第五章 存储管理

章成源

湖南大学-信息科学与工程学院-计算机科学系

办公室：院楼403

Email：cyzhangcse@hnu.edu.cn

第5章 存储管理

- 5.1 物理存储系统
- 5.2 数据组织
- 5.3 元数据存储
- 5.4 缓冲区管理
- 5.5 小结

第5章 存储管理

- 5.1 物理存储系统
- 5.2 数据组织
- 5.3 元数据存储
- 5.4 缓冲区管理
- 5.5 小结

5.1 物理存储系统

- 5.1.1 存储介质概述
- 5.1.2 存储介质-磁盘
- 5.1.3 磁盘I/O性能的提升策略

5.1 物理存储系统

- 5.1.1 存储介质概述
- 5.1.2 存储介质-磁盘
- 5.1.3 磁盘I/O性能的提升策略

存储介质概述

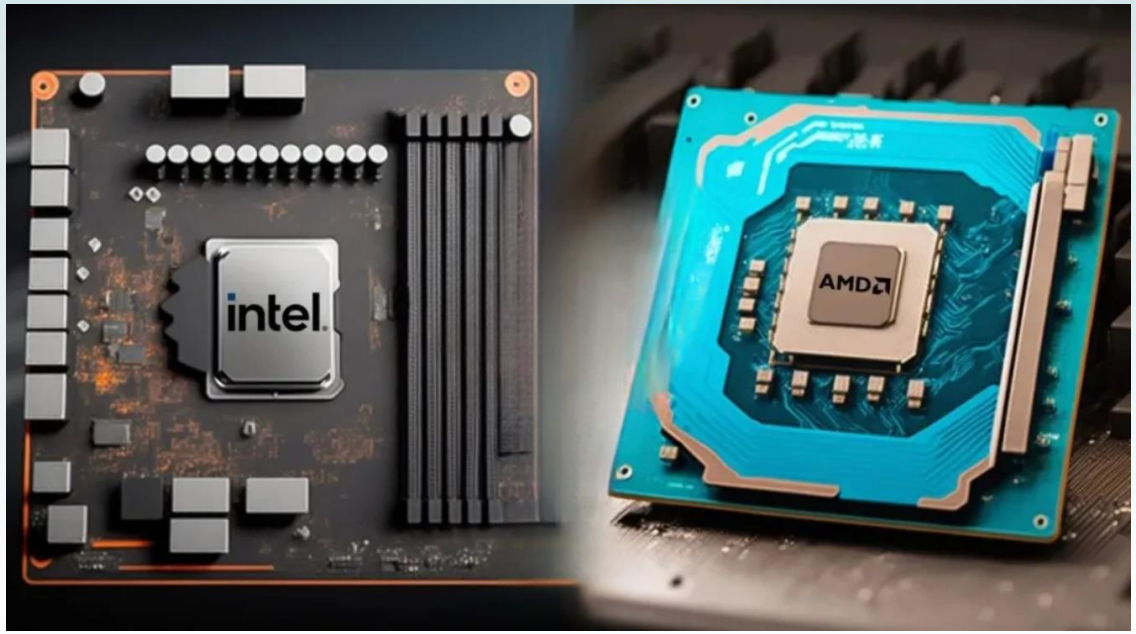
计算机系统中代表性的存储介质包括：

- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage）等

存储介质概述

计算机系统中代表性的存储介质包括：

- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage）等



- 高速缓冲存储器是存在于主存与CPU之间的一级存储器， 由静态存储芯片(SRAM)组成，容量比较小但速度比主存高得多， 接近于CPU的速度。在计算机存储系统的层次结构中，是介于中央处理器和主存储器之间的高速小容量存储器。它和主存储器一起构成一级的存储器。高速缓冲存储器和主存储器之间信息的调度和传送是由硬件自动进行的。

存储介质概述

计算机系统中代表性的存储介质包括：

- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage）等

主存储器（Main memory），简称主存。是计算机硬件的一个重要部件，其作用是存放指令和数据，并能由中央处理器（CPU）直接随机存取。现代计算机是为了提高性能，又能兼顾合理的造价，往往采用多级存储体系。即由存储容量小，存取速度高的高速缓冲存储器，存储容量和存取速度适中的主存储器是必不可少的。主存储器是按地址存放信息的，存取速度一般与地址无关。

存储介质概述

计算机系统中代表性的存储介质包括：

- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage）等

固态硬盘(Solid State Drives)，简称固盘，固态硬盘(Solid State Drive)用固态电子存储芯片阵列而制成的硬盘，由控制单元和存储单元(FLASH芯片、DRAM芯片)组成。固态硬盘在接口的规范和定义、功能及使用方法上与传统硬盘的完全相同，在产品外形和尺寸上也完全与传统硬盘一致，但I/O性能相对于传统硬盘大大提升。被广泛应用于军事、车载、工控、视频监控、网络监控、网络终端、电力、医疗、航空、导航设备等领域。

存储介质概述

计算机系统中代表性的存储介质包括：

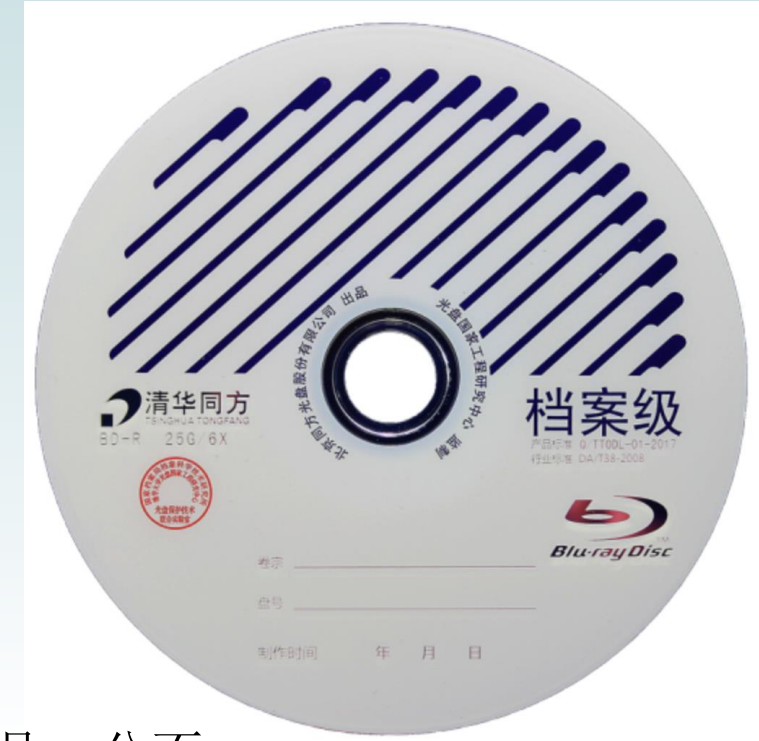
- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage）等

磁盘（disk）是指利用磁记录技术存储数据的存储器。磁盘是计算机主要的存储介质，可以存储大量的二进制数据，并且断电后也能保持数据不丢失。早期计算机使用的磁盘是软磁盘（Floppy Disk，简称软盘），如今常用的磁盘是硬磁盘（Hard disk，简称硬盘）。

存储介质概述

计算机系统中代表性的存储介质包括：

- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage）等



光盘以光信息做为存储物的载体，用来存储数据的一种物品。分不可擦写光盘，如CD-ROM、DVD-ROM等;和可擦写光盘，如CD-RW、DVD-RAM等。

光盘是利用激光原理进行读、写的设备，是迅速发展的一种辅助存储器，可以存放各种文字、声音、图形、图像和动画等多媒体数字信息。

存储介质概述

计算机系统中代表性的存储介质包括：

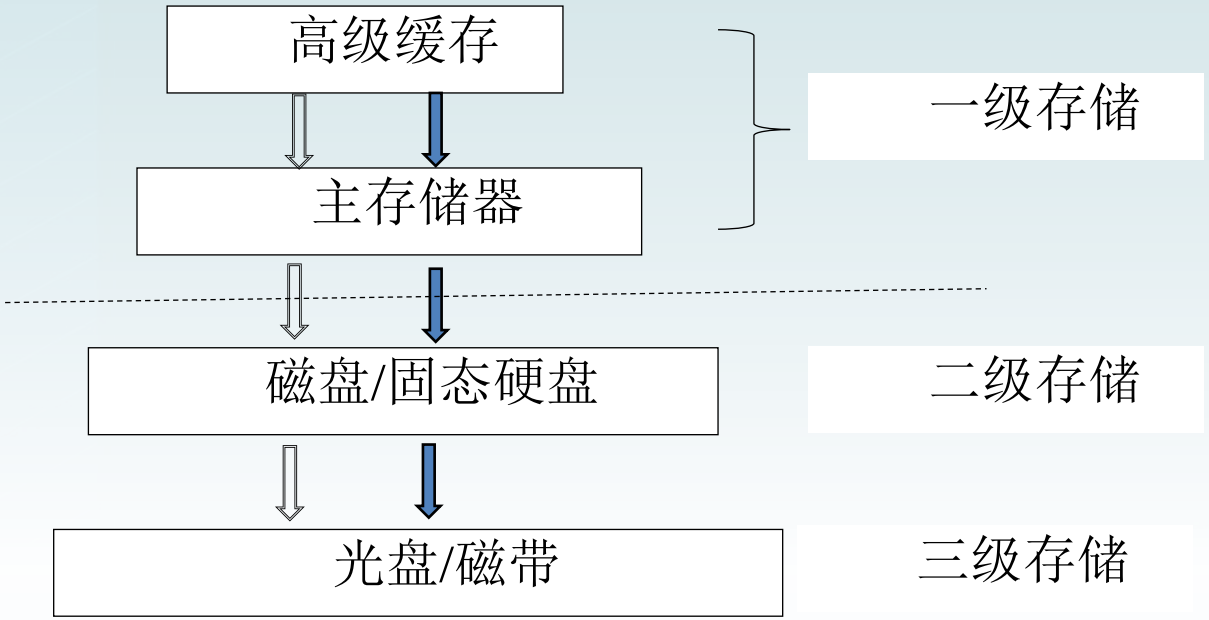
- 高速缓存（cache）
- 主存储器（main memory）
- 固态硬盘（solid-state drive）
- 磁盘（magnetic-disk storage）
- 光盘（optical storage）
- 磁带（tape storage） 等

磁带是一种用于记录声音、图像、数字或其他信号的载有磁层的带状材料，是产量最大和用途最广的一种磁记录材料。通常是在塑料薄膜带基（支持体）上涂覆一层颗粒状磁性材料或蒸发沉积上一层磁性氧化物或合金薄膜而成。曾使用纸和赛璐珞等作带基，现主要用强度高、稳定性好和不易变形的聚酯薄膜。

存储介质概述

- 三级存储架构
  - 层次越高，价格越贵，速度越快。
  - 高速缓存和主存储器称为一级存储。
  - 磁盘/固态硬盘通常用于联机存储数据，称为二级存储。
  - 光盘和磁带用于脱机存储，称为三级存储。
  - 虚线以上是易失存储介质，虚线以下是非易失（持久）存储介质。

