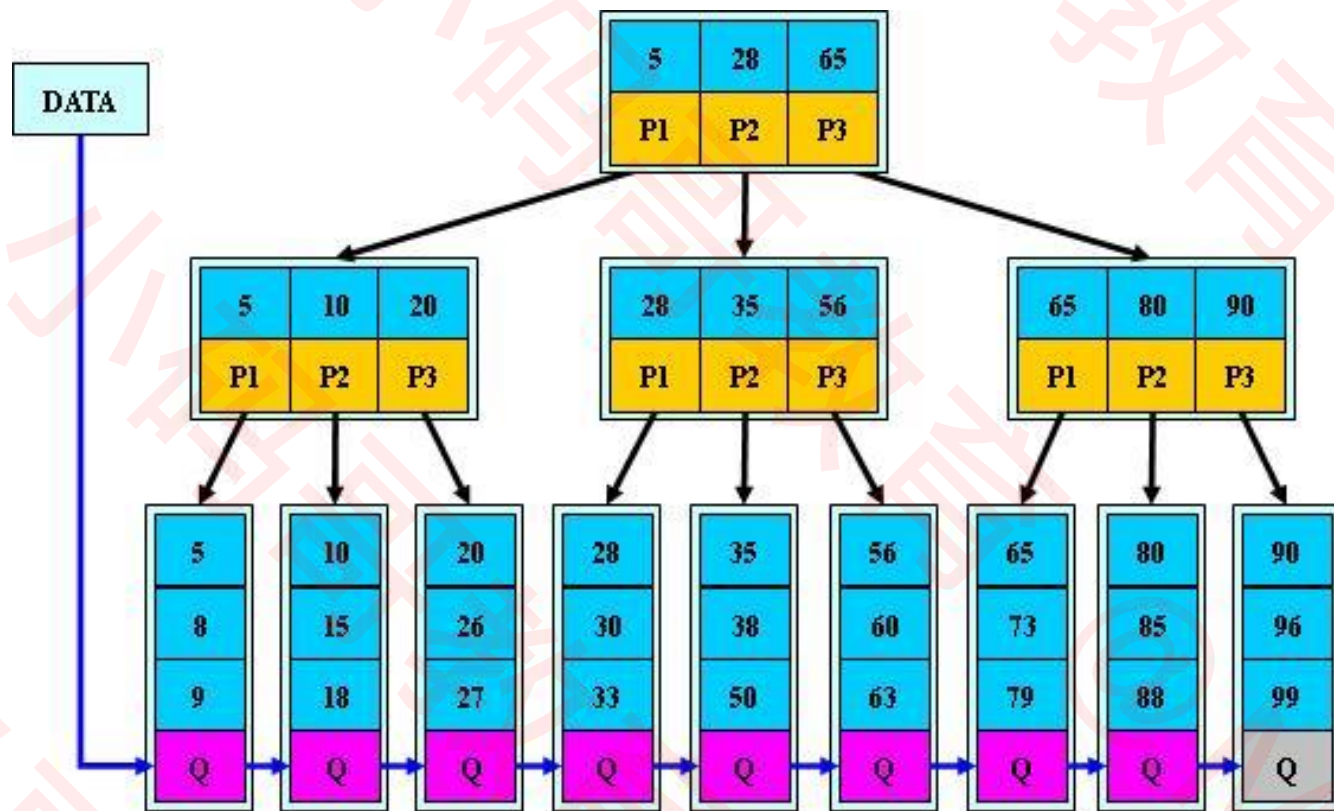


B+树

- B+树是B树的变体，常用于数据库和操作系统的文件系统中
- MySQL数据库的索引就是基于B+树实现的



■ B+树的特点

□ 分为内部节点（非叶子）、叶子节点2种节点

✓ 内部节点只存储key，不存储具体数据

✓ 叶子节点存储key和具体数据

□ 所有的叶子节点形成一条有序链表

□ m 阶B+树非根节点的元素数量 x

✓ $\lceil m/2 \rceil \leq x \leq m$

关于MySQL发音的官方说明

■ <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>

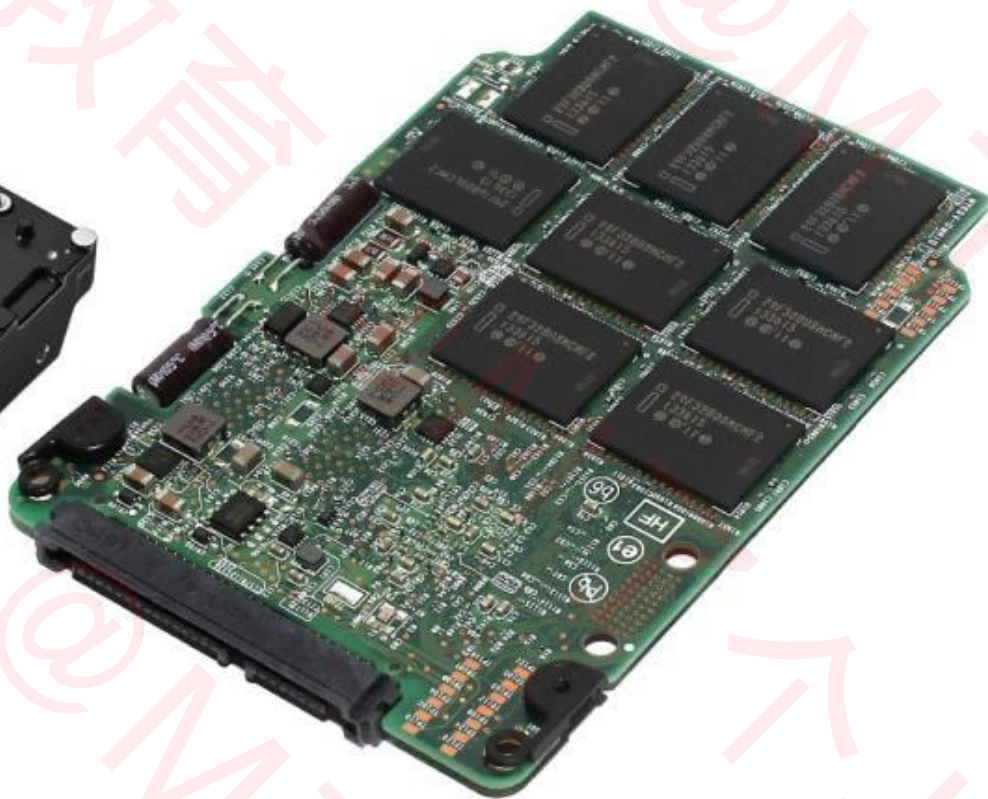
- ❑ The official way to pronounce “MySQL” is “**My Ess Que El**” (not “my sequel”),
- ❑ but we do not mind if you pronounce it as “my sequel” or in some other localized way.

硬盘的分类

- 市面上常见的硬盘有：机械硬盘（Hard Disk Drive, HDD）、固态硬盘（Solid State Drive, SSD）
- 我们主要介绍一下机械硬盘相关概念

