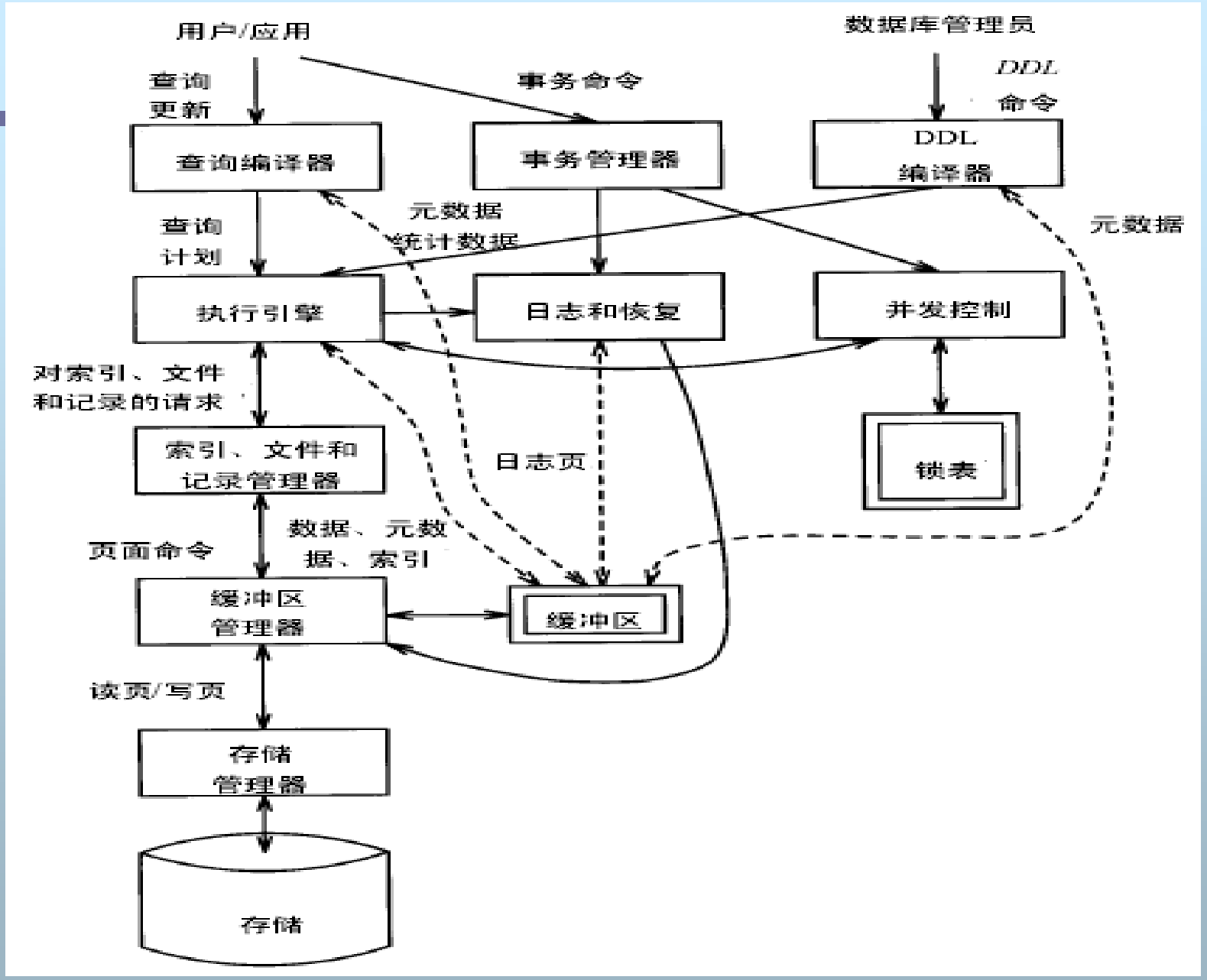
Data Representation

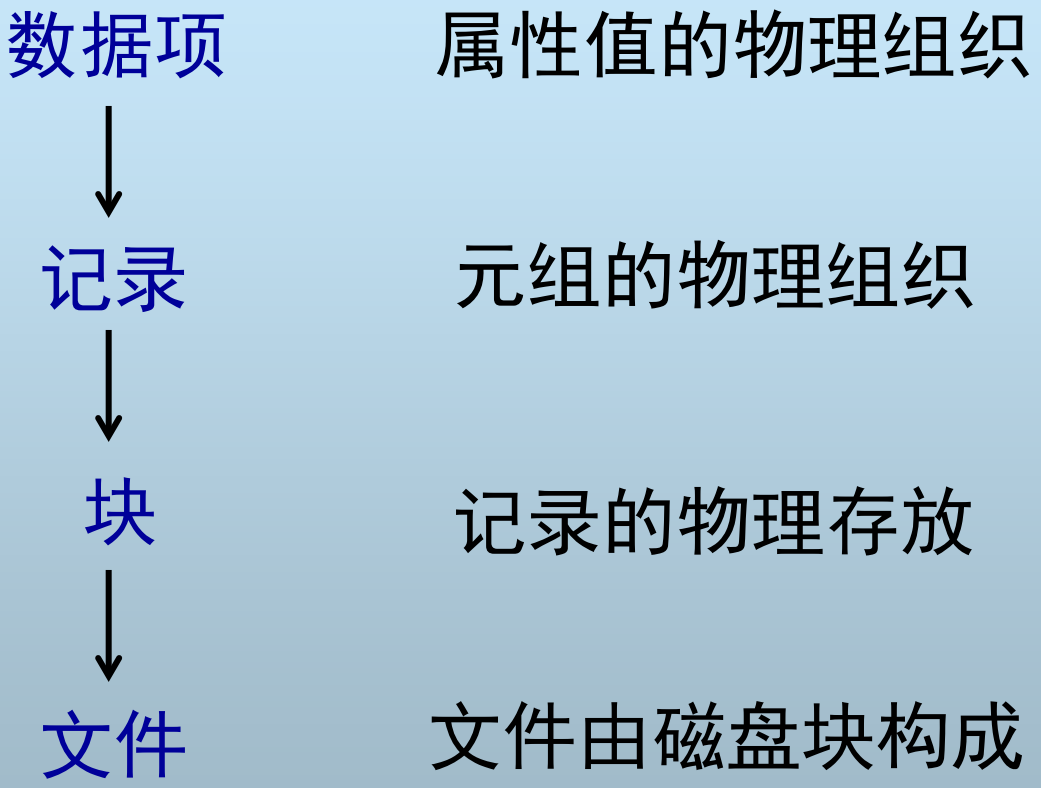
金培权（jpq@ustc.edu.cn）


用户/应用
数据库管理员
查询
更新
事务命令
DDL
命令
查询编译器
事务管理器
DDL
编译器
元数据
统计数据
元数据
查询
计划
执行引擎
日志和恢复
并发控制
对索引、文件
和记录的请求
索引、文件和
记录管理器
日志页
锁表
数据、元数
据、索引
页面命令
缓冲区
管理器
缓冲区
读页/写页
存储
管理器
存储