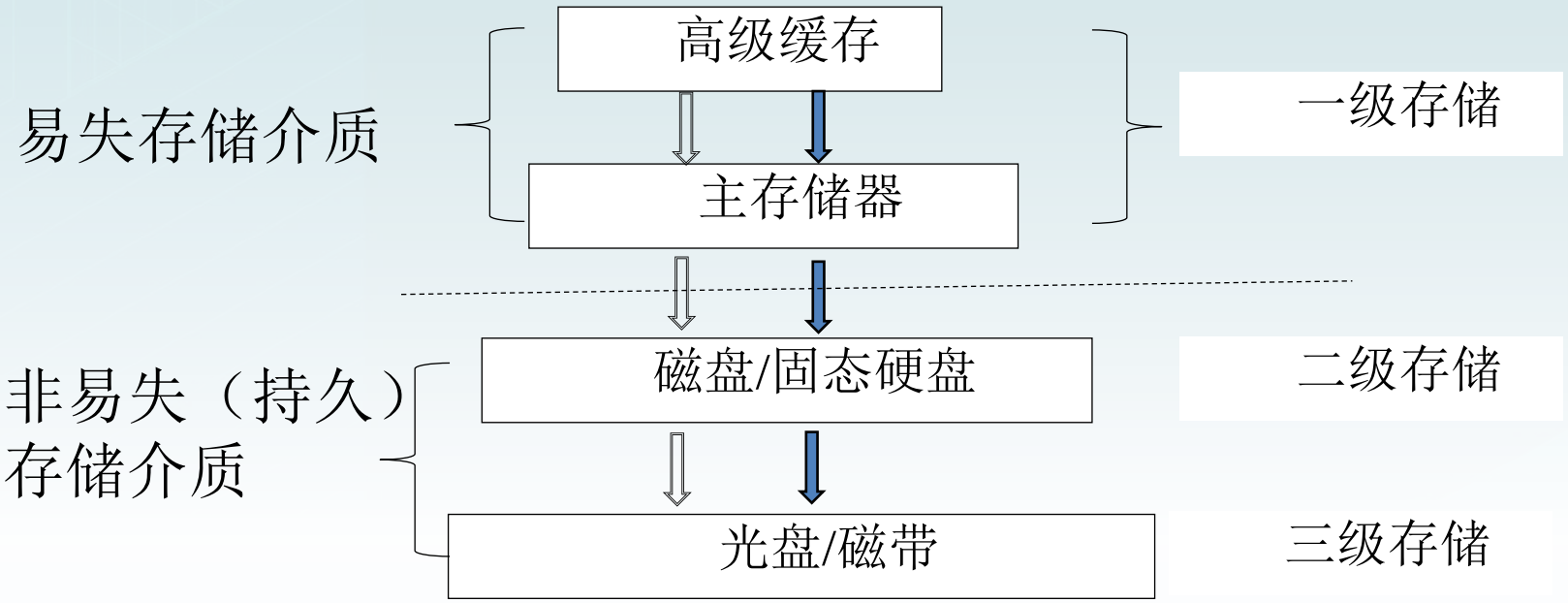
存储介质概述



- 速度更快
- 容量更小
- 价格更贵

- 速度更慢
- 容量更大
- 价格更便宜

存储介质概述



- 速度更快
- 容量更小
- 价格更贵

- 速度更慢
- 容量更大
- 价格更便宜

5.1 物理存储系统

- 5.1.1 存储介质概述
- 5.1.2 存储介质-磁盘
- 5.1.3 磁盘I/O性能的提升策略

存储介质-磁盘

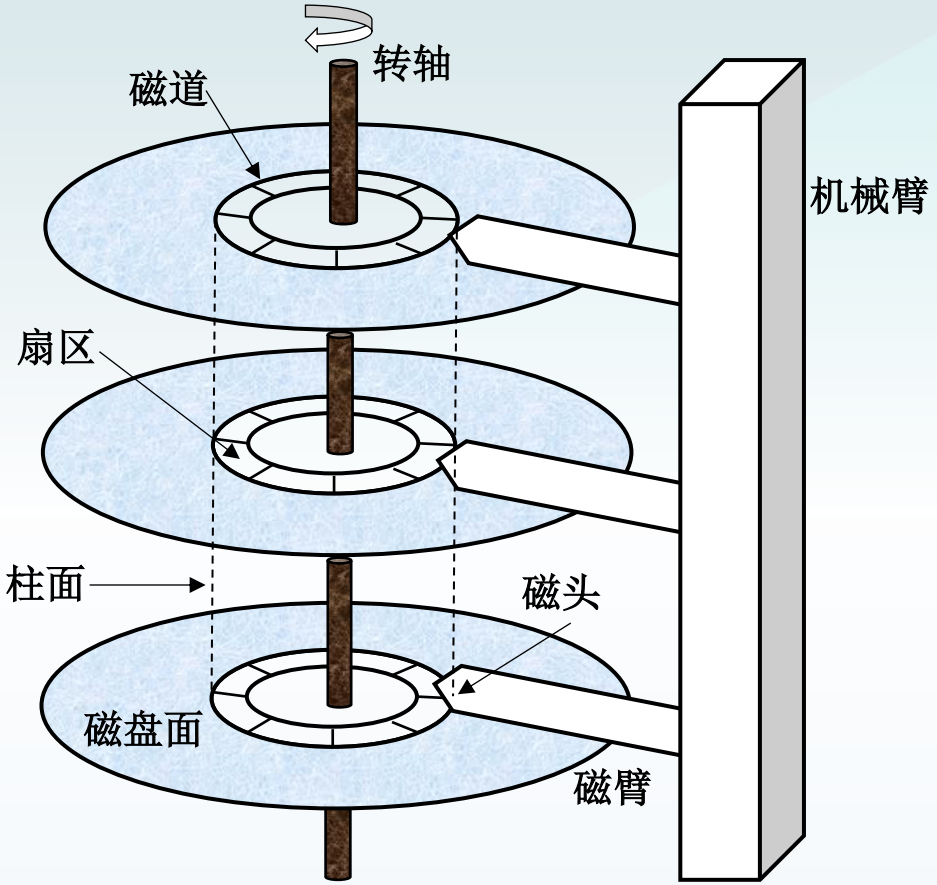
磁盘目前是计算机系统最主要的持久存储介质，对磁盘的访问效率直接影响系统的性能。

磁盘逻辑：

- 磁盘（track）
- 磁道（sector）
- 扇区（磁盘I/O最小的单位）

磁盘质量：

容量、访问时间、数据传输率、可靠性

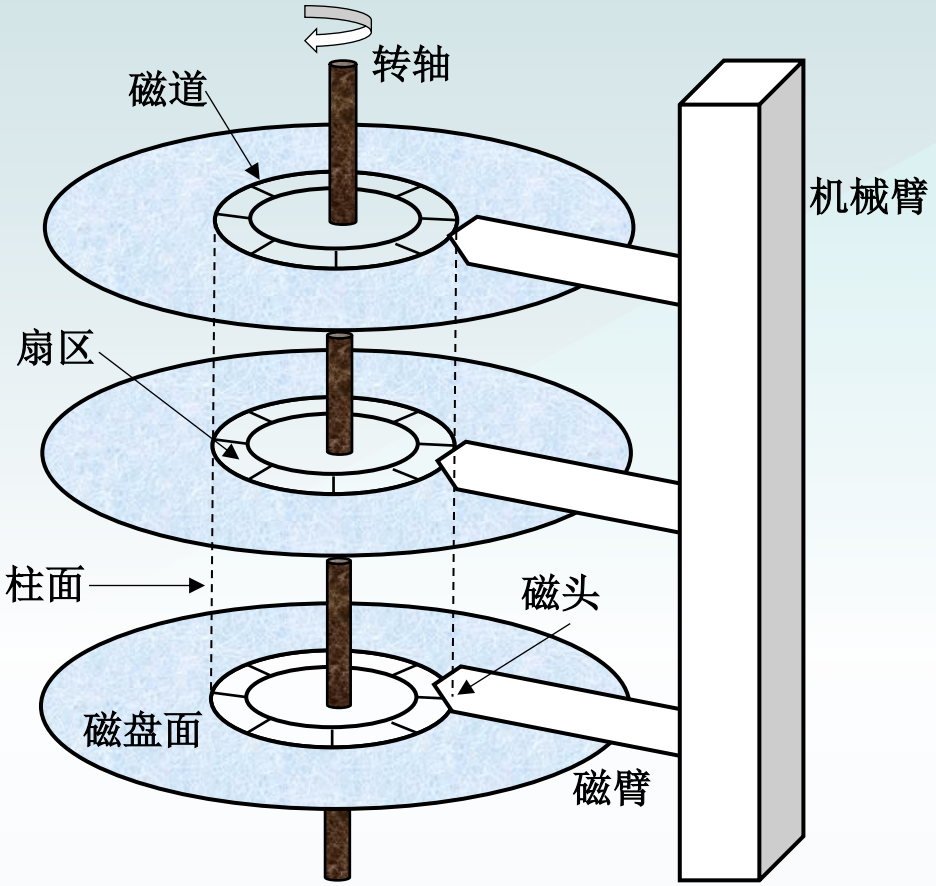


存储介质–磁盘

磁盘读写：

- 磁盘臂先移动到正确磁道的上方；
- 等待磁盘旋转到它下方指定的扇区；
- 磁盘控制器（disk controller）

  接受高层次的读写扇区命令。



存储介质-磁盘

访问时间：

- 寻道时间：磁盘臂定位的时间，依赖于目标磁道与磁盘臂初始位置之间的距离，大约2-20ms。
- 旋转延迟时间：等待磁盘旋转的时间，平均延迟时间应该是磁盘旋转一周时间的一半。
- 数据传输时间。

存储介质-磁盘

磁盘I/O请求：

- 通常由文件系统/数据库系统发出；
- 分为顺序读写和随机读写两种模式，顺序读写模式寻道时间短。

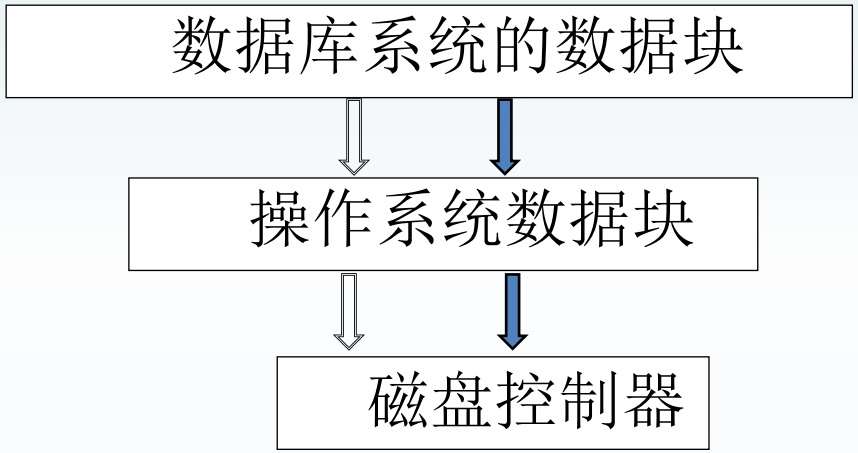


图. I/O请求的层次

5.1 物理存储系统

- 5.1.1 存储介质概述
- 5.1.2 存储介质-磁盘
- 5.1.3 磁盘I/O性能的提升策略

磁盘I/O性能的提升策略

- 缓解I/O瓶颈的常用措施包括：
  - 使用缓冲区，把常用的数据块缓存在内存中，减少物理I/O的发生。
  - 采用合适的数据组织方式以减少I/O的发生，提高I/O效率。
  - 有针对性地预读或使用系统提供的异步I/O能力。

第5章 存储管理

- 5.1 物理存储系统
- 5.2 数据组织
- 5.3 元数据存储
- 5.4 缓冲区管理
- 5.5 小结

5.2 数据组织

- 5.2.1 数据库的逻辑和物理组织方式
- 5.2.2 记录表示
- 5.2.3 块的组织
- 5.2.4 关系表的组织

5.2 数据组织

- 5.2.1 数据库的逻辑和物理组织方式
- 5.2.2 记录表示
- 5.2.3 块的组织
- 5.2.4 关系表的组织

数据库的逻辑和物理组织方式

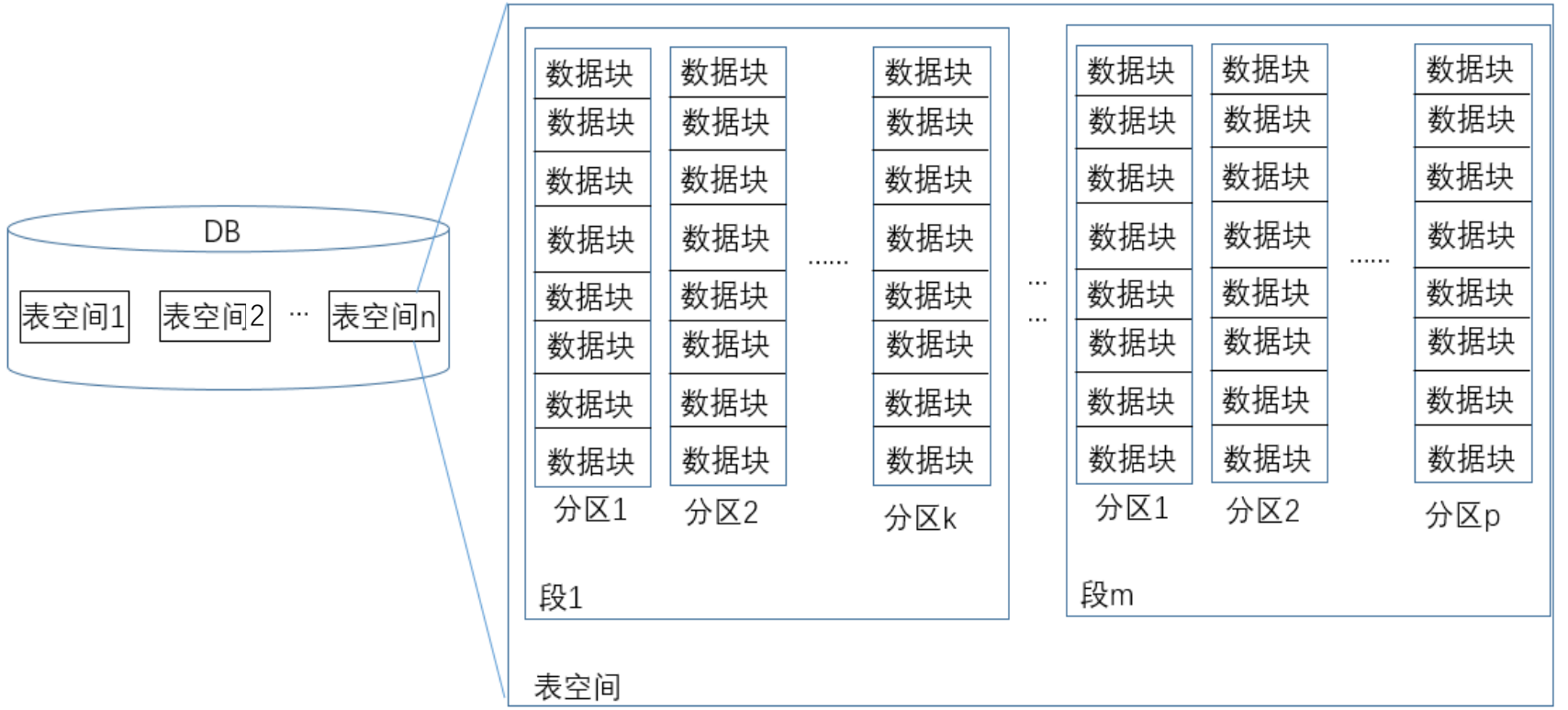
- DB以文件方式存储
  1. 一个表或者索引对应一个文件
     - 存储管理交由OS
  2. 整个DB对应一个或若干个文件（段页式存储方式）
     - 由DBMS进行存储管理

数据库的逻辑和物理组织方式

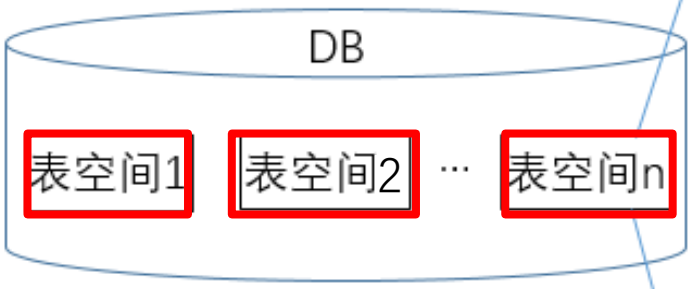
- 由DBMS进行存储管理
  - DB通常被划分成更小的单位，例如段(Segment或Area。
  - 一种常见的划分方式是将数据库组织成“**表空间–段–分区–数据块**”的形式
  - 整个DB对应一个或若干个文件，由DBMS进行存储管理
  - 不同DBMS划分方法不同
    - 示例：ORACLE

数据库的逻辑和物理组织方式

- “表空间-段-分区-数据块”形式
  - 分区和段相结合，能够增加灵活性；
  - 表和块相结合，能够更好地平衡性能，方便管理。



数据库的逻辑和物理组织方式

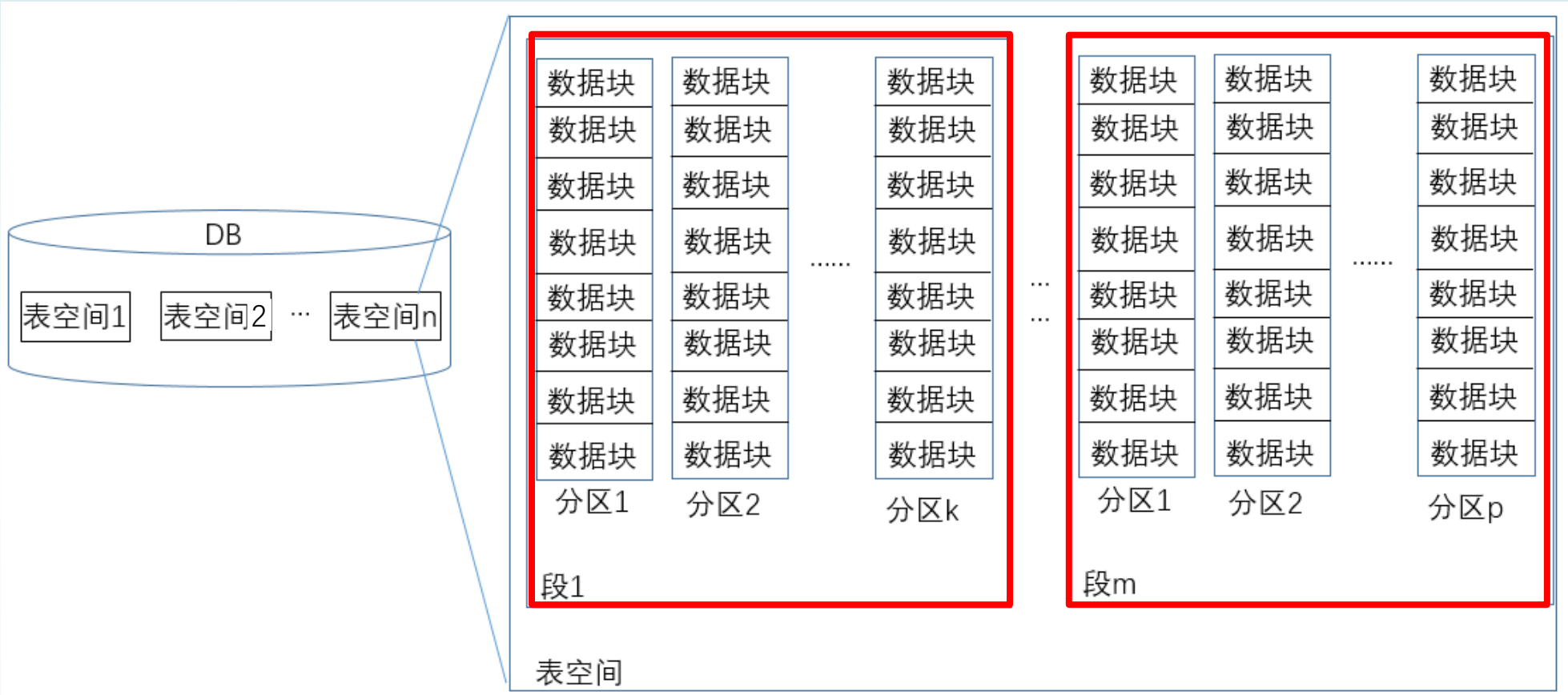


- 表空间(逻辑设备)
  - 对应磁盘上一个或多个物理数据文件
  - 一个物理文件只能属于一个表空间
  - 可以有多个表空间，逻辑地和物理地组织数据库中的数据存储。
  - 系统表空间、联机表空间、脱机表空间、永久表空间、临时表空间