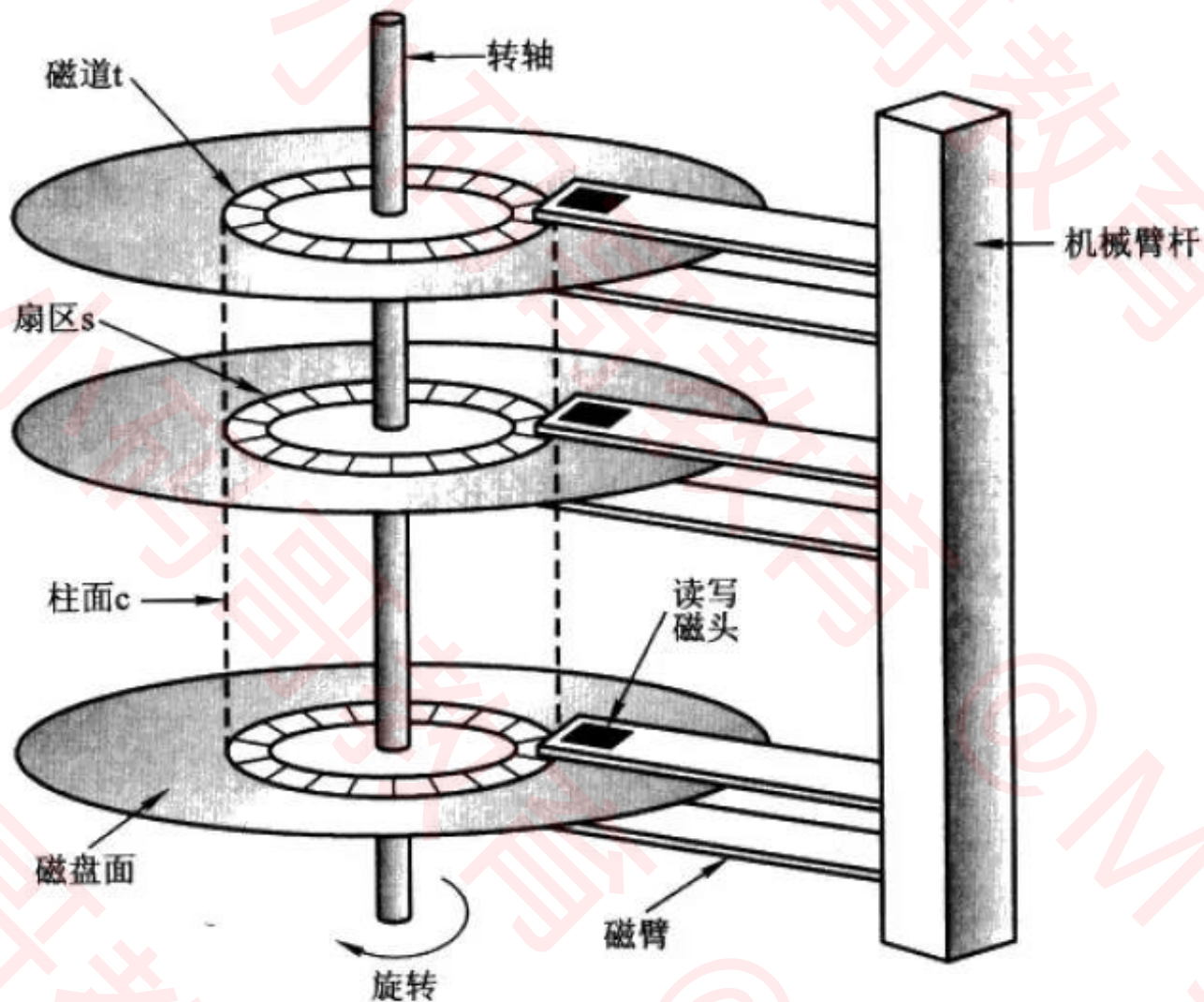


机械硬盘内部结构



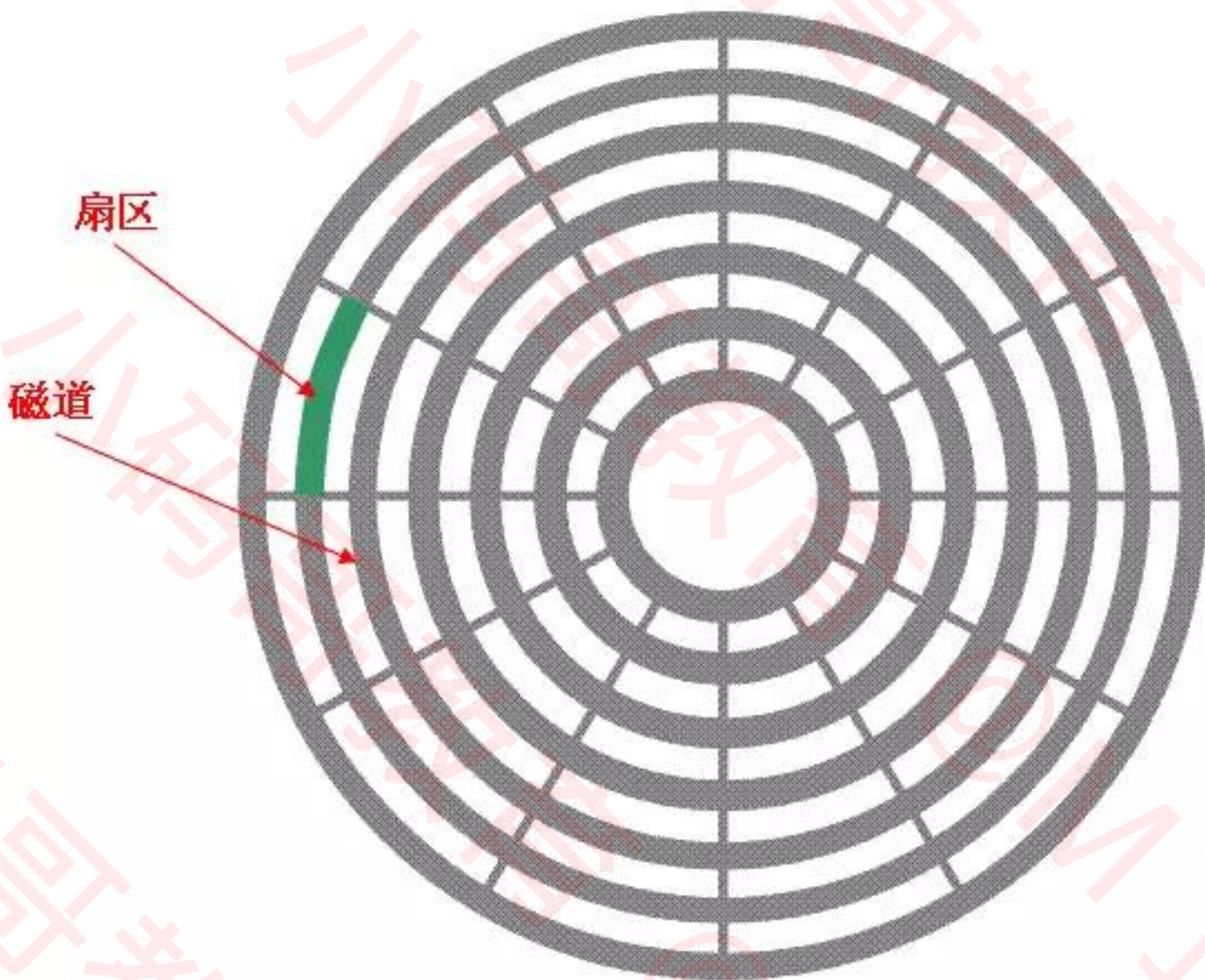
