第9讲：存储管理

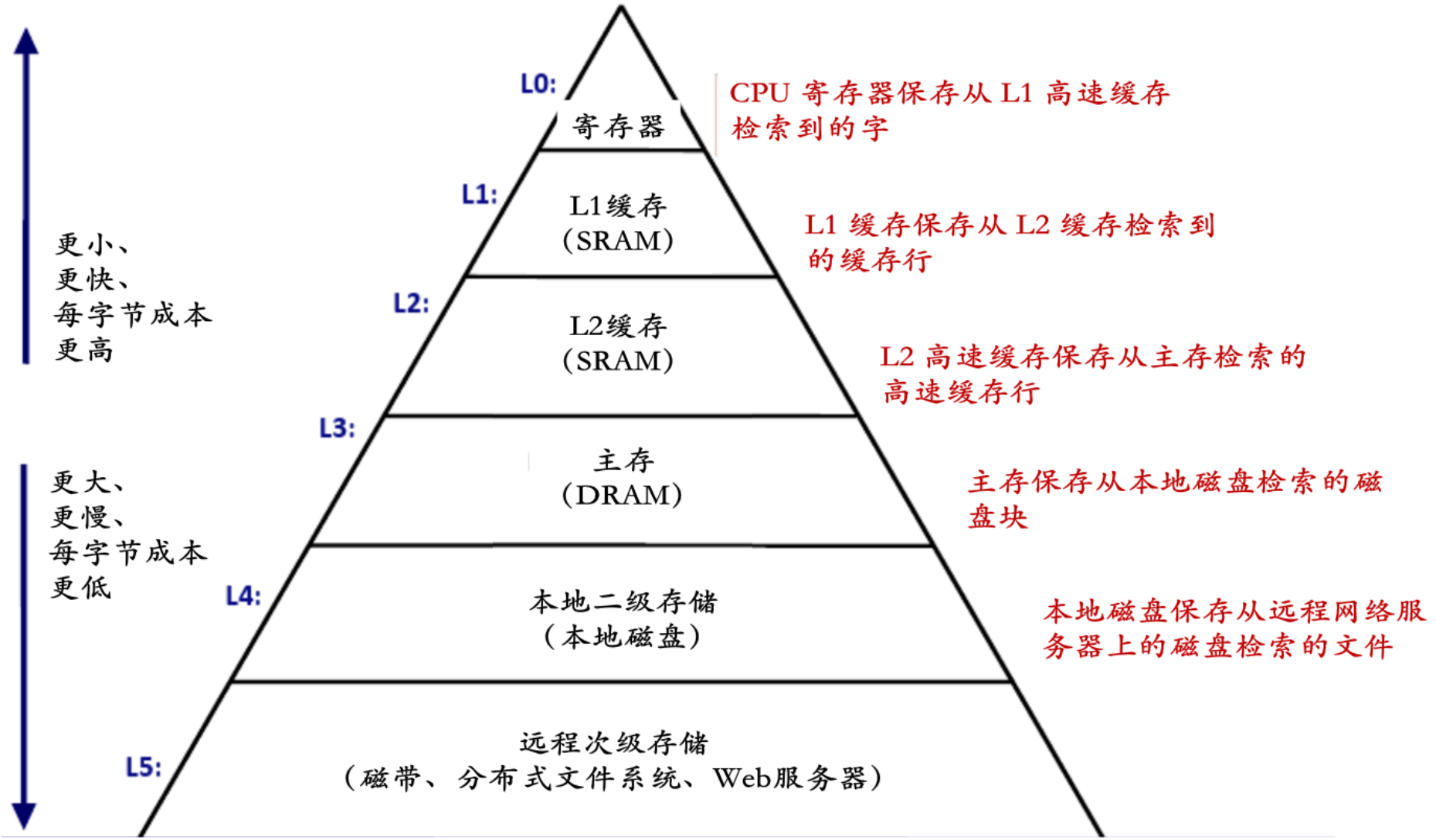
大纲

1. 存储介质（Storage Media）
2. 数据库在磁盘上的表示(Representation of Databases on Disks)
   - 元组布局（Tuple Layout）
   - 页面布局（Page Layout）
     - 面向元组的页面布局（Tuple-Oriented Page Layout）
   - 文件组织（File Organization）
3. 缓冲区管理（Buffer Management）

存储介质

存储层级

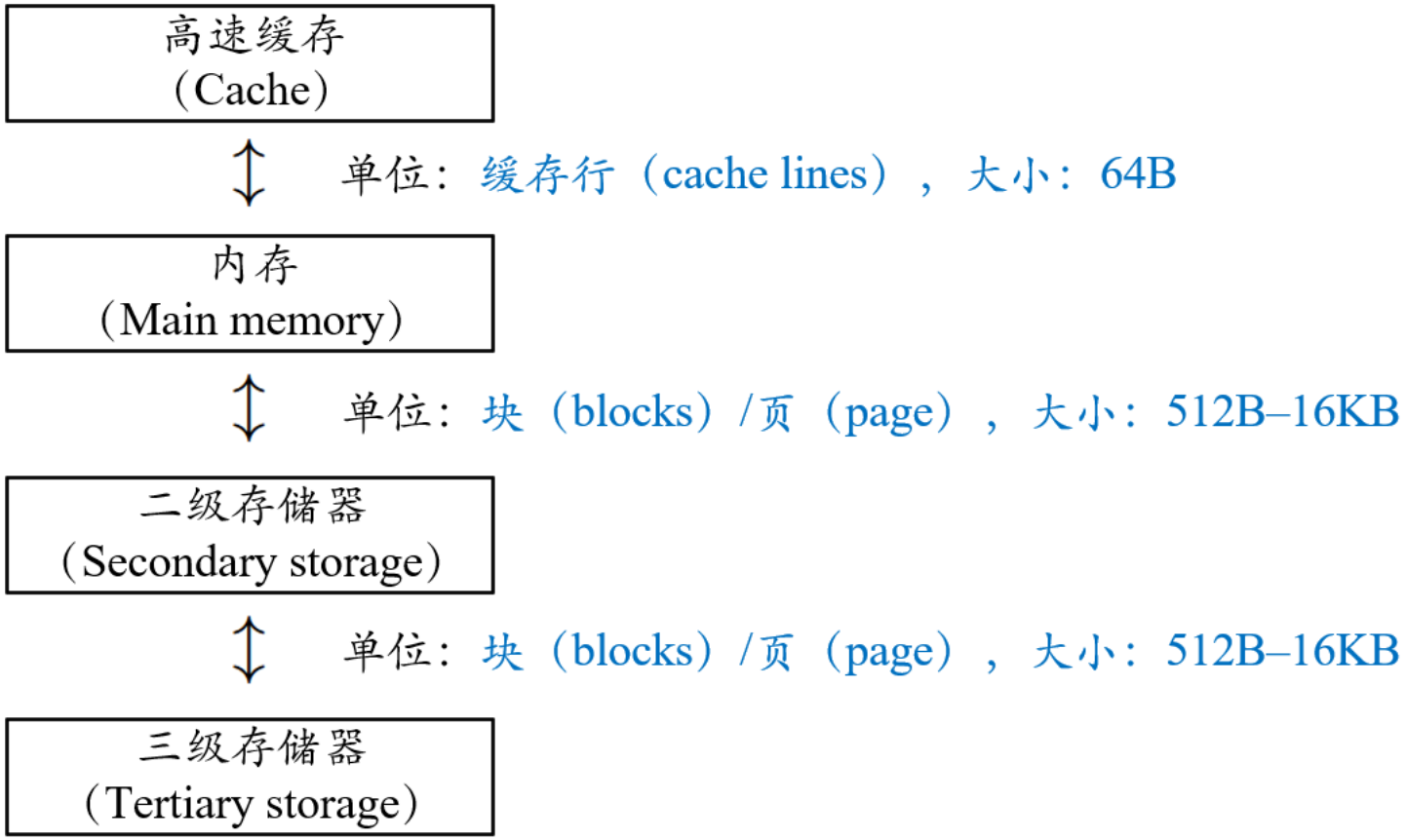
计算机系统中的存储器按层次结构排列



访问时间（Access Time）

访问时间 (ns)	存储介质
0.5	L1缓存
7	L2缓存
100	动态随机存取存储器（DRAM）
150,000	固态硬盘（SSD）
10,000,000	机械硬盘（HDD）
30,000,000	网络存储
1,000,000,000	磁带（Tape）

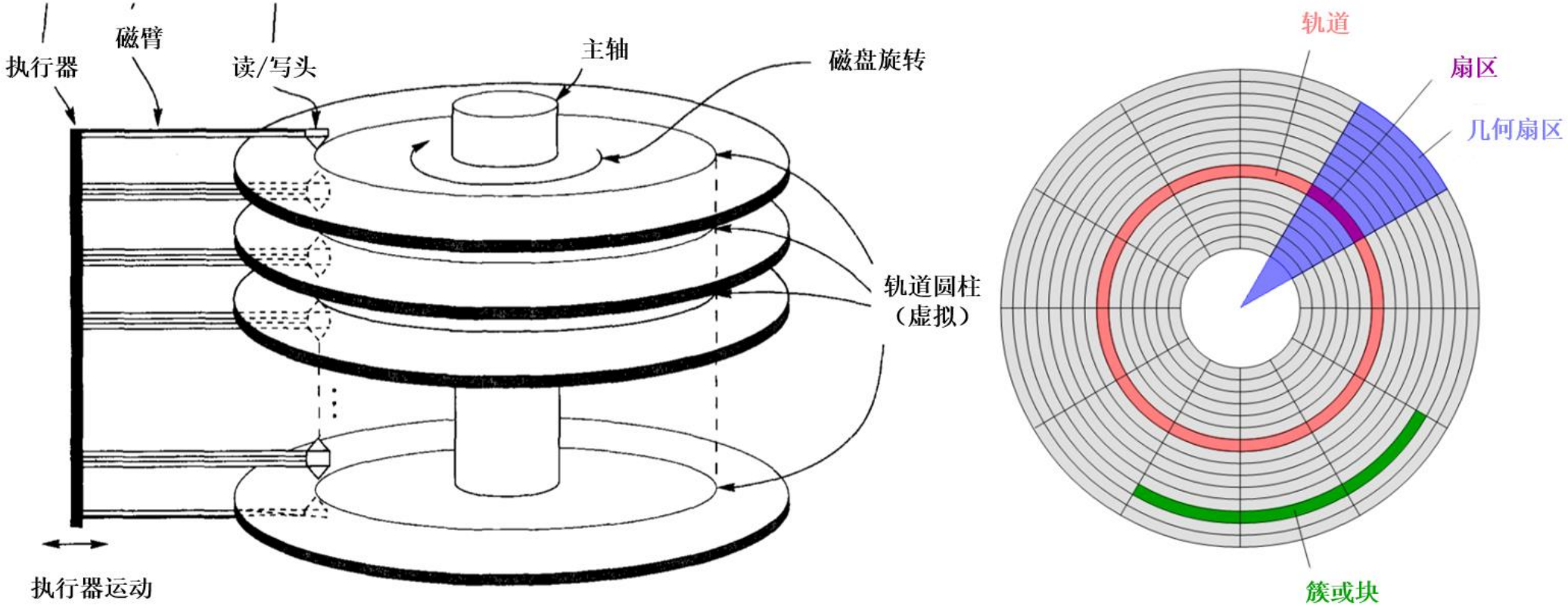
层级之间的数据传输


高速缓存
(Cache)
单位：缓存行（cache lines），大小：64B
内存
(Main memory)
单位：块（blocks）/页（page），大小：512B–16KB
二级存储器
(Secondary storage)
单位：块（blocks）/页（page），大小：512B–16KB
三级存储器
(Tertiary storage)

磁盘（Magnetic Disks）

磁盘由两部分组成

- 磁盘组件：扇区（sectors）⊂ 磁道（tracks）⊂ 柱面（cylinders）
- 磁头组件：磁头（disk heads）和 磁臂（disk arms）



磁盘块（Disk Blocks）/ 页（Pages）

操作系统在磁盘格式化时将轨道划分为大小相等的磁盘块（或页）

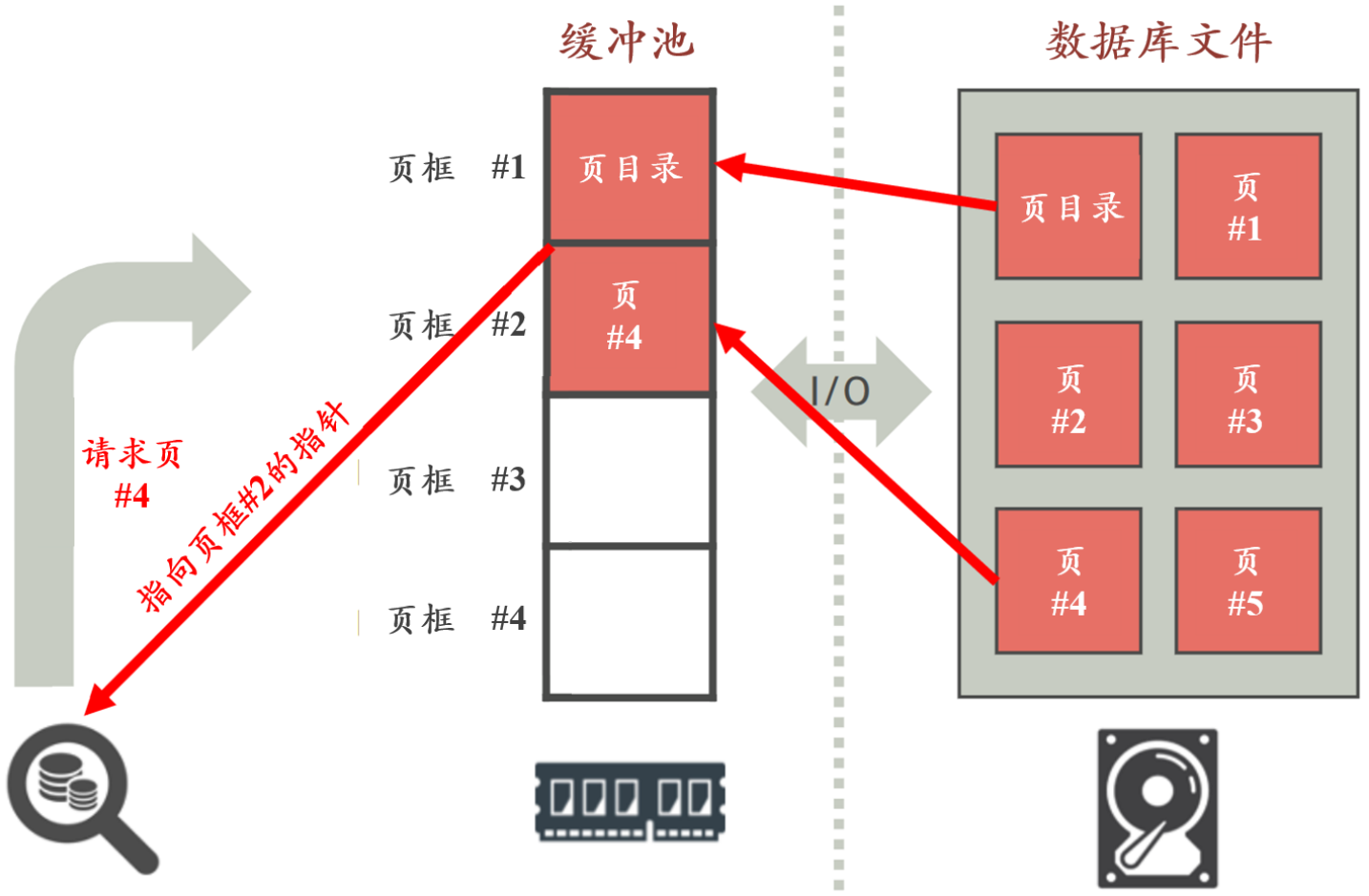
- 磁盘块的大小可以设置为扇区大小的倍数
- 块大小在磁盘格式化时固定，无法动态更改
- 典型磁盘块大小是 512B – 4KB