表空间

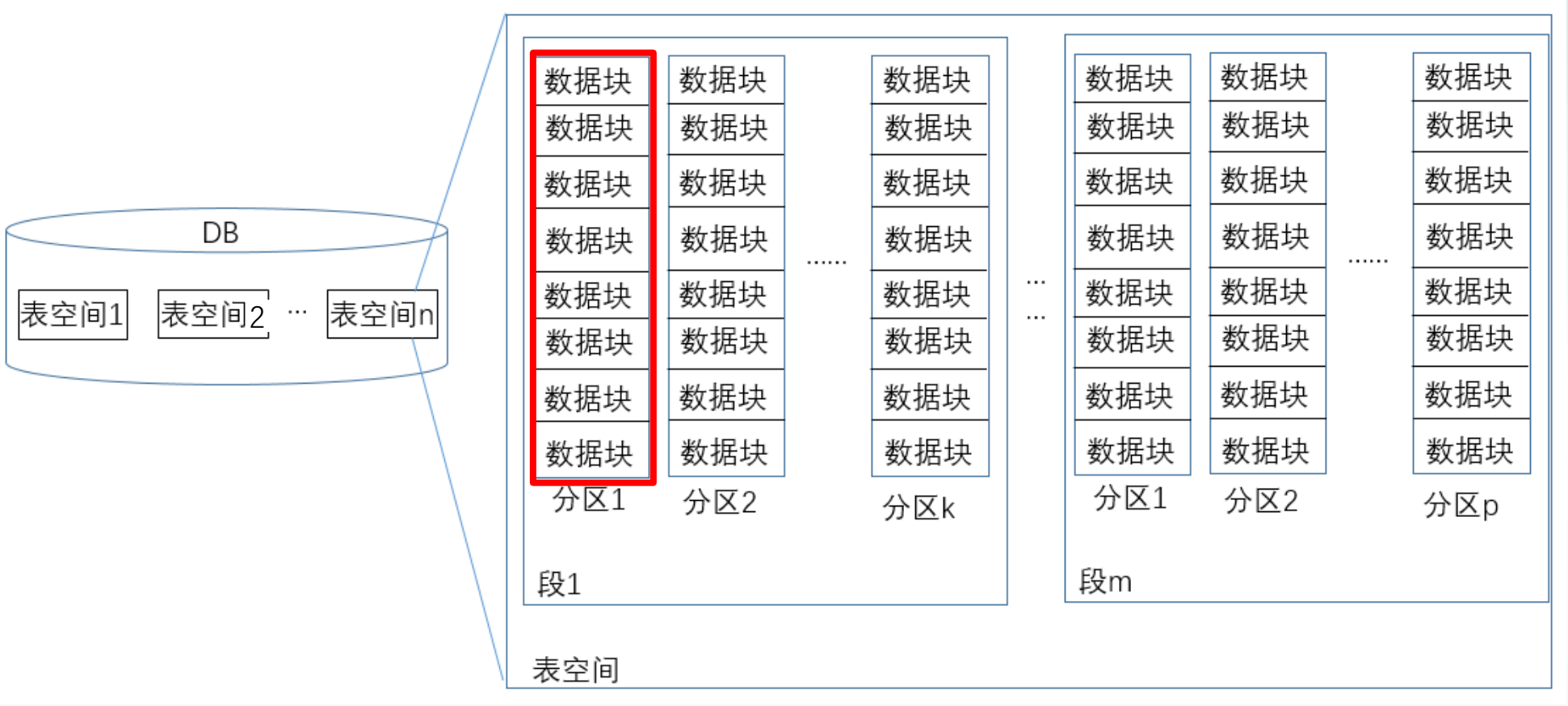
数据库的逻辑和物理组织方式

- 段
  - 由多个分区组成。
  - 数据段、索引段、LONG段、回滚段、临时段等



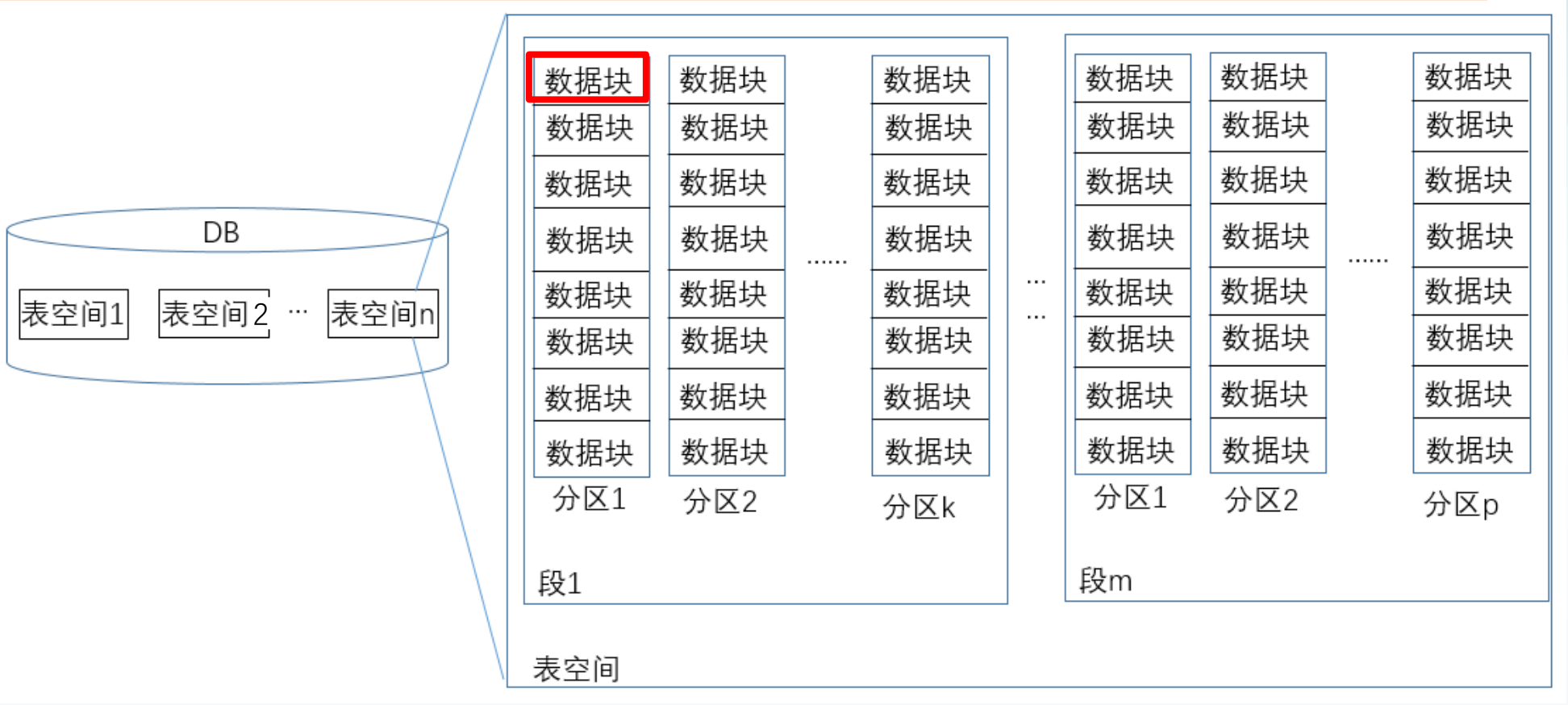
数据库的逻辑和物理组织方式

- 分区
  - 由一组连续的数据块组成

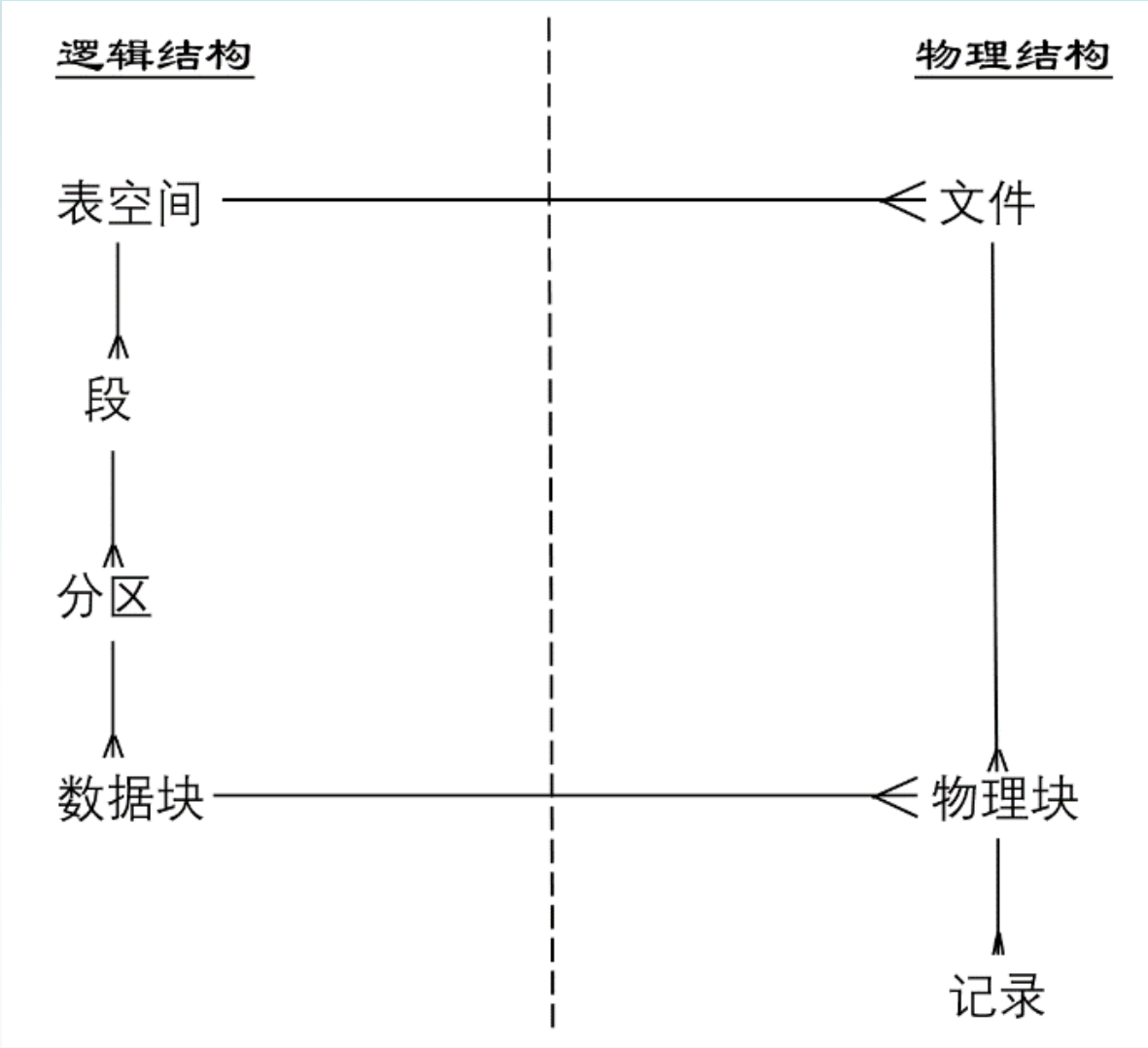


数据库的逻辑和物理组织方式

- **数据块**
  - 数据块是磁盘存取单元
  - 数据块的大小必须等于服务器操作系统块的大小或倍数



数据库的逻辑和物理组织方式



物理结构：

文件---块---记录

- 每个文件物理上分成定长的存储单元，即操作系统的物理块。
- 物理块是存储分配和I/O处理的基本单位。
- 一个物理块可以存放表中的多条元组（记录）。
- 一个表通常会占用多个块 。

5.2 数据组织

- 5.2.1 数据库的逻辑和物理组织方式
- 5.2.2 记录表示
- 5.2.3 块的组织
- 5.2.4 关系表的组织

记录表示

- 主要内容
  - 定长记录
  - 变长记录

定长记录

- 1）定长记录的构造

例：CREATE TABLE Products(

PID VARCHAR(32) PRIMARY KEY,

PName VARCHAR(128) NOT NULL,

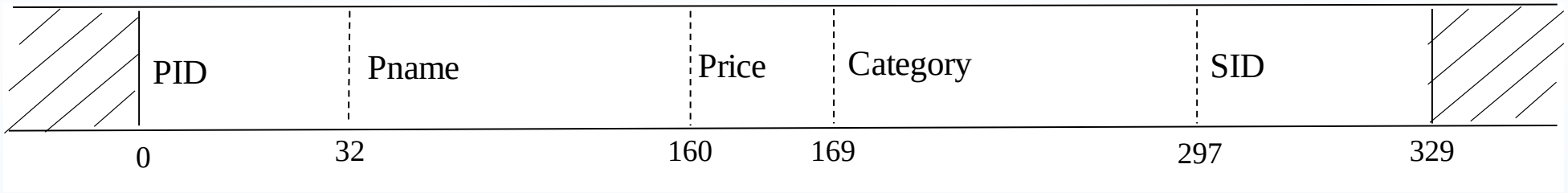
Price DECIMAL,

Category VARCHAR(128),

SID VARCHAR(32) NOT NULL,

FOREIGN KEY (SID) REFERENCES Suppliers(SID)

);



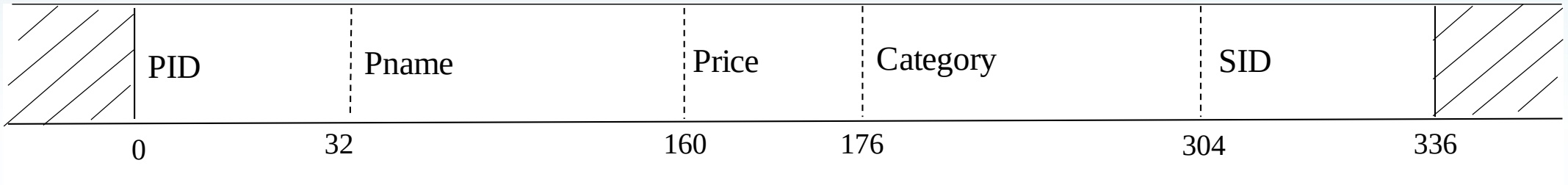
定长记录

- 某些机器对内存中数据地址的要求
  - 有些机器可以对内存中地址为4的倍数的字节开始的数据进行更有效的读写(64位处理器时则为8的倍数)
  - 在某些机器上，整数等数据类型要求必须从4的倍数的地址处开始
  - 在某些机器上，双精度实数要求从8的倍数处开始

