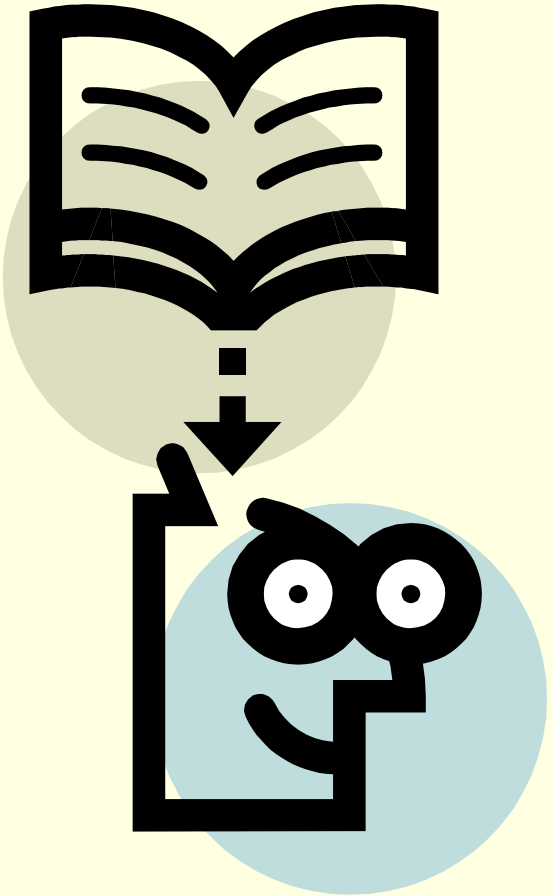
第5章 数据库的存储结构
Chapter 5 Database Storage Structure

目录 Contents　本章从用户角度来介绍

- **5.1 数据库存储结构的特点**
  - **多级存储**
  - **物理结构**
  - **逻辑结构**
- **5.2 关系表的典型存储机制**
  - **索引**
  - **散列**
  - **簇集**



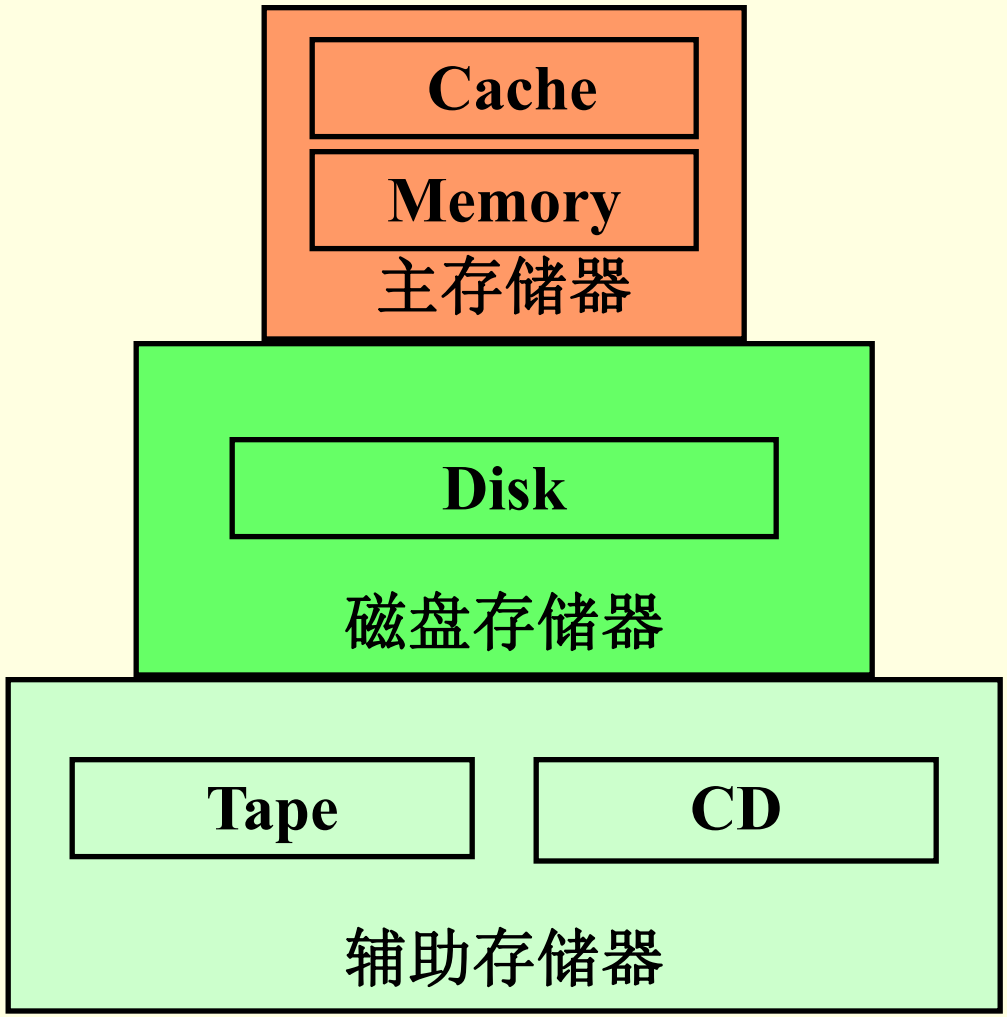
5.1 数据库存储结构的特点

- 一、多级存储
  - 用**内存（RAM）**作为数据库的存储介质是不合适的。
    - 内存的容量尚不够以存储数据库中的全部数据
    - 内存是易失性存储器（volatile storage），不适合持久存储数据
    - 内存存储单位数据的成本要比外存高得多（约1000倍）
  - 因此，目前的数据库系统主要采用多级存储结构。
    - 二级存储：内存（主存）— 外存（磁盘）
    - 三级存储：内存（主存）— 外存（磁盘）— 辅存（磁带等）

5.1 数据库存储结构的特点

- 三级存储结构
  - 第一级：主存储器 （main memory）
    - 高速缓冲存储器（cache）
    - 主存储器（memory）
  - 第二级：次级存储器（secondary storage）
    - 也称：二级存储器，通常是磁盘存储器
  - 第三级：辅助存储器（tertiary storage）
    - 也称：三级存储器，是辅助存储设备，如：磁带存储器、自动光盘机。

5.1 数据库存储结构的特点



5.1 数据库存储结构的特点

	存储容量	访问速度	访问类型	存取单位
第一级存储器	100MB ~ 10GB	10^{-8}秒 ~ 10^{-7} 秒	随机	字节
第二级存储器	10GB ~ 10^{3}GB	5毫秒 ~ 30 毫秒	随机	物理块
第三级存储器	10^{6}GB 以上	几秒钟 ~ 几分钟	顺序	数据块

5.1 数据库存储结构的特点

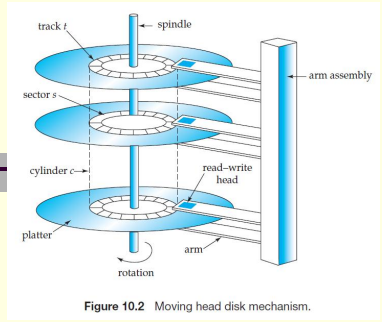


Figure 10.2 Moving head disk mechanism.