逻辑块地址（LBA, Logical Block Address）

磁盘块的逻辑块地址（LBA）是一个介于0和n-1之间的数字（假设磁盘的总容量为n个块）

面向磁盘的数据库存储

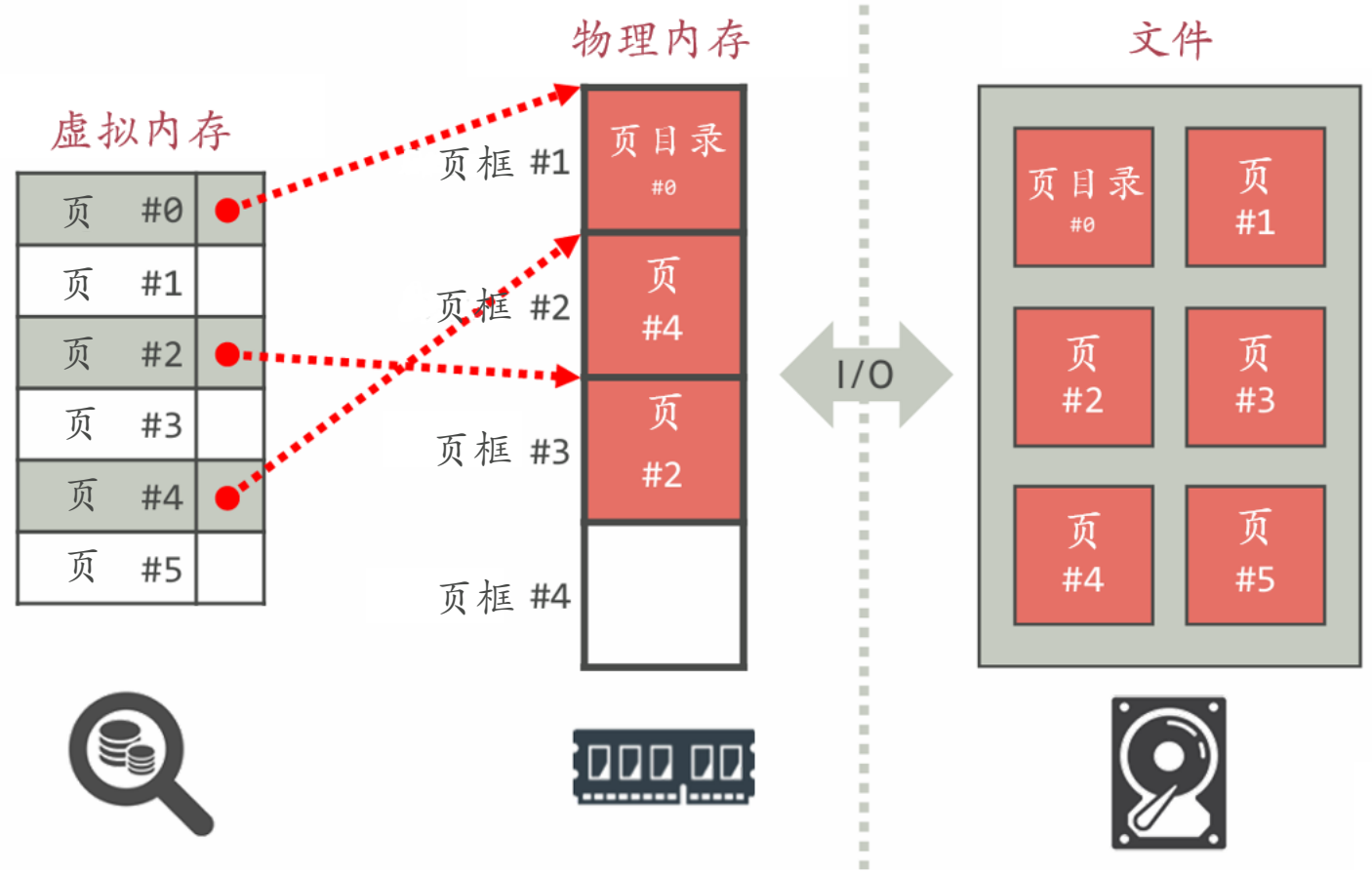
- 数据持久性地存储在二级存储器中
- 当数据要被读取或修改时（但尚未在主存储中找到），数据会被加载到主存储中



对比：OS内存映射

为什么不选择操作系统?

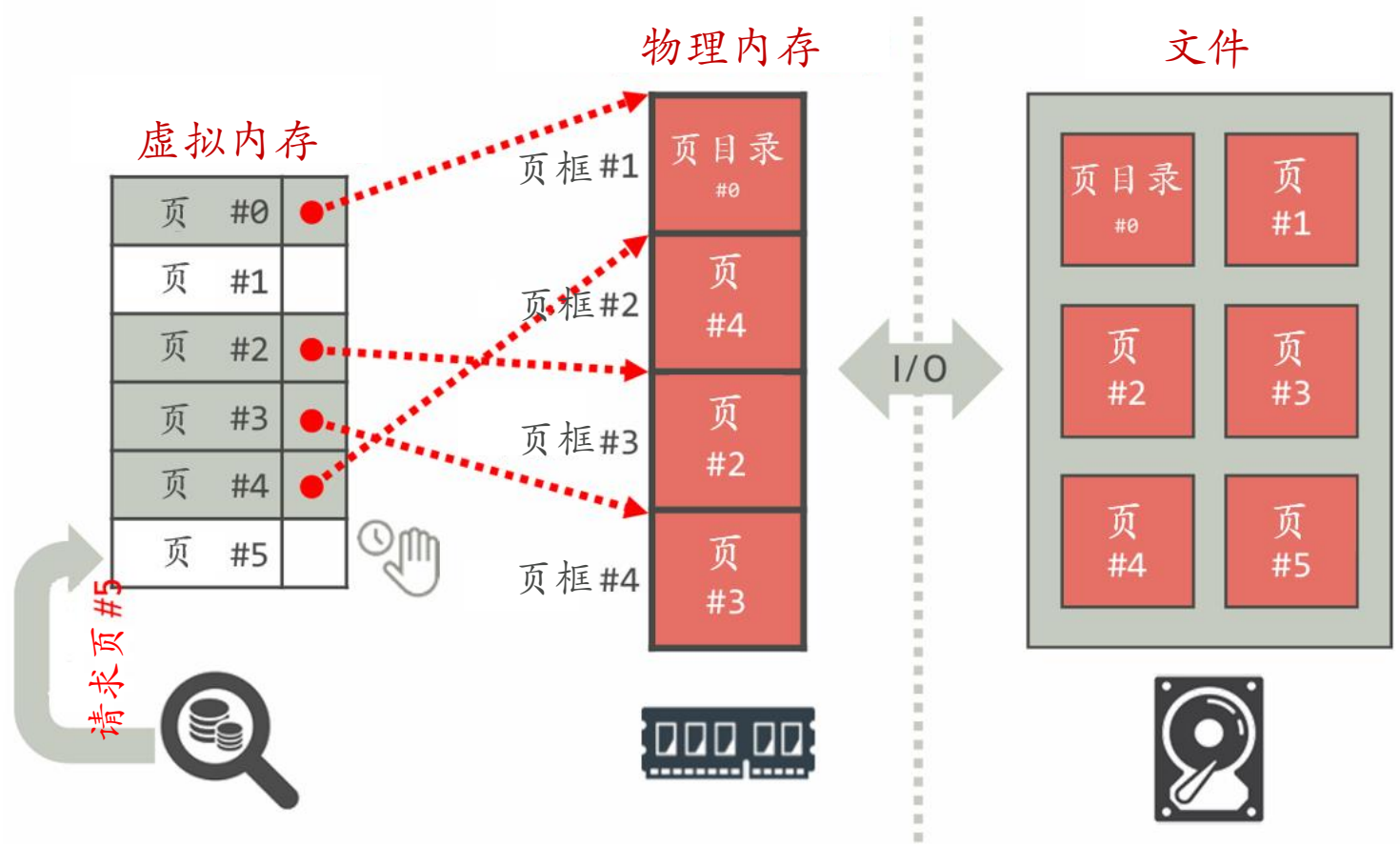
我们可以使用内存映射（mmap）将文件内容存储到进程的地址空间中。



对比：OS页面调度

为什么不选择操作系统?

操作系统负责页面调度。



面向磁盘的数据库存储

设计目标:

- 允许数据库管理系统（DBMS）管理超过可用内存量的数据库
- 读写磁盘的代价高(I/O Cost)，因此管理系统需要减少磁盘读写以避免大的延迟和性能下降

空间控制（Spatial Control）：

- 在磁盘上写入页面的位置
- 尽可能使经常同时使用的页面在磁盘上存储在邻近位置（数据局部性原理）
- 顺序读写 vs. 随机读写，减少寻址时间。

时间控制（Temporal Control）：

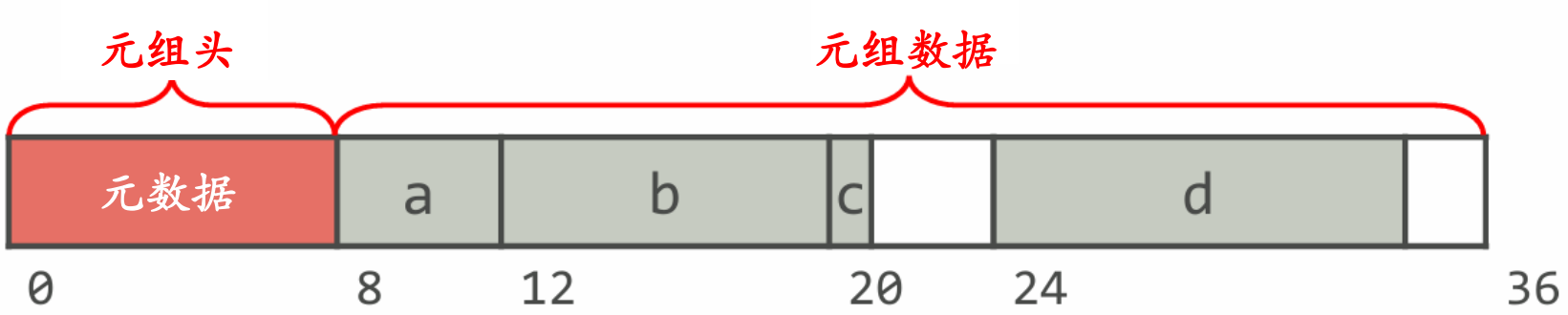
- 何时将页面读入内存，何时将它们写入磁盘
- 最小化因从磁盘读取数据而导致的延迟数

磁盘上数据库的表示：元组布局

元组布局（Tuple Layout）

元组（Tuple）本质上是字节序列

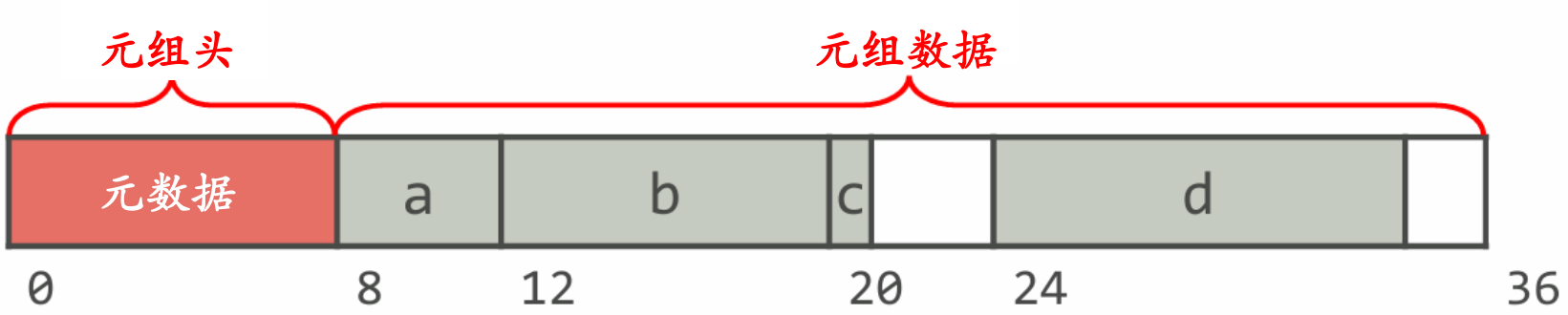
- 数据库管理系统（DBMS）的工作是将这些字节序列解释为对应的数据类型和值
- DBMS 的目录中包含关于表（Tables）的模式信息（Schema Information），DBMS可以用这些信息来确定元组布局



元组头（Tuple Header）

每个元组都有一个元组头，包含了关于它的元数据。

- 指向DBMS目录的中存储该类型元组的模式（Schema）的指针
- 元组的长度
- 可见性信息（并发控制）
- 表示Null值位置的位图

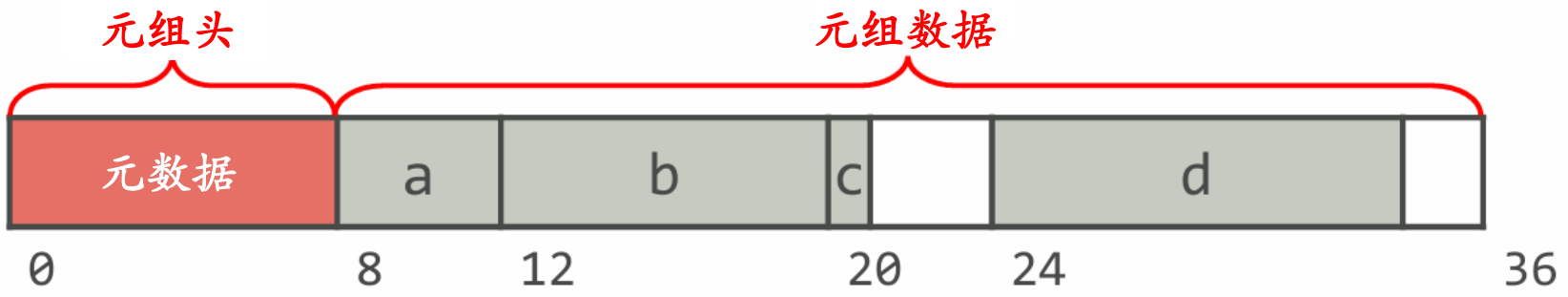


元组数据（Tuple Data）

元组数据部分连接元组的所有数据值。

- 通常以创建表时指定的顺序来存储数据值
- 每个数据值从Data部分起始位置，按照4字节/8字节的倍数对齐

```
CREATE TABLE Foo(
    a INT NOT NULL,
    b BIGINT NOT NULL,
    c CHAR NOT NULL,
    d VARCHAR(10) NOT NULL );
```