运行于这些机器上的DBMS，转换字段在块内的偏移量时必须小心。

定长记录

- 为简化地址转换，并提高可移植性，可规定
  - 每一条记录在块内从4的倍数的字节处开始
  - 记录中所有的字段都从与记录偏移量为4的倍数的字节处开始



修改后的记录中，每个字段都是从8的倍数的地址起始，虽然Decimal是9字节，但实际分配给它16字节

5.2.2 记录表示

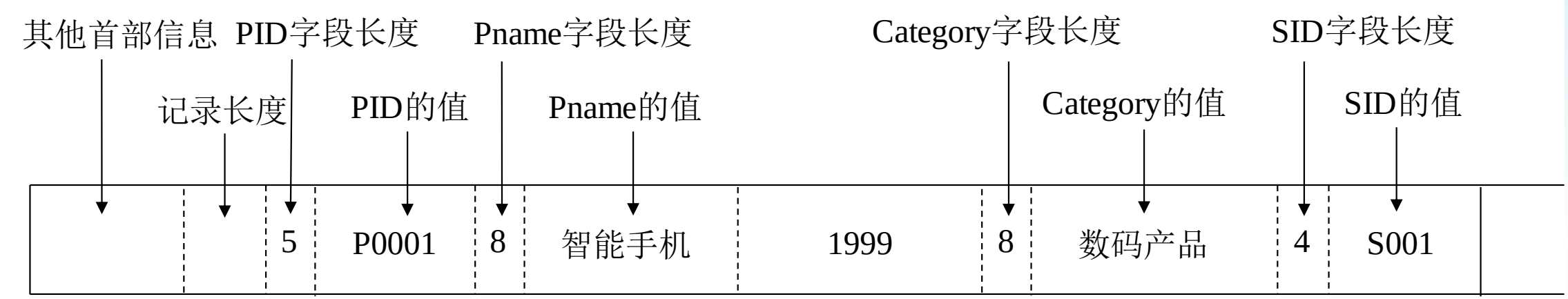
- 主要内容
  - 定长记录
  - 变长记录

变长记录

- **具有变长字段的记录（3种表示策略）**
  - 变长字段前加长度
  - 先放定长字段，后放变长字段
  - 变长字段单独存放在一个块中

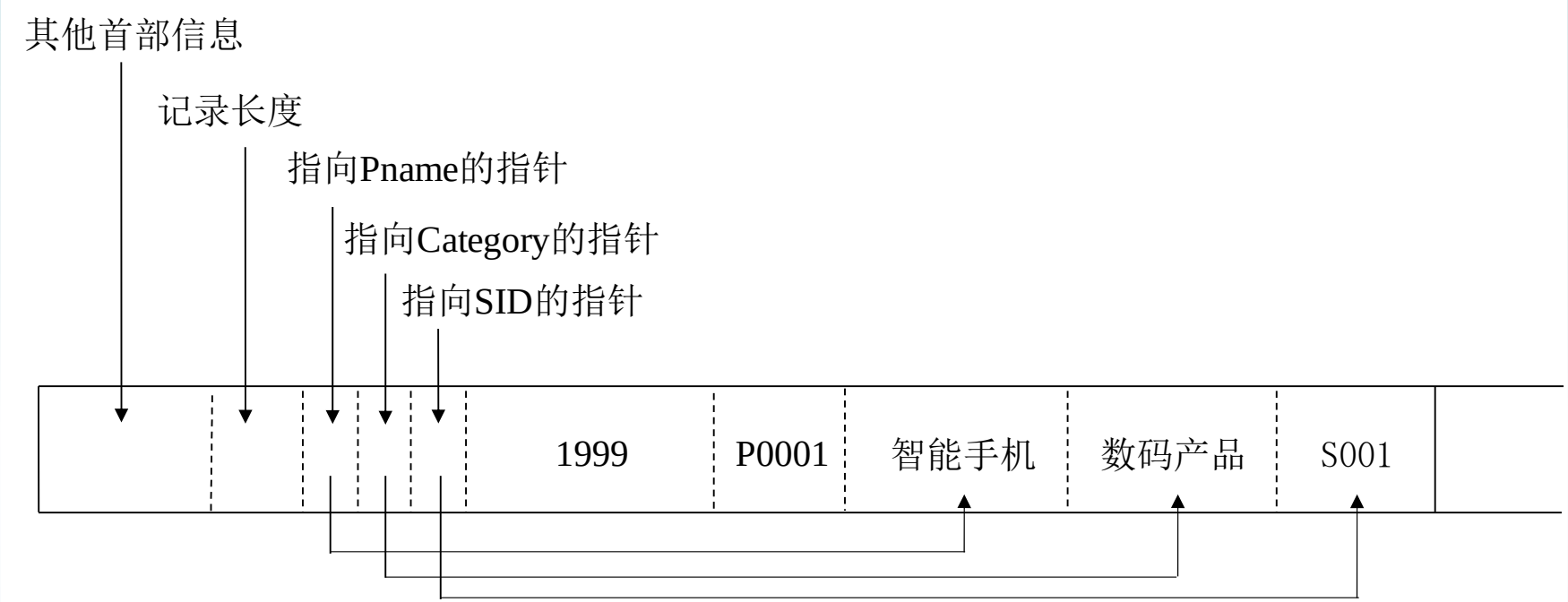
变长记录

- 变长字段前加长度



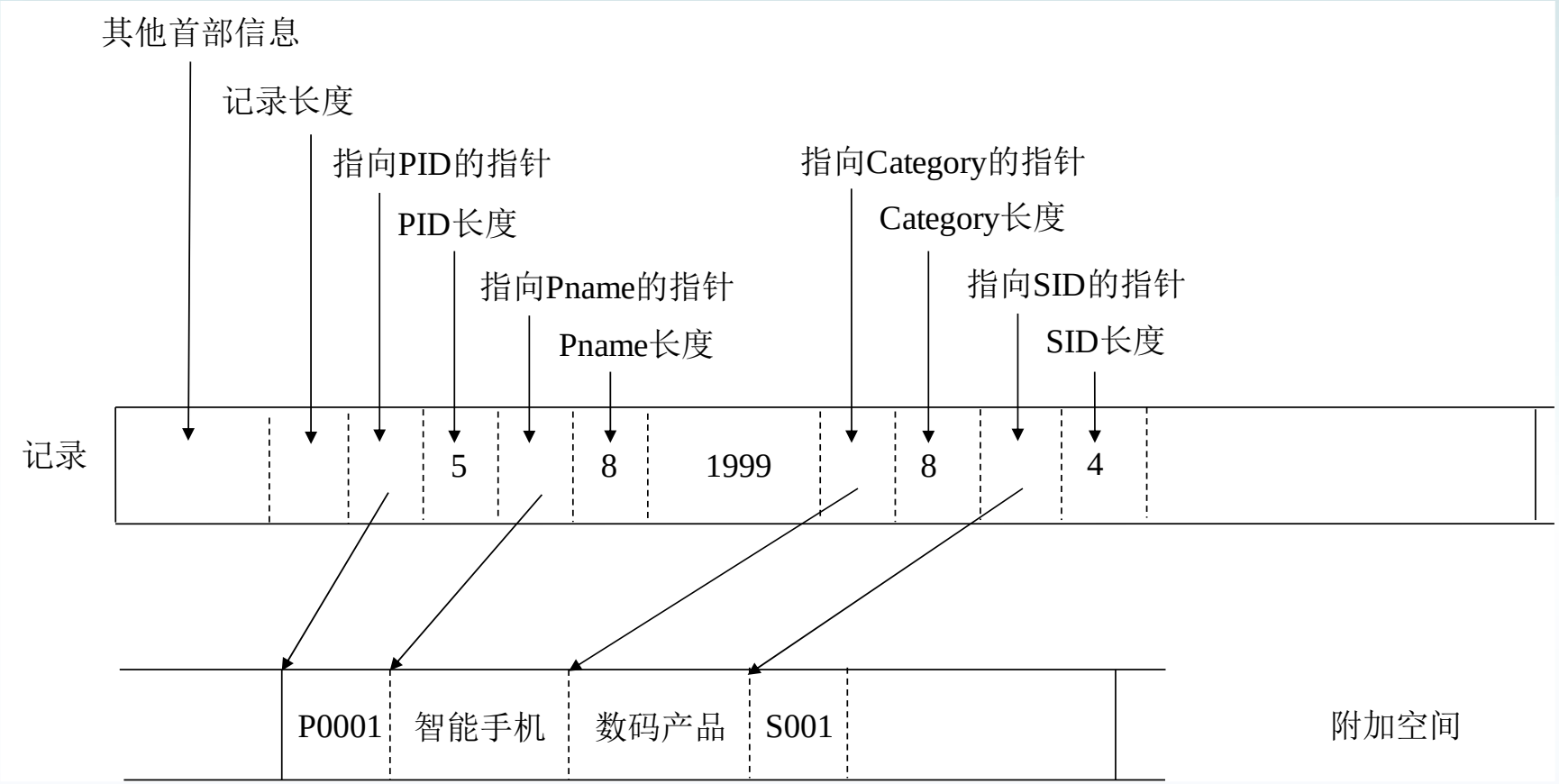
变长记录

- 先放定长字段，后放变长字段



变长记录

- 变长字段单独存放在一个块中



5.2 数据组织

- 5.2.1 数据库的逻辑和物理组织方式
- 5.2.2 记录表示
- 5.2.3 块的组织
- 5.2.4 关系表的组织

5.2.3 块的组织

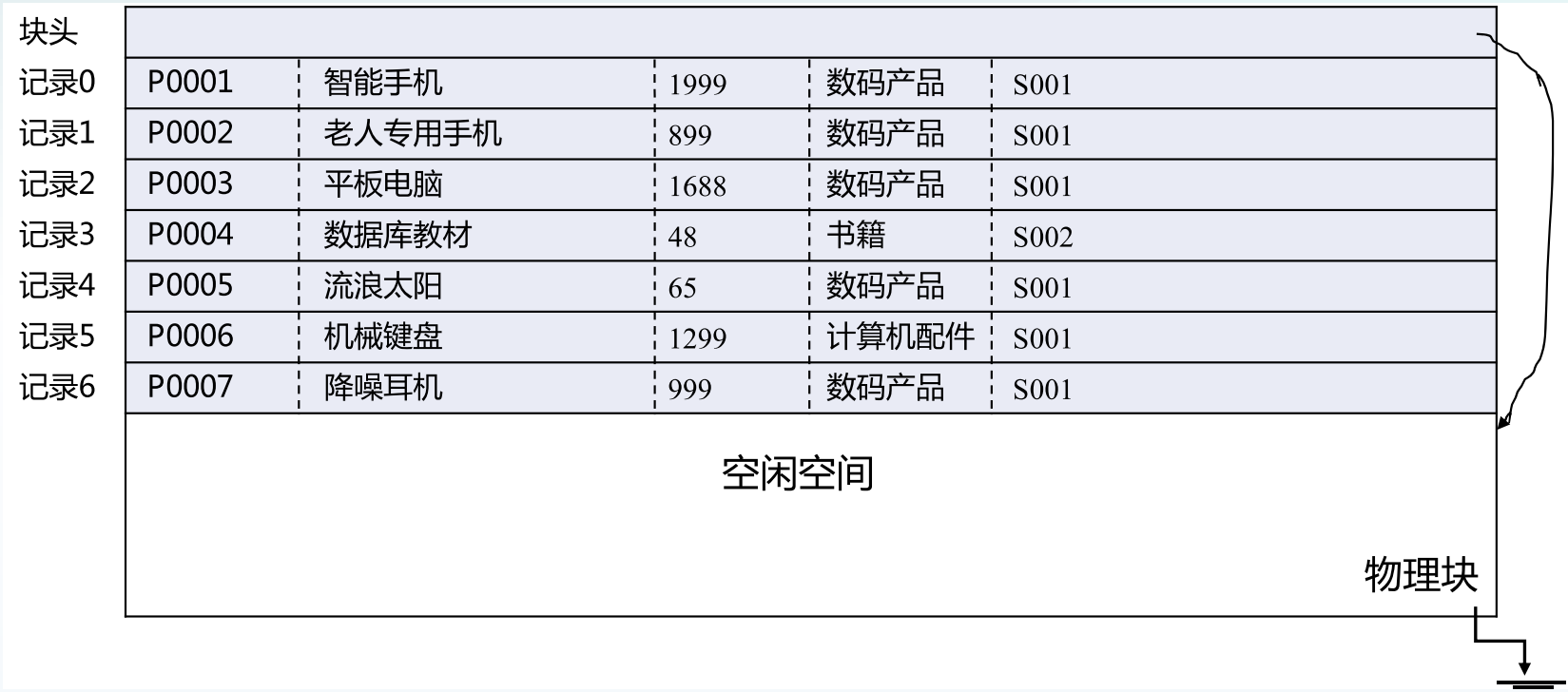
- 定长记录的块组织
- 变长记录的块组织

5.2.3 块的组织

- 定长记录的块组织
- 变长记录的块组织

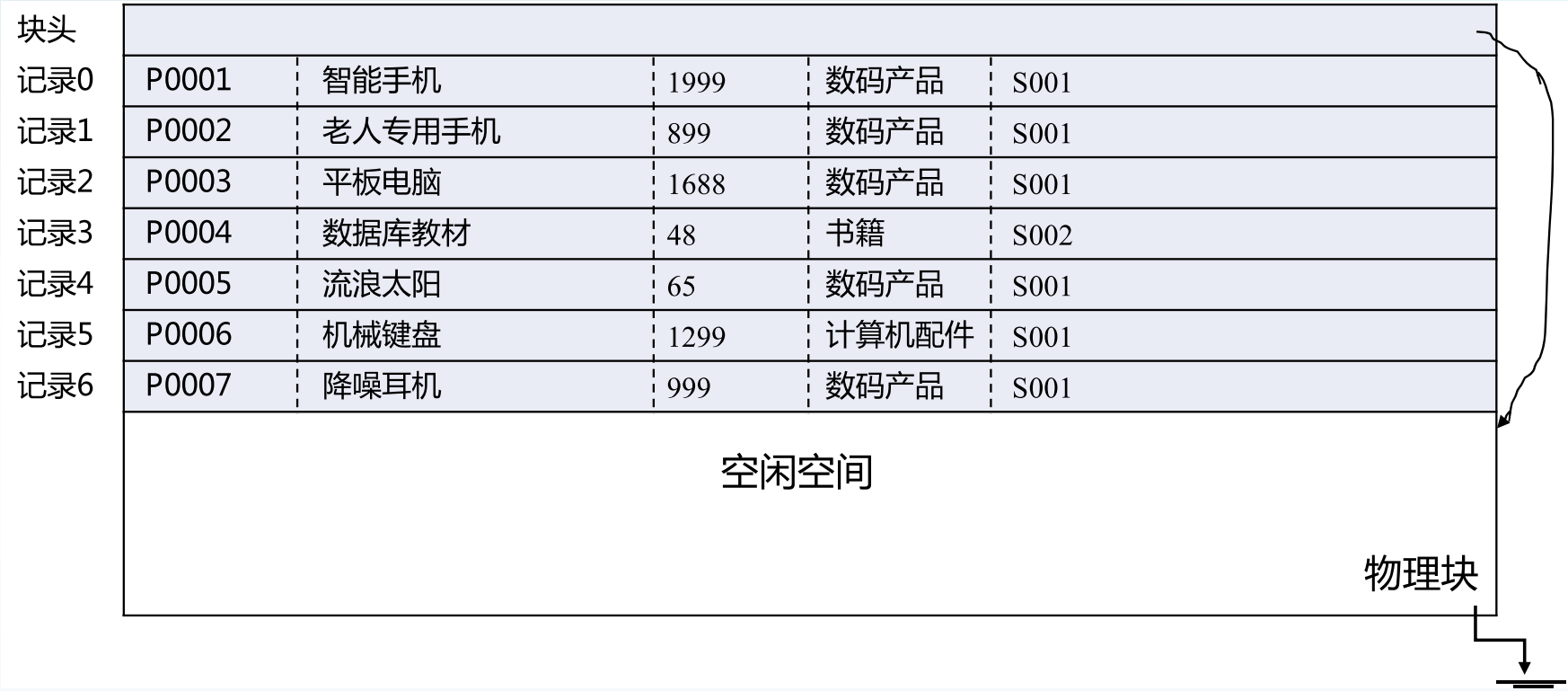
定长记录的块组织

- 定长记录的块组织
  - 表中元组依次存放在块中
  - 首部+记录+空闲空间
    - 首部：块ID、最后一次修改和访问该块的时间戳、每条元组在块内的偏移量、空闲空间头指针



