Study Notes for DATABASE SYSTEMS The Complete Book - Chapter 13 (Sections 13.1-13.3)

Rui LIN

22 August, 2023

1 Byte、16 进制、JPEG、编码

1. **“文件大小”与“所占空间”的差别?**
   “文件大小”是文件内容实际具有的字节数，它以 Byte 为衡量单位，只要文件内容和格式不发生变化，文件大小就不会放生变化。文件在磁盘上的所占空间不是以 Byte 为衡量单位的，它最小的计量单位是簇(cluster)。

   文件系统是操作系统与驱动器之间的接口，当操作系统请求从硬盘里读取一个文件时，会请求相应的文件系统（FAT 16/32/NTFS）打开文件。扇区是磁盘最小的物理储存单元，但由于操作系统无法对数目众多的扇区进行寻址，所以操作系统就将相邻的扇区组合在一起，形成一个簇，然后再对簇进行管理。每个簇可以包括 2、4、8、16、32 或 64 个扇区。显然，簇是操作系统所使用的逻辑概念，而非磁盘的物理特性。

   为了更好地管理磁盘空间和更高效地从硬盘读取数据，操作系统规定一个簇中只能放置一个文件的内容，因此文件所占用的空间，只能是簇的整数倍；而如果文件实际大小小于一簇，它也要占一簇的空间。所以，一般情况下文件所占空间要略大于文件的实际大小，只有在少数情况下，即文件的实际大小恰好是簇的整数倍时，文件的实际大小才会与所占空间完全一致。

2. **JPEG 算法**

JPEG 是 Joint Photographic EXPerts Group(联合图像专家组) 的缩写，文件后辍名为”. jpg”或”. jpeg”，是最常用的图像文件格式，由一个软件开发联合会组织制定，是一种有损压缩格式，能够将图像压缩在很小的储存空间，图像中重复或不重要的资料会被丢失，因此容易造成图像数据的损伤。[blog]

不同颜色模型有不同的应用场景。RGB 模型适合于像显示器这样的自发光图案，而在印刷行业，使用油墨打印，图案的颜色是通过反射光线产生的，通常使用 CMYK 模型。在 JPEG 算法中，需要把图案转换成 YCbCr 模型，这里的 Y 表示亮度（Luminance），Cb 和 Cr 分别表示绿色和红色的色差值，色差信号可以用来传输颜色信息。

在实际的 JPEG 压缩过程中，由于图像本身的连贯性，一个 8x8 的图像中的数值一般不会出现大的跳跃，经过 DCT 转换会有类似的效果，左上角的直流分量保存了一个大的数值，其他分量都接近于 0。

JPEG 算法提供两张标注你的量化系数矩阵，分别用于处理亮度数据 Y 和色差数据 Cr 以及 Cb。量化之后，一大部分数据变成了 0，非常有利于后面的压缩存储。DCT 系数矩阵中的不同位置的值代表了图像数据中不同频率的分量，JPEG 的两个量化矩阵是人们根据人眼对不同频率的敏感程度的差别所积累下的经验指定的，一般来说人眼对于低频的分量比高频分量更加敏感，所以两张量化系数矩阵左上角的数值明显小于右下角的区域（量化系数矩阵是作为除数的）。JPEG 通过 zigzag 方式读取数据的原因，是尽可能的把 0 放在一起，因为 0 大部分集中在右下角，所以才取这种由左上角到右下角的顺序，经过这种顺序变换，最终矩阵变成一个整数数组。（频率是 u，v 到原点的距离，高频变化剧烈，对应图像的细节；低频变化平稳，对应图像的轮廓）

Huffman coding（哈夫曼编码）：[CSDN 博客]

3. **像素与字节**

一个字节包含 8 个 bits，像素即图片元素，通常表示位图中的一个点。
1. 对于黑白图像（即二值图像），一个像素点用一个 bit 表示即可，即用 0 和 1 来表示黑白。这样一个字节可以表示 8 个像素。
2. 对于灰度图像，一个像素要求 256 种状态，即 2 的 8 次方，因此需要 8bit，即 1 个 byte 来表示一个像素。