磁盘上数据库的表示：页面布局

页面布局（Page Layout）

数据库页面

页是固定大小的数据块（512B-16KB）

- 它可以包含元组、元数据、索引、日志记录等。
- 大多数数据库管理系统不能同一个页面上存储不同类型的数据
- 一些数据库管理系统要求页面是自包含的

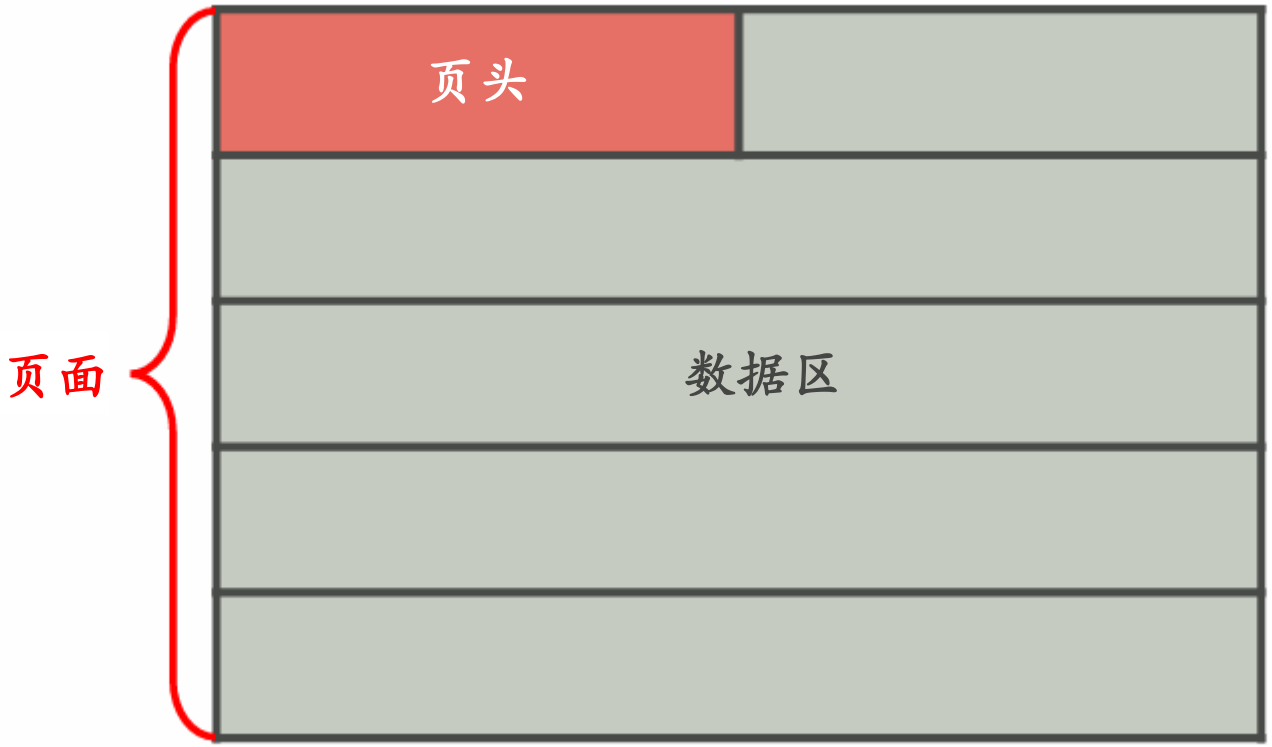
每个页面都有唯一的标识符作为其页号（page ID）

- 数据库管理系统（DBMS）使用一个间接层来映射页号到物理位置。

页面布局（Page Layout）

页面布局

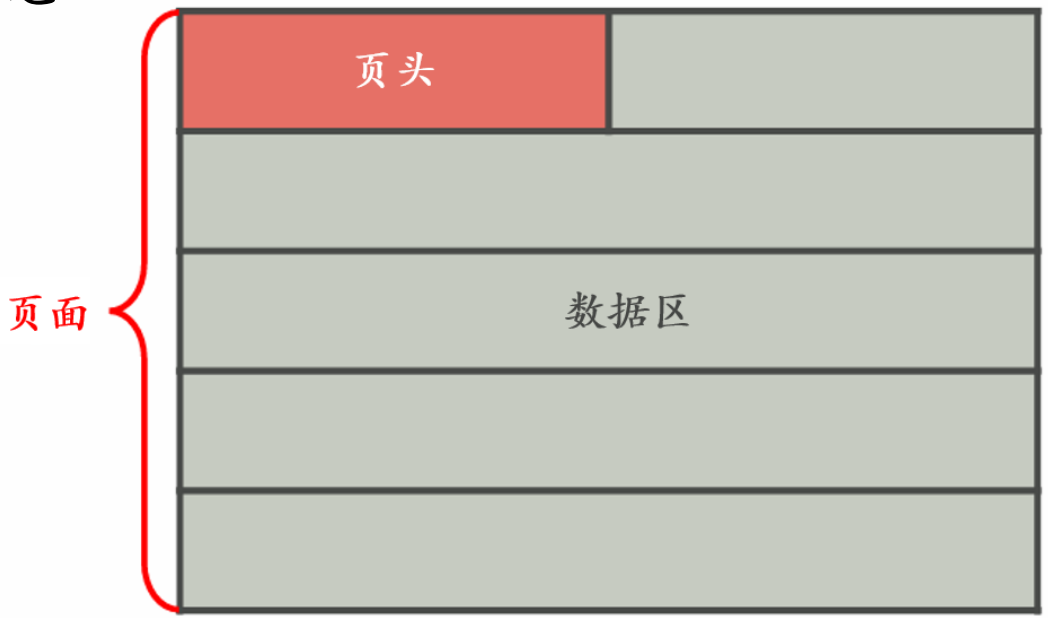
页面由页头和数据区组成



页头（Page Header）

每个页面都包含一个有关页面内容元数据的页头

- 页面大小
- 校验和（Checksum）
- DBMS版本
- 事务可见性
- 压缩信息



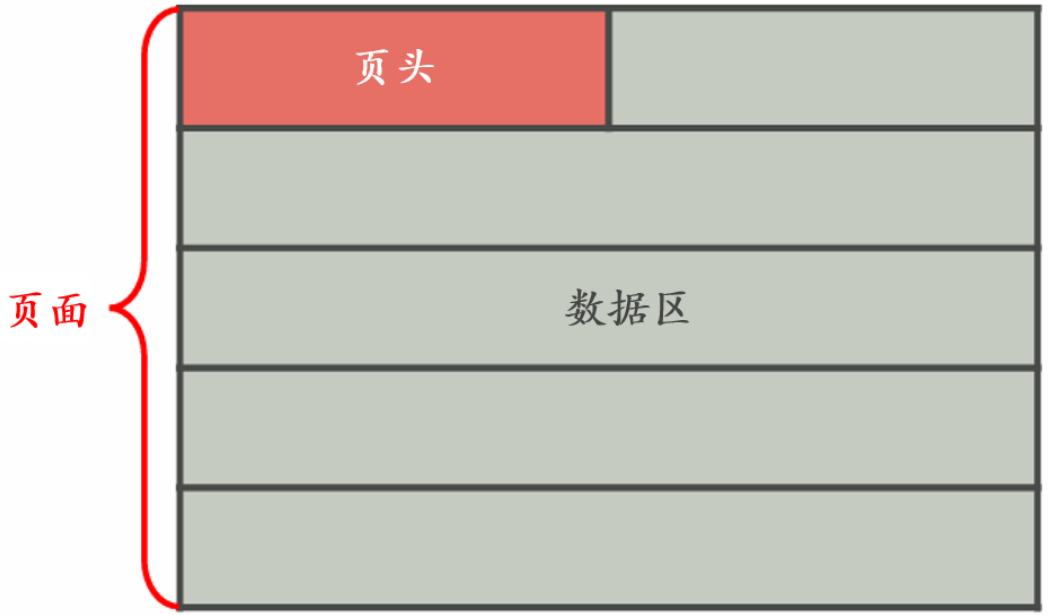
数据区（Page Data）

这里我们只考虑存储一种元组的页面。

如何在页面内组织存储的数据?

方法1➢面向元组（Tuple-oriented）

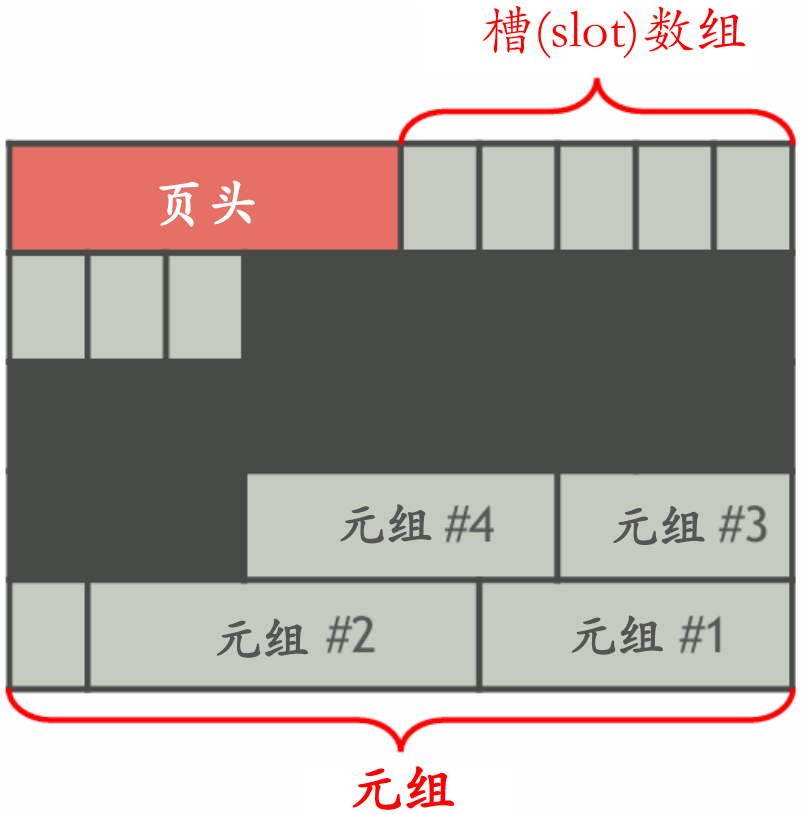
方法2➢日志结构化（Log-structured）（自行了解，如bigtable）