主要内容

- 数据项的表示（Data Items）
- 记录的表示（Records）
- 记录在块中的组织（Block）
- 记录的修改
- 块在文件中的组织

数据元素的表示层次



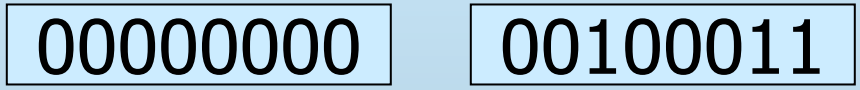
一、数据项的表示

- 数据项
  - 字节序列
  - 表示关系数据库中元组的属性值

1、数据项表示的内容

- 表示什么？
  - 姓名
  - 年龄
  - 出生日期
  - 照片
  - ……
- 用什么表示？
  - Bytes

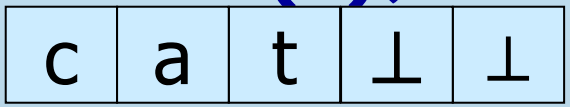
2、数据项表示方法：SQL数据类型

- **Integer (short)**
  - 2 bytes
  - 例如，35 表示为 

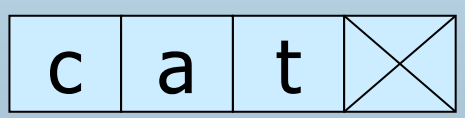
- **Real，Float**
  - 4 bytes （32 bits）
  - N bits表示小数，M bits表示指数

2、数据项表示方法：SQL数据类型

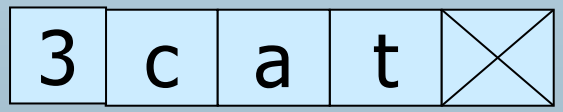
- Char(n) 或 Character(n) 定长字符串
  - 小于n时使用特殊填充符
  - 例如，若属性类型为Char(5)，则属性值'cat' 表示为



- Varchar(n) 变长字符串
  - NULL终止符，例 Varchar(5)



  - 带长度



  - 定长表示，n+1 bytes
    Varchar(4):

2、数据项表示方法：SQL数据类型

- **Boolean**
  - **TRUE** 1111 1111
  - **FALSE** 0000 0000
- **枚举类型**
  - **{RED,GREEN,YELLOW}**
  - **整数表示**
    - RED ↔ 1，GREEN ↔ 2，YELLOW ↔ 3
    - 若用两个字节的短整型来表示，则可以表示 2^{16} 个不同值

2、数据项表示方法：SQL数据类型

- Date
  - 10字符(SQL92)：’YYYY-MM-DD’字符串表示
  - 8字符：’YYYYMMDD’
  - 7字符：’YYYYDDD’, NOT ‘YYMMDD’!
  - Integer，自1900-01-01以来的天数
- Time
  - 8字符(SQL92)：’HH:NN:SS’ ——整数秒
  - Varchar(n)： ‘HH:NN:SS.FF’——带小数秒
  - Integer，自00:00:00以来的秒数

2、数据项表示方法：SQL数据类型

- **Bit**
  - 带长度的二进制位串

Length	Bits

  - 按字节表示，例如 **010111110011**

01011111	00110000

3、两种不同的数据项表示

- 定长数据项
- 变长数据项
  - 带长度 （常用!）
  - Null Terminated

数据项表示总结

类　型	表 示 方 法		
整数和实数	**字节串**		
定长字符串	***n*字节的数组**		
变长字符串	**用*n*+ 1字节**		
VARCHAR(n)	长度加内容	空值-终止字符串	
日期和时间	某种格式的定长字符串	变长值	整数
二进制位序列	长度加内容	字节表示	
枚举类型	使用整数编码表示一个枚举类型的值		

Where are we?

二、记录的组织

- 记录
  - 数据项 [字段，Fields] 的集合

E.g.: Employee record:

name field,

salary field,

date-of-hire field, ...

1、记录的类型

- 固定格式 vs. 可变格式
  Fixed Format vs. Variable Format
- 定长 vs. 变长
  Fixed Length vs. Variable Length

2、固定格式定长记录

- 所有记录具有相同的逻辑结构（模式）
- 记录的模式（Schema）
  - # fields
  - Name of each field
  - Type of each field
  - Order in record
  - Offset of each field in the record

E.g. 固定格式定长记录

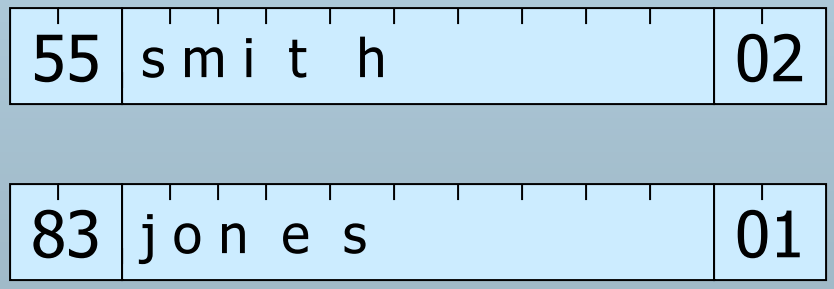
Employee record

(1) E#, 2 byte integer

(2) Ename, 10 char.