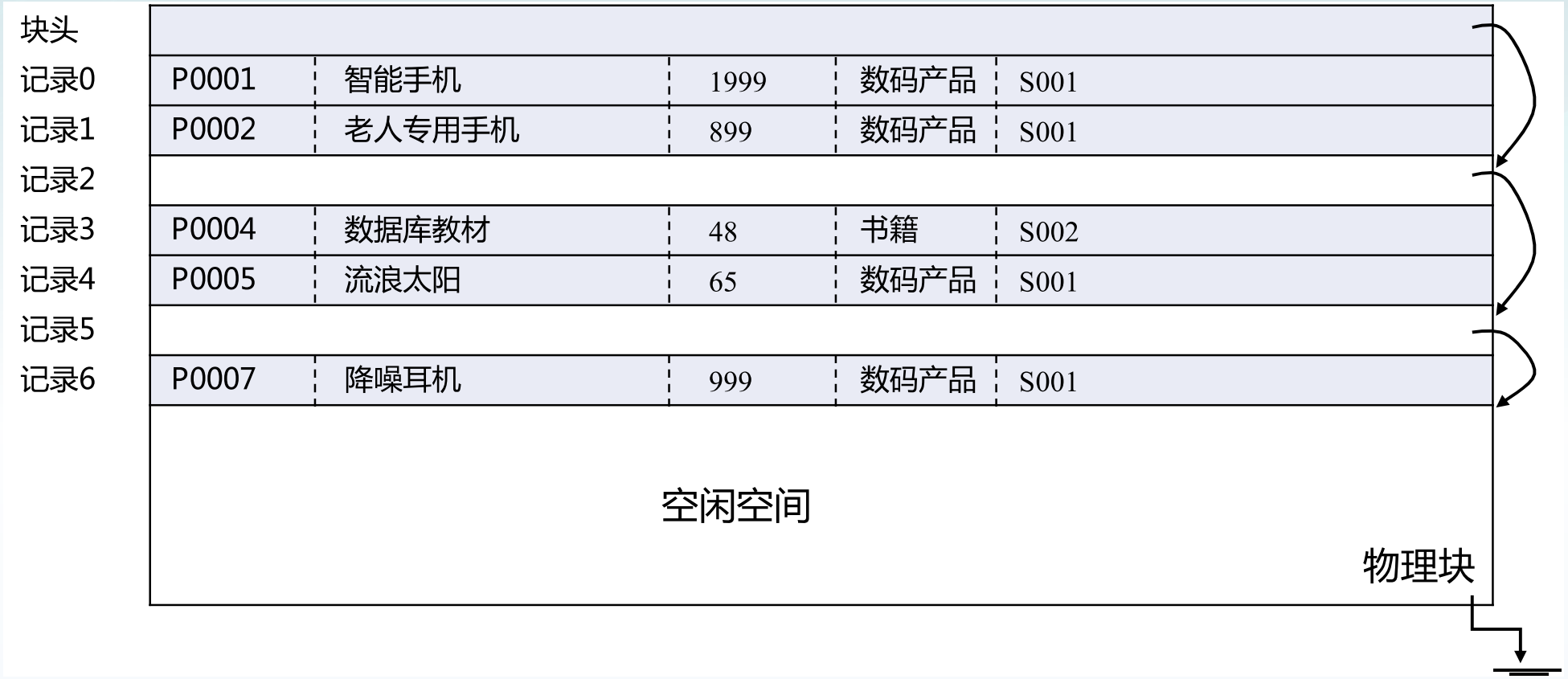
定长记录的块组织

- **定长记录块的维护**
  - 增：在空闲空间直接插入新元组
  - 改：直接在原位置修改
  - 删：回收空间，将空闲空间加入空闲空间链表



定长记录的块组织

- 例：在Products表中删除P0003和P0006两条元组

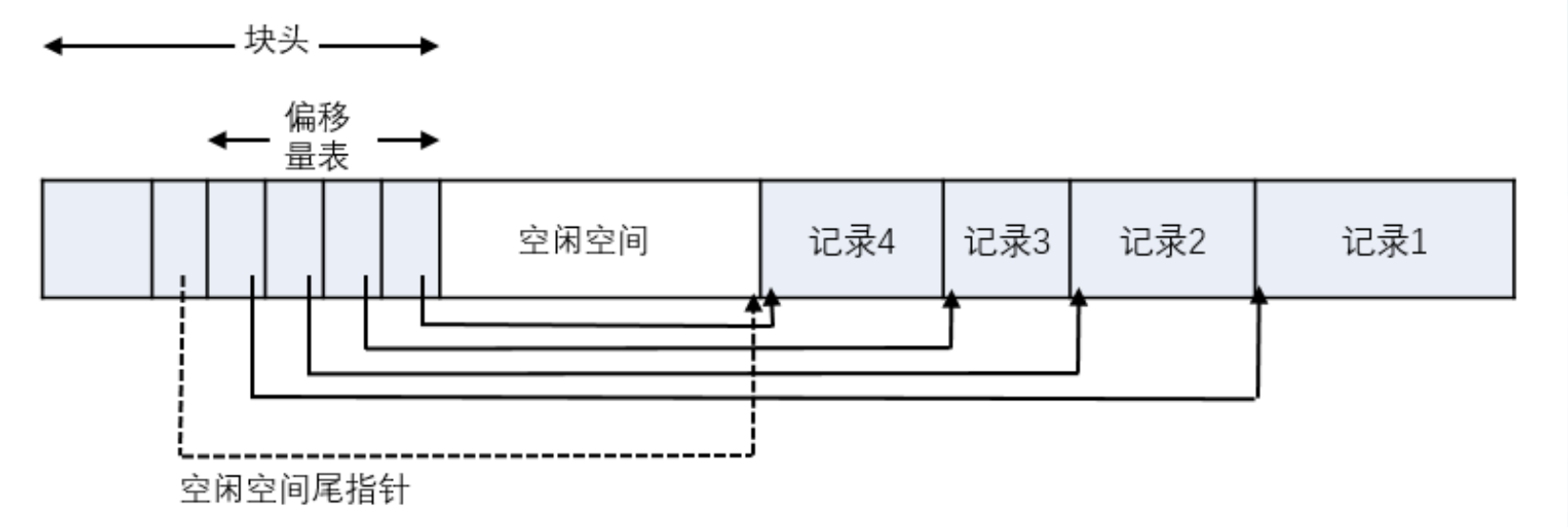


5.2.3 块的组织

- 定长记录的块组织
- 变长记录的块组织

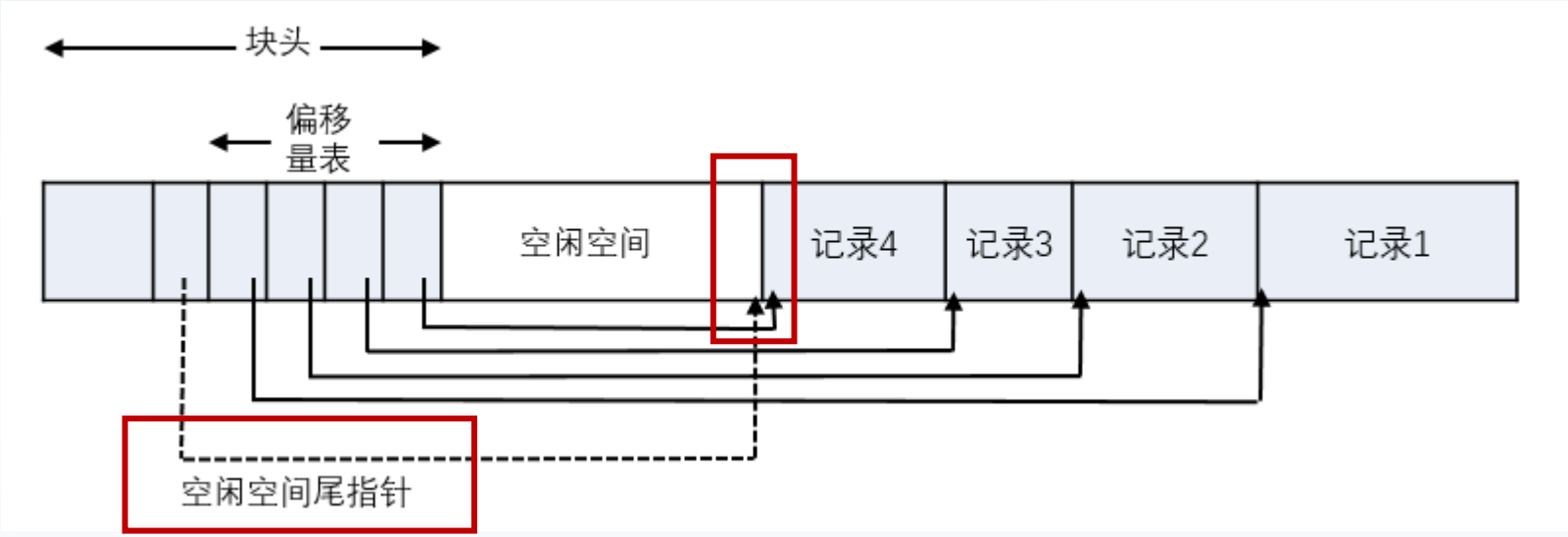
变长记录的块组织

- 变长记录的块组织
  - 表中元组从块的尾部连续存放
  - 首部+空闲空间+记录
    - 首部：各记录的指针（块内偏移量）、空闲空间为尾指针（块内偏移量）



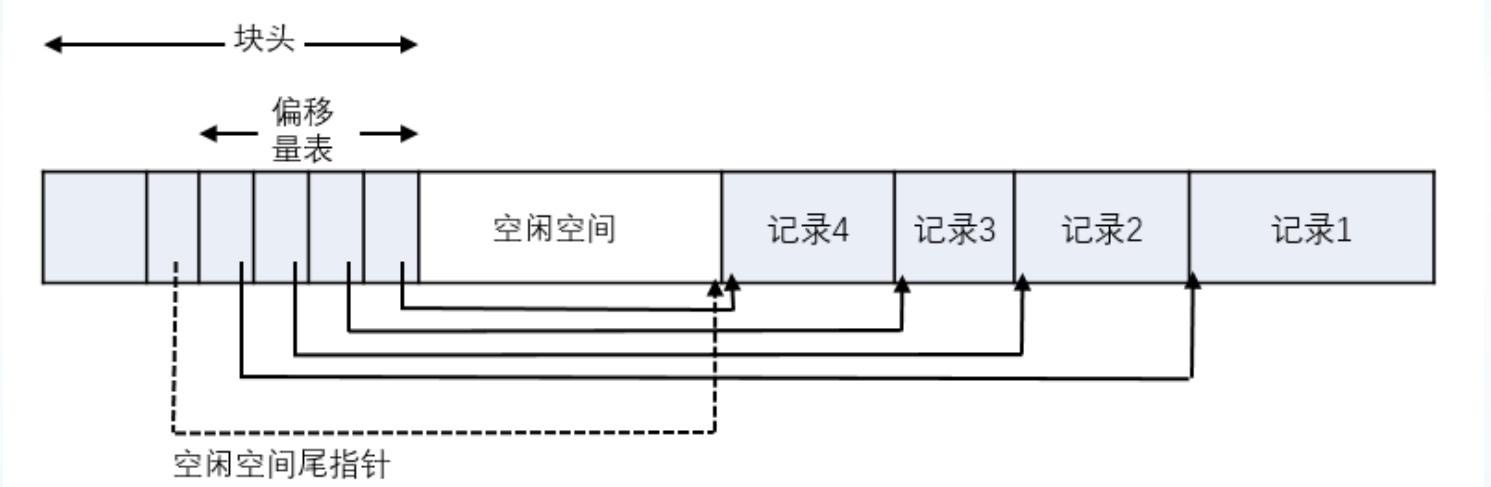
变长记录的块组织

- 变长记录块的维护
  - 增
    - 从空闲空间尾部分配空间
    - 在偏移量表中记录该元组的起始位置
    - 调整空闲空间尾指针



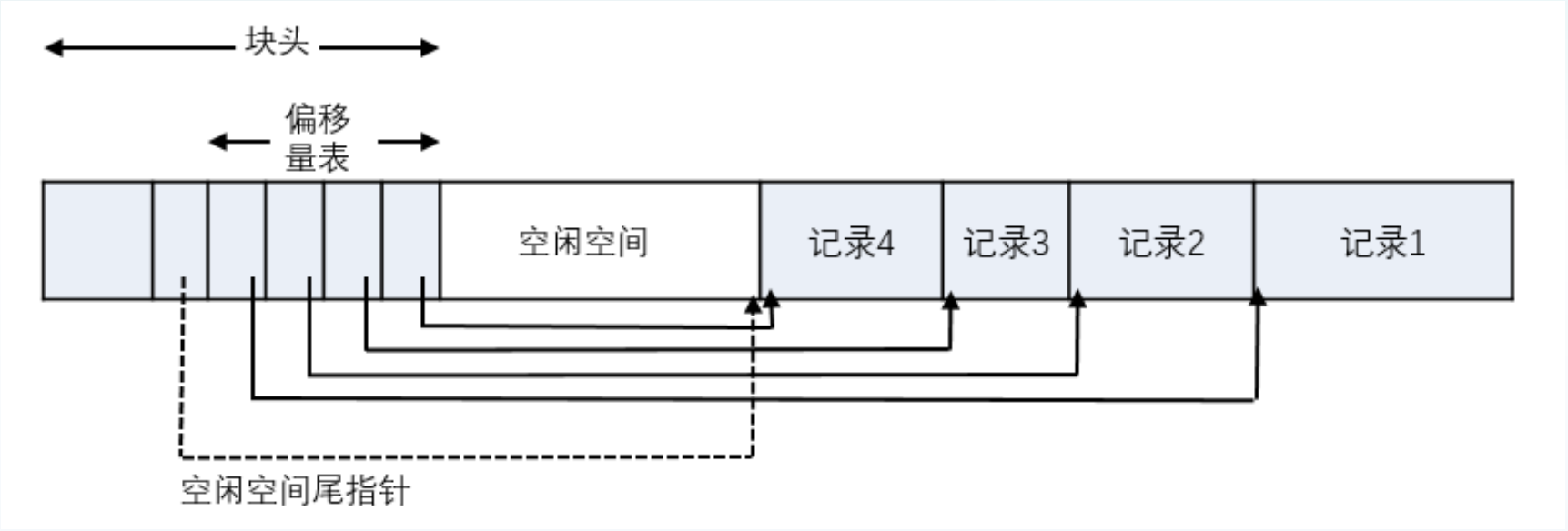
变长记录的块组织

- **变长记录块的维护**
  - 删
    - 在偏移量表中为该元组指针置删除标记
    - 释放元组空间，移动物理位置在其前面的元组（保证空间连续）
    - 修改指针
      - 被移动元组在偏移量表中的指针
      - 空闲空间尾指针



变长记录的块组织

- **变长记录块的维护**
  - 改
    - 在原位置修改
    - 若修改后记录在原位置放不下，会带来记录的迁移



5.2 数据组织

- 5.2.1 数据库的逻辑和物理组织方式
- 5.2.2 记录表示
- 5.2.3 块的组织
- 5.2.4 关系表的组织

5.2.4 关系表的组织

- **关系表的6种存放方式**
  - 堆存放方式
  - 顺序存放方式
  - 多表聚簇存放方式
  - B+树存放方式
  - 哈希存放方式
  - LSM树存放方式

堆存放方式

- **堆存放方式**
  - 表中的一条记录可以存放在该表的任何块中，没有顺序要求
  - 插入元组时，在该表的块中找到合适的空闲空间即可
  - 如果没有足够的空闲空间，就为该表申请新的块

顺序存放方式

- 顺序存放方式
  - 一个表中的各条记录根据指定的属性或属性组的取值大小顺序的存放
  - 同一个表的不同块中通过指针链接实现有序

例：如果Products表按专业Category的升序存放，则可以快速回答如下的SQL语句

```
SELECT PID, Pname, Price
FROM Products
WHERE Category=’数码产品’;
```

问题：插入或修改元组时，为保持记录顺序，有可能需要在块内或块间迁移已有记录，因而代价较高。

多表聚簇存放方式

- 多表聚簇存放方式
  - 不同表的元组聚簇存放在同一组块中

例：两个具有主外码的参照关系表Products和OrderItems，按照商品ID相等聚簇存放

多表聚簇存放方式

- **多表聚簇存放方式的优劣**
  - 优势：
    - 减少连接操作带来的开销
      - 可快速回答如下查询：

```
SELECT OrderItems.PID,Products.Pname, count(*)
FROM Products, OrderItems
WHERE OrderItems.PID=Products.PID AND
OrderItems.PID=’P0001’;
```

多表聚簇存放方式

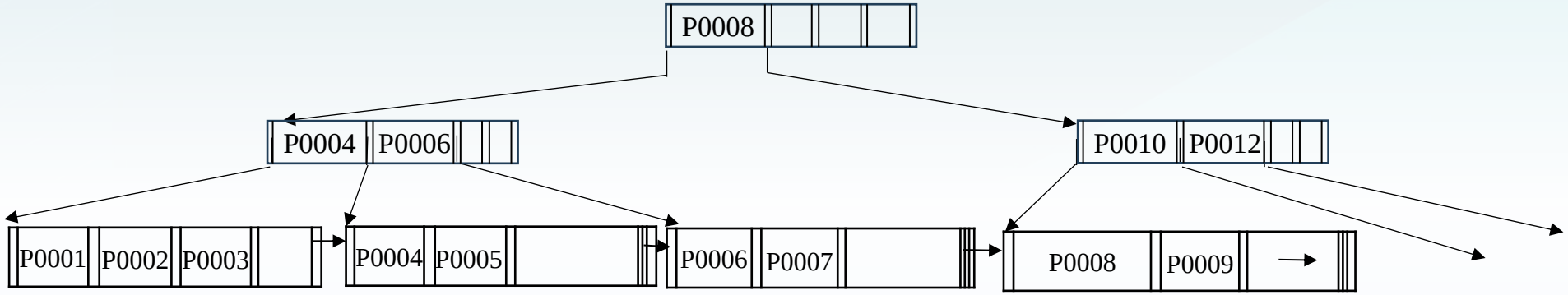
- **多表聚簇存放方式的优劣**
  - 劣势：
    - 降低某些查询的查询效率（同一表中的元组会分散在更多块中）
      - 如：

```
SELECT *
FROM Products
WHERE SID=’ S001’ AND
Category=’数码产品’;
```

    - 在更新操作时会带来更频繁的数据迁移
      - 如在每笔新交易结束后，为了保证个订单项紧跟在其商品信息之后，会频繁地迁移记录，以腾出空间插入新的订单项记录。

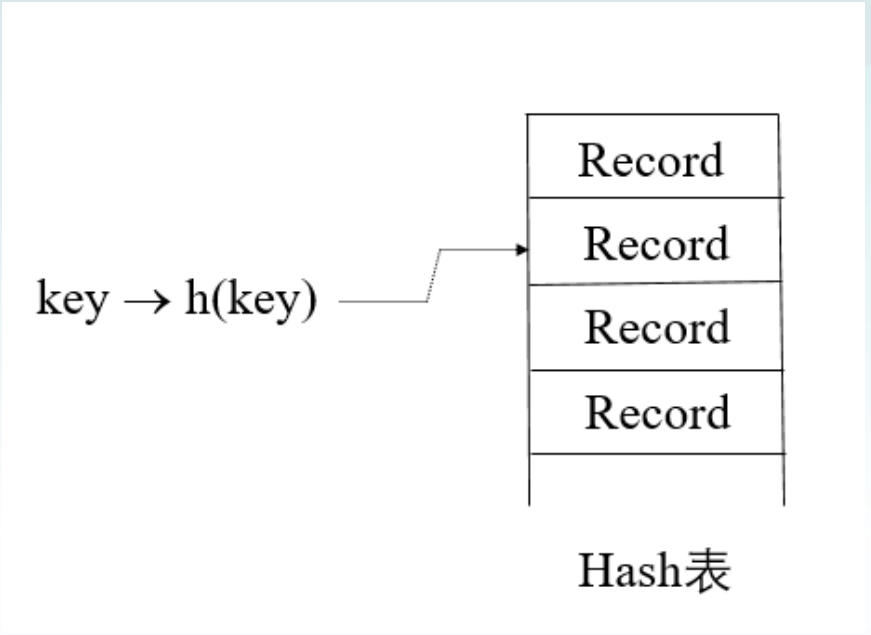
B+树存放方式

- B+树存放方式
  - 以B+树索引的方式确定记录存放在哪个数据块中
  - 保持较高的访问效率的同时，降低数据维护开销

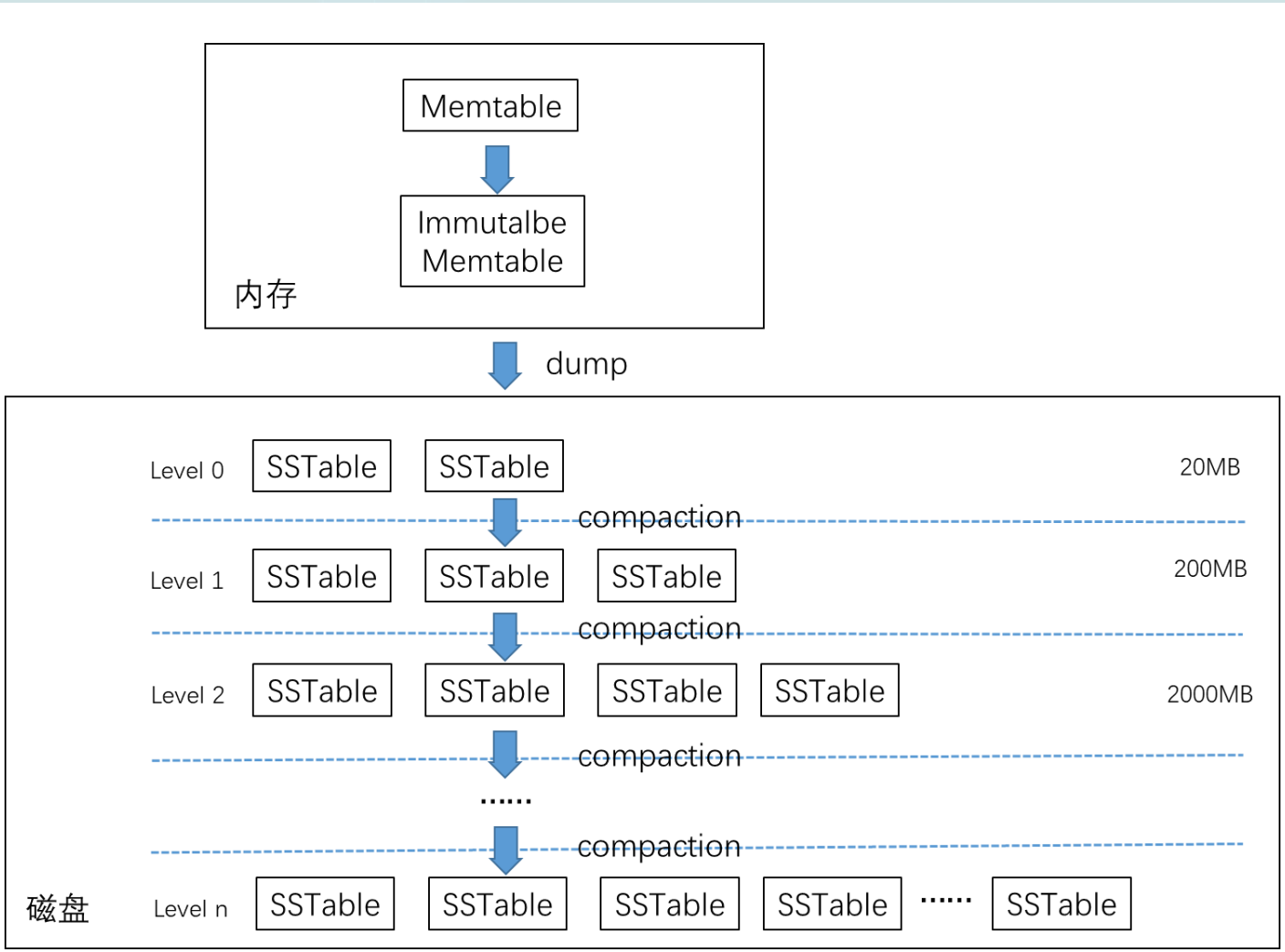


哈希存放方式

- **哈希存放方式**
  - 用哈希函数计算表中指定属性的哈希值，以此确定相应记录放在哪个块中
    - 哈希表：由B个哈希桶组成，每个桶编号0到B-1，对应一个或多个物理块，存放一条或多条记录
    - 哈希函数：输入为记录的哈希属性，输出为介于0到B-1之间的整数，对应哈希桶号