分槽页面（Slotted Pages）

页面布局

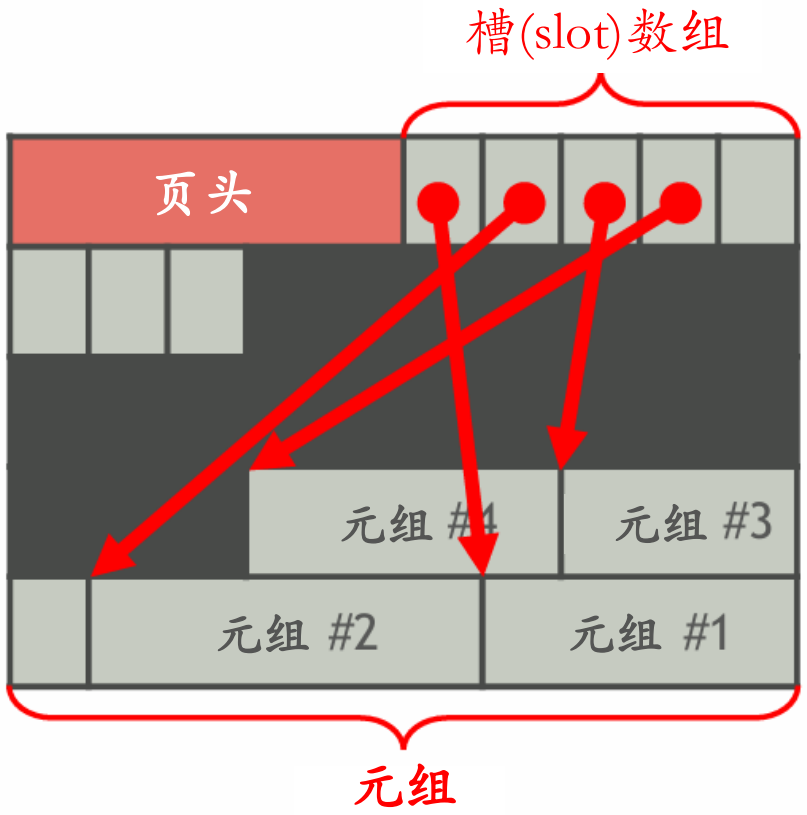
最常见的页面布局方案被称为分槽页面（Slotted Pages）



分槽页面（Slotted Pages）

分槽页面

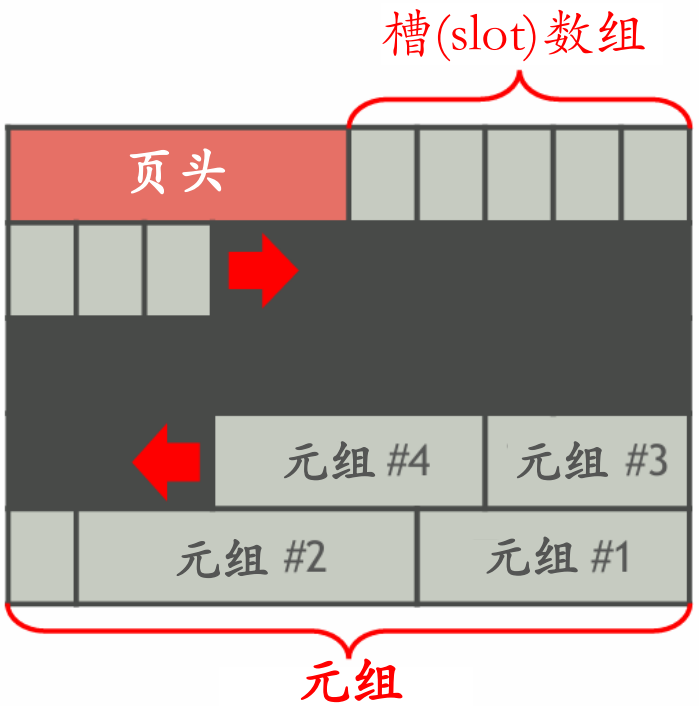
Slot 数组，将已使用的“槽”映射到元组起始位置的偏移量



分槽页面（Slotted Pages）

分槽页面的表头中需要维护：

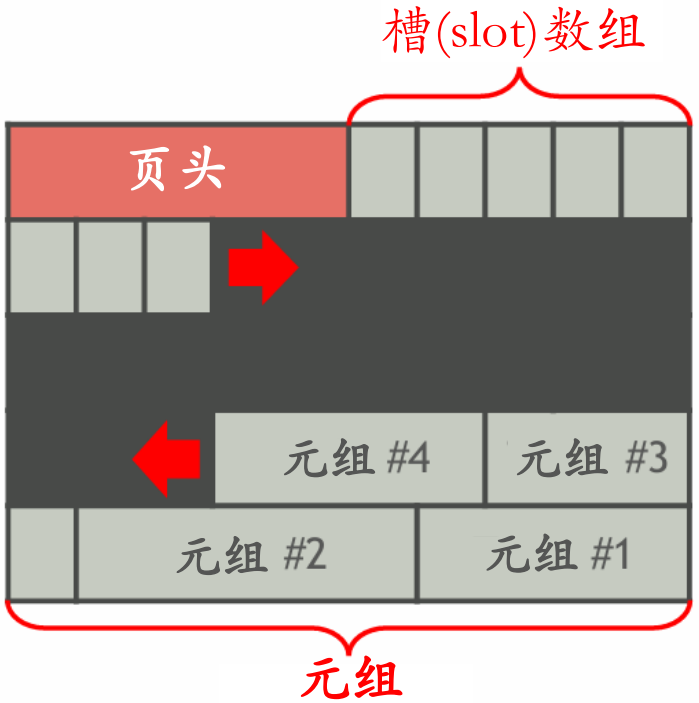
- 已使用槽的数量
- 最后一个使用槽的起始位置偏移量



记录号（Record ID）

为了追踪单个元组，DBMS为每个元组分配了一个唯一的*记录标识符*。
最常见的记录号是（页号，槽号）：

- 页号：包含该元组的页面的ID
- 槽号：元组在页面中存储的槽位编号



磁盘上数据库的表示
文件组织

文件组织-文件存储：

数据库管理系统（DBMS）将数据库存储为磁盘上的一个或者多个文件

- 操作系统（OS）对文件内容一无所知

存储管理器负责维护数据库的文件，将文件组织成一系列页面的集合

- 追踪页面的数据读/写
- 追踪可用空间

文件组织-页面存储架构

不同的数据库管理系统在磁盘上的文件中管理页面方式各不相同，
主要分为三种方法：

- 堆文件组织
- 顺序/排序文件组织
- 哈希文件组织

文件组织-堆文件组织：

堆文件（Heap File）是一个无序的页面集合，其中存储的元组是随机排序的

- 创建/获取/写/删除页面
- 必须支持遍历所有页面

需要 Meta-data 跟踪哪些页面存在以及空闲页面

表示堆文件的两种方式：

- 链表法
- 页目录法

文件组织-堆文件组织：

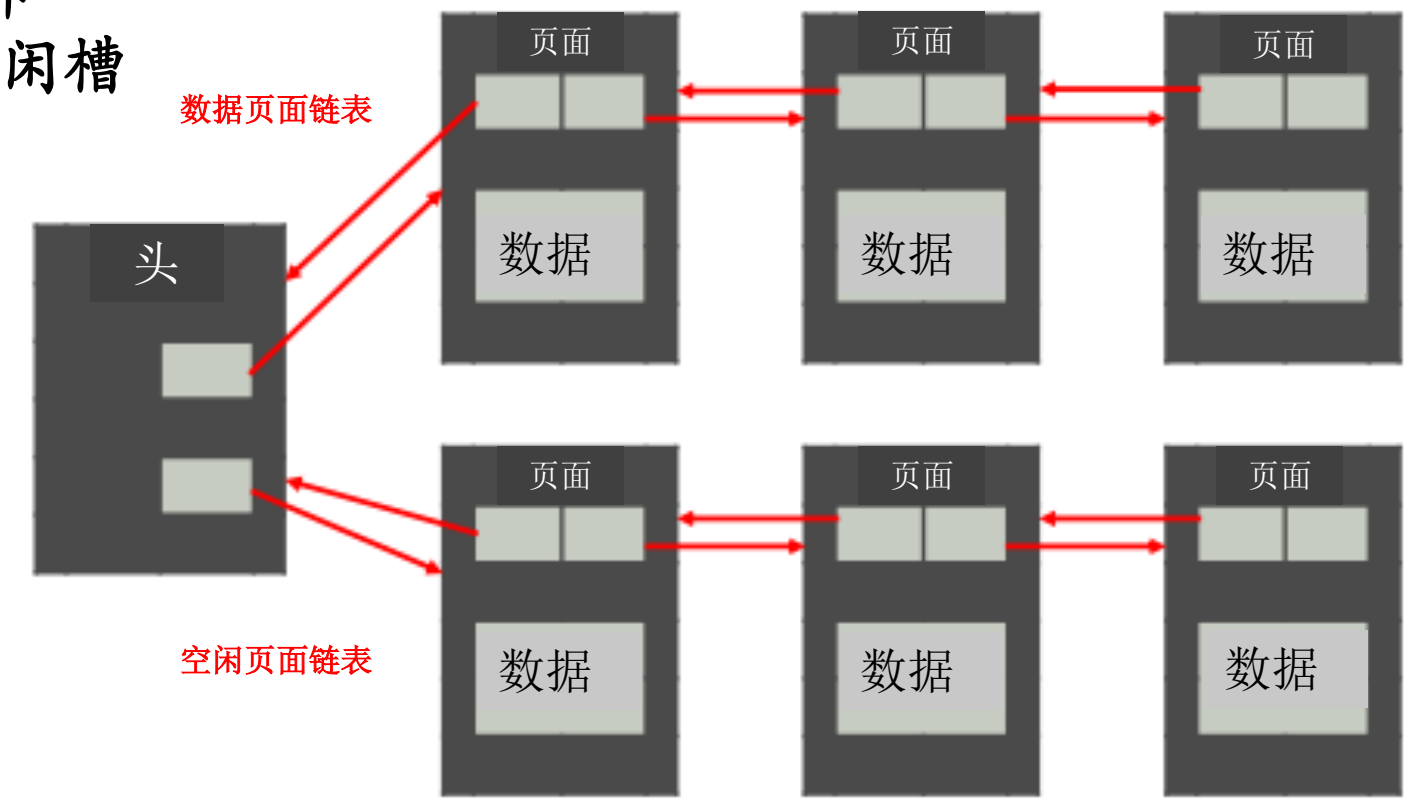
堆文件组织：链表法

在文件开头维护一个头页面（head page），该页面存储两个指针：

（1）空闲页面链表的头部

（2）数据页面链表的头部

每个页面都跟踪自身中空闲槽



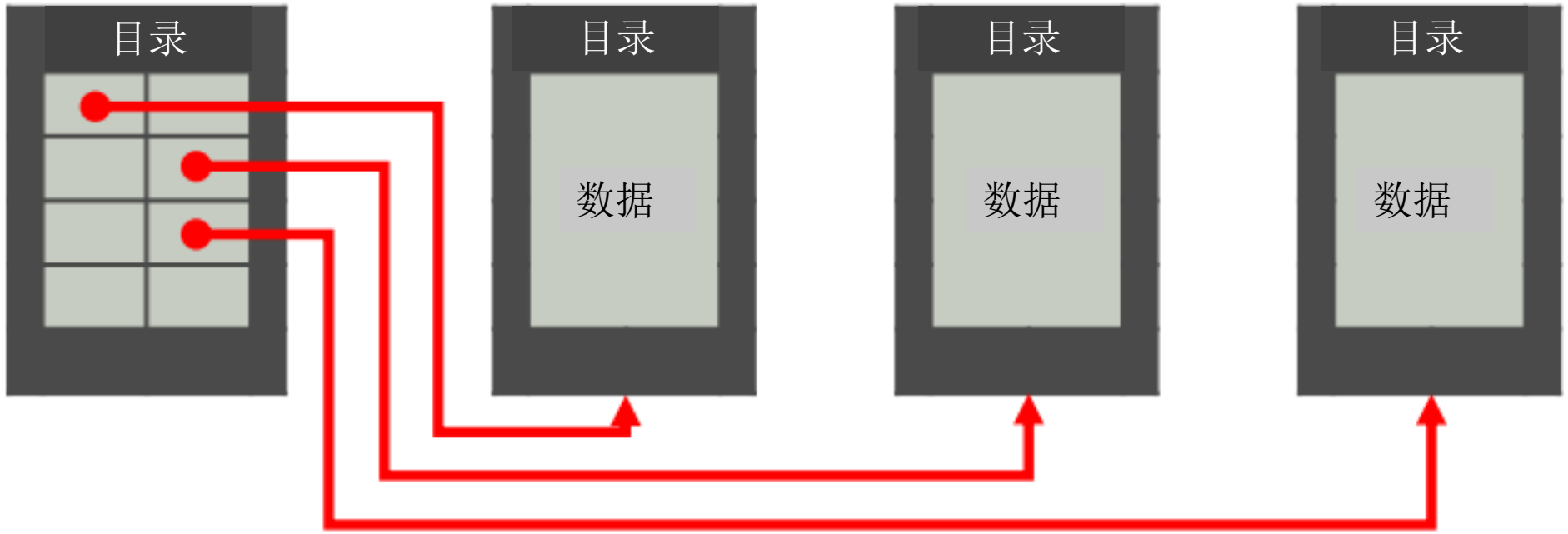
文件组织-堆文件组织：

堆文件组织：页目录法

(1)DBMS维护特殊页面，负责追踪数据库文件中数据页面位置

(2)目录还记录每个页面的空闲槽数量

(3)DBMS必须确保目录页面与数据页面保持同步



文件组织-顺序/有序文件组织：

顺序/排序文件是一个有序的页面集合，其中存储的元组按照排序键的顺序排列

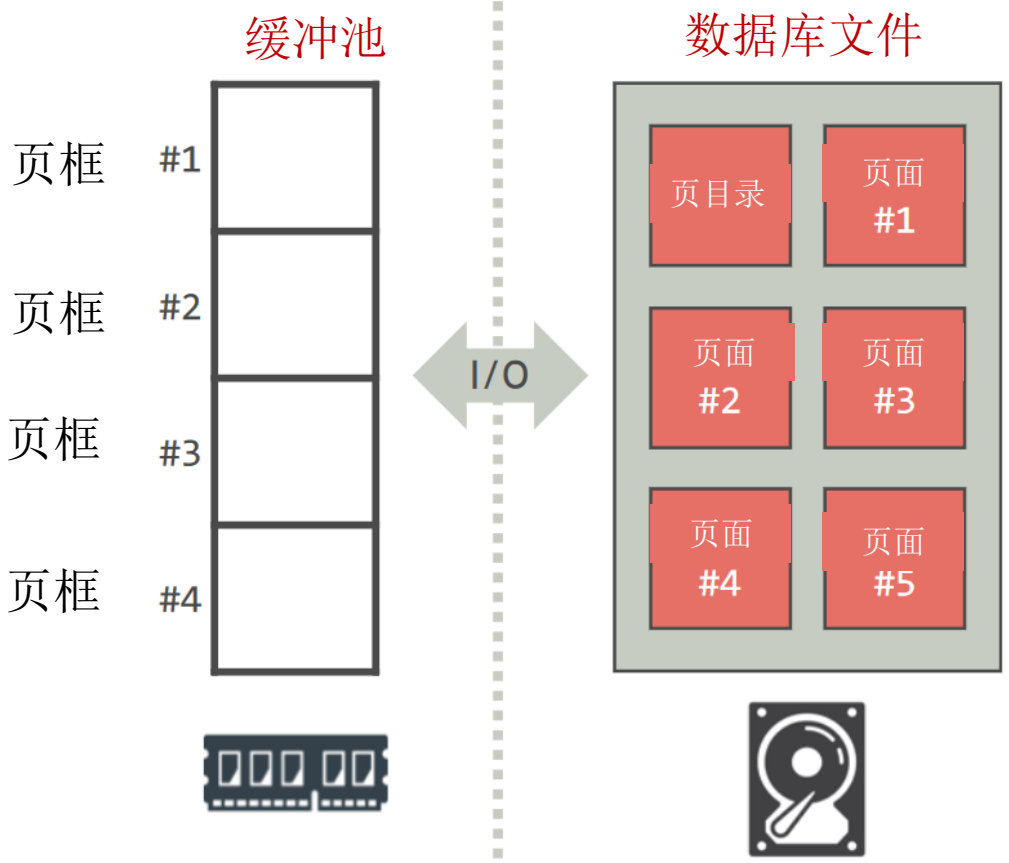
■除非使用了主索引，否则很少使用有序文件。

缓冲区管理

缓冲区管理-缓冲池：

缓冲池：

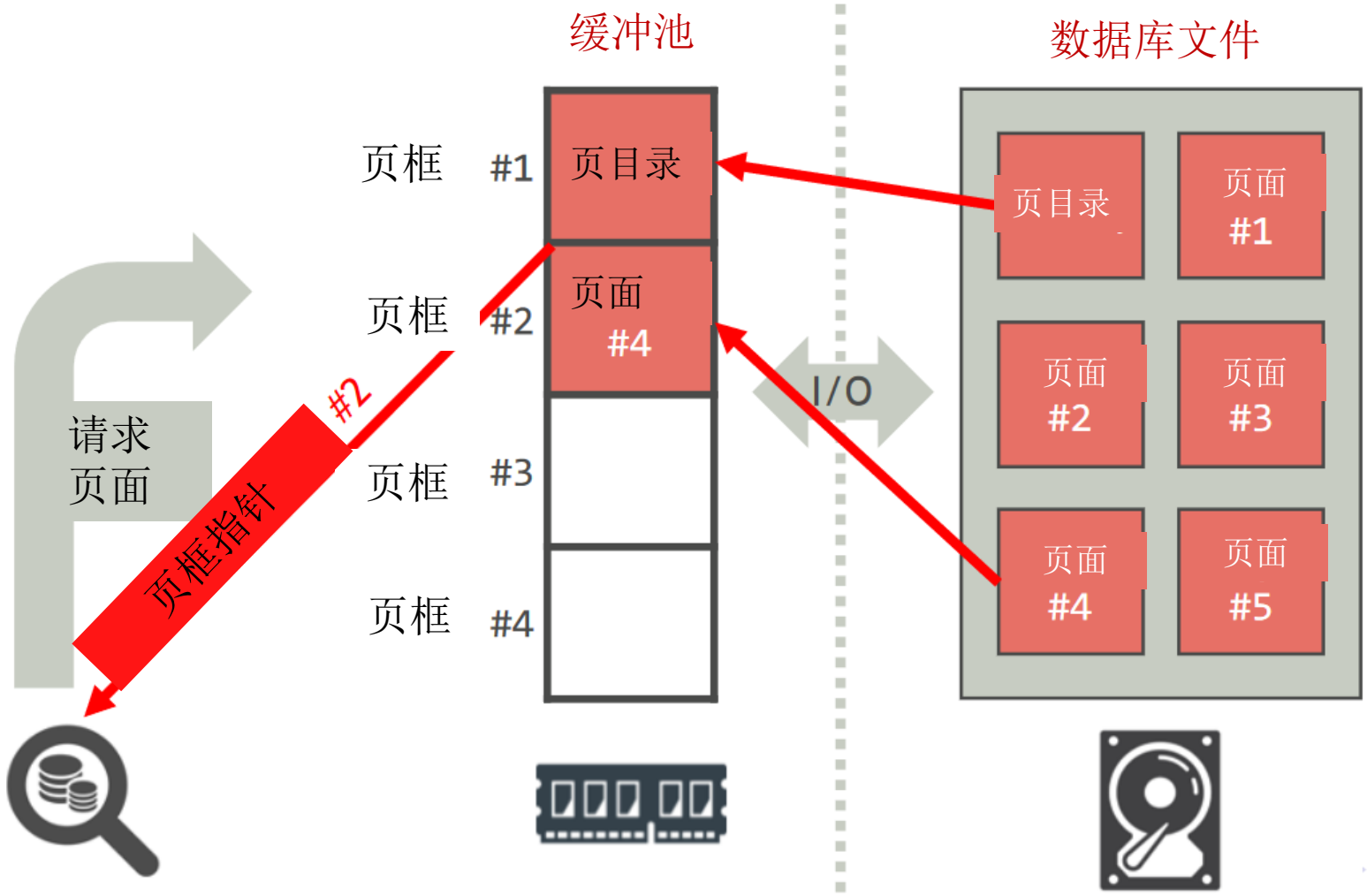
- 可用的内存区域被划分成一个固定大小页面的数组，这些页面统称为缓冲池（buffer pool）
- 缓冲池中的页面被称为页框（frame）



缓冲区管理-缓冲区管理器:

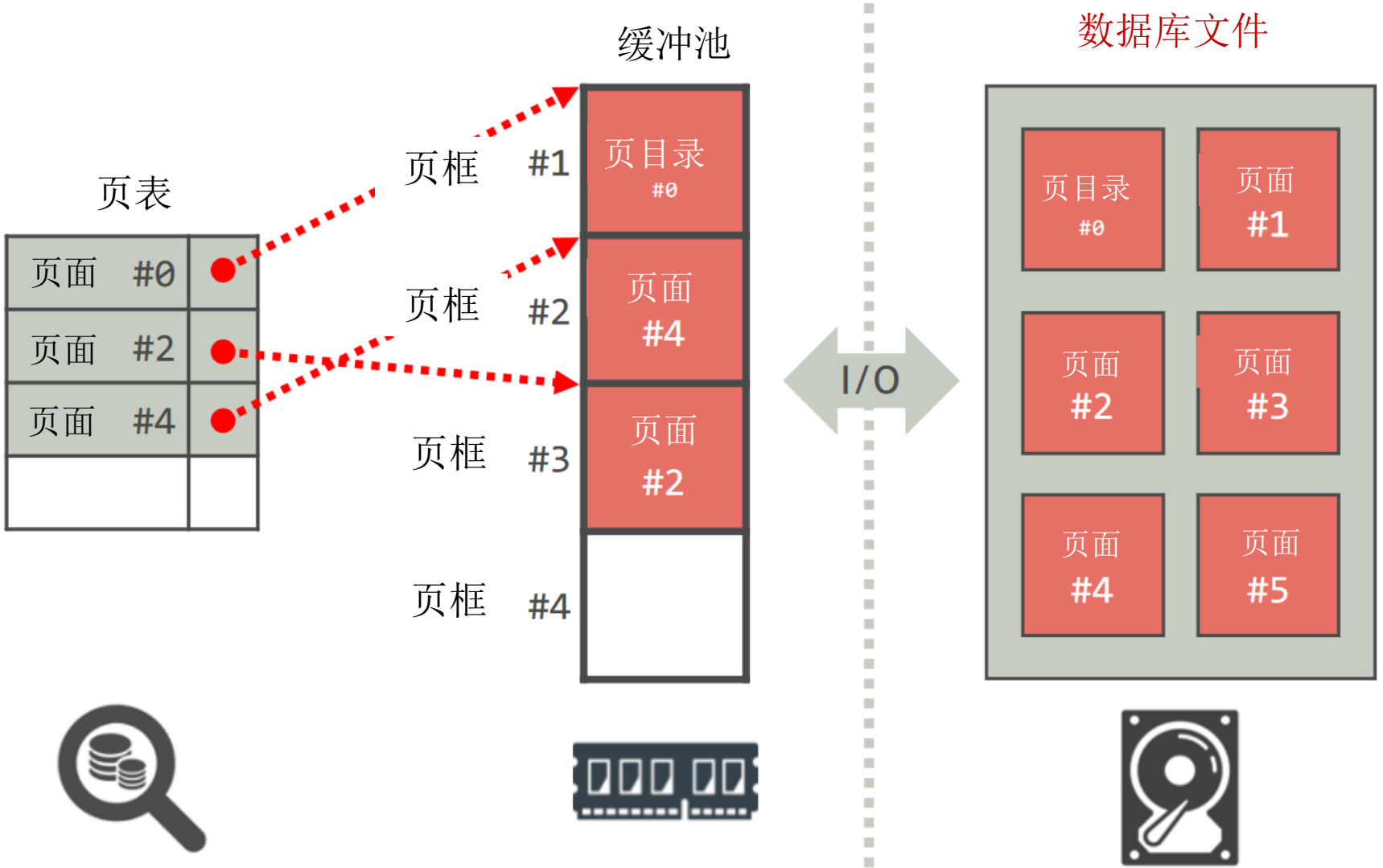
缓冲区管理器负责根据需要将页面page带入缓冲池

■如果缓冲池已满，缓冲管理器决定替换缓冲池中哪一个page来容纳新page



缓冲区管理-页表

页表跟踪在缓冲池内的页Page



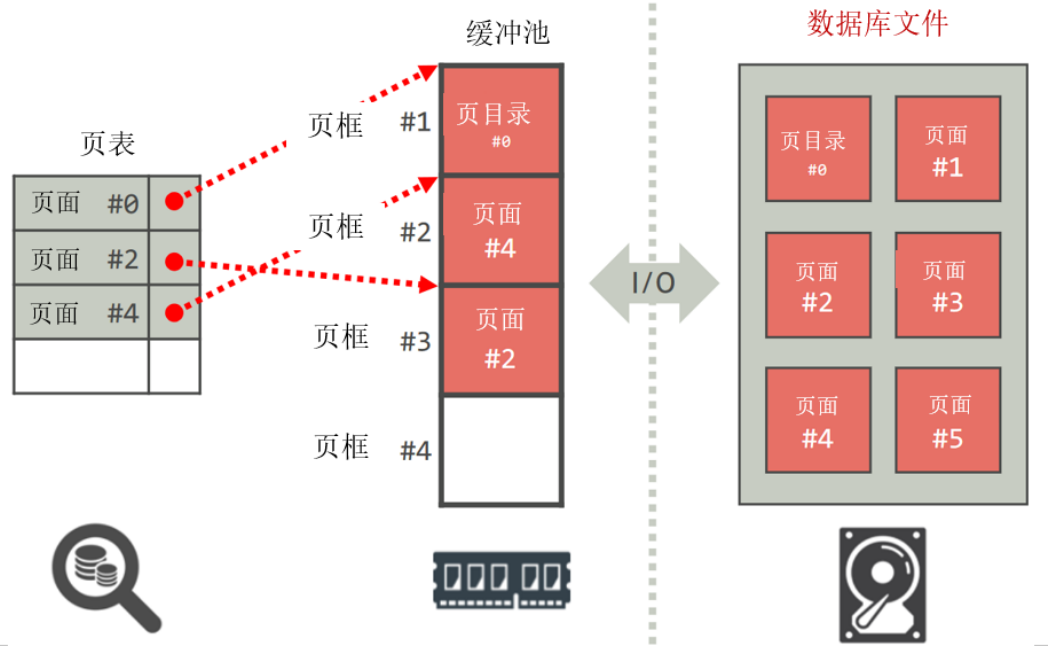
缓冲区管理-页表VS页目录

页目录：将页面ID映射到数据库文件中页面位置的映射表

- 所有的更改必须记录在磁盘上，以方便DBMS在重启时能够找到

页表：将页面ID映射到缓冲池页框中页面副本的映射表

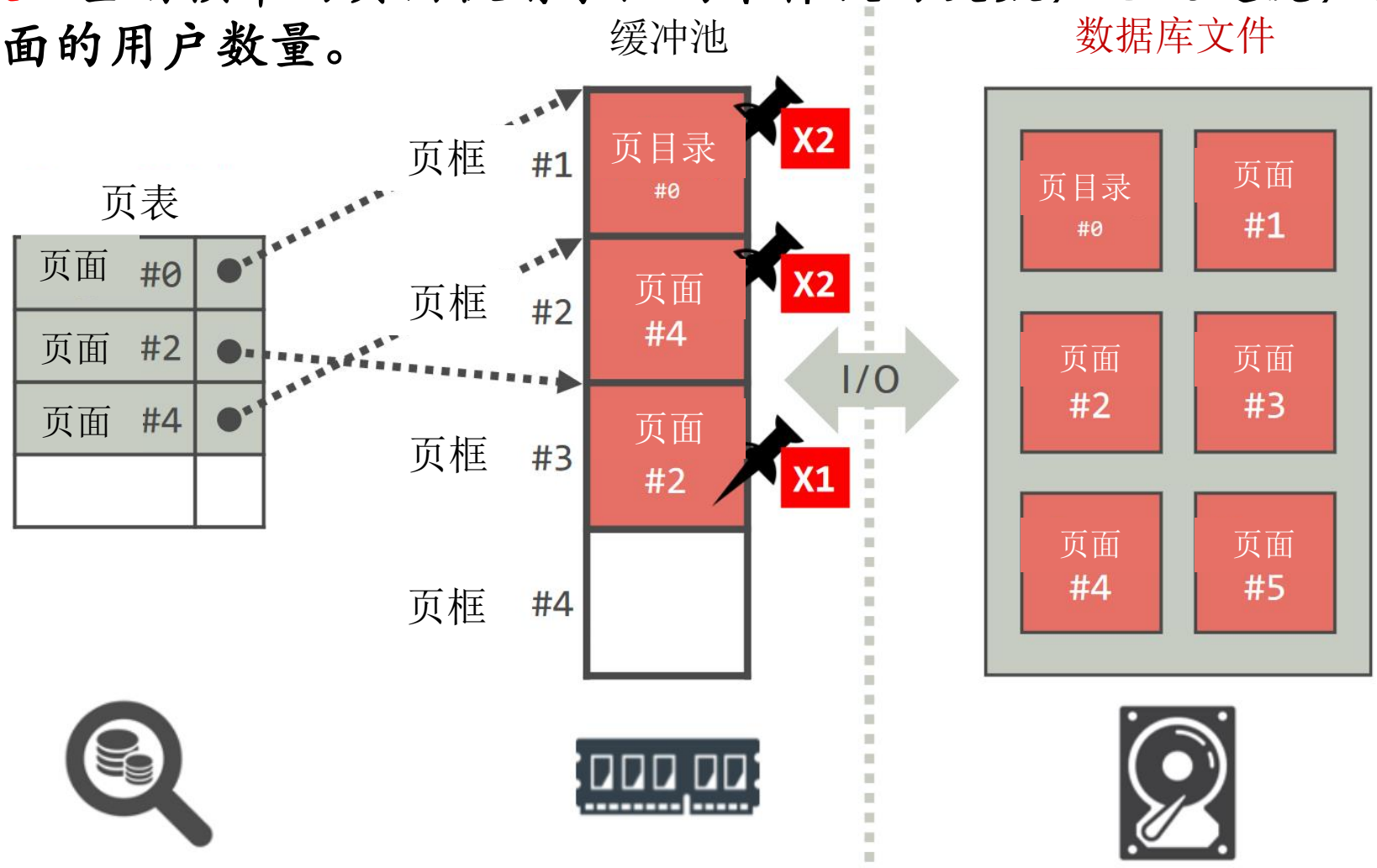
- 是内存中数据结构，不需要存储在磁盘上



缓冲区管理-缓冲池页框元数据

缓冲区管理器为每个页框维护两个变量：

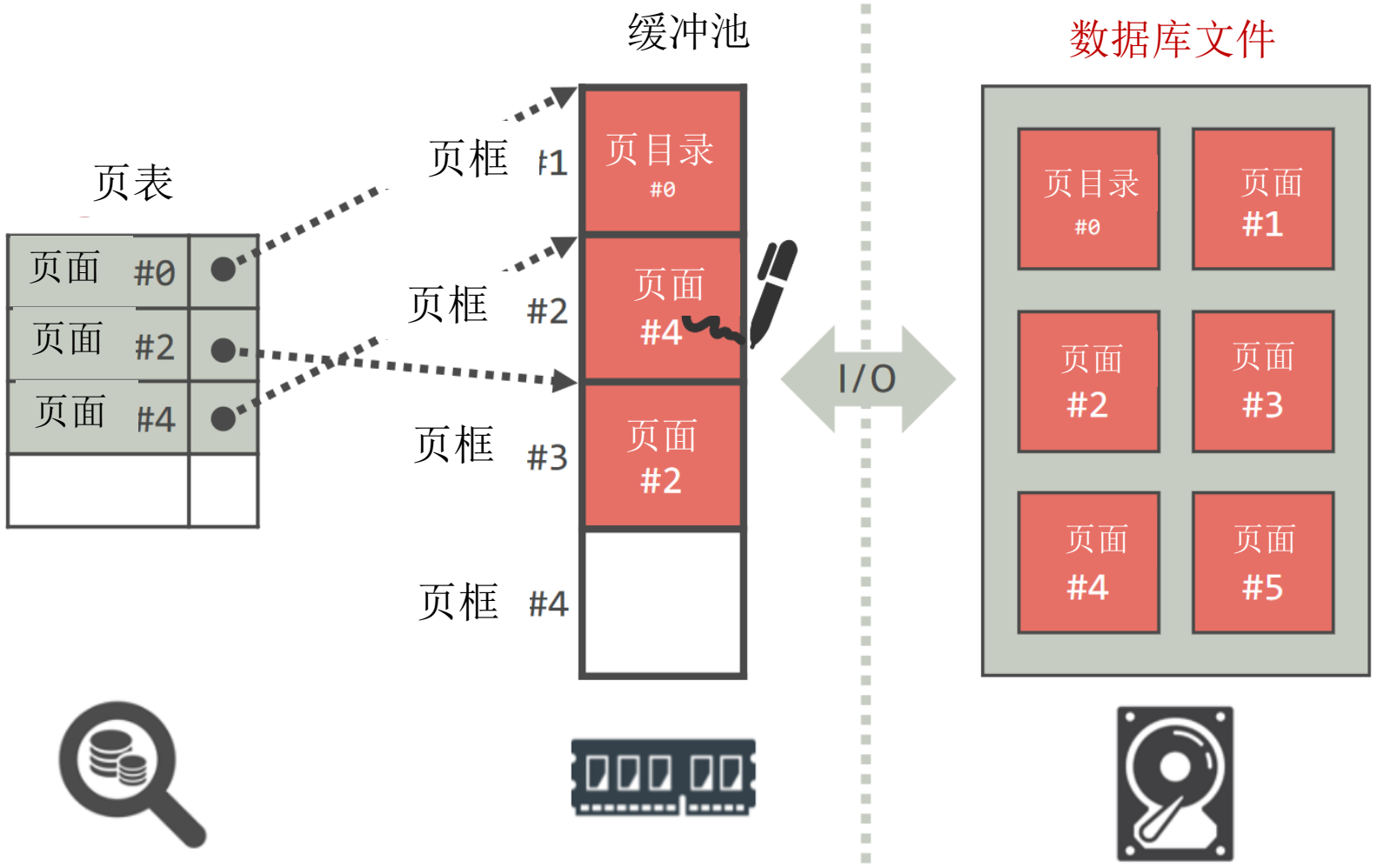
- **pin计数**：当前帧中的页面被请求但尚未释放的次数，也就是说，当前使用该页面的用户数量。