(3) Dept, 2 byte code

Schema



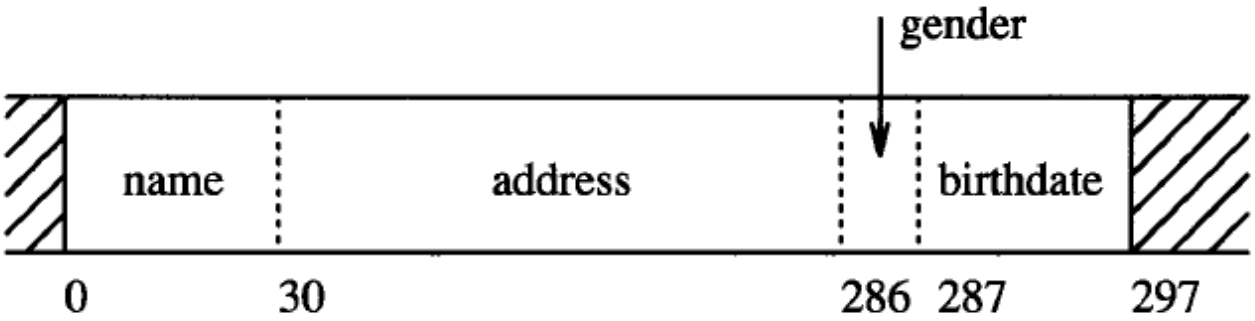
Records

2、固定格式定长记录

- 构造

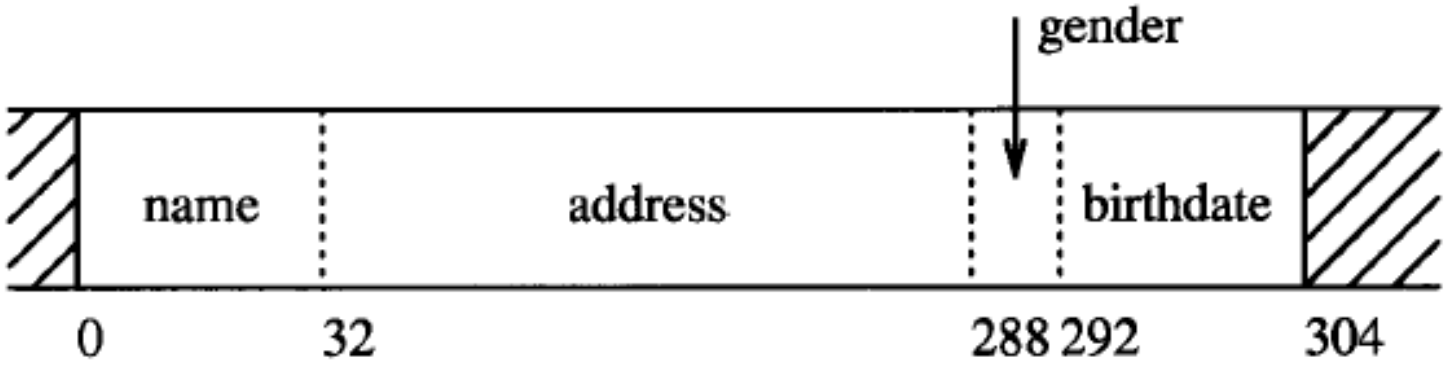
```
CREATE TABLE MovieStar(
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE
);
```