SSD固态硬盘内部结构

盘片 (platter)、盘面 (side)、读写磁头 (head)



- 硬盘一般由多个盘片组成
- 每个盘片包含2个盘面
- 每个盘面有1个对应的读写磁头
- 盘面、磁头由上到下从0开始编号

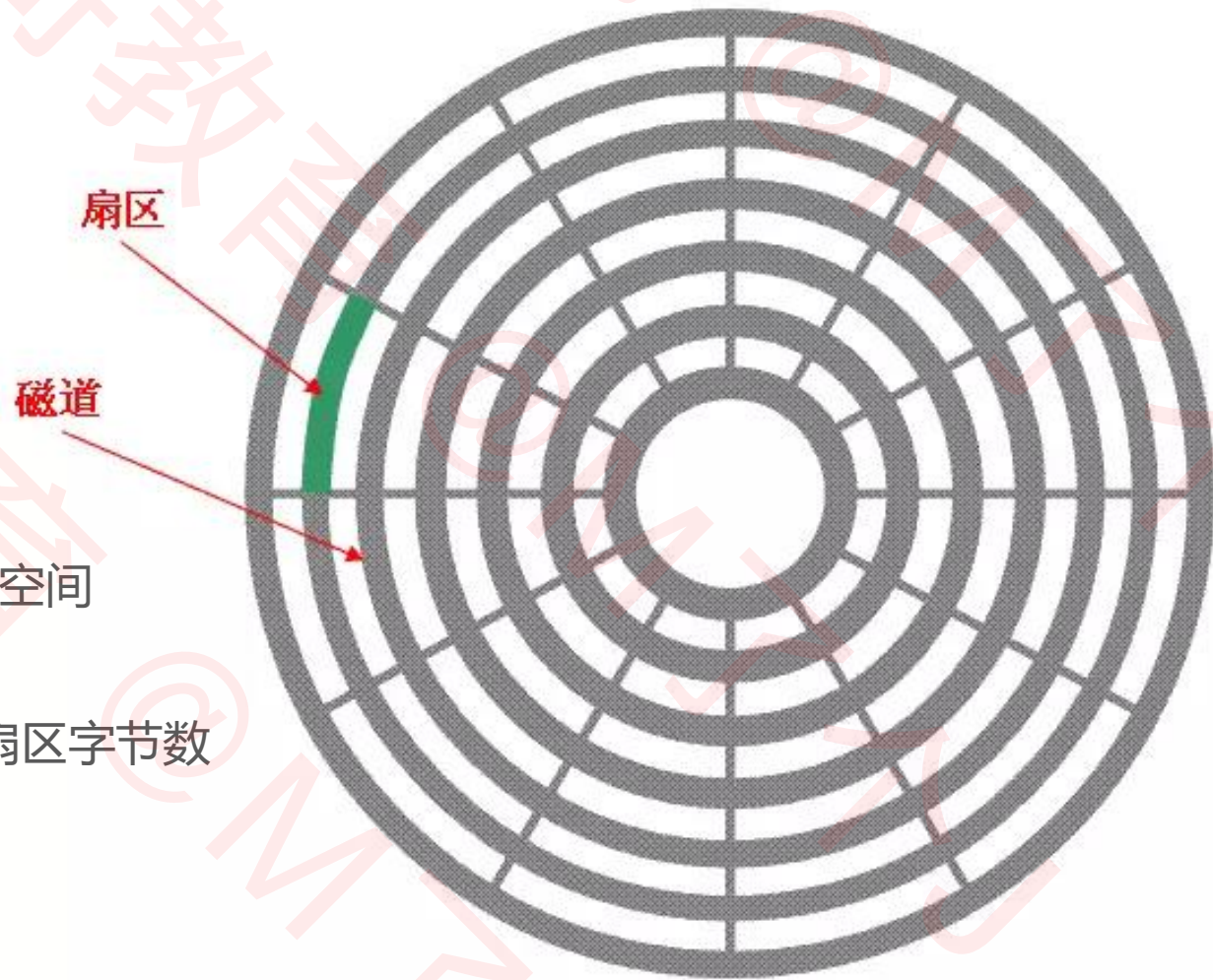
磁道 (track)、扇区 (sector)