3. 对于 RGB 图，一个像素占 24 个 bit，即 3 个字节表示一个像素。上面说的都是位图 (bitmap)，所谓位图即是用像素点表示的图

4. **bytes 和 bytearray**

- byte：表示字节序列，是一个不可变的数据类型

- bytearray：表示字节数组，是一个可变的数据类型

- 16 进制：英文字母 A、B、C、D、E、F 分别表示数字 10 15。$0xAF = 16^0 \times 15 + 16^1 \times 10 = 175$, 16 进制数是计算机常用的一种计数方法，它可以弥补二进制数书写位数过长的不足，也用于电视机中。

5. **编码**

- 计算机语言（机器语言）：二进制，0 和 1,1 代表有一个信号，0 代表没有信号。

- ASCII: 可以理解成人的语言翻译成机器的语言的方式。BIN 是 0 和 1 组成的编码，Symbol 是人类可以识别的符号。从 BIN 到 Symbol 就是解码（decode），从右到左就是编码（encode）。

- 多字节编码：单字节编码，如果想匹配多于 256 个字符的语言，一个字节显然不够，用两个字节的话，16 比特，可以编码 65536 个字符，BIG-5 是一种双字节编码方式，它包括大多数中文繁体字，GB18030 包括繁体和简体。

- 统一编码 Unicode：Unicode (Universal Multiple-Octet Coded Character Set) 其实不是一种编码，而是定义了一个表（是一个字符集），表中为世界上每种语言中的每个字符设定了统一并且唯一的码位（code point），以满足跨语言、跨平台进行文本转换的要求。在表示一个 Unicode 的字符时，通常会用“U+”然后紧接着一组十六进制的数字来表示这一个字符。它没有规定如何存储，一个编号为 65 的字符，只需要一个字节就可以存下，但是编号为 40657 的字符需要两个字节的空间才可以装下，而更靠后的字符可能会需要三个甚至四个字节的空间。

- UTF-8: UTF-8 代表 Unicode Transformation Format -8 bits，是 Unicode 的储存方式。UTF-8d 代表每次 8 个位传输数据，而 UTF-16 就是每次 16 个位。UTF-8 是在互联网上使用最广的一种 unicode 的实现方式，就是为了传输而设计的编码。看起来编码规则是人为设定的。[blog][知乎]

- 16 进制：x 和 0x 的区别。前者用在字符串里，即前面有个 b 的时候使用。16 进制在 Python 里会出现 F，J 等字母，此时使用的是 ASCII 值。

6. **三级存储结构**

- 高速缓存存储器：高速缓存存储器是 CPU 内部集成的一种高速存储器，用于暂时存放 CPU 频繁使用的数据和指令。由于高速缓存存储器的访问速度非常快，可以大大提高 CPU 访问数据的速度，从而提高整个系统的性能。

- 主存储器：主存储器是计算机系统中存储器的主要组成部分，用于存储程序和数据。主存储器的访问速度比高速缓存存储器慢，但容量比高速缓存存储器大得多。主存储器一般采用 DRAM（动态随机存储器）实现，容量通常为几个 GB。

- 辅助存储器：辅助存储器是计算机系统中容量最大的存储器，用于长期存储数据和程序。常见的辅助存储器包括硬盘、光盘、U 盘等，它们的容量比主存储器大得多，但访问速度比主存储器慢得多。

2 落盘

2.1 The Memory Hierarchy

1. **虚拟内存 (Virtual Memory)**

- 典型的软件在虚拟内存中运行

- 操作系统对虚拟内存进行管理,使一部分虚拟内存在 main memory,一部分在盘上。Memory 和 Disk 的传输发生在 disk 的分区（blocks/pages）里

- 虚拟内存是操作系统的伪影，它使用了机器的硬件，不能算是存储结构中的一层

2. **摩尔定律（Moore's law）**

- 摩尔定律是由英特尔（Intel）创始人之一戈登·摩尔提出的。其内容为：集成电路上可容纳的晶体管数目，约每隔两年便会增加一倍；而经常被引用的“18 个月”，则是由英特尔首席执行官大卫·豪斯（David