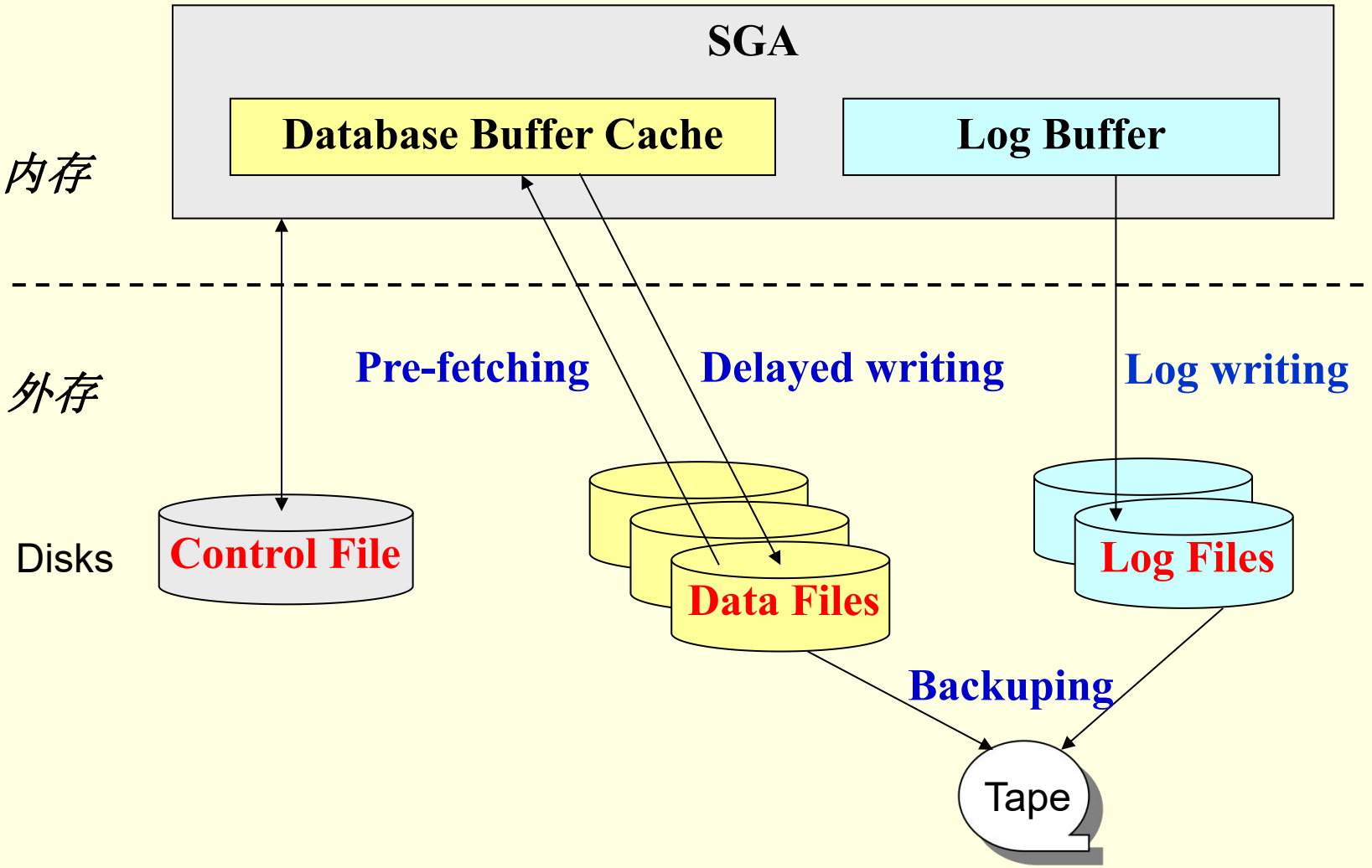
- 磁盘的存取速度与内存的存取速度严重不匹配。
- 磁盘的I/O操作：首先根据磁盘物理块（磁盘与内存之间进行数据交换的基本单位）的地址来定位，然后读/写物理块中的数据，存取时间（access time）包括三个部分：
  - 寻道时间（seek time）：磁头臂的机械移动时间
  - 旋转延时时间（rotational delay time）：磁盘片的旋转定位时间
  - 传输时间（transfer time）：读/写数据时间
- 为了有效支持数据库的数据读写、提高数据库性能， DBMS必须在内存开辟物理块的缓冲区（buffers），并采用数据预取（pre-fetching）和延时写（delayed writing）技术，以减少I/O操作次数、提高外存数据的存取速度。

5.1 数据库存储结构的特点

二、物理结构

- 数据库以多个文件（files）的形式进行组织，并物理地存储于磁盘介质上。存储空间及文件由DBMS的存储管理器进行管理。（注：OS为DBMS提供底层支持）
- 通常，一个数据库有三种文件：
  - 数据文件（Data Files）：用于保存数据库中的数据与元数据，一个数据库对应一个或多个数据文件。
  - 日志文件（Log Files）：用于保存用户存取数据库的日志记录，一个数据库对应一个或多个日志文件。
  - 控制文件（Control Files）：用于保存与数据库有关的若干参数（如：数据库名、数据文件和日志文件的名字和位置，数据库的建立日期，等），一个数据库对应一个控制文件。

5.1 数据库存储结构的特点



5.1 数据库存储结构的特点

三、逻辑结构

- 数据库用户并非直接与数据库的物理结构（即：物理存储介质或物理文件）打交道；
- DBMS的存储管理器提供物理←→逻辑的映像（mapping），使得用户直接面对数据库的逻辑结构。