缓冲区管理-缓冲池：页框元数据

缓冲区管理器为每个页框维护两个变量：

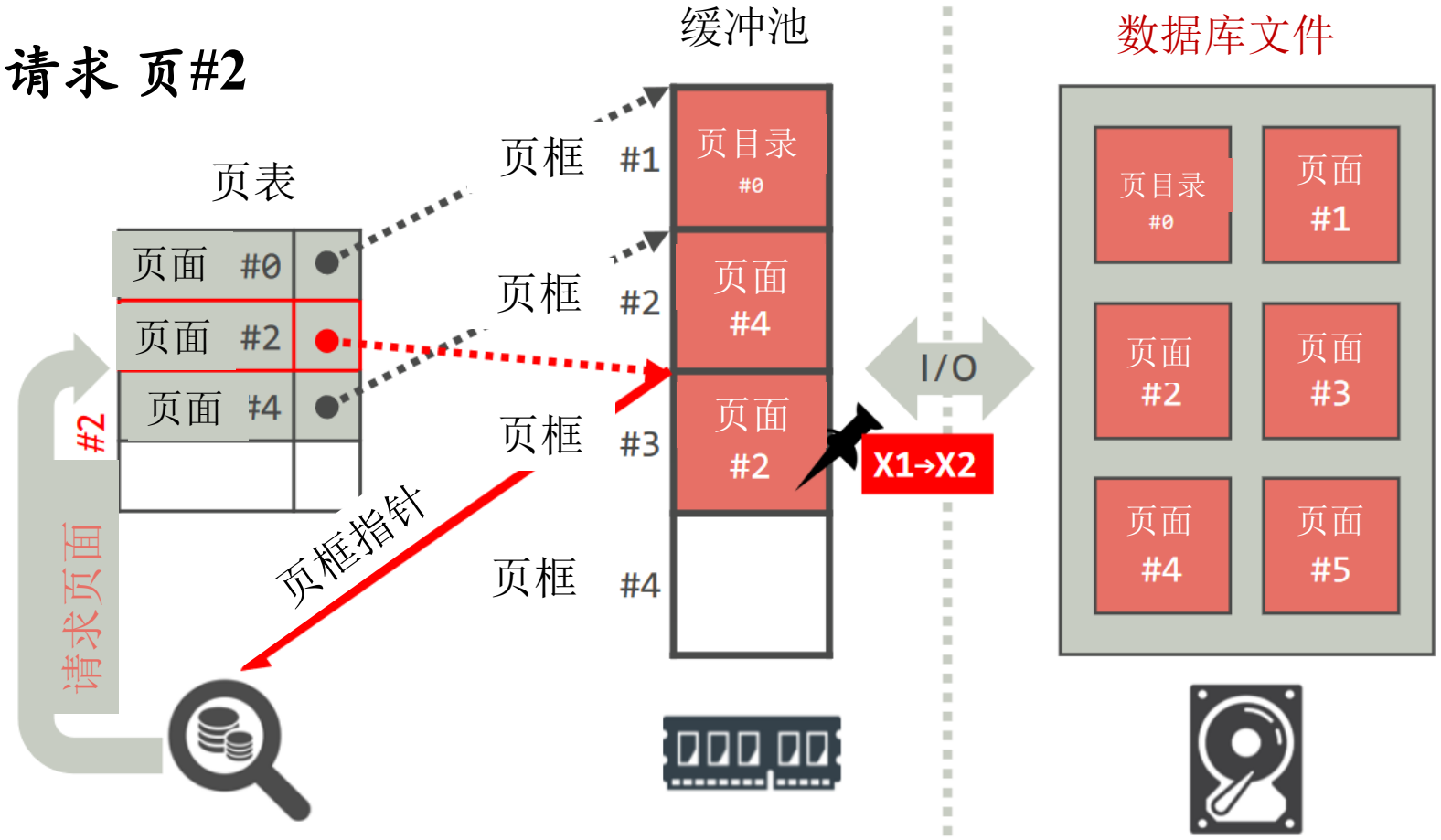
- **dirty**：自从页面被带入缓冲池以来，该页面是否已被修改的状态。



请求页（Page Requests）

① 检查页表，看看某个帧是否包含所请求的页面 P
② 如果 P 在缓冲池中，则 pin 页面 P，即增加包含 P 的帧的 pin 计数
③ 返回包含 P 的帧的指针

➢ 例如: 请求 页#2



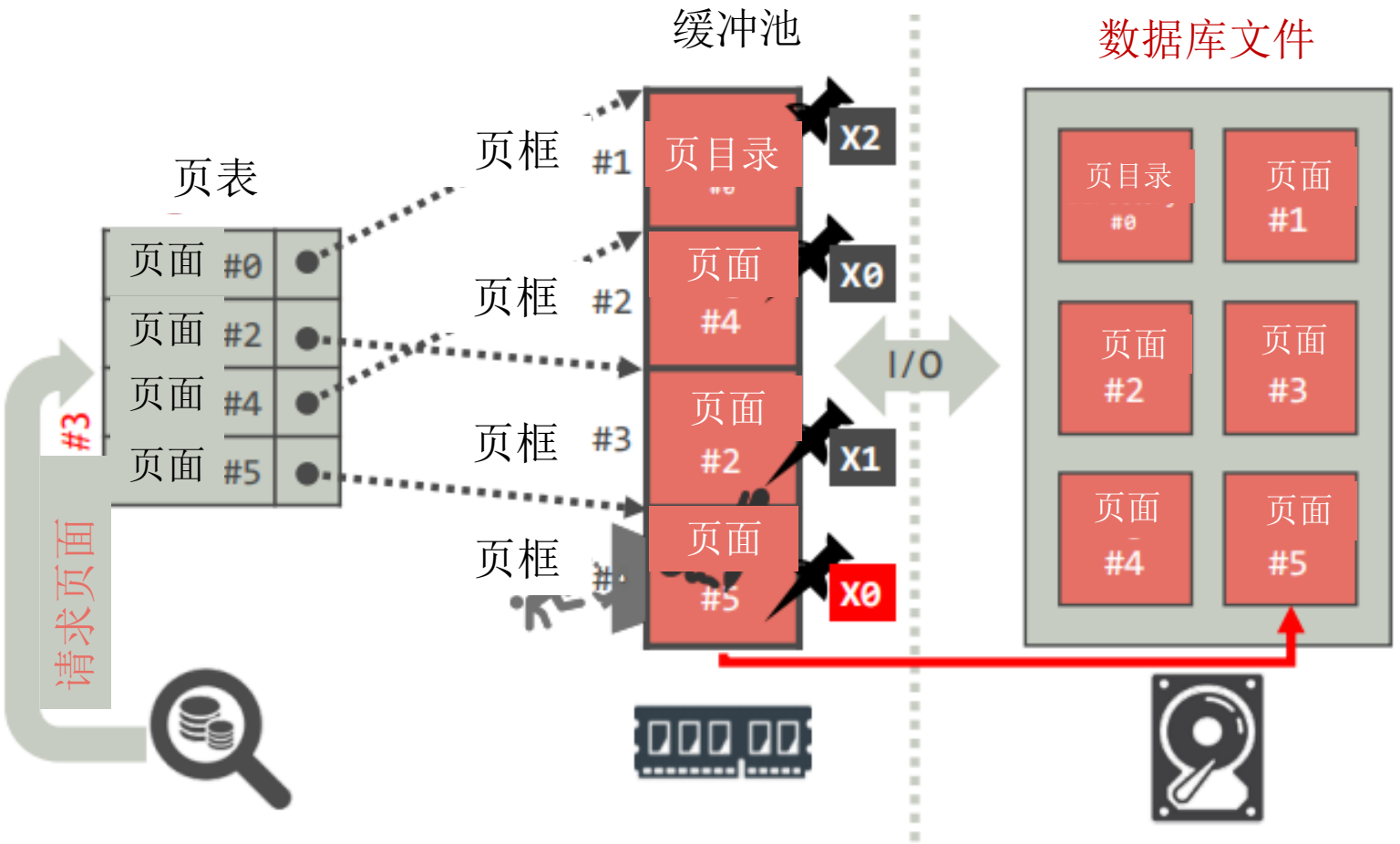
请求页（Page Requests）

如果请求的页面不在缓冲池中，

① （选替换页）选择一个 pin_count 为0的帧进行替换，使用替换策略，
并增加其 pin_count
② （写回脏页）如果替换帧的脏位是开启的，则将其包含的页面写入磁盘
③ （读请求页）将请求的页面读入替换框架