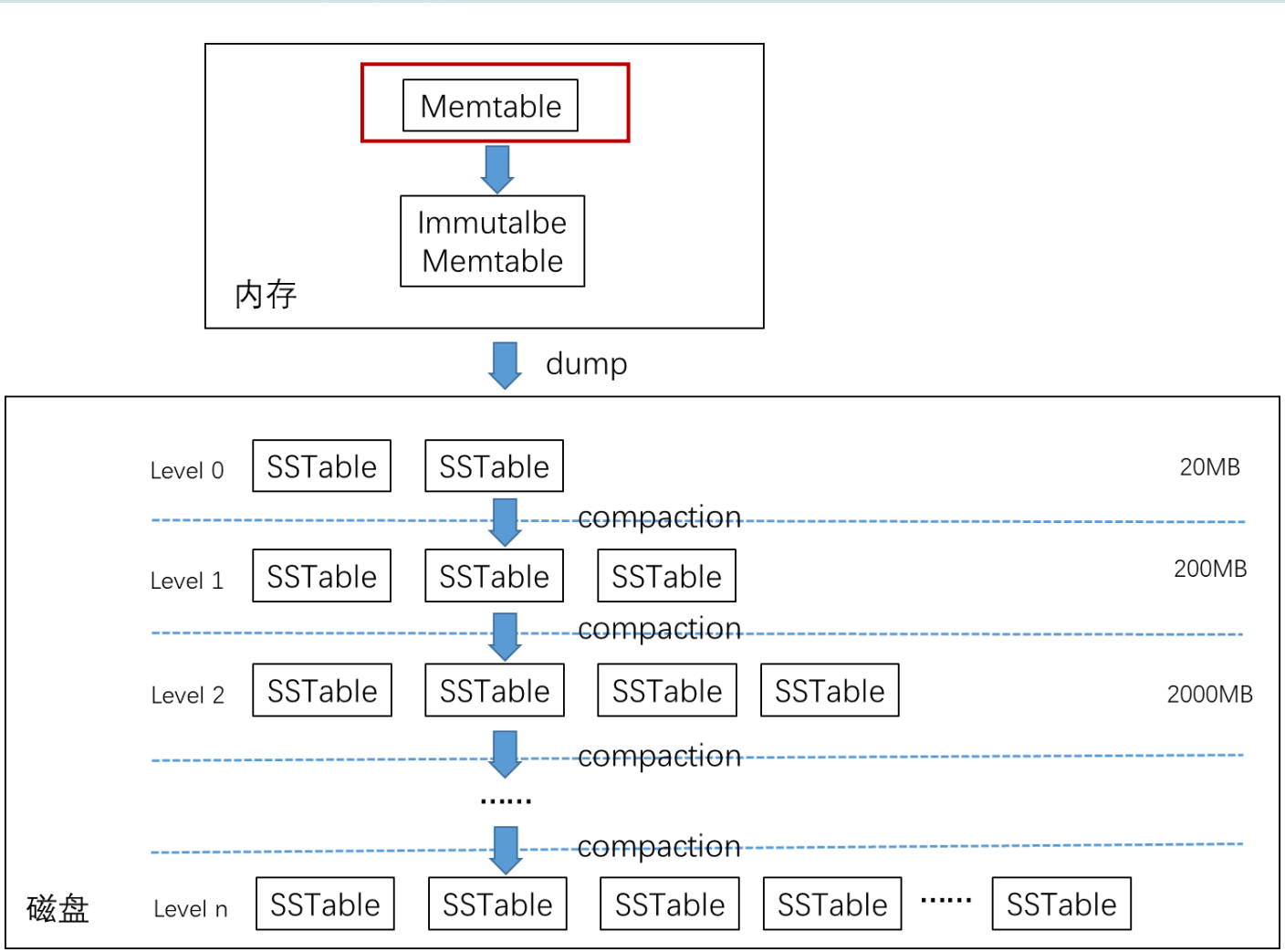


LSM树存放方式



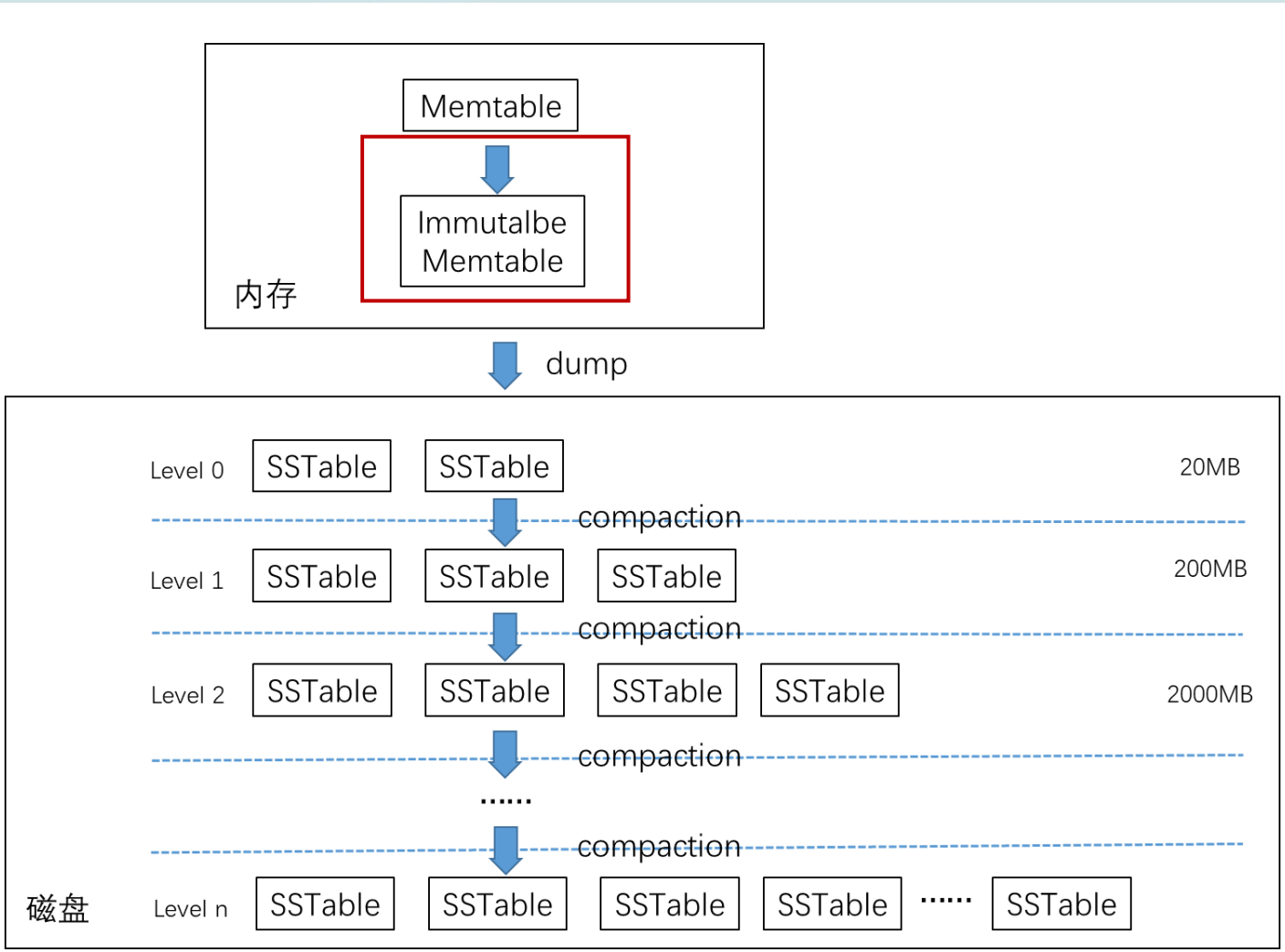
- LSM树：Log-Structured-Merge Tree
- LSM树是一种分层有序、硬盘友好的数据存储方式
- 它采用内存+磁盘的多层存储结构，即将数据存储分成内存和磁盘，并将硬盘分为多个层级（Level 0, Level 1, Level 2, ……, Level N）

LSM树存放方式



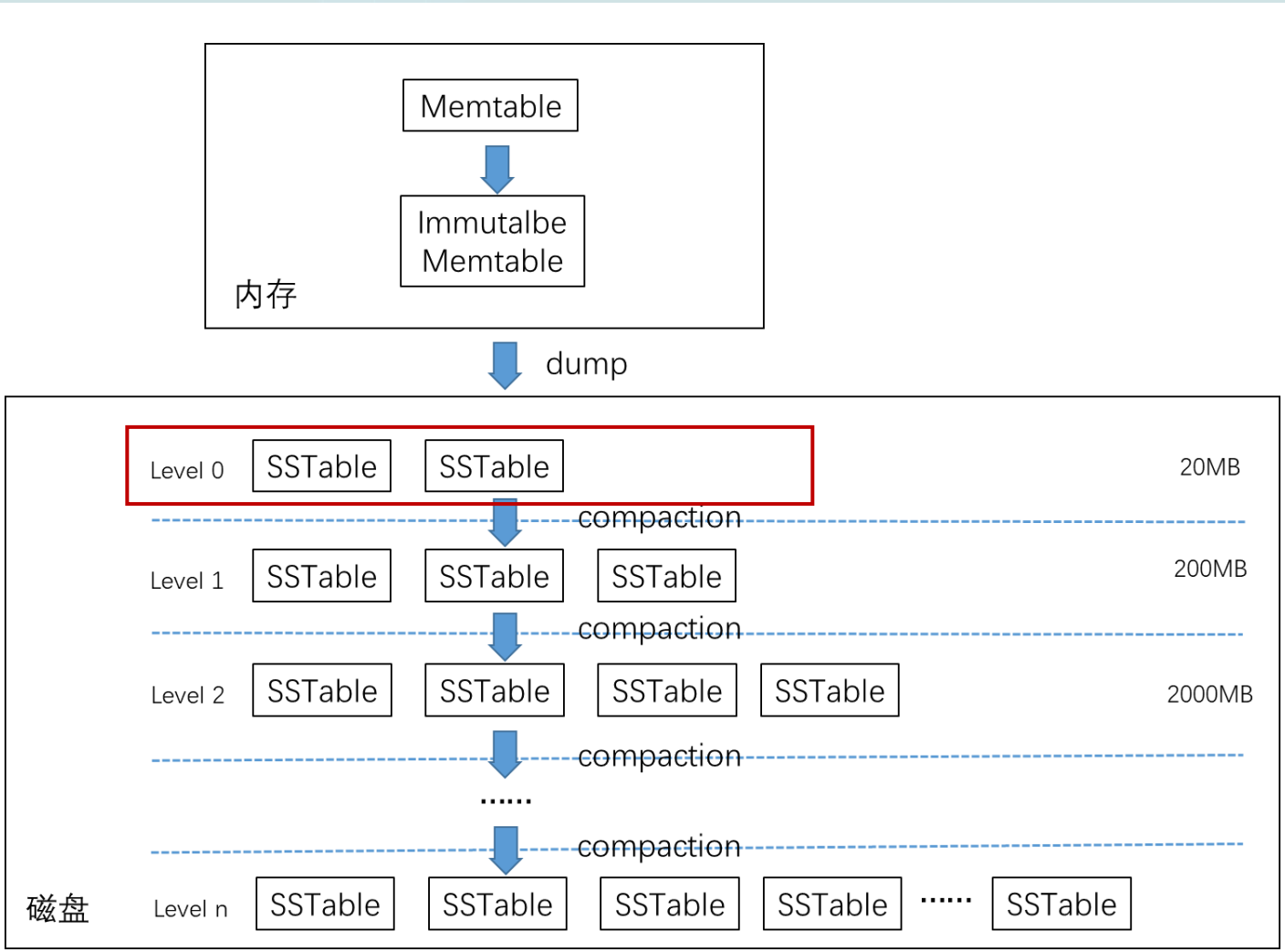
- DBMS首先将数据写到内存数据结构MemTable中，并保证数据是按键值有序的。

LSM树存放方式



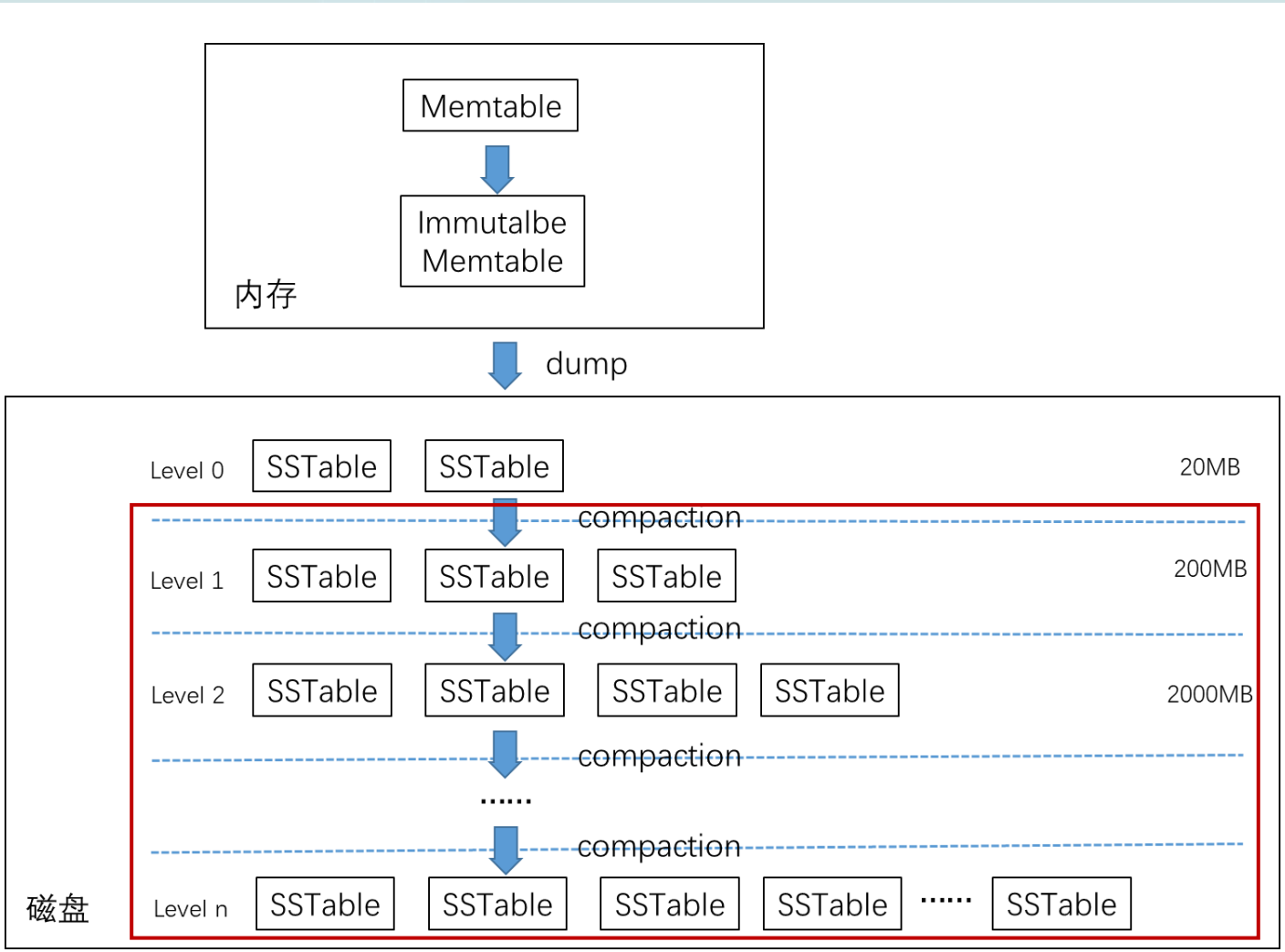
- 当MemTable的大小达到某个阈值时，会转化为immutable MemTable
- Immutable MemTable 是将 MemTable 转为磁盘上的 SSTable 的一种中间状态
- 转化过程中写操作由新的 MemTable 处理，过程中不阻塞数据更新操作。

