针链的多表聚簇文件结构

10.7 数据字典存储

元数据：关于数据的数据（如关系的模式）

元数据存储在称为数据字典或系统目录中的结构中

系统必须存储的信息类型有：

- 关系的名字。
- 每个关系中属性的名字。
- 属性的域和长度。
- 在数据库上定义的视图的名字和这些视图的定义。
- 完整性约束(例如, 码约束)。

此外, 很多系统为系统的用户保存了下列数据:

- 授权用户的名字。
- 关于用户的授权和账户信息。
- 用于认证用户的密码或其他信息。

## 10.9 总结

见P264