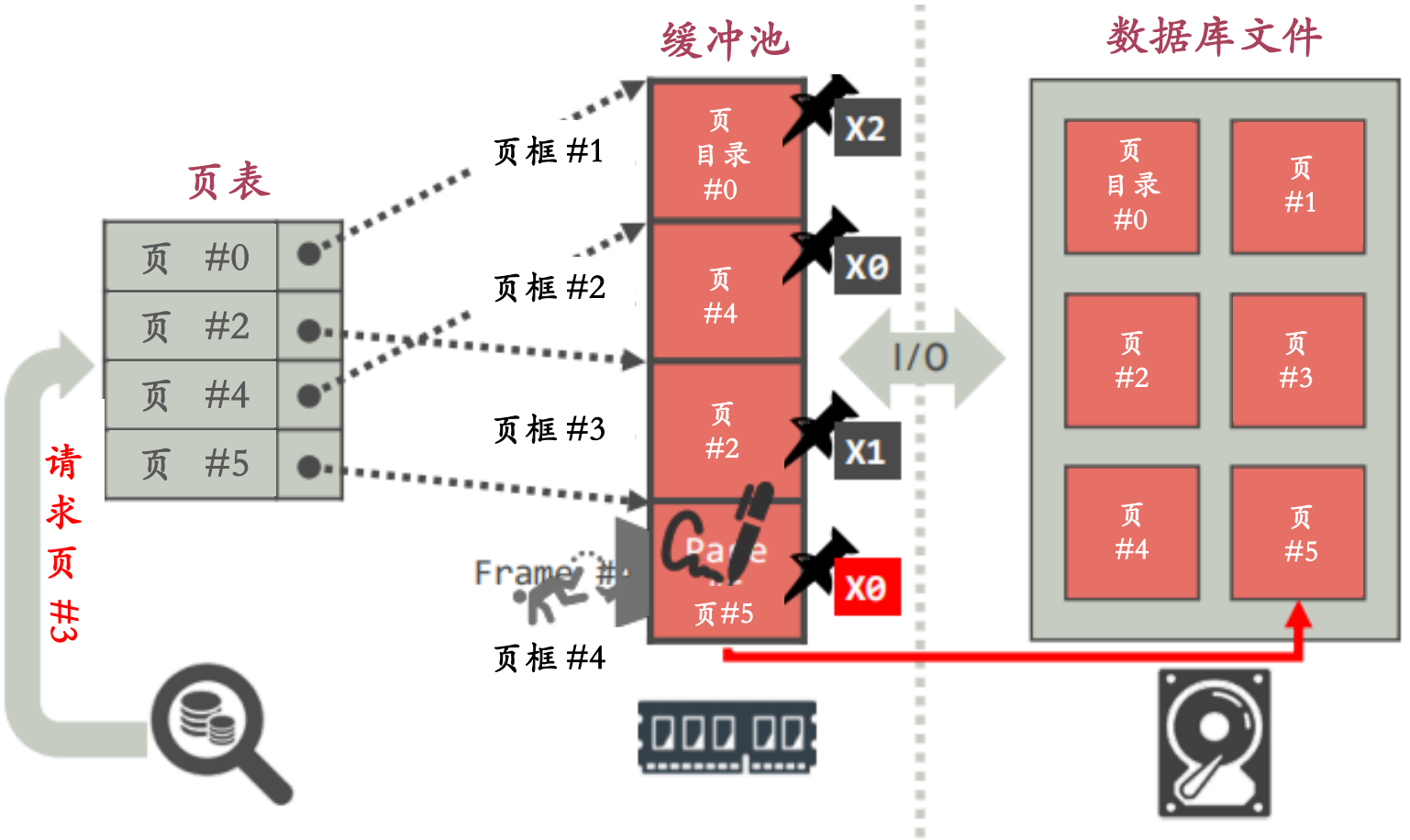
请求页（Page Requests）

- 例如: 请求 页#3



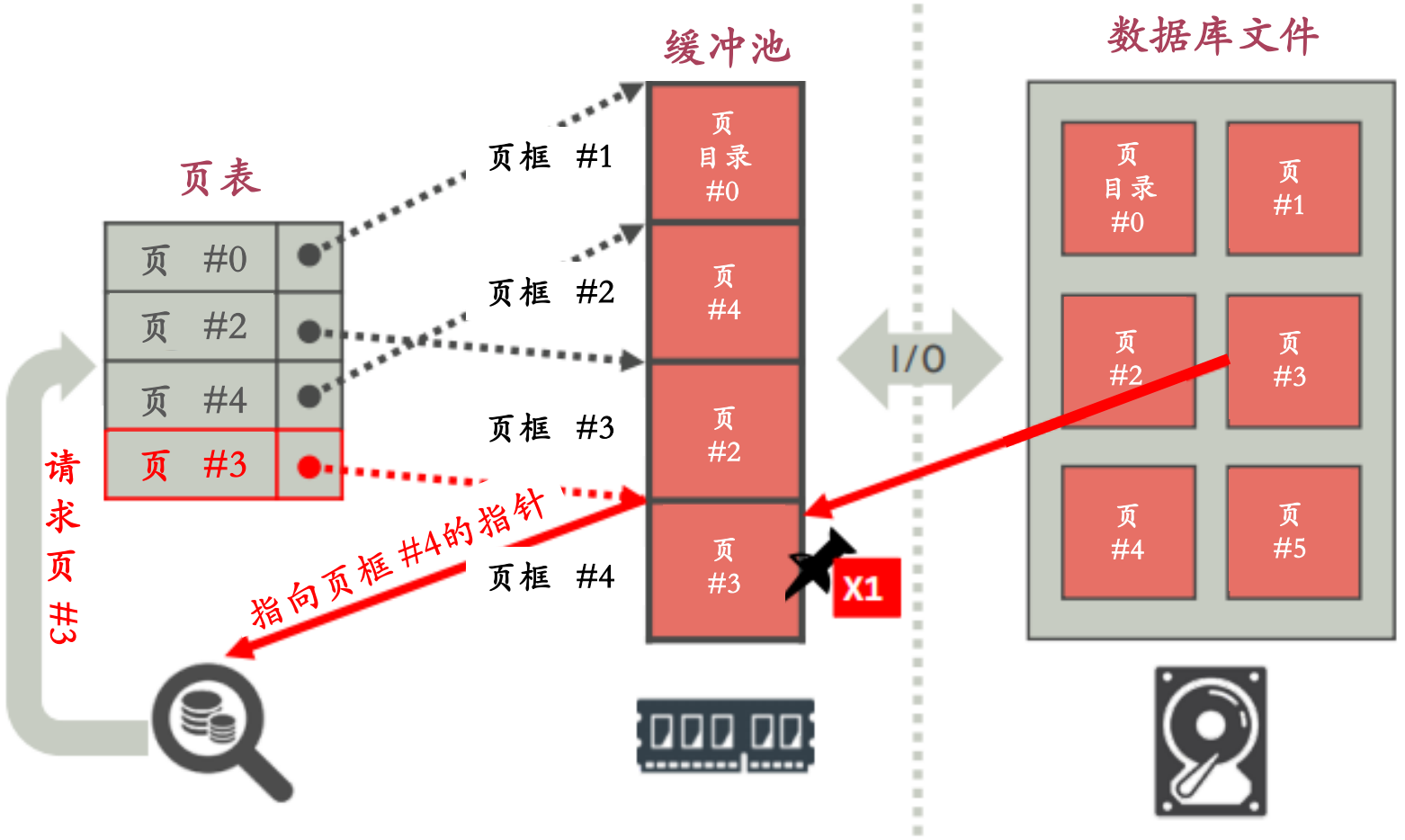
请求页（Page Requests）

- 例如: 请求 页#3



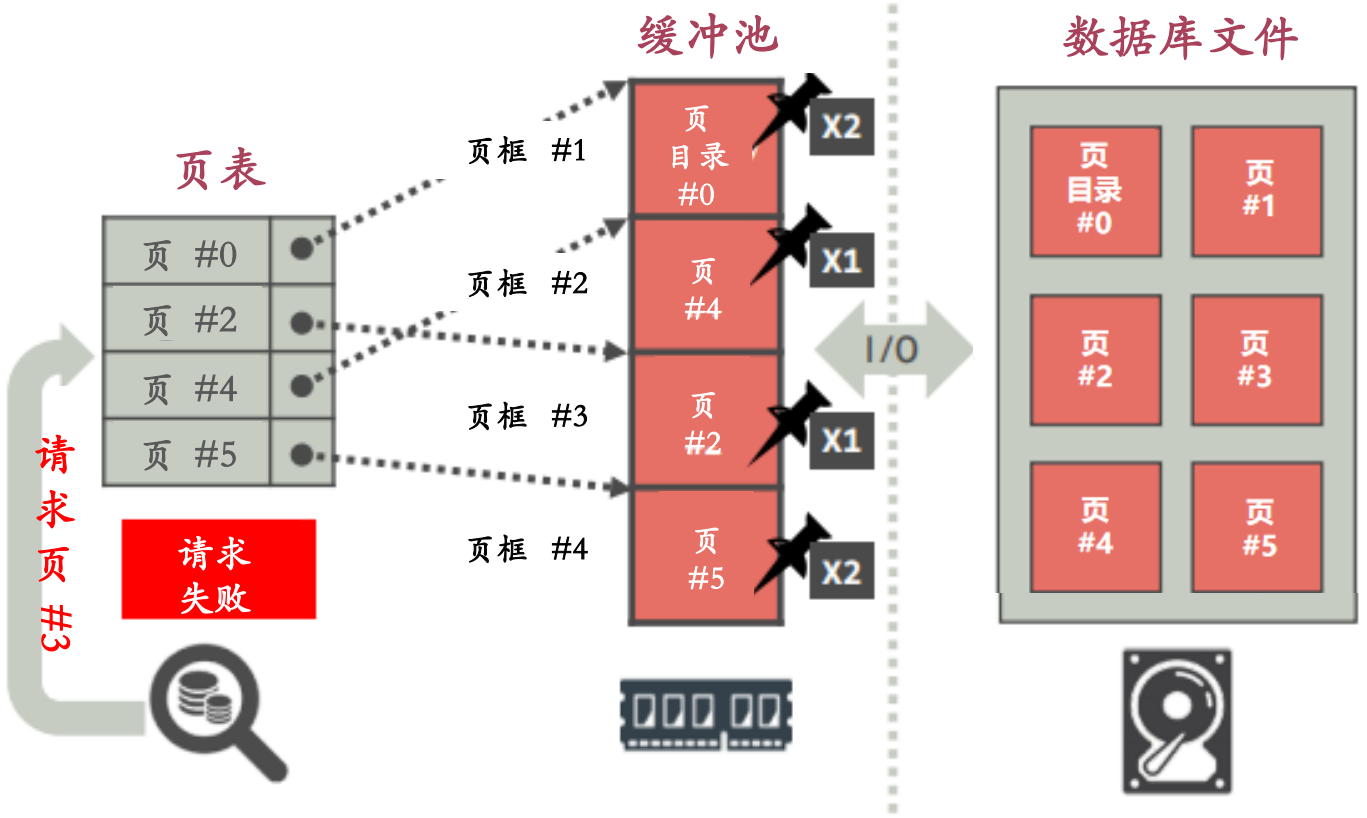
请求页（Page Requests）

- 例如: 请求 页#3



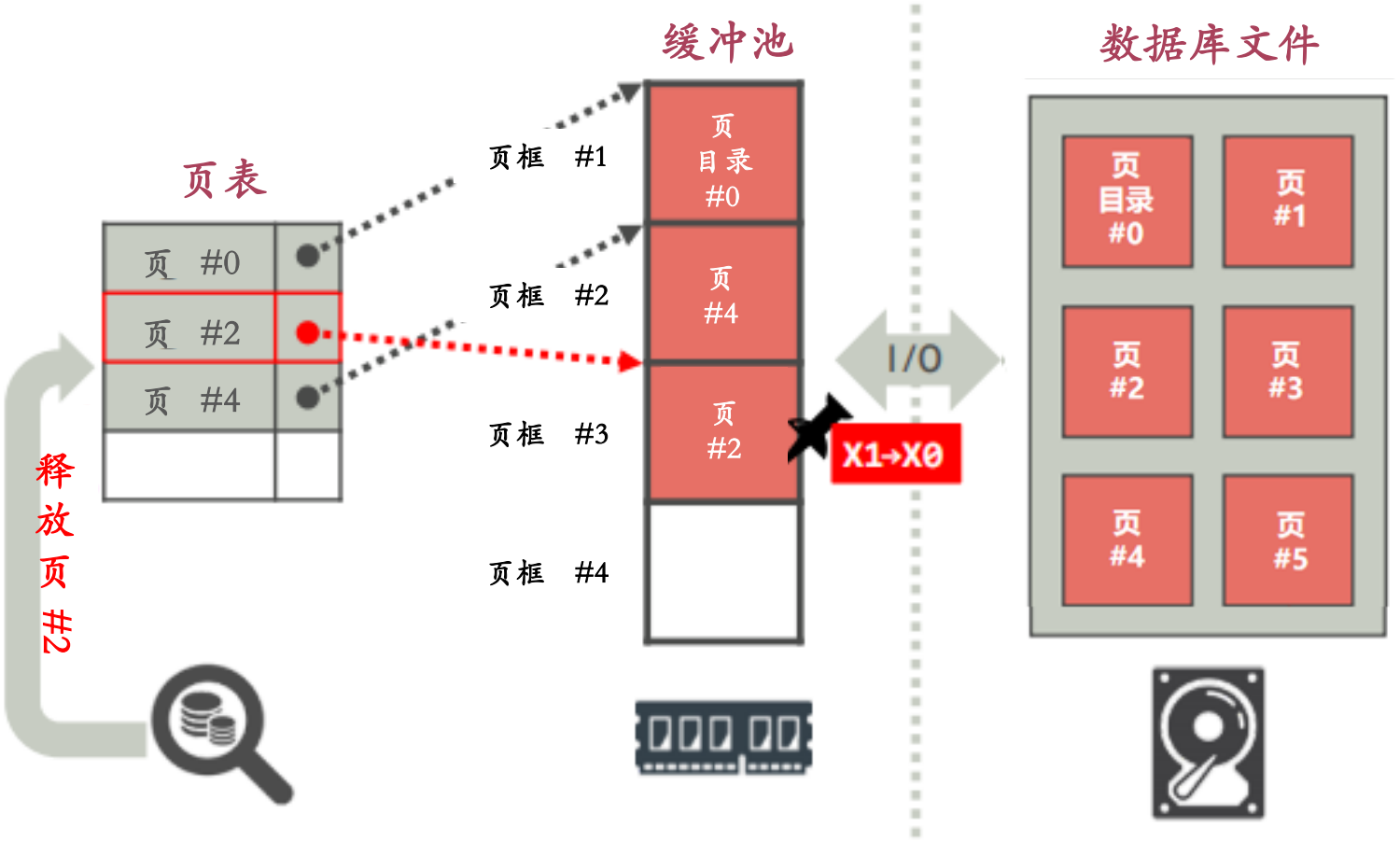
请求页（Page Requests）

- 如果缓冲池中没有页面的 pin_count 等于 0，缓冲管理器必须等待某些页面被释放后才能响应页面请求。
- 实际上，请求该页面的事务可能会直接被中止。



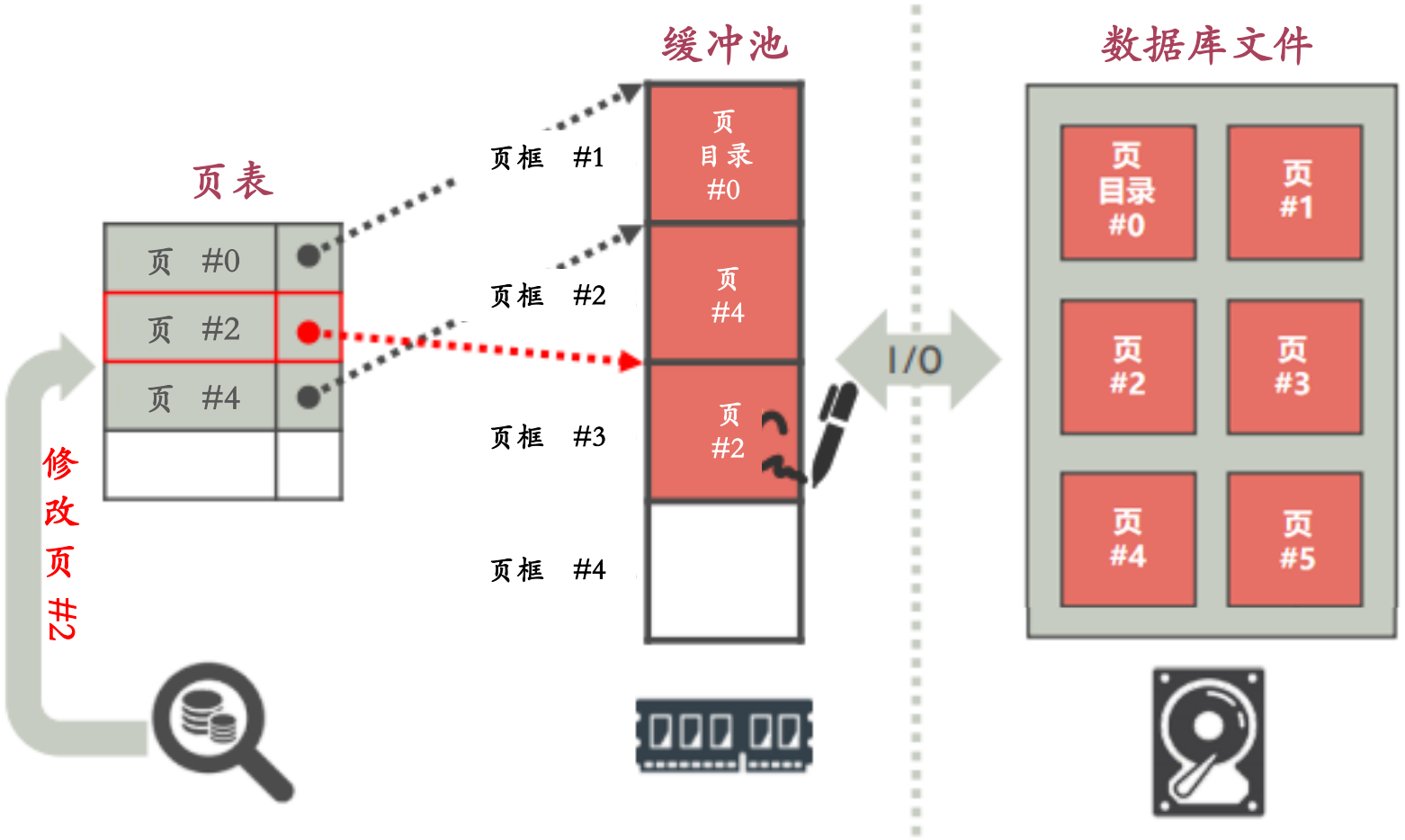
页释放（Page Releases）

- 解除已释放页面的固定状态，即减少包含该页面的帧的 pin_count。
- 示例：释放页面 #2。



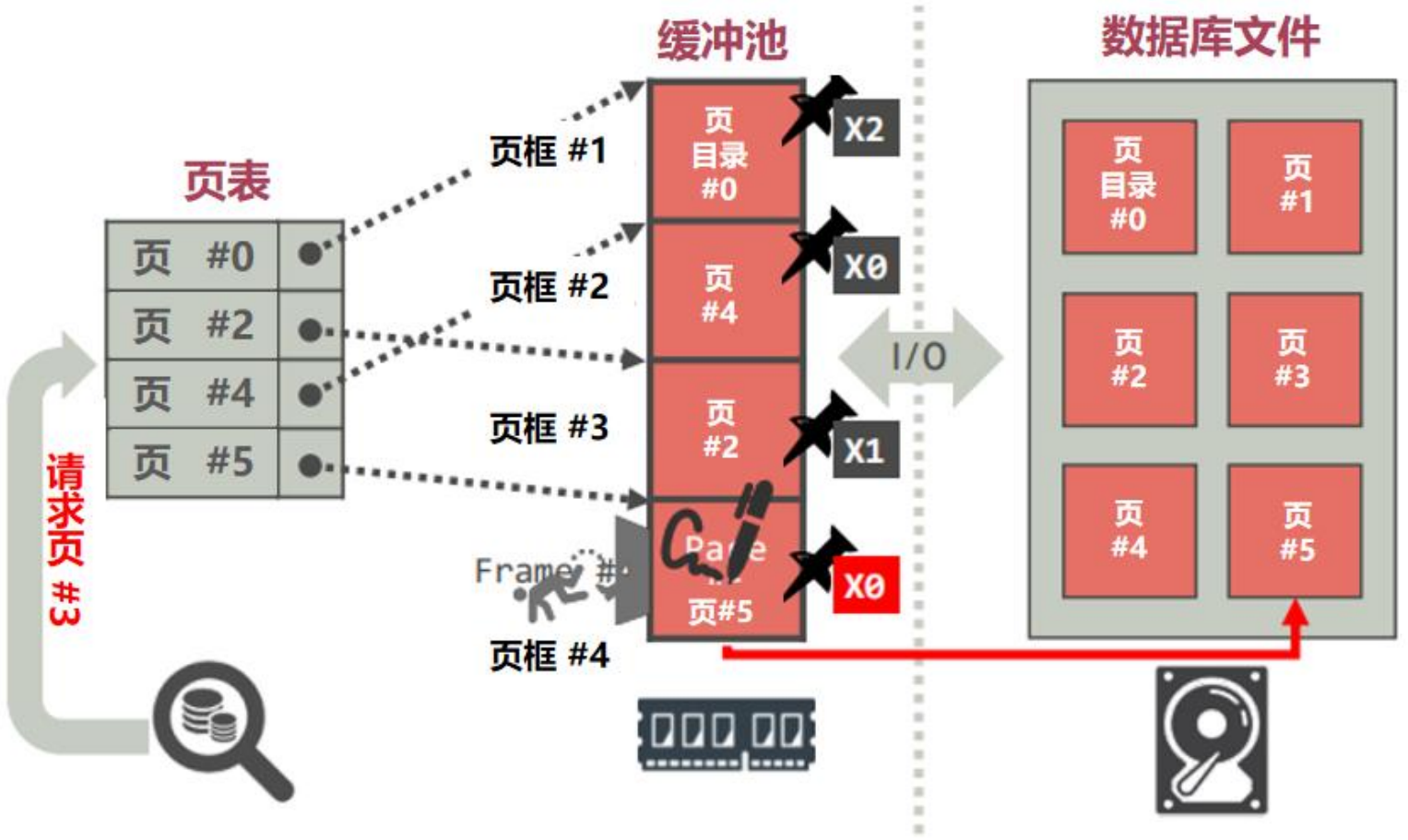
页修改（Page Modifications）

- 把包含被修改页面的帧设置脏位（dirty bit）。
- 示例：修改页面 #2。



页替换策略(Page Replacement Policies)