- 逻辑结构涉及两个方面：
  ① 数据库的存储空间如何被逻辑地划分与组织（即逻辑存储空间）；
  ② 用户如何使用数据库中的数据（即用户模式及其对象）。

5.1 数据库存储结构的特点

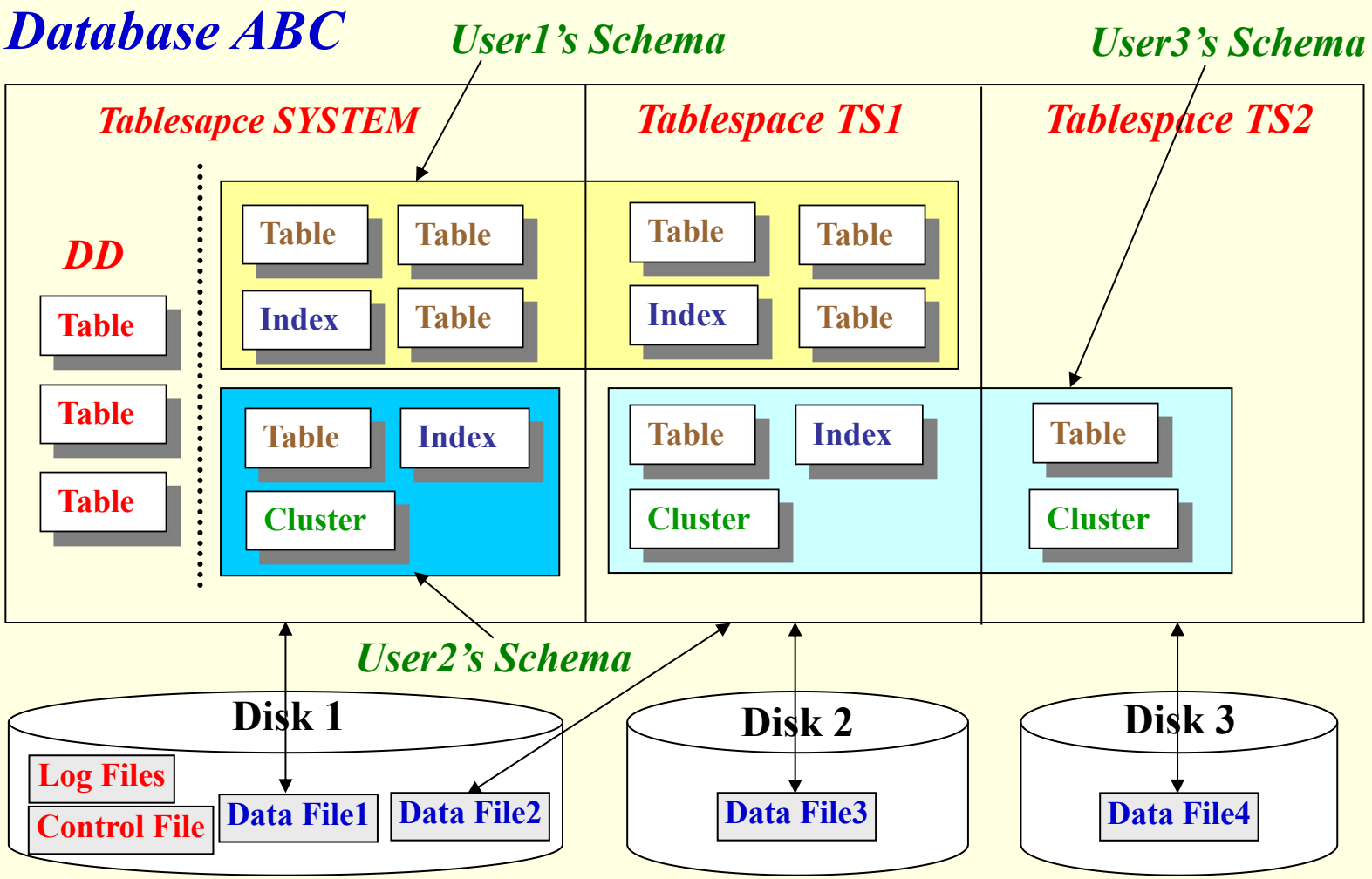
三、逻辑结构（续）

- 逻辑存储空间：（以Oracle为例介绍）
  - 表空间（Table Space）：表空间是数据库的逻辑存储单位。一个数据库可包含一个或多个表空间；一个表空间可跨多个磁盘进行分配。一般地，在数据库初始化时，DBMS自动建立一个缺省表空间（如Oracle中SYSTEM表空间），DBA事后可定义其他表空间。
  - 段（Segment）：段是表空间中一种指定类型的逻辑存储结构。有：
    - 数据段：每个表/簇集有一个数据段，用于存储其中的数据。
    - 索引段：每个索引有一个索引段，用于存储索引数据。
    - 回滚段：由DBA建立，用于临时存储事务回滚中需撤消的信息。
    - 临时段：当SQL语句需临时工作区时，由DBMS建立，用完后回收。
  - 范围（Extent）：一个段由一组范围组成，是数据库进行存储空间分配的逻辑单位。
  - 数据块（Data Block）：一个范围由一组连续的数据块所组成，数据块是DBMS进行I/O的最小物理单位，其大小可异于OS的标准I/O块大小。

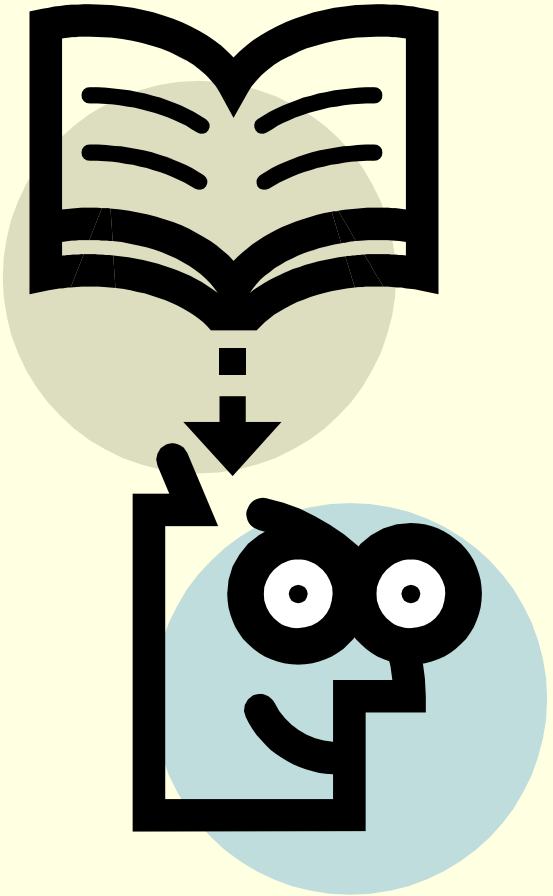
5.1 数据库存储结构的特点

- 三、逻辑结构（续）
  - 用户模式及其对象
    - 模式（Schema）： 每个数据库用户对应一个模式。
    - 模式对象（Schema Objects）： 一个模式中包含的数据逻辑结构对象包括：表，视图，索引，簇集，过程、触发器，等。

5.1 数据库存储结构的特点



目录 Contents



5.2 关系表的典型存储机制

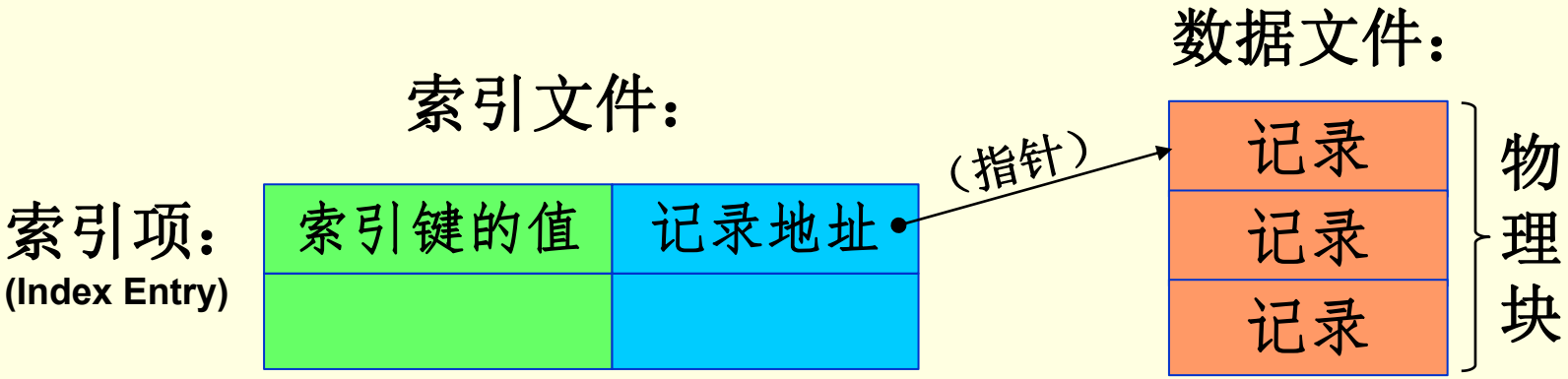
- 数据文件中记录的组织方式：
  - 堆文件（Heap File）组织：数据记录按照其插入的先后顺序进行存放（“堆放”）
    - 堆文件中的记录往往存放在不连续（地址不相邻）的物理块
    - 在堆文件中查找数据只能采用顺序扫描（sequential scanning）
  - 顺序文件（Sequential File）组织：数据记录按照某个属性值（如主键值）进行排序后再存放
    - 在顺序文件中查找数据可以采用更为快速的【时间复杂度为O(log n)】二分搜索（binary search）又称折半搜索（half-interval search）又称对数搜索（logarithmic search）：搜索键是排好序的属性值（如主键值）
  - 散列文件（Hashing File）组织：在每个记录的某些属性上计算哈希函数，函数的结果指定应将记录放置在文件的哪个块中。这种组织方式与索引结构密切相关。
    - 在Hash文件中查找数据记录通常非常直接快速。

5.2 关系表的典型存储机制

- 数据查找速度对数据库的性能至关重要！
- 当数据库的数据量较大（很大）时，数据查找速度就会受到（很大）影响。因此数据库中需引入高效率的表数据存储机制/存取路径：
  - 索引（Indexing）
  - 散列（Hashing, a.k.a. Hashing file organization ）
  - 簇集（Clustering, a.k.a. Multitable clustering file organization）

5.2 关系表的典型存储机制>>索引

- **索引**（Indexing）是与表（或簇集）相关的一种可选存储机制，它通过一棵有序树（如B树）将**索引键**（Index Key）的值与该值所对应的数据记录的物理地址（简称：记录地址）或地址集建立联系，以提高数据查找速度。