- 不考虑寻址特点



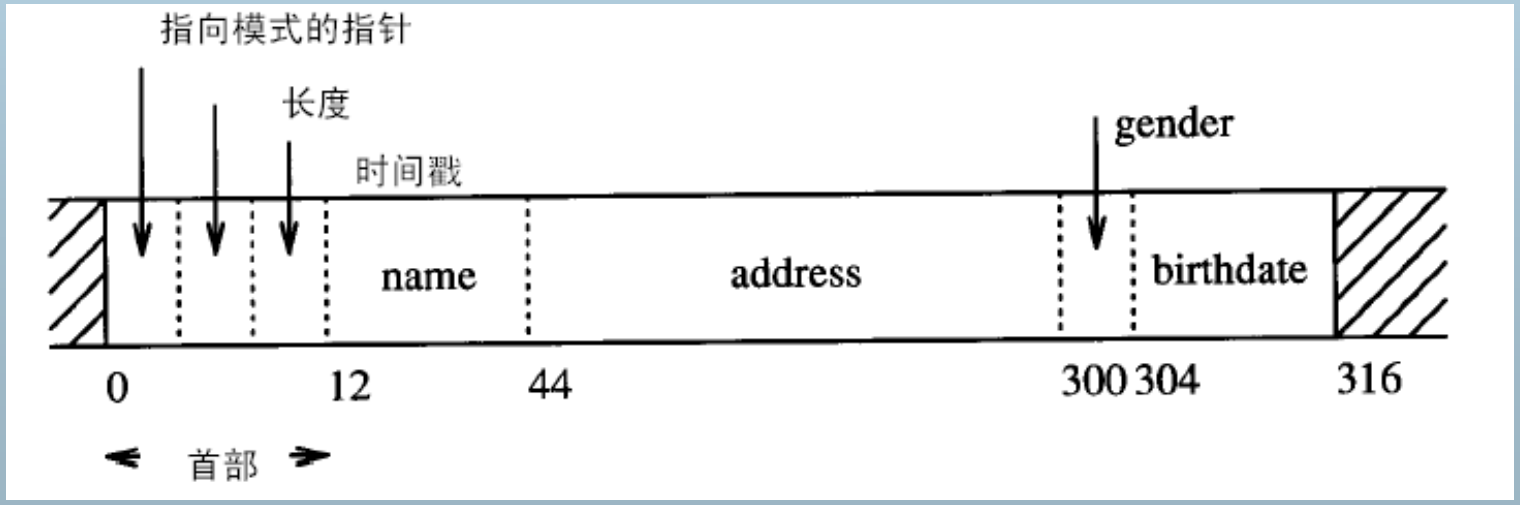
2、固定格式定长记录

- 考虑寻址特点
  - 假设记录和字段的开始地址必须是4的倍数



3、记录首部

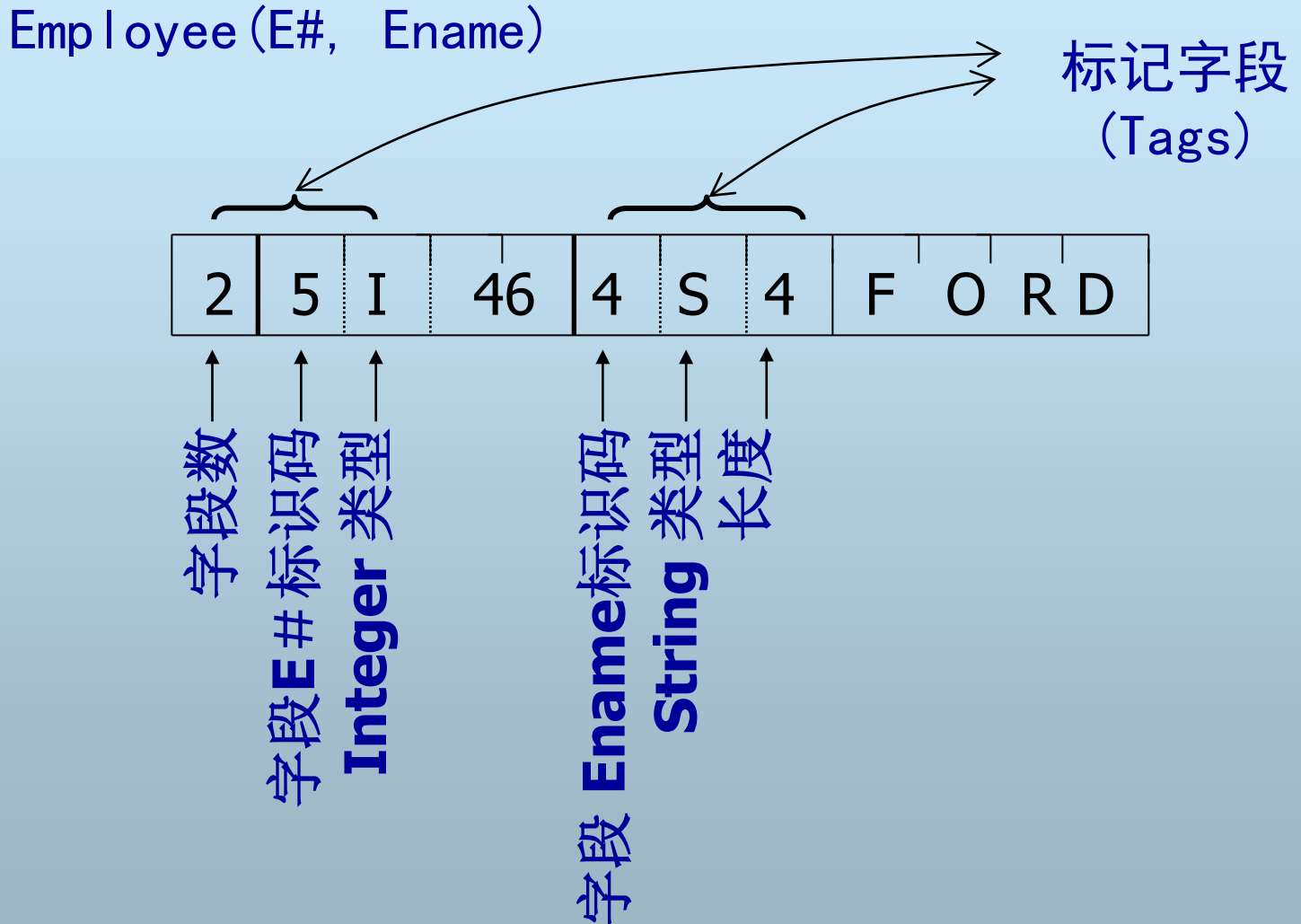
- 在记录首部（Head）的描述记录的信息
  - 记录类型（模式信息）
  - 记录长度
  - 时间戳
  - 其它信息



4、可变格式记录

- 每个记录的格式不同
- 记录的格式存储于记录中

E.g. 可变格式变长记录表示



E.g. 可变格式变长记录表示

- Key-Value
- 记录都以“KEY+VALUE”方式表示
- KEY与VALUE都以字节流（byte string）存储，如下：

```
typedef struct {
    void *data; //字节流指针
    int size; //字节流长度
} DBT;
```

- BerkeleyDB
- Memcached
- Redis
- LevelDB

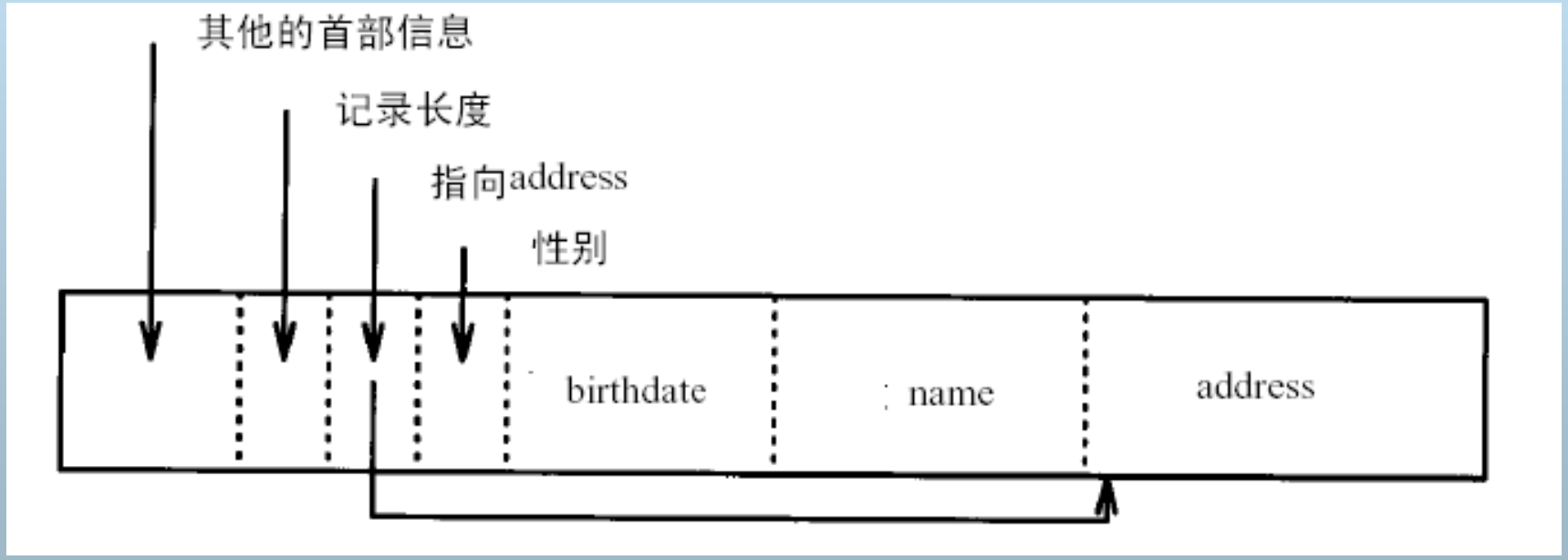
- 数据类型没有限制
- 应用与数据库之间不需转换数据格式
- 不提供KEY和VALUE的内容和结构信息
- 应用必须知道所用的VALUE的含义