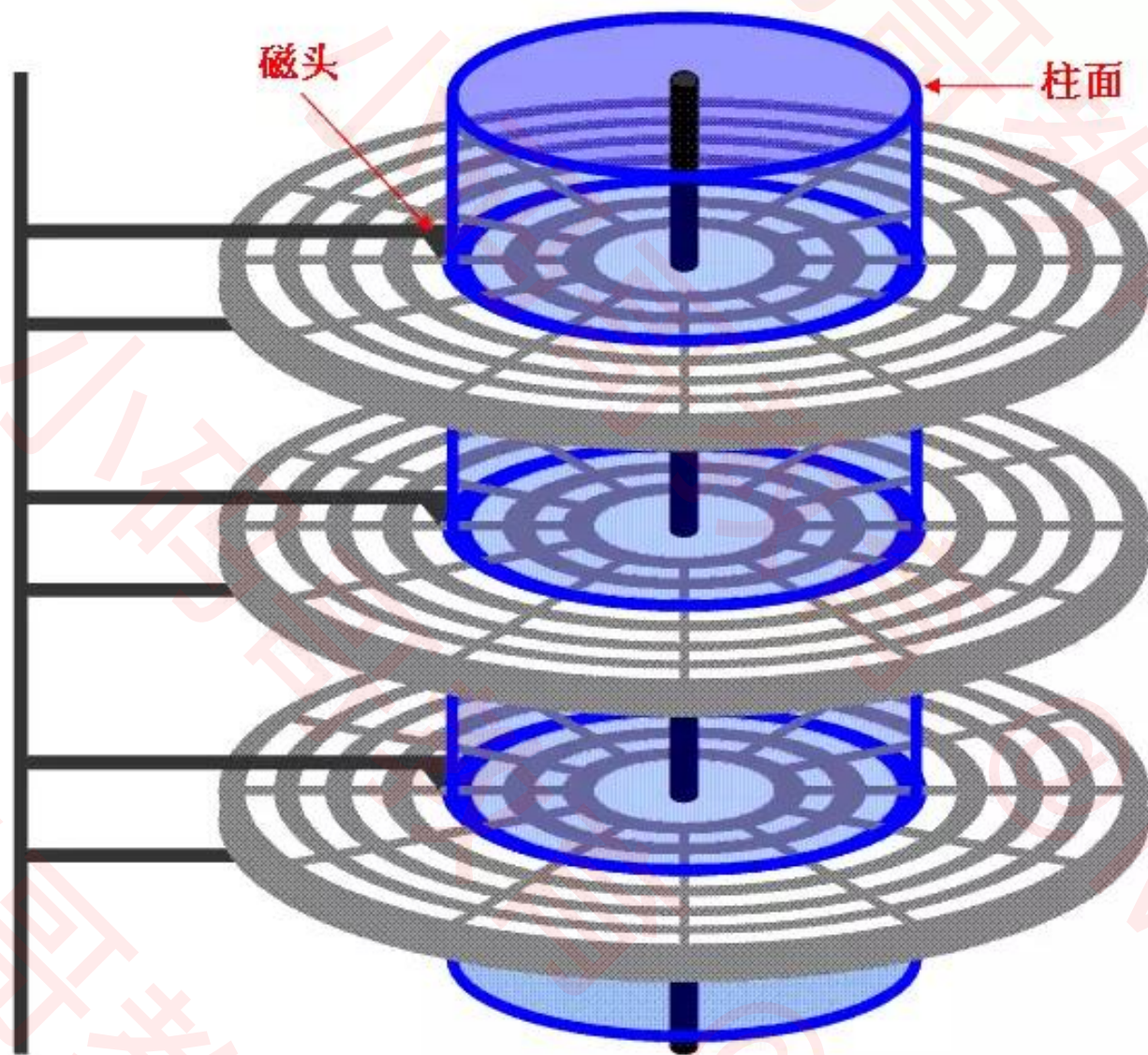
- 盘面中的一圈圈灰色圆环为是一条条的磁道
- 磁道由外到内从0开始编号
- 每条磁道上的一个弧段叫做一个扇区
- 扇区是磁盘的最小读写单位
- 一个扇区的大小通常是512字节
- ✓ 也有4096字节的