House）提出：预计 18 个月会将芯片的性能提高一倍（即更多的晶体管使其更快），是一种以倍数增长的观测。

- 摩尔定律的定义递归后的三个版本：

  - 积体电路上可容纳的电晶体数目，约每隔 18 个月便增加一倍
  - 微处理器的性能每隔 18 个月提高一倍，或价格下降一半
  - 相同价格所买的电脑，性能每 18 个月增加一倍

3. **Disk** [blog]

   - Disk 有两个组成部分：盘组件（disk assembly）和头部组件（head assembly）

   - 盘组件有一个或者多个圆形盘片（circular platters）, 这些盘片都围绕着中心主轴。这些圆形盘片的上下两面都被磁性材料覆盖，这些磁性材料是用来储存 bit 信息的。数字 0 和 1 在磁性材料上通过不同的 pattern 储存。

   - Track 是单个盘片上的同心圆, 不同 surface 上的同心圆可以形成一个圆柱体。Track 占据了表面上的大部分空间，除了靠近轴的部分。一条 track 上的密度比半径上的 track 的密度要大得多。举例：从圆心出发的 1 英尺，含有 100000 个 tracks；但是在一个 tracks 上划定 1 英尺的距离，存了大约有百万个 bits。

   - Tracks 被切割为扇形区域（sectors），相邻的扇形区域之间都用 gaps 隔开，这些 gaps 区域没有使用磁性材料来表达 0 和 1。Sector 是一个不可切割的单元, 如果 sector 里面有一部分被损坏了, 那么整个 sector 都不能储存数据了，无法被使用。

   - 从一个同心圆的角度看，gaps 在一个同心圆上占据了大约 10% 的位置，它们的用途是帮助确定 sectors 开始的地方。一个 disk block 含有一个以上的 sectors。

   - 每一个 surface 都有一个读写头，这些读写头 move in and out together, being part of the rigid head assembly.

4. **Disk Controller**

   - Controlling the mechanical actuator that moves the head assembly, to position the heads at a particular radius

- Selecting a sector from among all those in the cylinder at which the heads are positioned
- Transferring bits between the desired sector and the computer's main memory
- buffering an entire track or more in local memory of the disk controller, hoping that many sectors of this track will be read soon, and additional accesses to the disk can be avoided.

5. **Disk Access Characteristics**

   - latency 包含三个时间：seek time, rotational latency, transfer time.

   - seek time（寻道时间）：需要移动头部组件在圆柱体上寻找 track，要确定 sector 的位置还有半径的大小

   - rotational latency（旋转延迟）：在找到 sector 和半径之后，要定位到第一个 sector，需要硬盘转动

   - transfer time（传输时间）：同一个半径下，所有的 sector 都要旋转一圈来读取或者写入

   - 磁盘上写入的方式：从上到下，由外向内。所有盘面上的同一磁道构成一个圆柱，通常称做柱面（Cylinder），每个圆柱上的磁头由上而下从“0”开始编号。数据的读/写按柱面进行，即磁头读/写数据时首先在同一柱面内从“0”磁头开始进行操作，依次向下在同一柱面的不同盘面即磁头上进行操作，只在同一柱面所有的磁头全部读/写完毕后磁头才转移到下一柱面（同心圆的再往里的柱面），因为选取磁头只需通过电子切换即可，而选取柱面则必须通过机械切换。电子切换相当快，比在机械上磁头向邻近磁道移动快得多，所以，数据的读/写按柱面进行，而不按盘面进行。也就是说，一个磁道写满数据后，就在同一柱面的下一个盘面来写，一个柱面写满后，才移到下一个柱面开始写数据。

6. **Accelerating Access to Secondary Storage**

   - Place blocks that are accessed together on the same cylinder, so we can often avoid seek time, and possibly rotational latency as well.

   - Divide the data among several smaller disks rather than one large one. Having more head assemblies that can go after blocks independently can increase the number of block accesses per unit of time.

- "Mirror" a disk: making two or more copies of the data on different disks. In addition to saving the data in case one of the disks fails, this strategy, like dividing the data among several disks, lets us access several blocks at once.

- Use a disk-scheduling algorithm, either in the operating system, in the DBMS, or in the disk controller, to select the order in which several requested blocks will be read or written.

- Prefetch blocks to main memory in anticipation of their later use.