4、可变格式记录

- 好处
  - 灵活的记录格式，适合“松散”记录
    - 尽管一个记录可能有大量字段，但某个记录通常只有有限的几个字段
    - 例如，病人的检验结果
  - 适合处理重复字段
  - 适合记录格式演变
- 缺点
  - 浪费存储空间

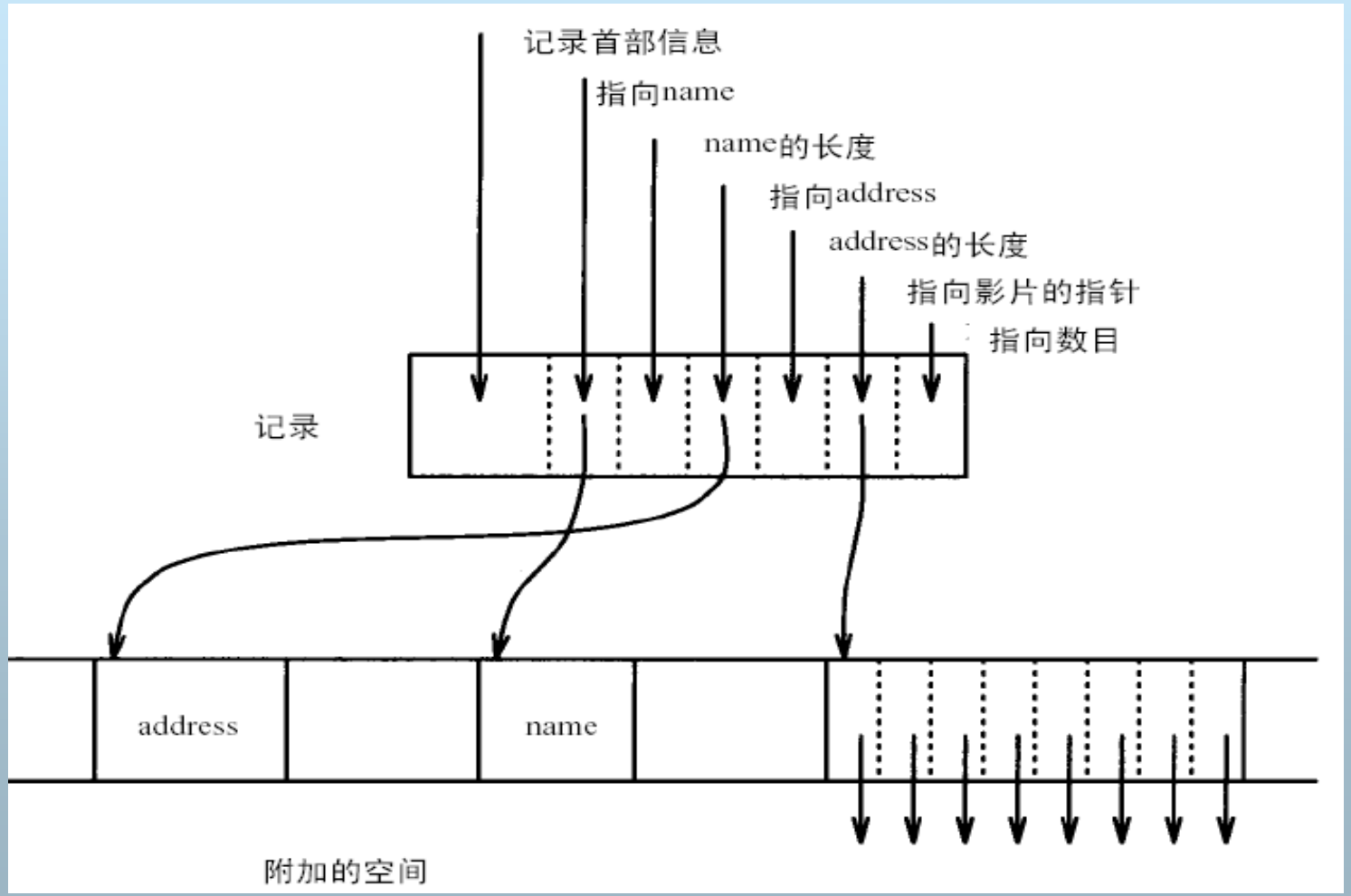
6、变长记录表示

- 首部指针法
  - 定长字段在前，变长字段在后
    name、address变长



6、变长记录表示

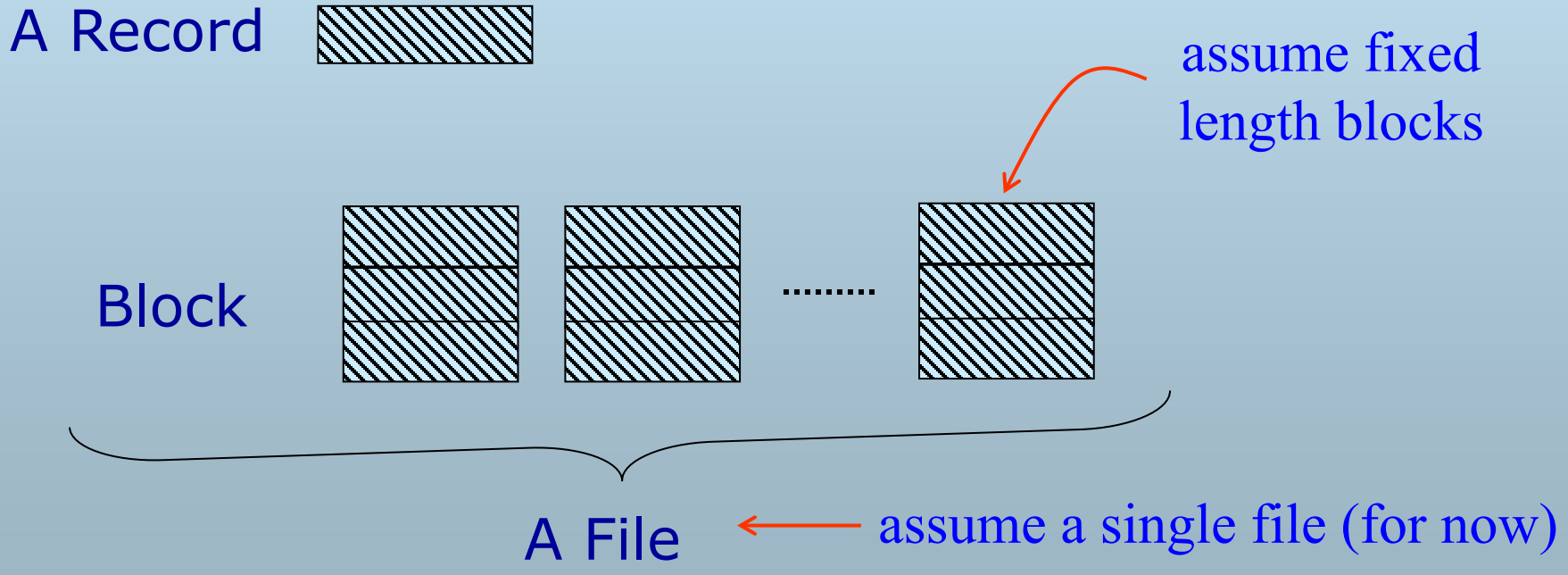
- 混合格式：定长记录＋变长记录



Where are we?