- 当缓冲管理器需要释放一个帧以腾出空间容纳新页面时，必须决定从缓冲池中驱逐哪个页面。
- 页面置换策略会显著影响数据库操作所需的时间。



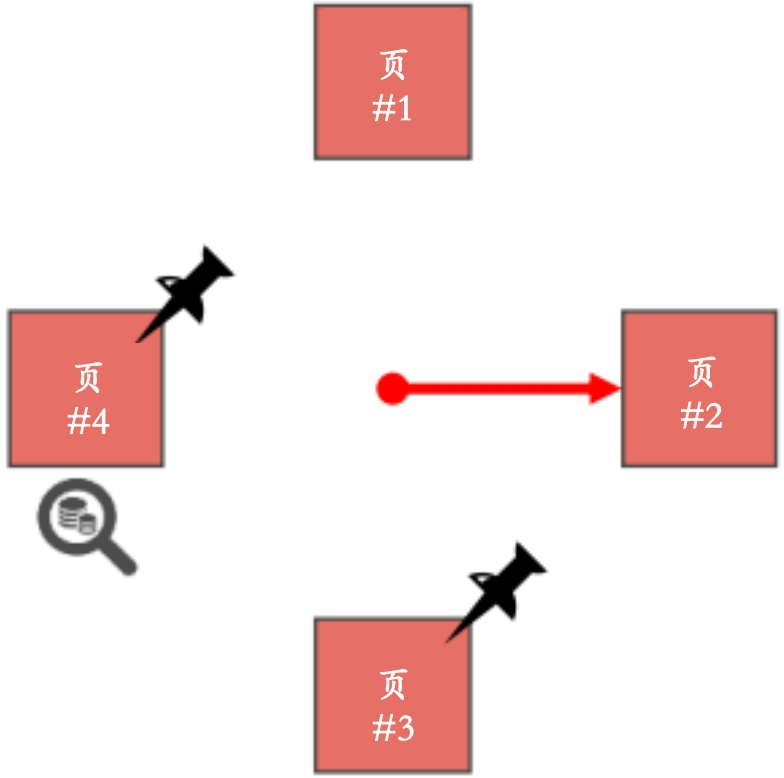
最近最少使用（LRU）置换策略：

- 维护每个页面上次访问时的时间戳。

- 当数据库管理系统（DBMS）需要驱逐页面时，选择时间戳最早的页面。
  - 将页面按顺序排列，以减少驱逐时的搜索时间。

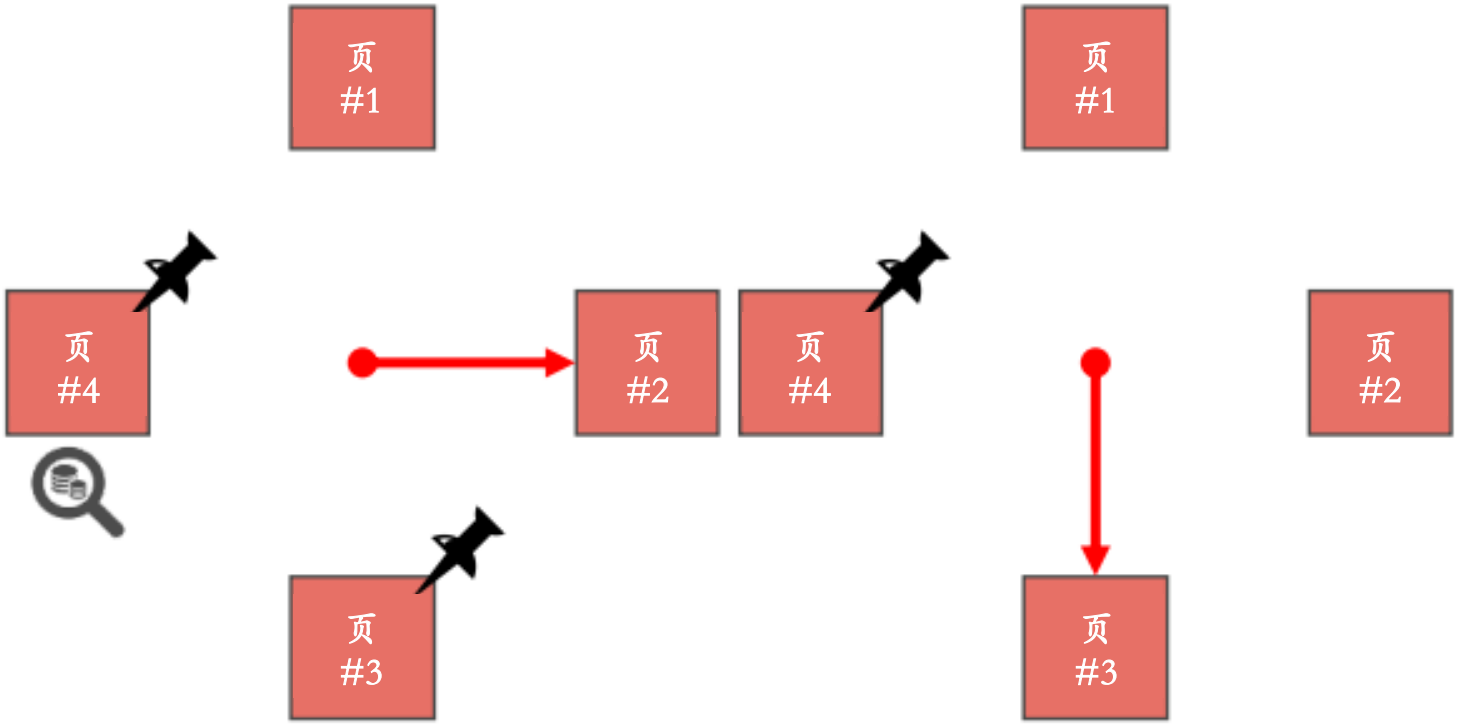
时钟替换策略（Clock Replacement Policy）

- 时钟置换策略是最近最少使用（LRU）的一种近似方法，它不需要为每个页面设置单独的时间戳。
  - 每个帧都有一个引用位，值为 0 或 1。
  - 当一个页面被读取到帧中，或者帧中的页面被访问时，它的引用位会被设置为 1。



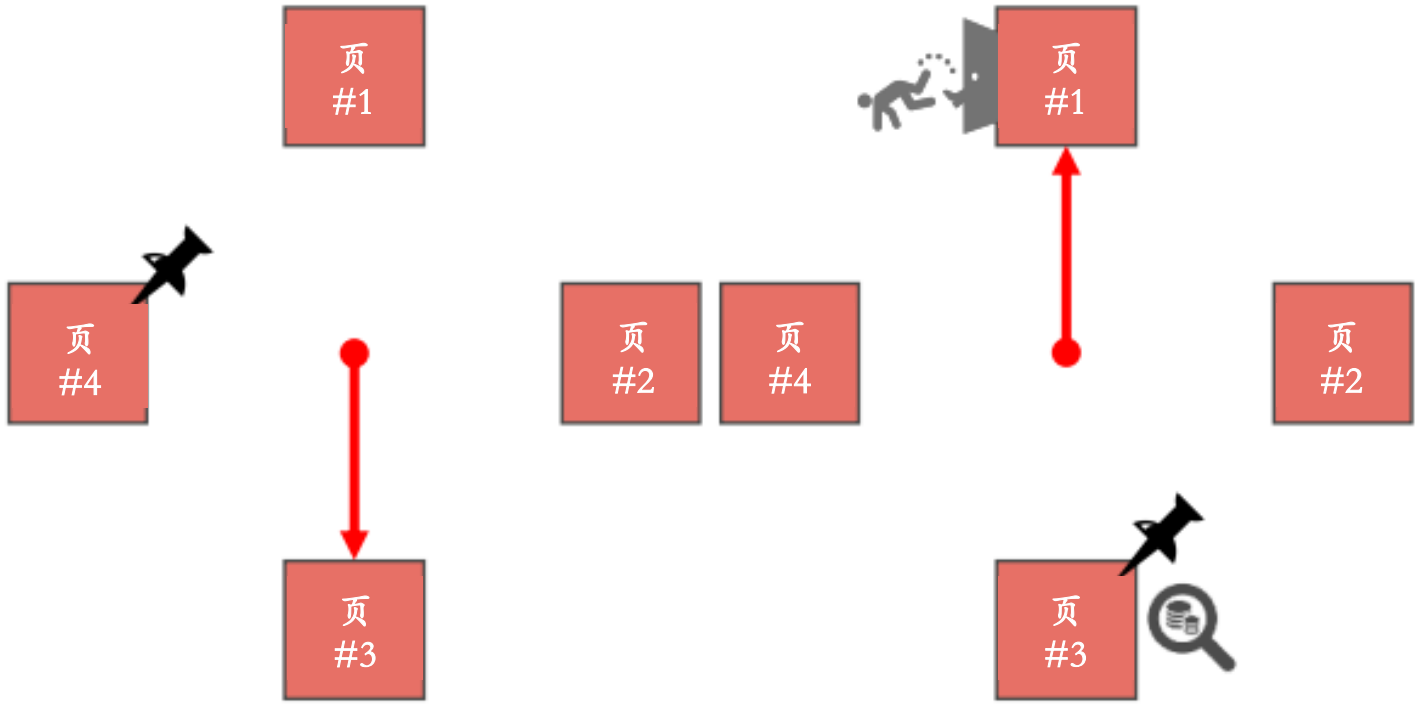
时钟替换策略（Clock Replacement Policy）

- 这些帧被组织成一个环形缓冲区，并有一个“时钟指针”。
  - 时钟指针顺时针旋转。
  - 当指针扫过时，检查页面的引用位是否为 1。
  - 如果是，则将引用位设置为 0。



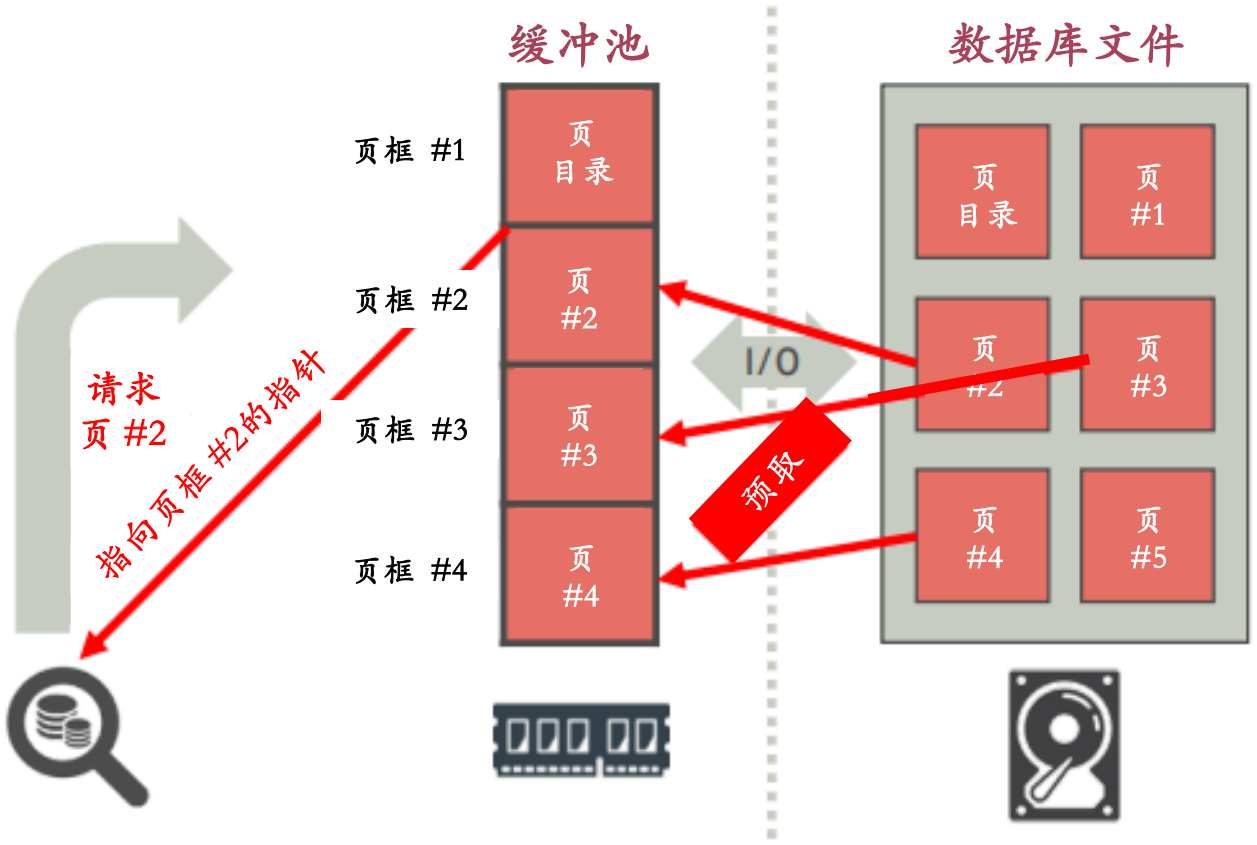
时钟替换策略（Clock Replacement Policy）

- 当缓冲管理器需要为新页面提供帧时
  - 寻找第一个引用位为 0 的帧，
  - 驱逐该帧中的页面。



预取（Prefetching）

➢ 为了更高效地处理页面请求，缓冲管理器通常会预取即将被请求的页面。

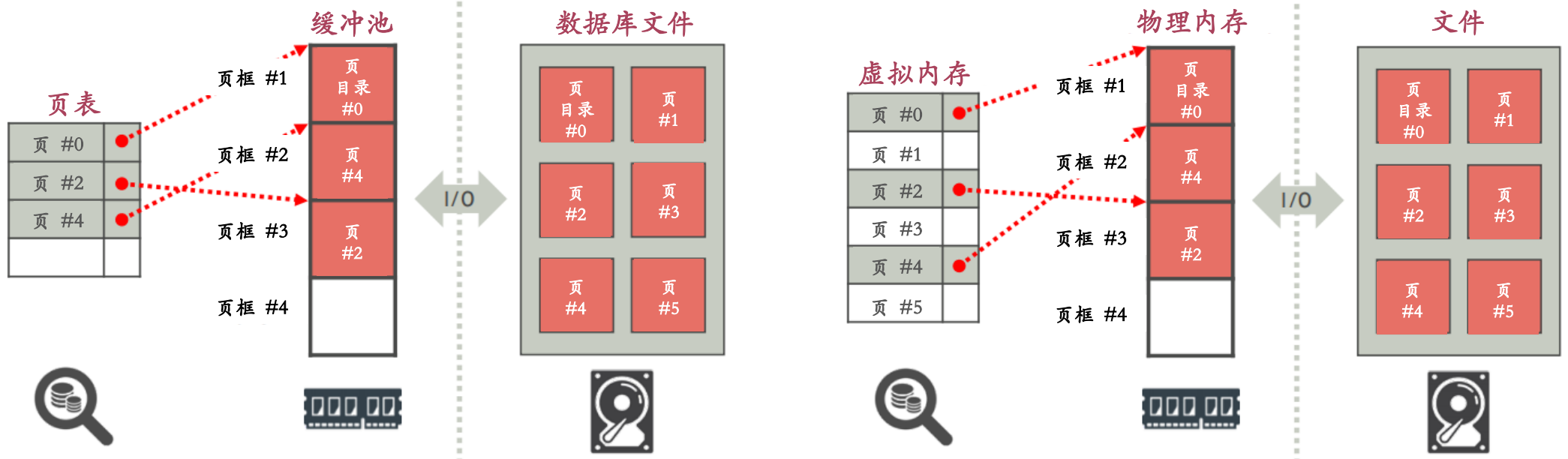


缓冲池 VS 虚拟内存

相似之处

- 相似的目标：两者都旨在提供访问更多数据的能力，这些数据无法完全放入主内存中。
- 相似的操作：两者都根据需要从磁盘将页面加载到主内存，并替换主内存中不再需要的页面。

➢ 为什么我们不能使用操作系统的虚拟内存能力来构建一个数据库管理系统（DBMS）？



总结（Summary）

1. 存储介质（Storage Media）
2. 数据库在磁盘上的表示(Representation of Databases on Disks）
   - 元组布局（Tuple Layout）
   - 页面布局（Page Layout）
     - 面向元组的页面布局（Tuple-Oriented Page Layout）
   - 文件组织（File Organization）
3. 缓冲区管理（Buffer Management）