早期硬盘的扇区细节

- 每条磁道的扇区数相同
- 所以外圈扇区的面积会比内圈扇区大
- 为了更好的读取数据，它们会存放相同的字节数
- 所以外圈扇区的记录密度要比内圈小，会浪费大量的存储空间
- 硬盘的存储容量 = 磁头数 * 盘面磁道数 * 磁道扇区数 * 扇区字节数



柱面 (cylinder)



- 相同编号的磁道形成一个圆柱，称为柱面
- 磁盘的柱面数与一个盘面的磁道数是相等的

磁盘块

- 磁盘块，在Windows中叫做簇（cluster），在Linux中叫做块（block）
- 相邻的 2^n 个扇区组合在一起，形成一个磁盘块
- 操作系统对磁盘进行管理时，以磁盘块作为最小读写单位
- ✓ 一般一个磁盘块是4096字节（4KB，由8个连续的512字节扇区组成）

■ 注意

- 磁盘块是操作系统中的一个虚拟概念
- 扇区是磁盘上真实存在的物理区域

查看硬盘信息

■ Windows

- 如果是查看D盘，管理员权限打开命令行工具，输入【fsutil fsinfo ntfsinfo d:】
- 或者搜索框输入【系统信息】

■ Mac

- 使用【磁盘工具】

操作系统读取硬盘数据的过程

① 操作系统将LBA传送给磁盘驱动器并启动读取命令

✓ LBA (Logical Block Address, 逻辑块地址)

✓ 比如类似设备号4、磁头号4、磁道号8、扇区号16、扇区计数8这样的信息

② 磁盘驱动器根据LBA将磁头移动到正确的磁道，盘片开始旋转，将目标扇区旋转到磁头下

③ 磁盘控制器将扇区数据等信息传送到一个处于磁盘界面的缓冲区

④ 磁盘驱动器向操作系统发出“数据就绪”信号

⑤ 操作系统从磁盘界面的缓冲区读取数据

✓ 既可以按照一个字节一个字节的方式读取

✓ 也可以启动DMA (Direct Memory Access, 直接内存访问) 命令读取

磁盘完成IO操作的时间

- 主要由寻道时间、旋转延迟时间、数据传输时间3部分构成

- 寻道时间 (seek)

- ✓ 将读写磁头移动至正确的磁道上所需要的时间，这部分时间代价最高

- 旋转延迟时间 (rotation)

- ✓ 盘片旋转将目标扇区移动到读写磁头下方所需要的时间，取决于磁盘转速

- 数据传输时间 (transfer)