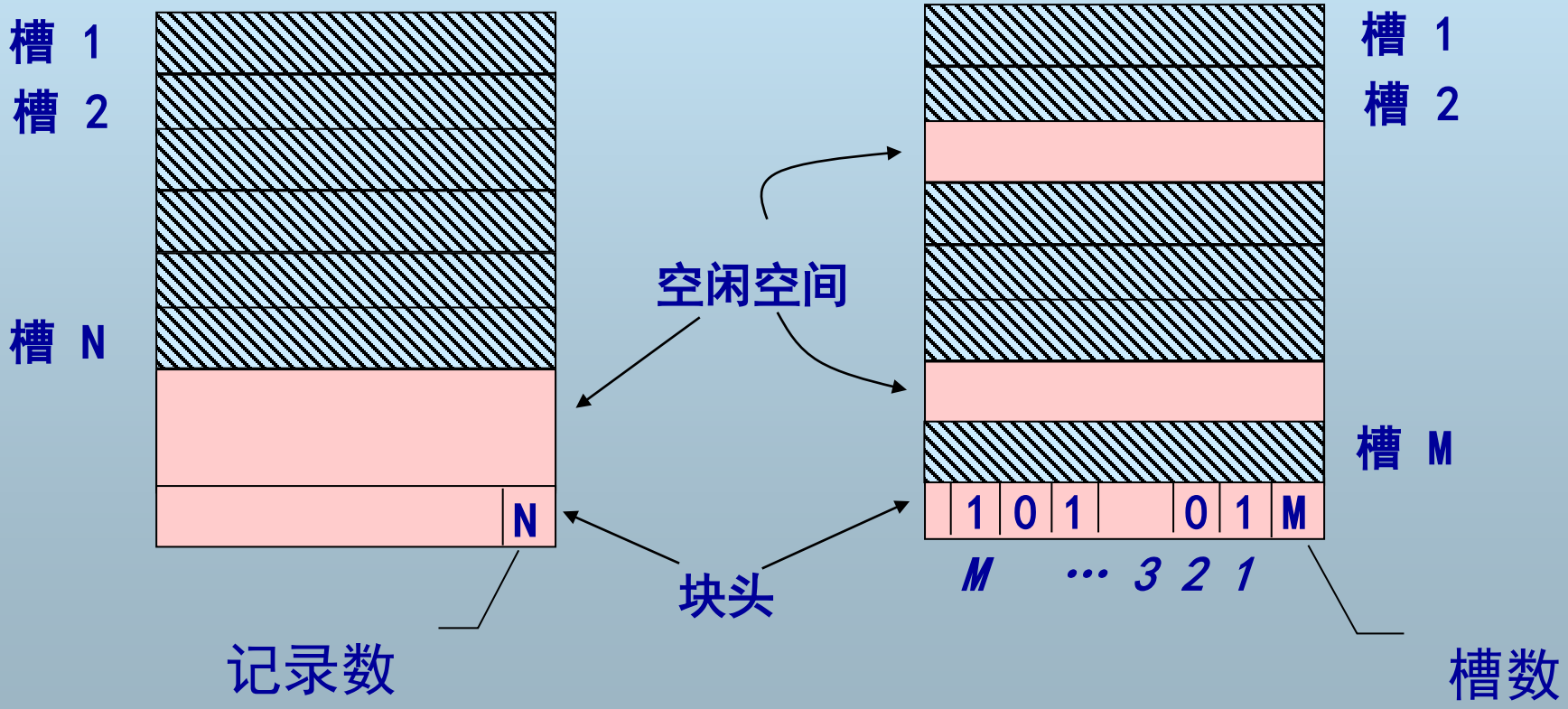
三、记录在块中的组织

- 假设
  - 块的大小固定
  - 记录组织成单个文件



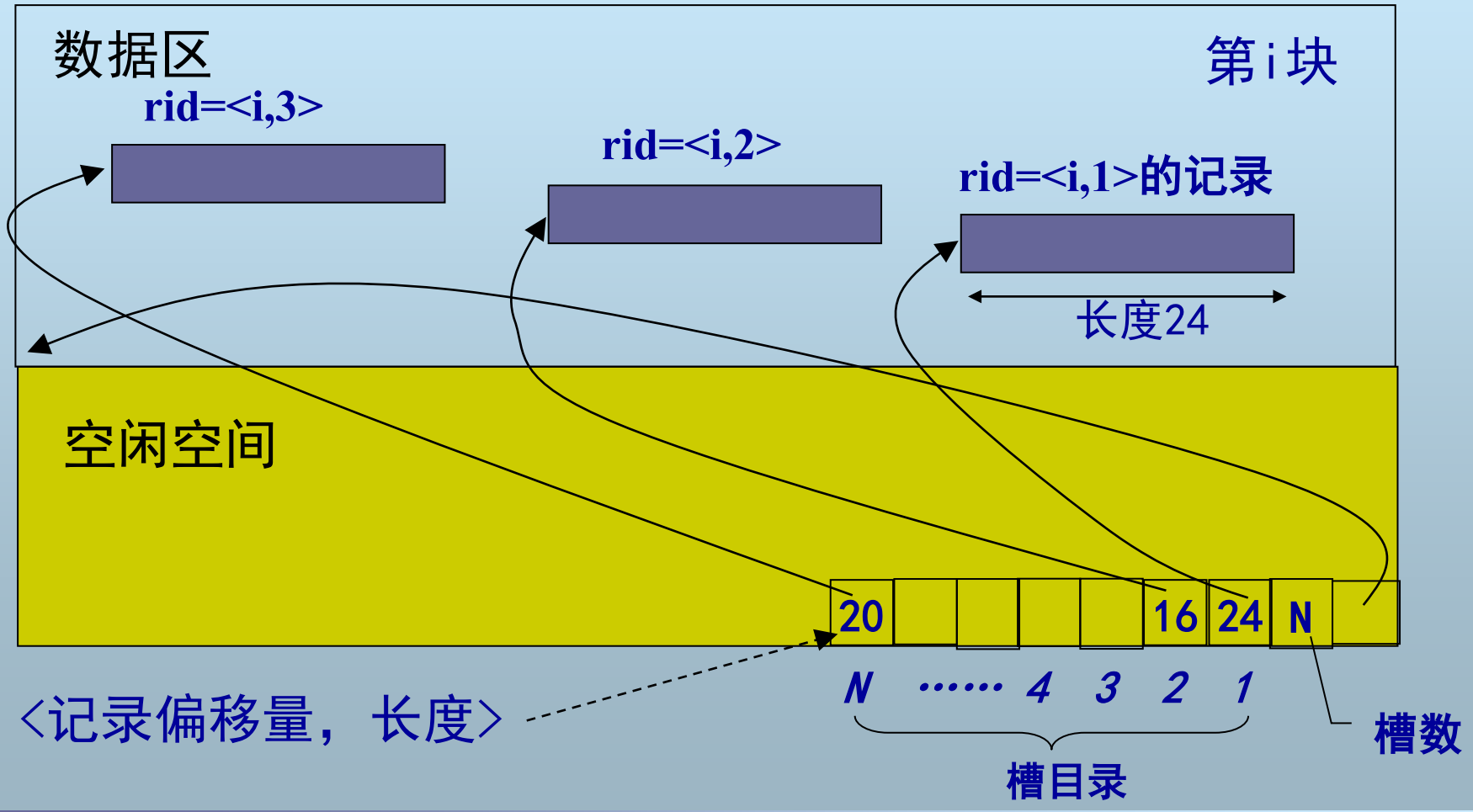
三、记录在块中的组织

- 定长记录的两种块内组织
  - 记录地址rid通常使用*<块号，槽号>*表示



三、记录在块中的组织

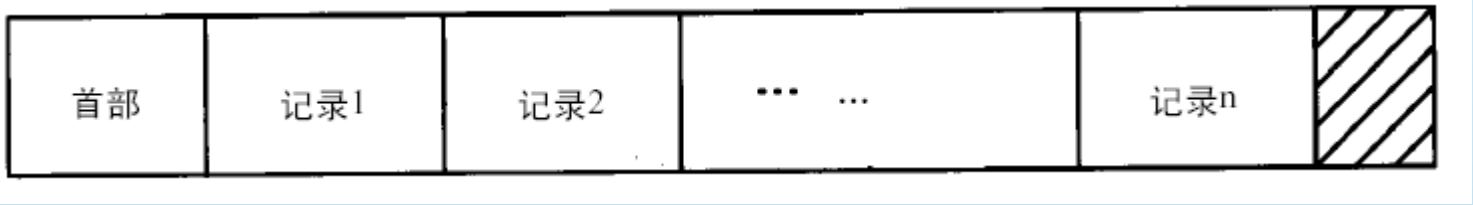
- 变长记录在块内的组织