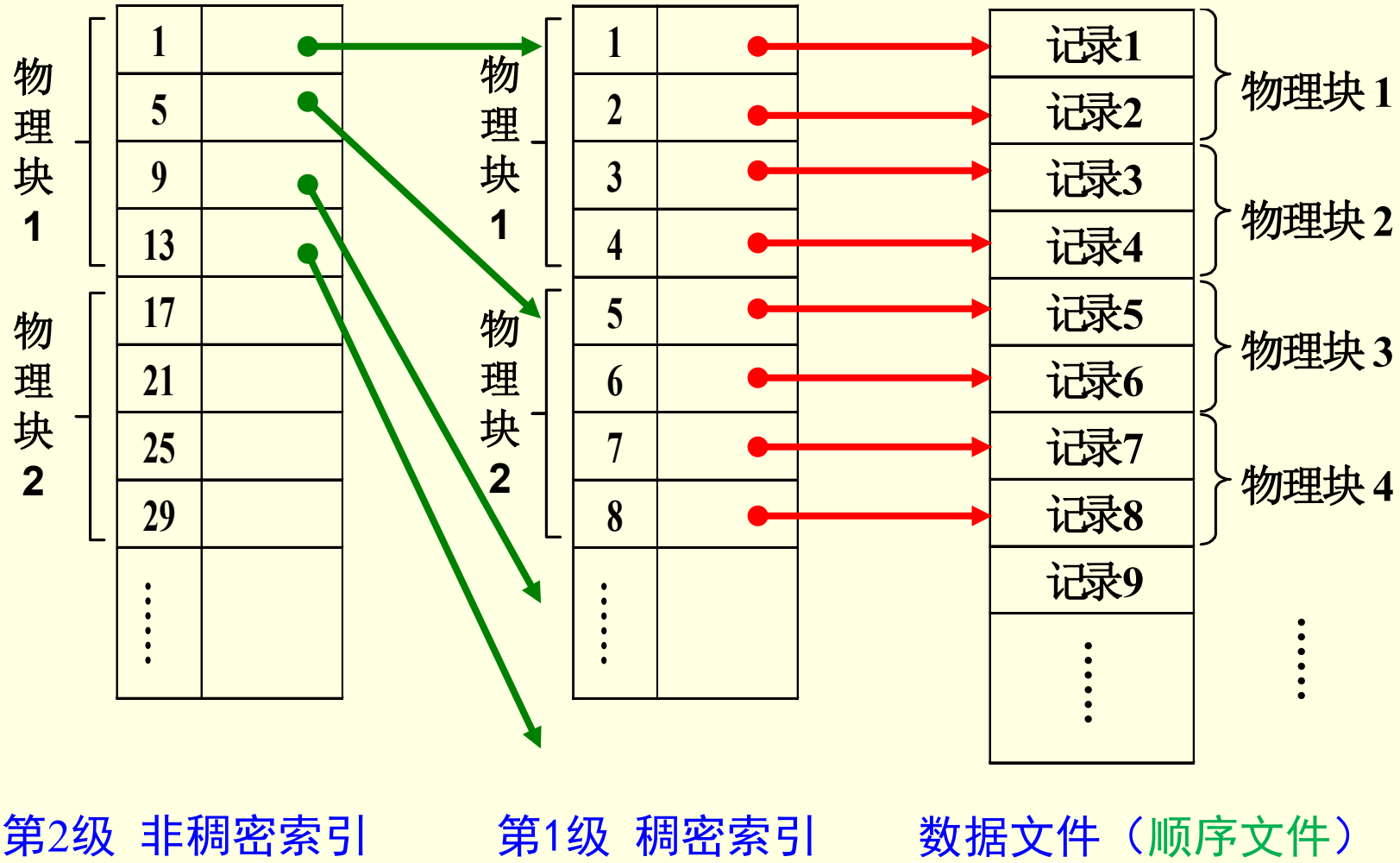
5.2 关系表的典型存储机制>>索引

- 索引的基本概念
  - 索引键如是PK或UNIQUE列，称主索引（primary index）或唯一索引（unique index），否则称次索引（secondary index）
    - 一般地，主索引由DBMS自动建立
  - 索引键可以是由多个属性所组成的一个属性组，此时称组合索引（composite index）
  - 若每个索引键值均有一个索引项，则称这样的索引结构为稠密索引（dense index），否则称为非稠密索引（nondense index）或稀疏索引（spare index）
  - 稠密索引 vs. 非稠密索引：
    - 非稠密索引的索引项所占的存储空间少于稠密索引。但是，非稠密索引只能用于对顺序文件进行索引
    - 稠密索引中的每个索引项对应数据文件中的一条数据记录（物理地址），而非稠密索引中的每个索引项对应数据文件中的一个物理块（地址）
    - 利用稠密索引可快速访问数据记录；非稠密索引需要额外的磁盘I/O操作，即需要将数据文件中的物理块读入内存后才能判别数据记录是否存在

5.2 关系表的典型存储机制>>索引

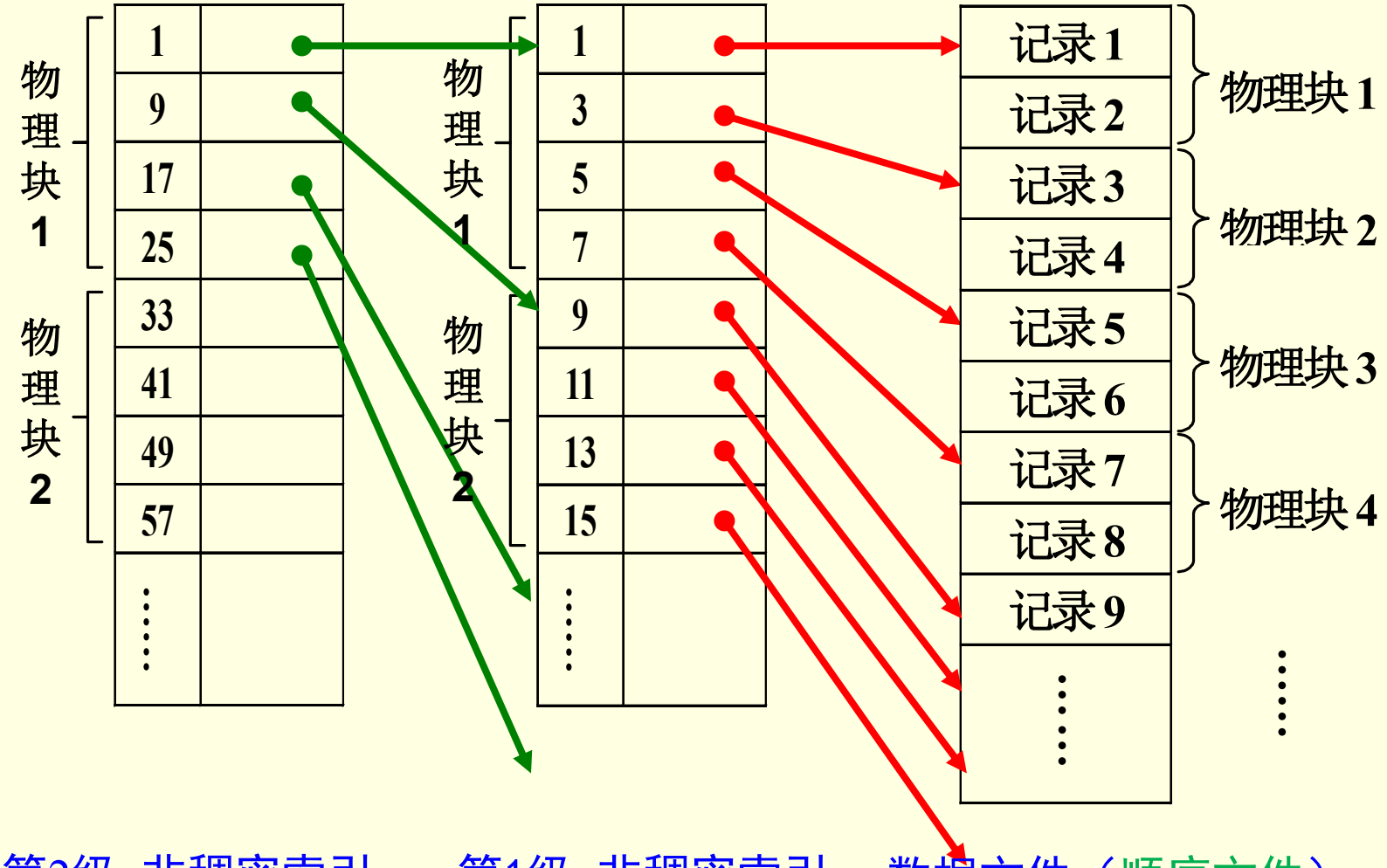
- 在顺序文件上的索引结构
  - 可选的索引结构：稠密索引；非稠密索引；多级索引
  - 多级索引：
    - 索引文件（即索引项数据）本身一般也会占据很多物理块。为了能快速定位到这些物理块的存储位置，需要为索引项建立索引，即：在索引文件上再建立索引，从而构成了多级索引
    - 由于索引文件本身是顺序文件，因此，在索引文件上通常再建立非稠密索引就可以了
    - 在多级索引中，第一级索引可以是稠密索引或非稠密索引（注：非稠密索引只能用于对顺序文件进行索引）；从第二级索引开始建立的都是非稠密索引