- ✓ 完成传输数据所需要的时间，取决于接口的数据传输率，通常远小于前两部分消耗时间

- 决定时间长短的大部分因素是和硬件相关的，但所需移动的磁道数是可以通过操作系统来进行控制的

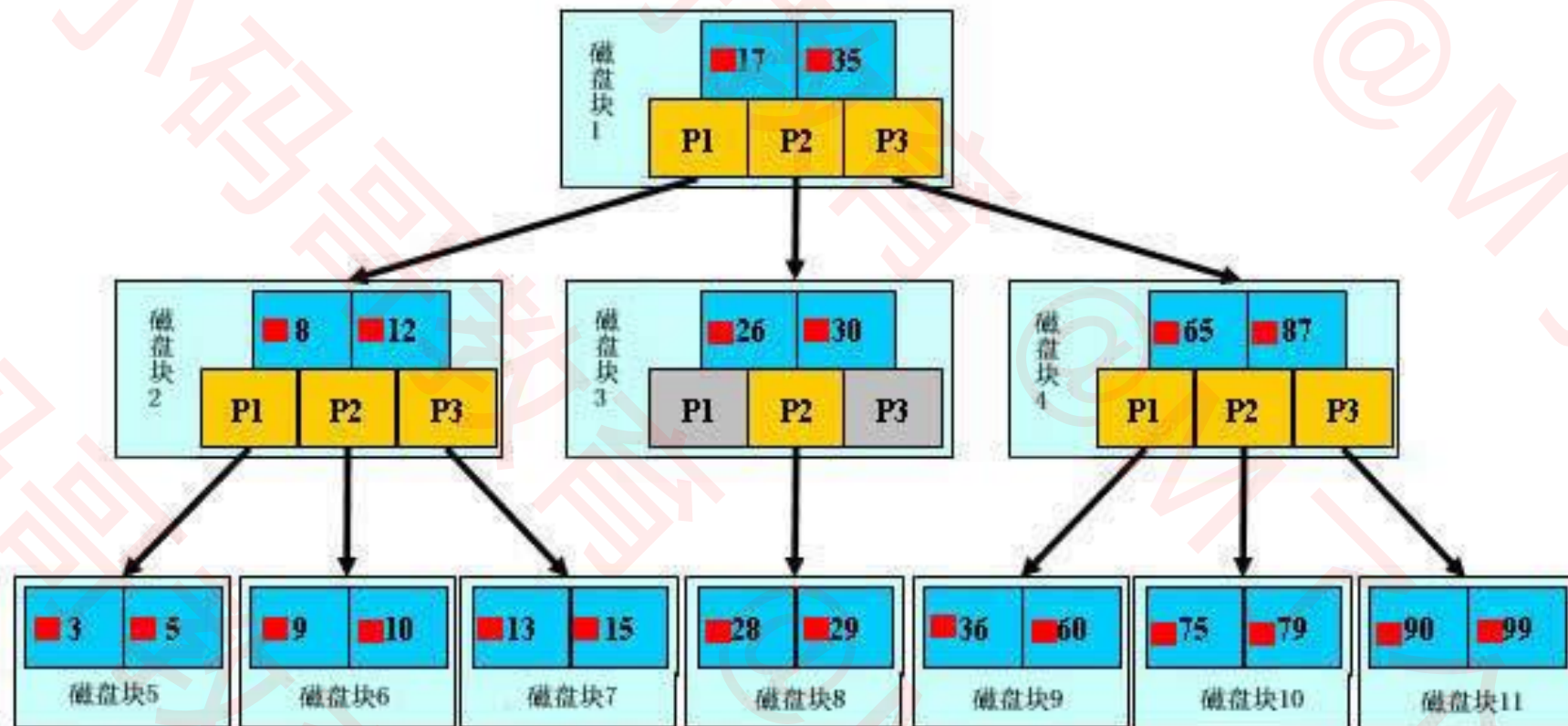
- 减少所需移动的磁道数是减少整个硬盘读写时间的有效办法

- 合理安排磁头的移动以减少寻道时间就是磁盘调度算法的目的所在

MySQL的索引底层为何使用B+树?

- 为了减小IO操作数量，一般把一个节点的大小设计成最小读写单位的大小
 - MySQL的存储引擎InnoDB的最小读写单位是16K
- 对比B树，B+树的优势是
 - 每个节点存储的key数量更多，树的高度更低
 - 所有的具体数据都存在叶子节点上，所以每次查询都要查到叶子节点，查询速度比较稳定
 - 所有的叶子节点构成了一个有序链表，做区间查询时更方便

B树



B*树

■ B*树是B+树的变体：给内部节点增加了指向兄弟节点的指针

□ m 阶B*树非根节点的元素数量 x

✓ $\lceil 2m/3 \rceil \leq x \leq m$