LSM树存放方式



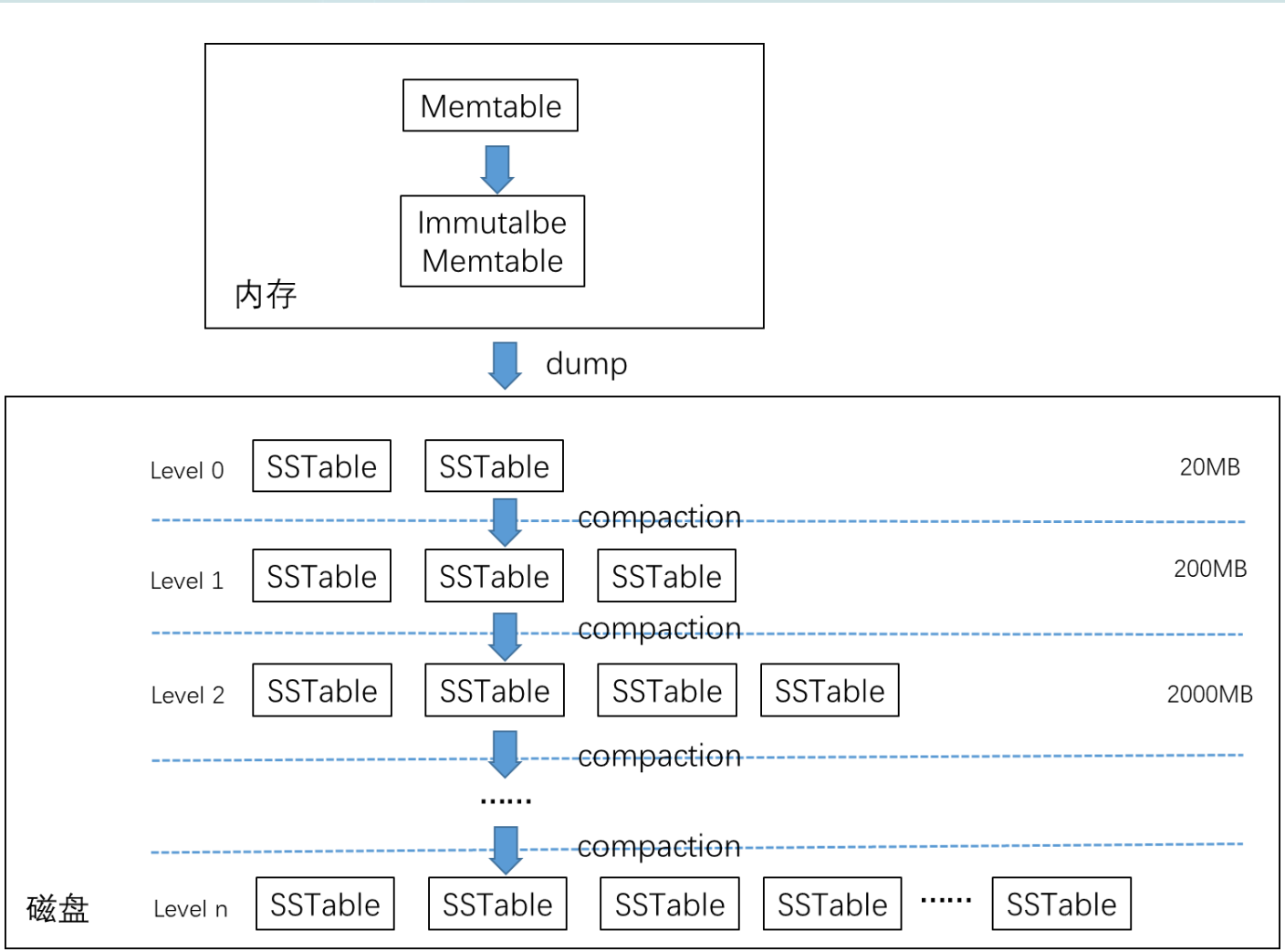
- SSTable（Sorted String Table）也称为有序字符串表，是一种持久化、有序且不可变的的磁盘键值存储结构。
- SSTable分为多个层级，从Immutable MemTable刷入的数据首先进入Level 0层，并生成相应的Sstable。

LSM树存放方式



- 当Level 0中的数据量达到某个阈值时，其中的SStable会合并到Level 1层，并以此方式逐层向下合并。
- Level层级数越小，表示处于该level的SSTable越新，level级数越大表示处于该level的SSTable越老
- 最大层级数由系统设定。

LSM树存放方式



- LSM树充分利用了磁盘批量顺序写性能远高于随机写性能
- NoSQL数据库，如Cassandra、RocksDB、HBase、LevelDB等，以及NewSQL数据库如TiDB，均使用LSM树来组织磁盘数据。

课堂思考

那些措施可以提高磁盘I/O性能？

第5章 存储管理

- 5.1 物理存储系统
- 5.2 数据组织
- 5.3 元数据存储
- 5.4 缓冲区管理
- 5.5 小结

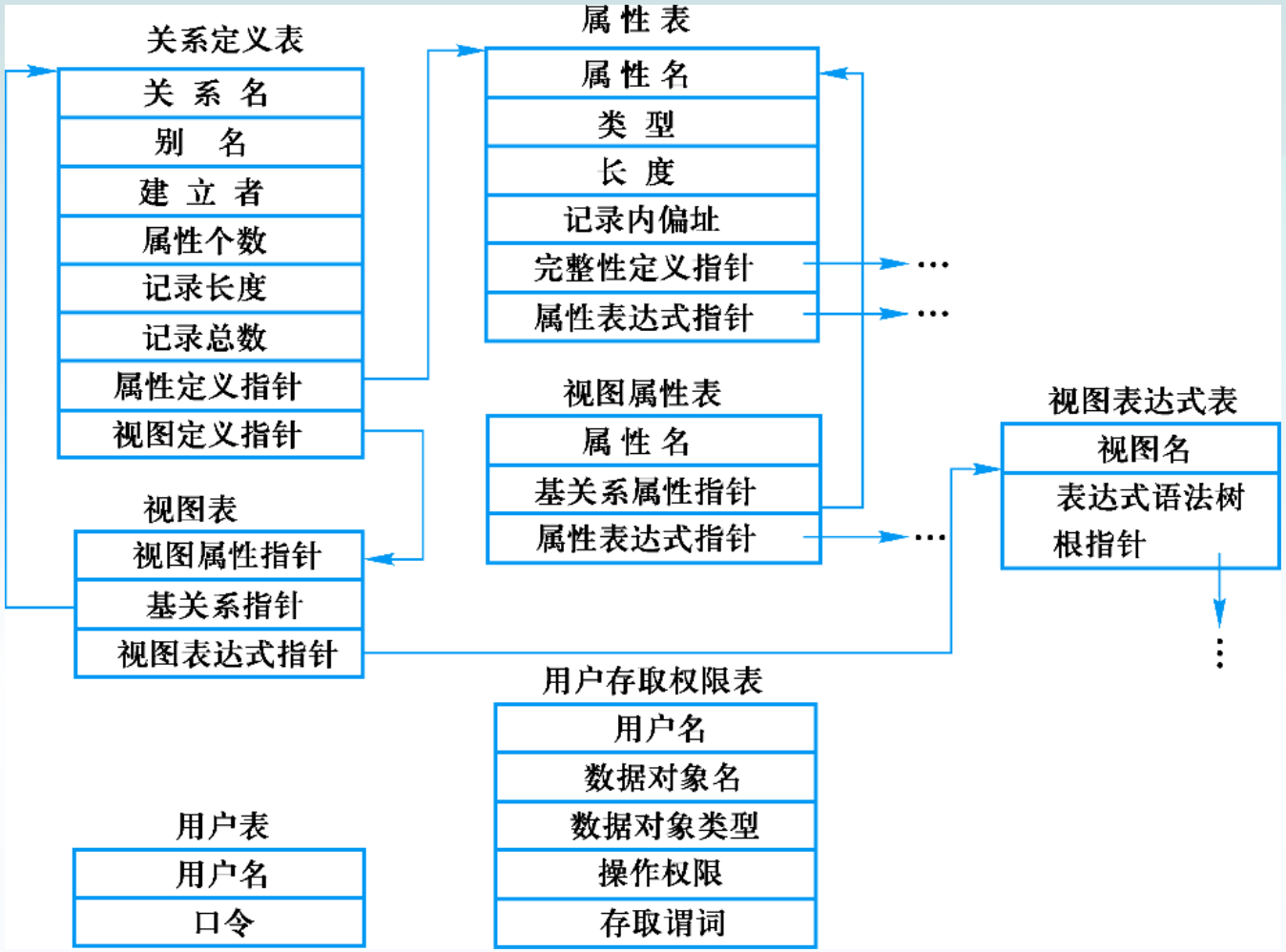
元数据存储

元数据：关于数据的数据

- 关系的信息，例如关系名称、类型、存储信息、属性信息、完整性约束等；
- 数据库中定义的索引、视图、触发器、存储过程、函数等信息；
- 用户的名称、密码、授权信息等；
- 关于关系和属性的系统统计信息等。

元数据存储

- DBMS通常把元数据存放在数据字典（data dictionary）或系统目录（system catalog）中。
- 在关系数据库中，数据字典的组织方式就是关系表。存放元数据的数据字典称为系统表。



第5章 存储管理

- 5.1 物理存储系统
- 5.2 数据组织
- 5.3 元数据存储
- 5.4 缓冲区管理
- 5.5 小结

缓冲区管理

- **DBMS设置缓冲区的原因**
  - 提供DBMS的设备独立性
    - 上层操作是基于系统缓冲的
    - 外存设备的变更不会对它们造成影响
  - 提高存取效率
    - 异步读写：预读，延迟写

缓冲区管理

- **缓冲区大小**
  - 太大：　占据内存空间
  - 太小：　频繁缺页调页，造成“抖动”，影响效率

缓冲区管理

- **两类缓冲区管理结构**
  - 缓冲区只在内存。RDBMS通常采用此方法。
  - 缓冲区在虚存中。由OS决定哪些缓冲区常驻内存，哪些在OS管理的磁盘上的“交换空间”中。内存DBMS和OODBMS通常采用此方法。

缓冲区的使用

- **读数据**
  - 当磁盘的页被缓存到缓冲区，DBMS用到的数据可以直接从缓冲区进行获取。
  - 如果请求的数据未能在缓冲区中找到，那么缓存管理器将从磁盘中缓存请求数据的页。
- **更新数据**
  - 修改数据也同样是在缓冲区里做修改，并标记该页为脏数据。
  - 脏数据在清空缓冲区以及页替换的时候必须被写回磁盘。
- **缓冲区置换**
  - 缓冲区满时，必须选择一个已缓存的页进行替换。替换策略对于缓存的性能会有很大的影响。

页面置换策略

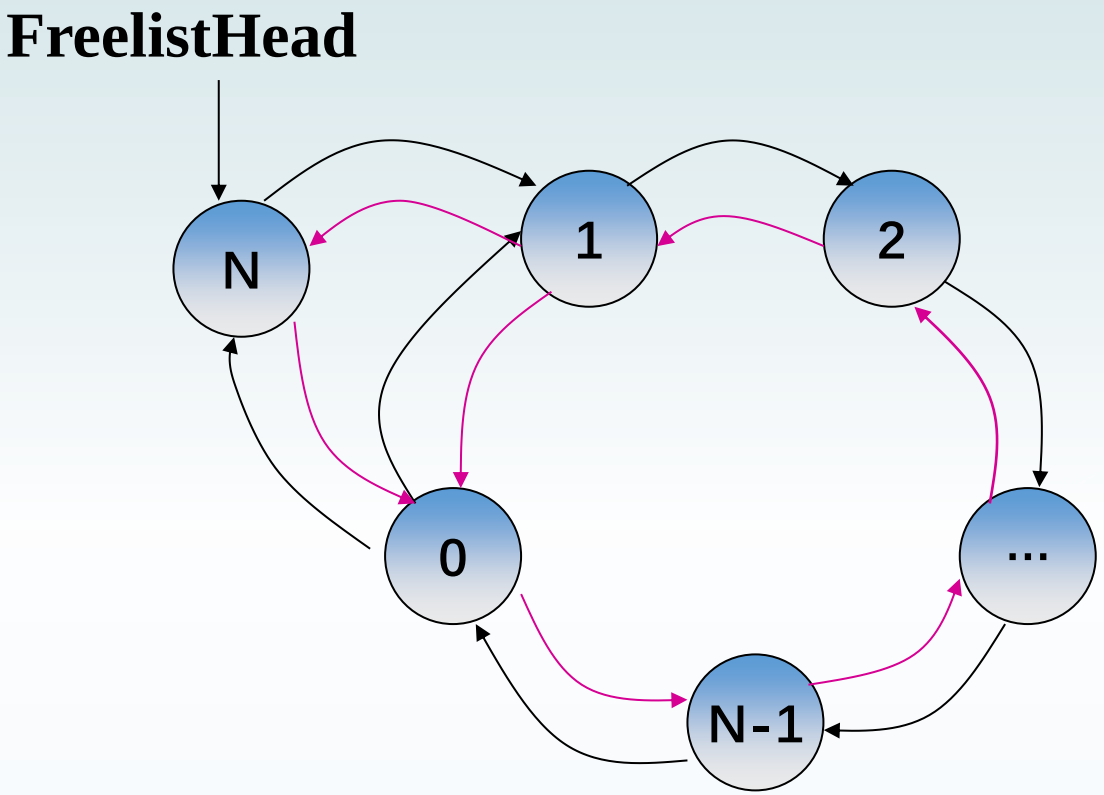
- **常用的缓冲区页面置换算法**
  - LRU算法
  - FIFO算法
  - 时钟算法(又称二次机会法SCH)：
  - RAND算法
  - 系统控制法
  - 混合算法

LRU算法

- 基本思想
  - 淘汰最长时间没有读或写过的缓冲块。
- 缓冲区管理器维护一张表，记录每个缓冲区块最后被访问的时间。或维护一个链表按访问时间来组织各缓冲块。
- LRU策略可以采用多种数据结构来实现，例如单链表、双链表、双链表和哈希相结合等。