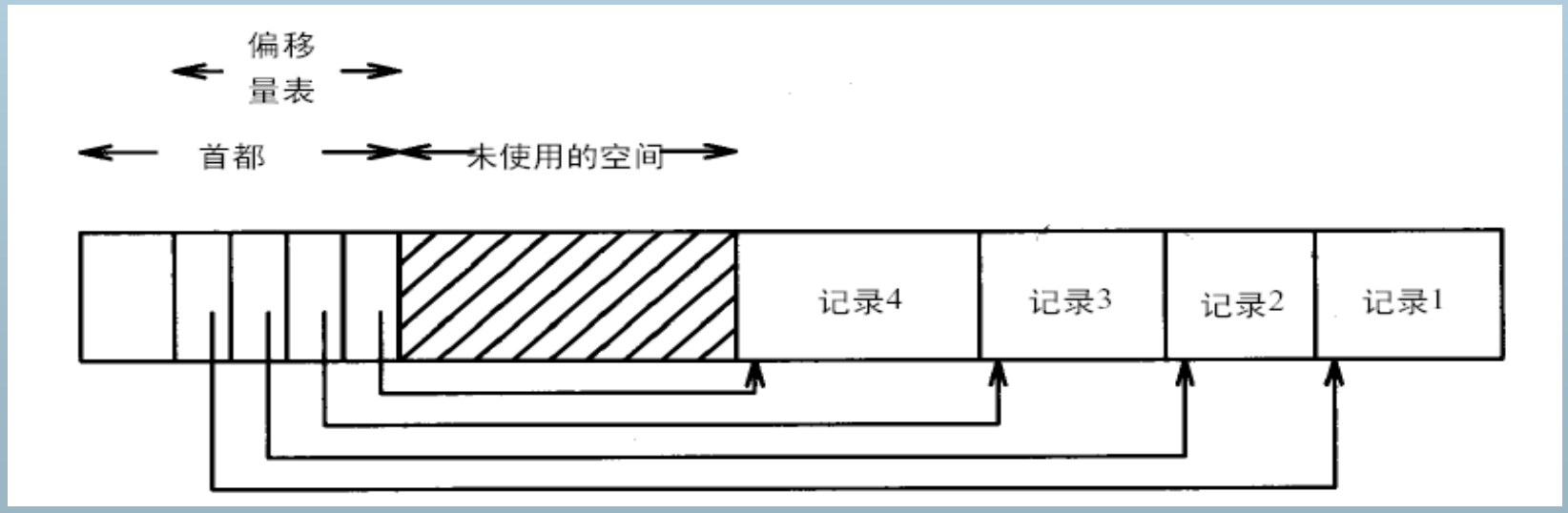
三、记录在块中的组织

- 其他问题
  - 记录在块中的分隔 (separating records)
  - 记录跨块 vs.记录不跨块 (spanned vs. unspanned)
  - 不同类型的记录聚簇 (mixed record types – clustering)
  - 按序组织 (sequencing)
  - 记录的分裂 (split records)
  - 记录地址 (record address)
  - 记录的修改