5.2 关系表的典型存储机制>>索引



第2级 非稠密索引　　第1级 稠密索引　　数据文件（顺序文件）

5.2 关系表的典型存储机制>>索引

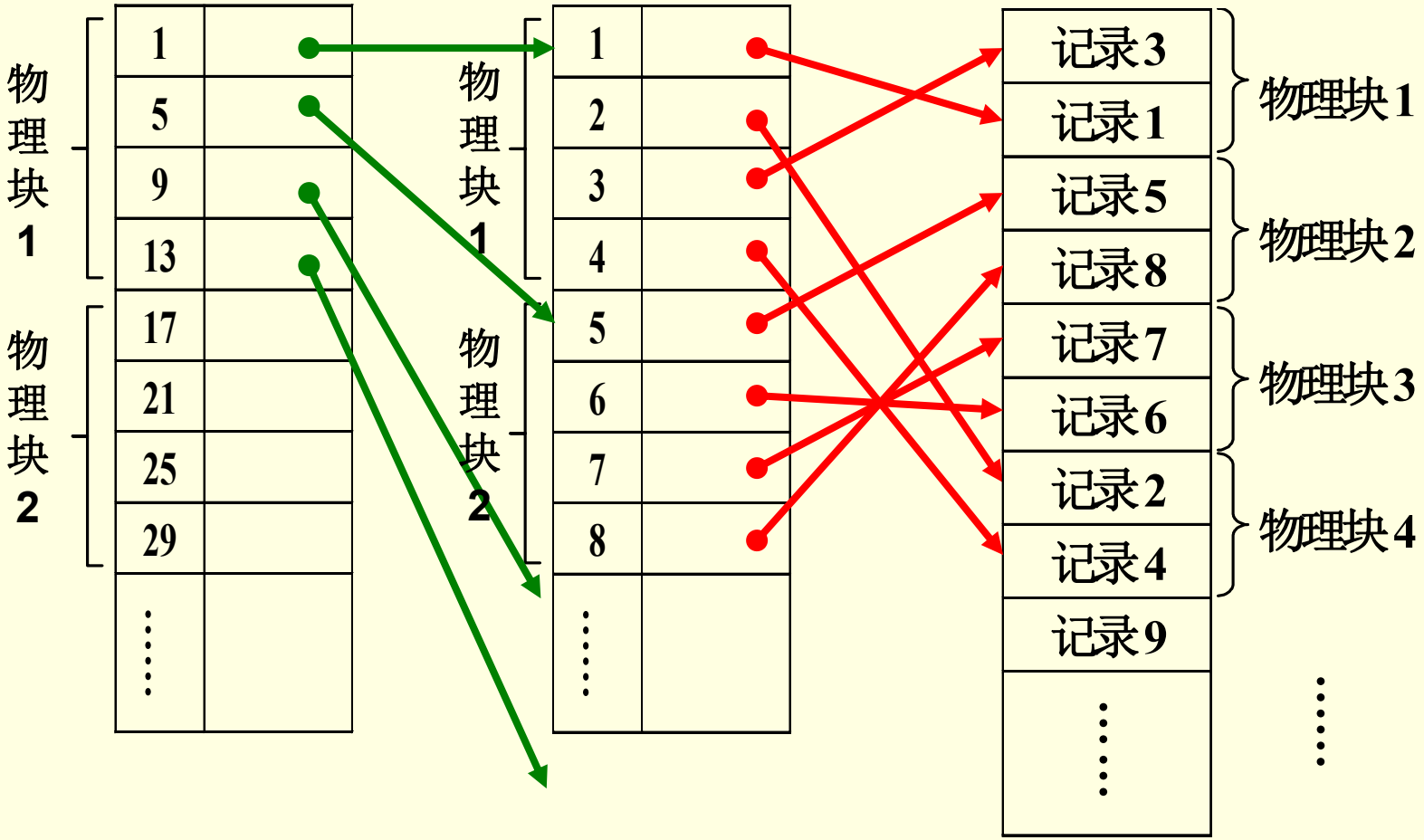


第2级 非稠密索引　　第1级 非稠密索引　　数据文件（顺序文件）

5.2 关系表的典型存储机制>>索引

- 在非顺序文件上的索引结构
  - 数据库系统中使用的数据文件常常是无序的（如堆文件），因此更加需要在非顺序文件上建立索引，以提高记录查找速度。
  - 如果索引键值在非顺序数据文件中具有唯一性，则可按下列步骤来建立索引文件：
    - 首先，为非顺序数据文件建立第一级稠密索引
    - 然后，再根据需要建立该稠密索引上的（多级）非稠密索引