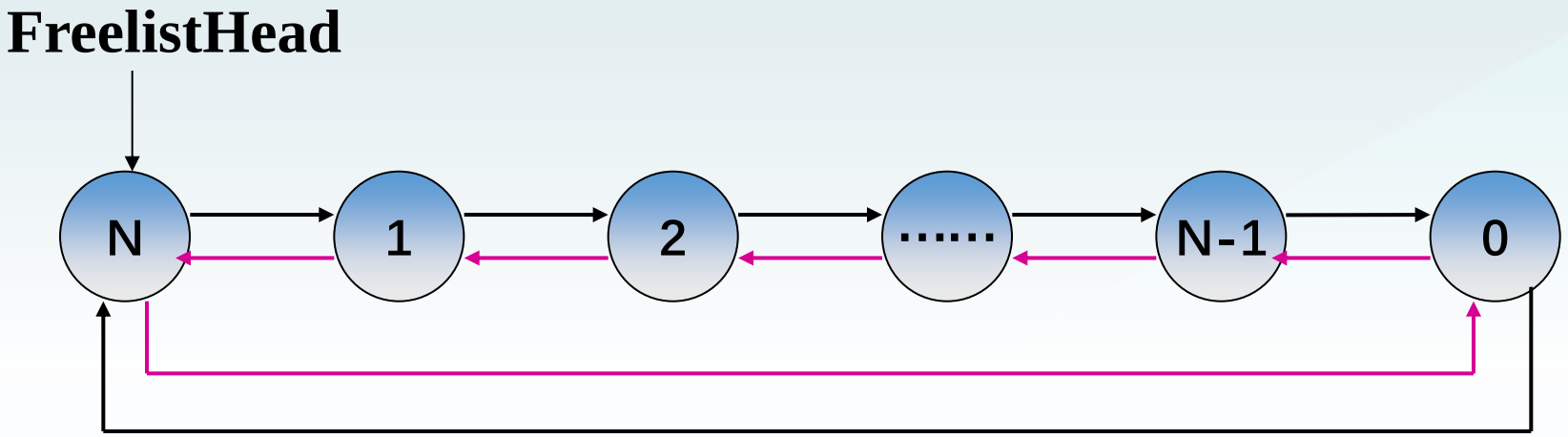
LRU算法–InitFreelist

例如，我们可以将系统中的N（0到N-1）个缓存块初始化成双向循环队列。链表中增加第N个节点，该节点是个虚拟节点，作为该链表的头FreelistHead。



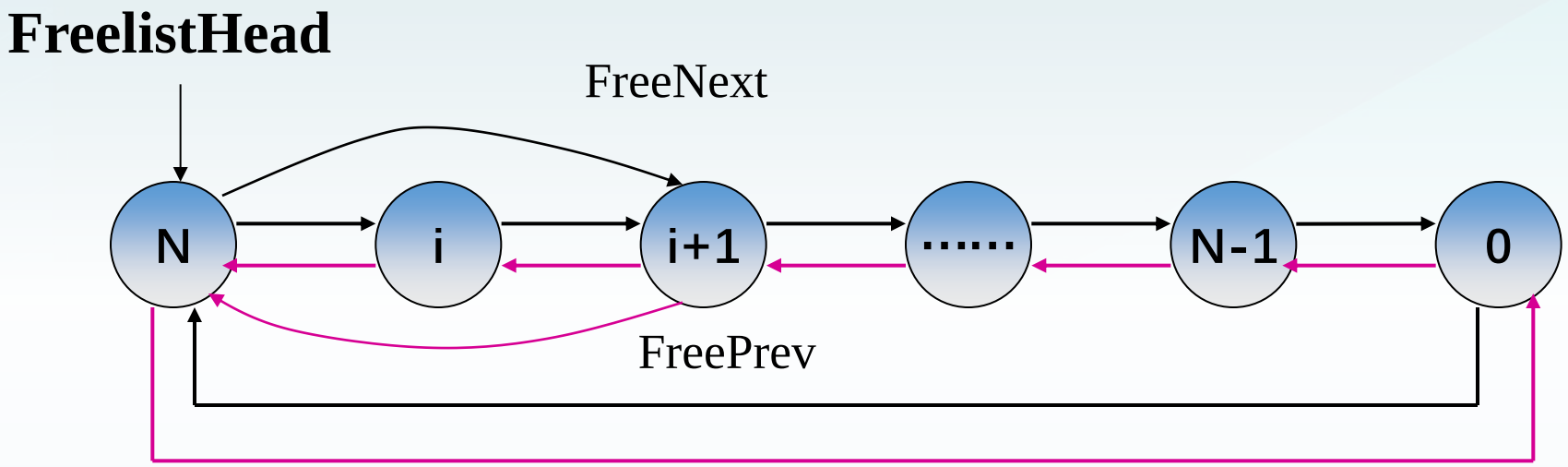
InitFreelist Cont.-变形

例如，我们可以将系统中的N（0到N-1）个缓存块初始化成双向循环队列。链表中增加第N个节点，该节点是个虚拟节点，作为该链表的头FreelistHead。



GetFreeBuffer(前->后)

当系统需要缓存块存放数据时，每次取走FreelistHead的FreeNext指向的位置，即下一个缓存块，把刚使用完的缓存块放在FreelistHead的FreePrev指向的位置。



FIFO算法

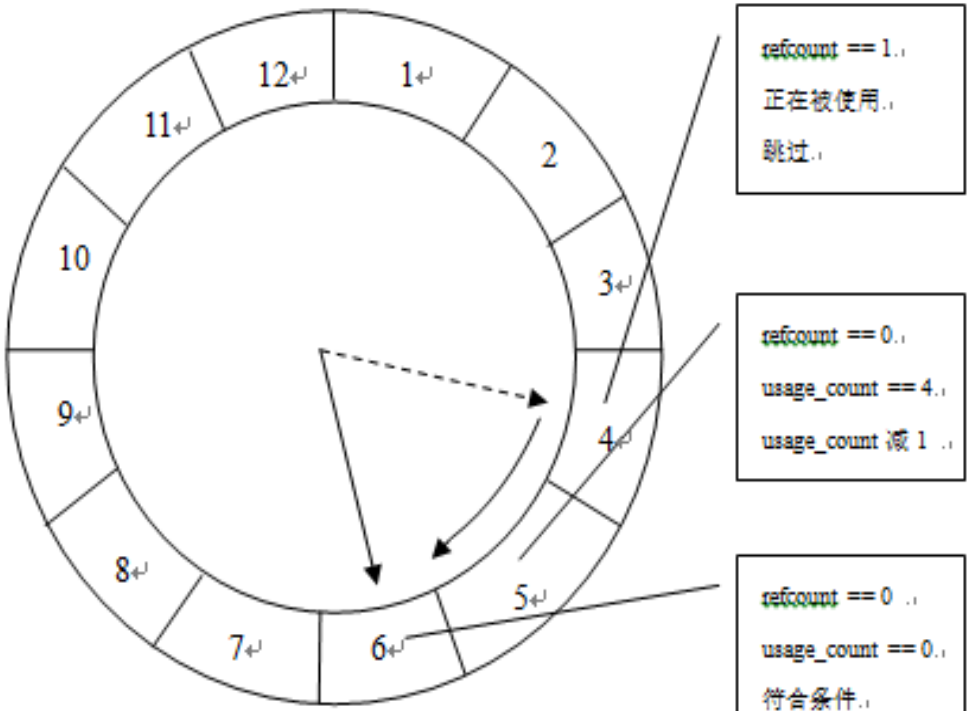
- **基本思想**
  - 淘汰最早读入缓冲区的块。

- **缓冲区管理器维护一张表，记录每个块读入缓冲区的时间。或维护一个链表，并用指针指向最老缓冲块。**

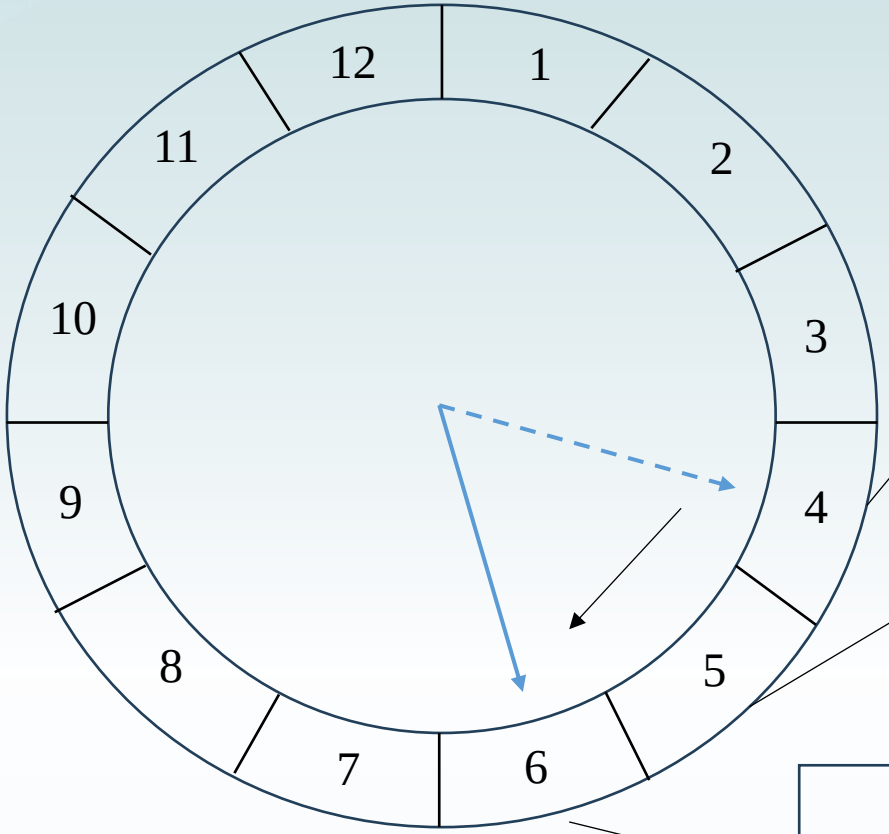
时钟算法(二次机会法SCH)

- **基本思想：FIFO+LRU**
  - 把所有的页面都保存在一个类似钟面的「环形链表」中，一个表针指向最老的页面。
  - 当一个块读入缓冲区时，其标志设为1
  - 当一个缓冲块被访问时，其标志设为1

- 淘汰时，按轮转方式(FIFO方式)查找缓冲区，淘汰第一个具有0标志的缓冲块。同时将查找过的标志为1的缓冲块置标志0。



时钟算法(二次机会法SCH)

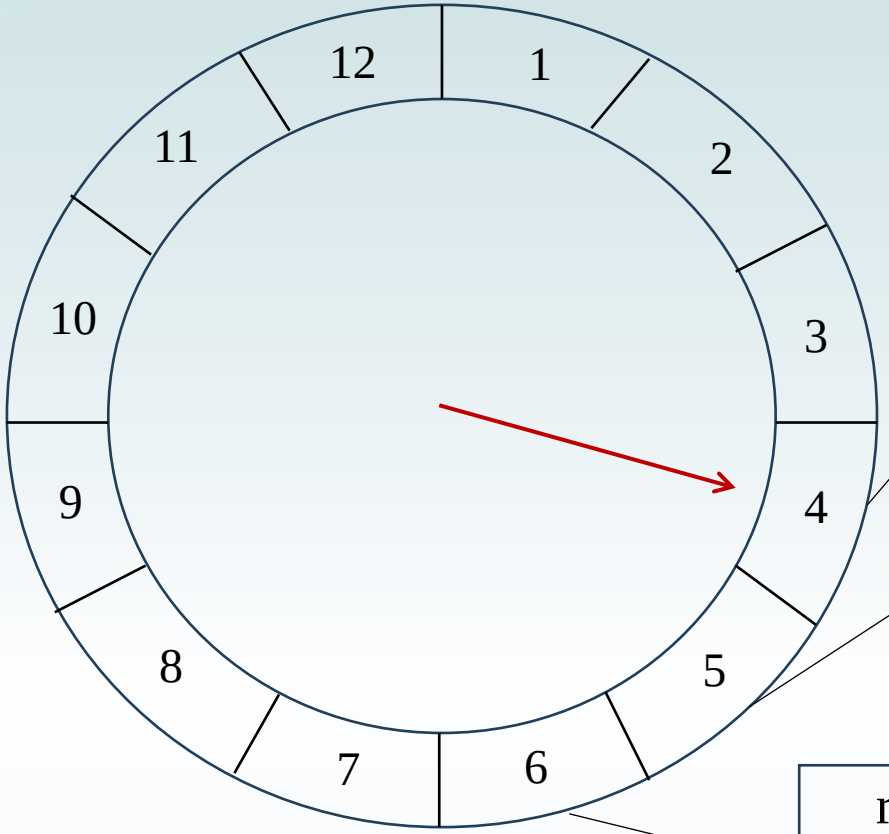


refcount == 1
正在被使用 跳过

refcount == 0
Usage_count == 4
Usage_count - 1

refcount == 0
Usage_count == 0
符合条件

时钟算法(二次机会法SCH)

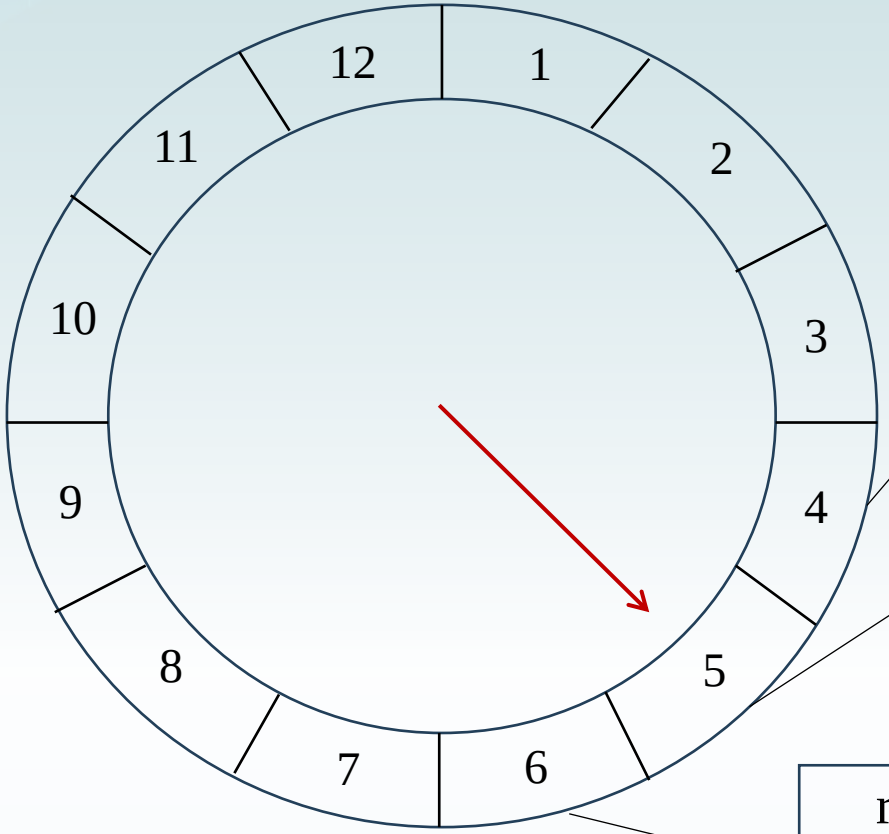


refcount == 1
正在被使用 跳过

refcount == 0
Usage_count == 4
Usage_count - 1

refcount == 0
Usage_count == 0
符合条件

时钟算法(二次机会法SCH)

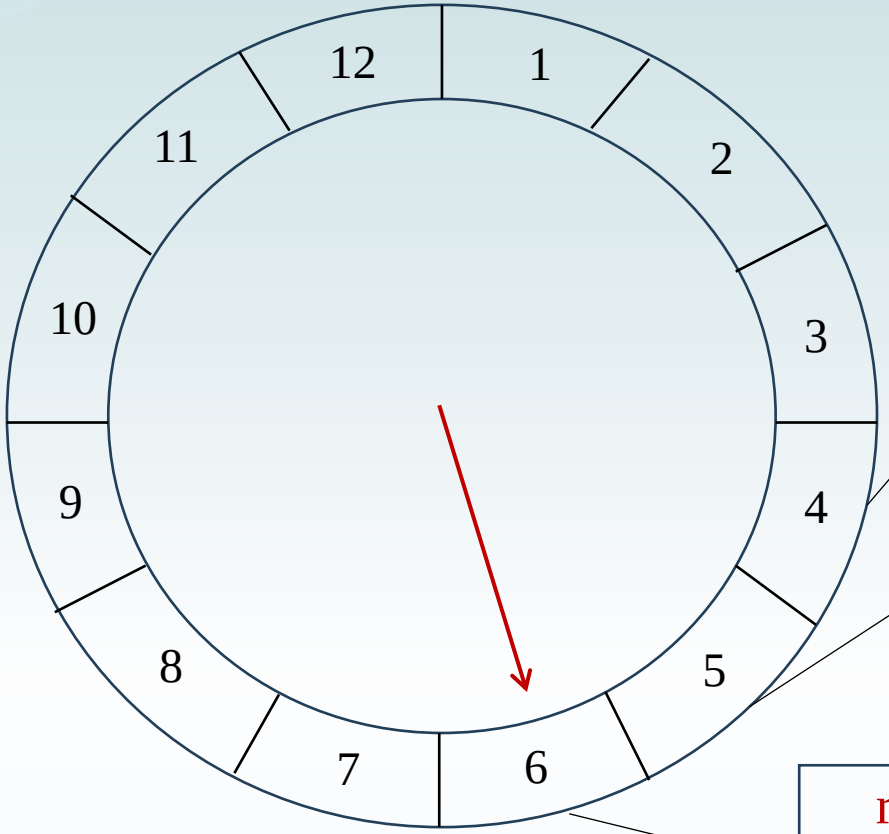


refcount == 1
正在被使用 跳过

refcount == 0
Usage_count == 4
Usage_count - 1

refcount == 0
Usage_count == 0
符合条件

时钟算法(二次机会法SCH)



refcount == 1
正在被使用 跳过

refcount == 0
Usage_count == 4
Usage_count - 1

refcount == 0
Usage_count == 0
符合条件

第5章 存储管理

- 5.1 物理存储系统
- 5.2 数据组织
- 5.3 元数据存储
- 5.4 缓冲区管理
- 5.5 小结

小结

- 磁盘I/O是影响系统性能的瓶颈，因此如何优化磁盘I/O受到极大关注。
- 数据库的数据组织方式包括逻辑组织方式和物理存储方式。逻辑上，常见方式是将数据库组织成“表空间-段-分区-数据块”的形式。物理上，数据库中的数据最终是以文件的形式存储在磁盘上。
- 缓冲区是数据库存储管理中的另一个重要问题，缓冲区的管理策略以及页面转换策略是其中的关键技术，也是本章的重点和难点内容。