5.2 关系表的典型存储机制>>索引

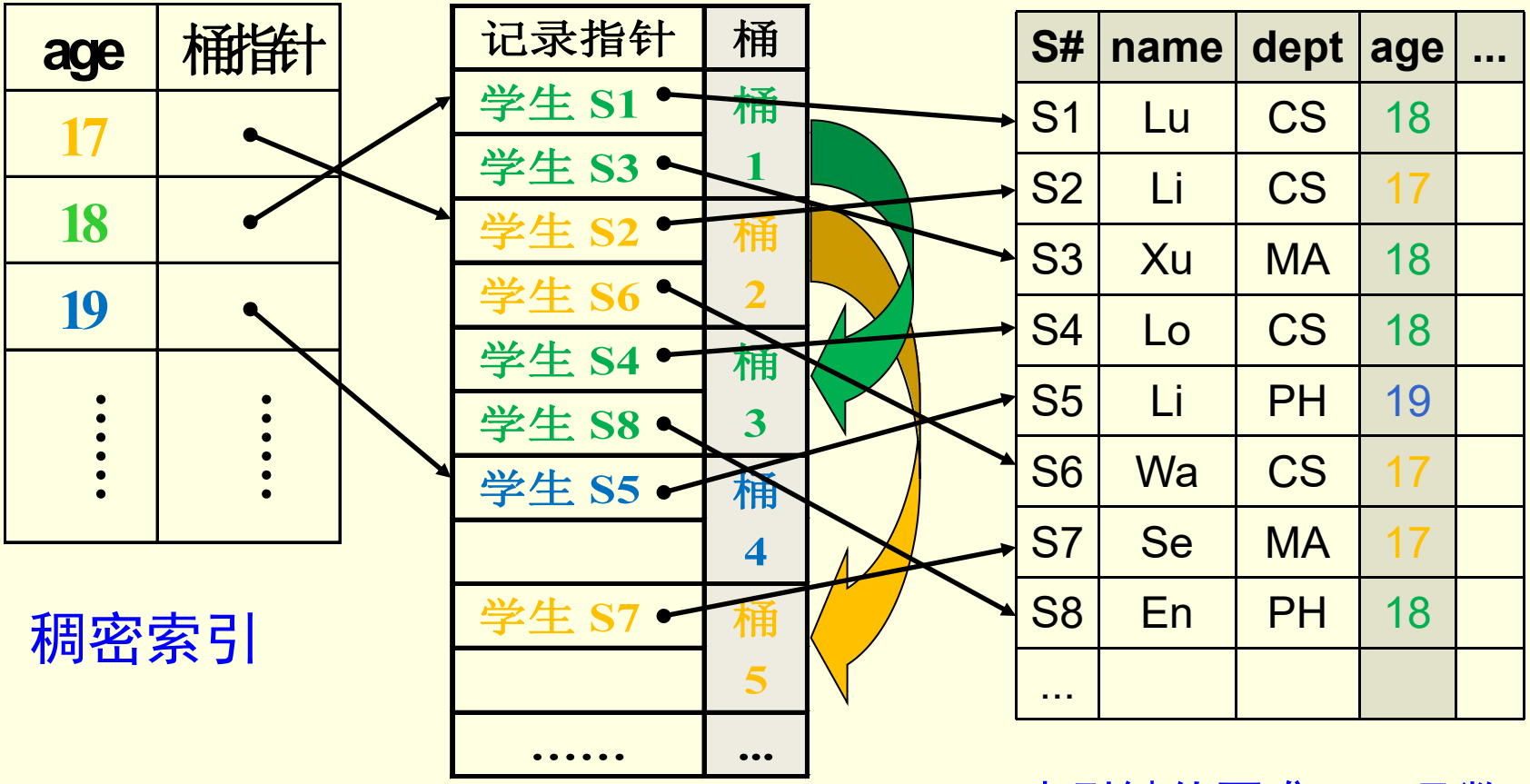


第2级 非稠密索引　　第1级 稠密索引　　数据文件（堆文件）

5.2 关系表的典型存储机制>>索引

- 在次索引中，索引键值不唯一，则可以在第一级稠密索引和数据文件之间加一个记录指针桶（bucket）。此时，索引项(K_i, P_i)中的记录指针P_i不再指向数据文件中的记录，而是指向一个记录指针桶，在该桶中存放着指向索引键值为K_i的记录的记录指针。
  - 记录指针桶的大小 BSize 是预先确定的。
  - 当在数据文件中插入第一条索引键值为 K_i 的记录 R_i 时，在索引文件中将生成一个新的索引项(K_i, P_i)，同时系统将自动申请一个大小为 BSize 的记录指针桶 B_i，将指针 P_i 指向该记录指针桶 B_i，同时将当前记录 R_i 的记录指针保存于记录指针桶 B_i 中。
  - 当新的索引键值为 K_i 的记录被添加到数据文件中时，只需将新记录的记录指针添加到记录指针桶 B_i 中。如果记录指针桶 B_i 已满，系统将申请一个新的记录指针桶，并链接到原来的记录指针桶的后面。

5.2 关系表的典型存储机制>>索引



稠密索引

记录指针桶

索引键值不唯一，且数据文件没有按照索引键值来排序存储

5.2 关系表的典型存储机制>>索引

- B 树
  - B树（B-tree）是一种动态平衡多叉树（dynamic balanced multiway tree）。B树有几个变种，如：B+树、B*树，等
  - 运用B树的索引结构总是动态索引
  - B树中叶结点所构成的最下层的一级索引总是采用稠密索引，而其它层次上的索引常常采用非稠密索引
  - B树（即索引）维护由DBMS完成，对数据库用户是透明的
  - B^+树中的结点：
    - 每个结点占用一个物理块，其中能容纳 n 个键和 n+1 个指针
      将 n 取得尽可能大，以便在一个物理块中存放更多的索引项
    - B^+树的结点的结构如下：

P_1	K_1	P_2	K_2	…	P_n	K_n	P_{n+1}

5.2 关系表的典型存储机制>>索引

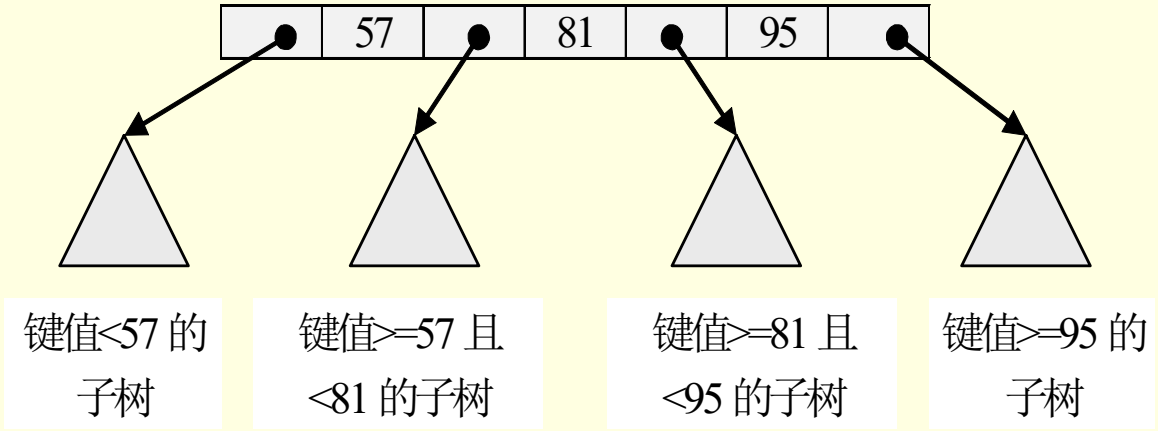


图 1　B⁺树的某个内部结点

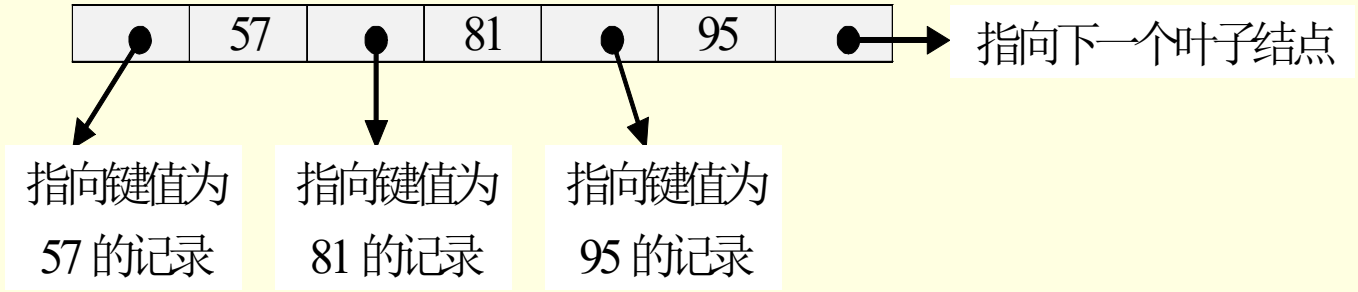
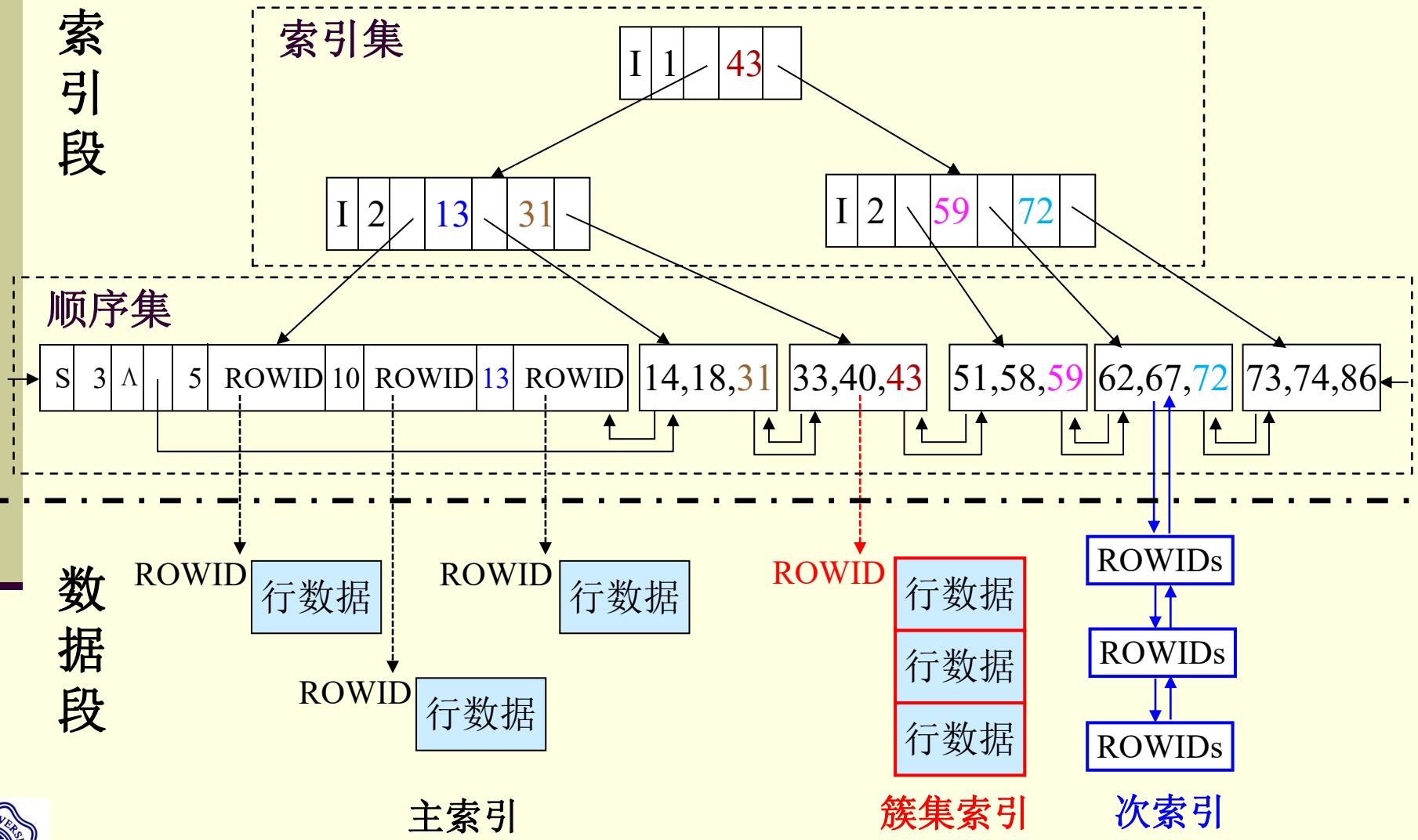


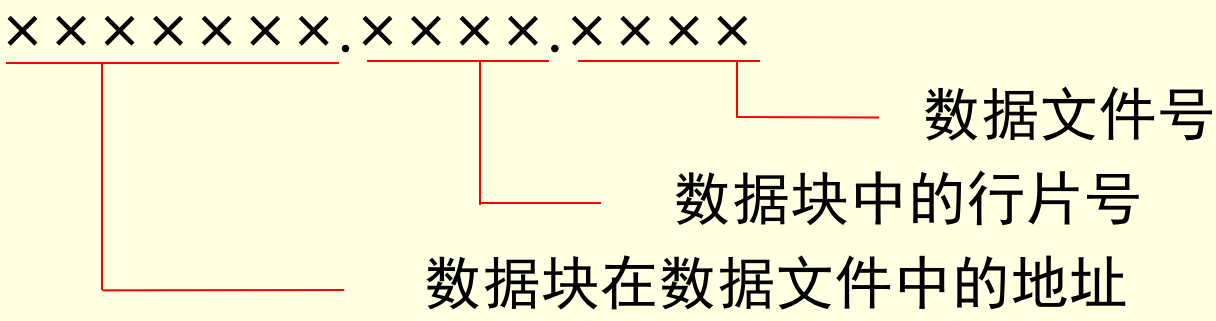
图 2　B⁺树的某个叶子结点

例：设索引键值集为：{5，10，13，14，18，31，33，40，43，51，58，59，62，67，72，73，74，86}，Oracle数据库的B*树索引结构：



5.2 关系表的典型存储机制>>索引

- 注：Oracle中的ROWID是16进制位串：



- ROWID是系统附加在表行上的伪列（pseudo column）：

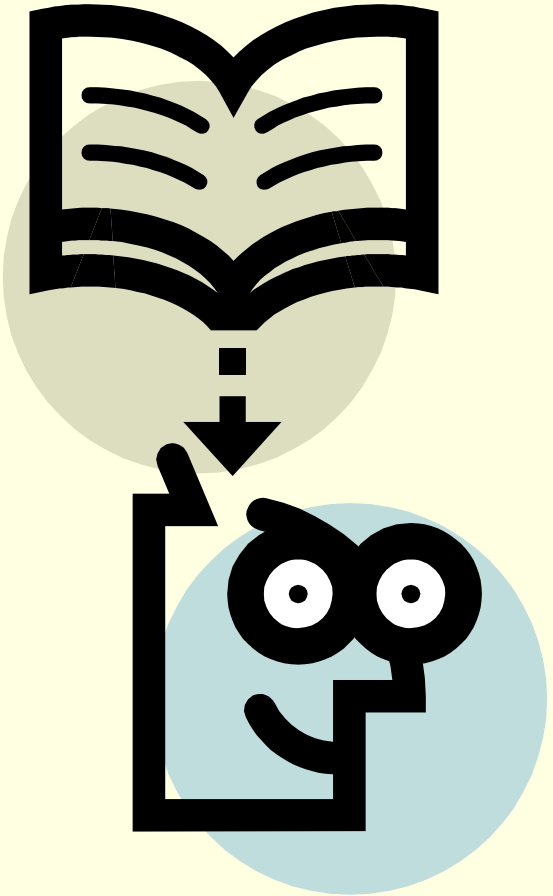
```
SELECT *  FROM emp
WHERE ROWID = '0000DC5.0001.0001';

SELECT ROWID, ename  FROM emp
WHERE empno=100;
```

5.2 关系表的典型存储机制>>索引

- **B树索引的利弊**
  - 利
    - 当表的容量较大时（需占用较多物理块时），索引能明显提高数据查询性能——最理想的块I/O次数为：B树深度+1。
    - 索引特别适合于指定行查询和范围查找。
  - 弊
    - 增加了系统维护的开销，特别是当表的数据需频繁更新时。

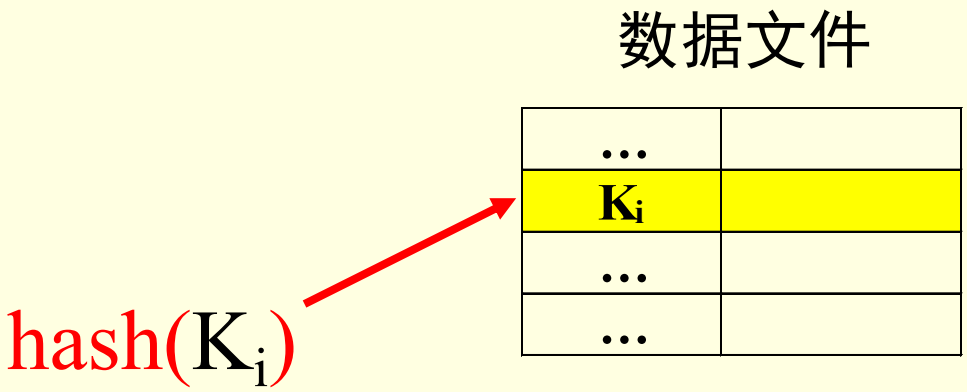
目录 Contents

- 5.1 数据库存储结构的特点
  - 多级存储
  - 物理结构
  - 逻辑结构
- 5.2 关系表的典型存储机制
  - 索引
  - 散列
  - 簇集



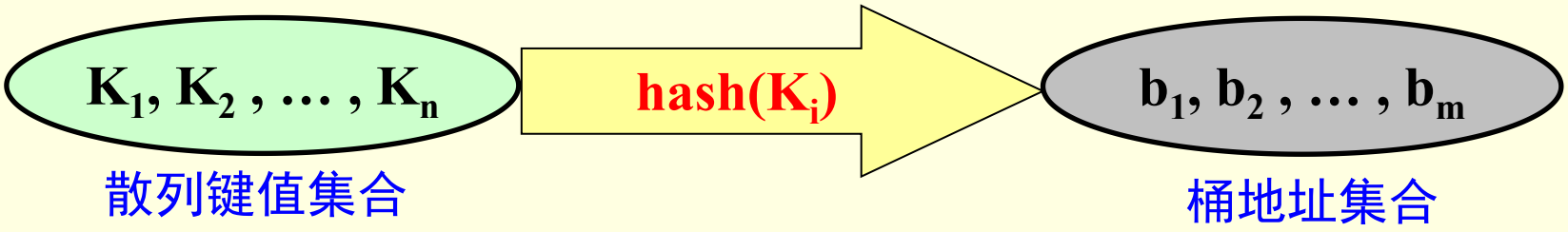
5.2 关系表的典型存储机制>>散列

- 散列（Hash）
  - 是与表（或簇集）相关的一种可选存储机制，由于可通过一个Hash函数将**散列键**（Hash Key）的值映射成一个物理块（数据块）的地址，因此，给出散列键的值K_i后立即可通过$hash(K_i)$得到其对应的物理块地址，从而可明显改进数据检索的性能。



5.2 关系表的典型存储机制>>散列

- 静态散列的实现方法
  - 确定数据文件的散列键K以及该键值的集合$\{K_1, K_2 \ldots, K_n\}$
  - 建立磁盘物理存储单位桶（bucket）以及桶地址的集合$\{b_1, b_2, \ldots, b_m\}$
    - 一个桶可以存放多条记录（或记录指针）
    - 一个桶可以是一个磁盘物理块，也可以是更大的物理空间
  - 设计散列函数$hash(K_i)$
    - 以便建立散列键值与桶（桶地址）之间的对应关系，即一个K_i通过$hash(K_i)$必能找到唯一的一个桶地址
    - 使得n个键值被平均分配到m个桶中去



5.2 关系表的典型存储机制>>散列

- 在散列技术中，桶的空间大小是固定的，即一个桶中可以存放的记录（指针）数是固定的。
- 但是在实际应用中，记录在散列键值上的分布往往是不均衡的，从而使得在某些桶中存在空间浪费现象，而另外一些桶则存在空间溢出问题。
- 当一个桶的空间溢出时，需要通过链接的方法申请“溢出桶”与其相连，以达到扩大桶空间的目的。

5.2 关系表的典型存储机制>>散列

- 静态散列技术的优缺点
  - 优点
    - 按Hash键值访问数据，速度快。
  - 缺点
    - Hash键值映射的地址空间有限
    - 用Hash键值寻址时，同一个Hash键值可能对应多个记录，不同的Hash键值可能映射到同一个地址
    - 只对从Hash键值到记录的访问有效，对其他类型的访问不一定有效
    - 处理变长记录不便
    - 很难找到通用的Hash函数
    - 由于当桶装满后的溢出处理较为复杂等原因，在数据经常变动的数据库环境中不宜使用静态散列，因此，使用动态散列（桶的数目可动态变化；桶可分裂/合并）较普遍

5.2 关系表的典型存储机制>>散列

- 在理想下，基于散列键的单行查询只需一次物理块I/O即可。但仅靠“散列”机制带来好处的情况并不多， 故一般在关系数据库系统中，“散列”只用于一种场合：散列簇集 。
——例如：Oracle就这样，以至于把“散列”就说成是散列簇集

- 散列簇集（Hash Cluster），就是对簇表中的行在簇集键列上运用Hash函数进行散列，由此决定相应物理块的地址。

- Oracle提供缺省的Hash函数，支持单列/列组上的散列簇集。用户也可以自己提供Hash函数，但此时有限制：簇集键/散列键必须由单列组成，且只允许取整数值。

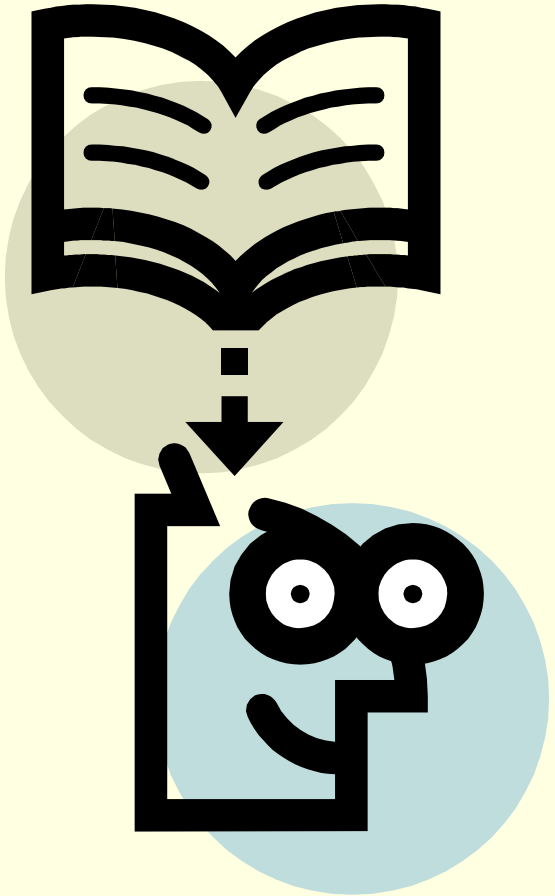
5.2 关系表的典型存储机制>>散列

- 散列簇集的利弊
  - 利
    - 若Hash值对每行是唯一的，此时使用散列簇集最为理想。

      ——只需1次物理块I/O。
  - 弊
    - 若Cluster Key或hash(Cluster Key)有许多相同的值，则用散列簇集并不好。

      ——因为地址冲突必须将数据块链到溢出表，会降低存取速度。

目录 Contents



5.2 关系表的典型存储机制>>簇集

- 簇集（Cluster）是存储表数据的一种可选方法。
  - 一个簇集是一个（组）表，其中具有同一公共列（组）值的所有表行（即元组）均存储在一起（即物理上相同或相邻的数据块中）。这些公共列（组）称簇集键（Cluster Key）。
  - 本人将簇集比喻成“葡萄串”：

5.2 关系表的典型存储机制>>簇集

- **例：**表emp与dept均有“部门编号”（deptno）列，可将deptno列作为簇集键创建一个簇集，将两个表的数据一起存储于该簇集中。

```
CREATE CLUSTER personnel ( dept_number INT ) ;  /* 创建簇集* /

CREATE TABLE emp
( empno INT PRIMARY KEY,
  ename VARCHER(12) NOT NULL,
  …
  deptno INT NOT NULL  )
CLUSTER personnel ( deptno ) ;    /* 称emp为（已）簇表 */

CREATE TABLE dept
( deptno INT PRIMARY KEY,
  dname VARCHER(10),
  loc VARCHER(12)  )
CLUSTER personnel ( deptno ) ;    /* 称dept为（已）簇表 */
```

5.2 关系表的典型存储机制>>簇集

- 簇集有两种实现方式
  - 索引簇集（Indexed Cluster）【缺省方式】
    - 对簇表在簇集键上再建索引，每个簇集键值有一个索引项。
    - 例：建立索引簇集personnel1：

```
CREATE CLUSTER personnel1 (dept_number INT) INDEX;
```

  - 散列簇集（Hash Cluster）
    - 对簇表的行在簇集键列上运用Hash函数进行散列，以决定相应物理块的地址。这样，具有同一散列值的行将存储在一起。
    - 例：建立散列簇集personnel2：

```
CREATE CLUSTER personnel2 ( dept_number INT )
HASH IS dept_number HASHKEYS 100;
                    /* Hash函数最多可产生100个不同的Hash值 */
```

5.2 关系表的典型存储机制>>簇集

- 簇集的利弊
  - 利
    - 可改进簇表之间在簇集键列上连接运算的性能。

      ——因为减少了磁盘I/O次数。
    - 节省存储空间。

      ——因为簇表中每个簇集键值只存储一次，不管这个/些表中有多少行包含此簇集键值。
  - 弊
    - 降低了簇表上更新操作（INSERT, UPDATE, DELETE）的性能。

      ——因为增加了系统维护开销。

The End

- 第5章作业：【补充的】
- 试解释关系数据库系统中基表的四种典型存储机制（方法）：(1)普通表；(2)索引的表；(3)索引簇表；(4)散列簇表。
- 提醒：请在截止时间（10月27日23:59）之前提交答案！