1、记录在块内的分隔

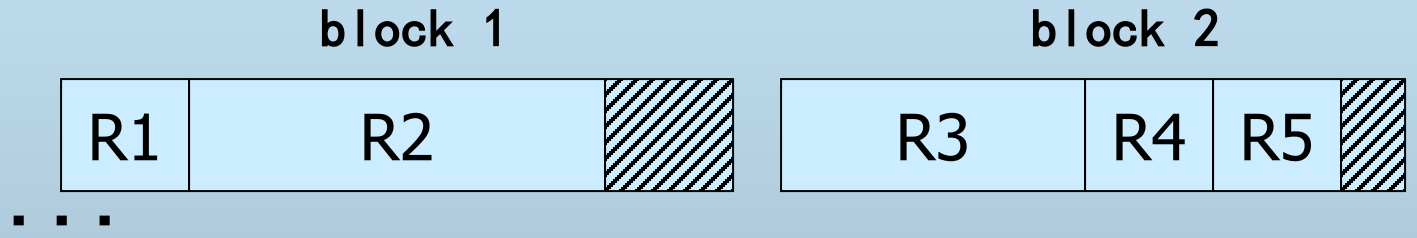


- 定长记录：不需分隔
- 使用特殊标记
- 通过块内偏移量

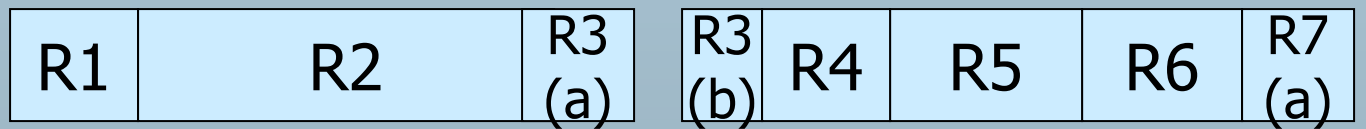


2、跨块 vs. 不跨块

- Unspanned： 记录必须在一个块中存储

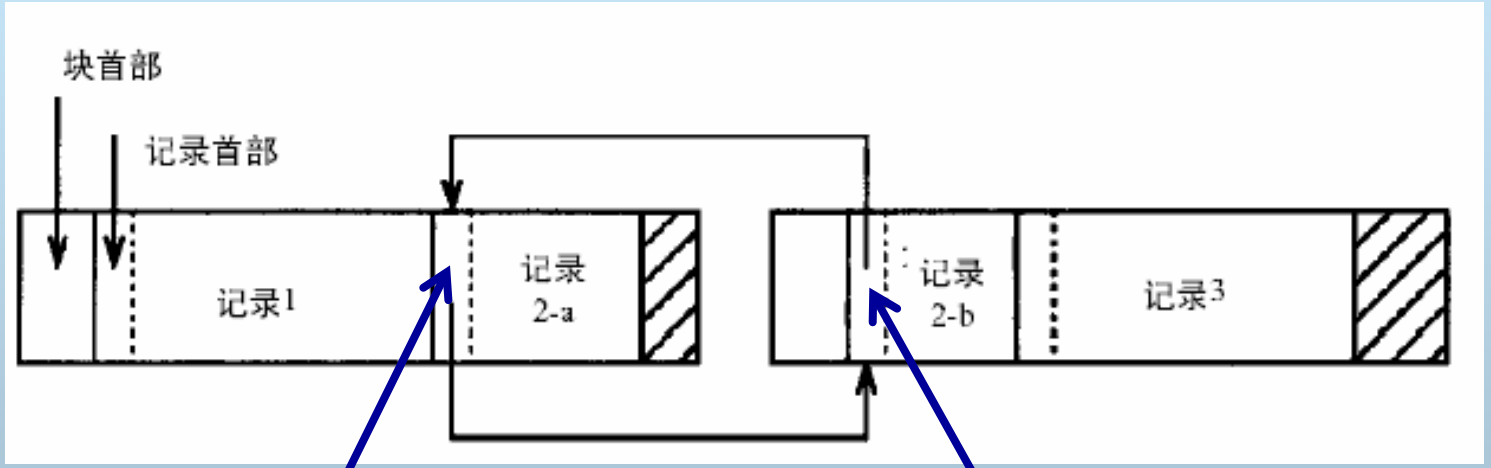


- Spanned：记录可跨块存储



2、跨块 vs. 不跨块

- 跨块

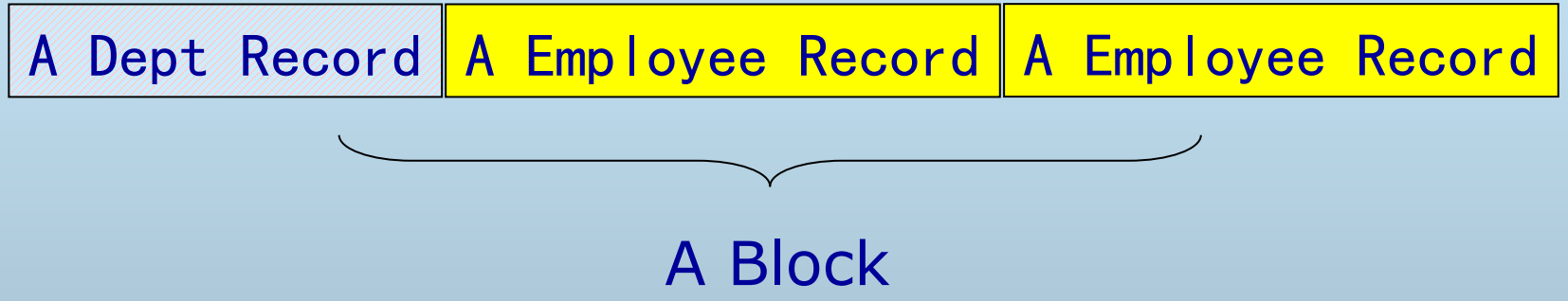


2、跨块 vs. 不可跨块

- 比较
  - unspanned：实现简单，但空间浪费
  - spanned：有效利用空间，实现更复杂
- But ……
  - If record size > block size，MUST be spanned

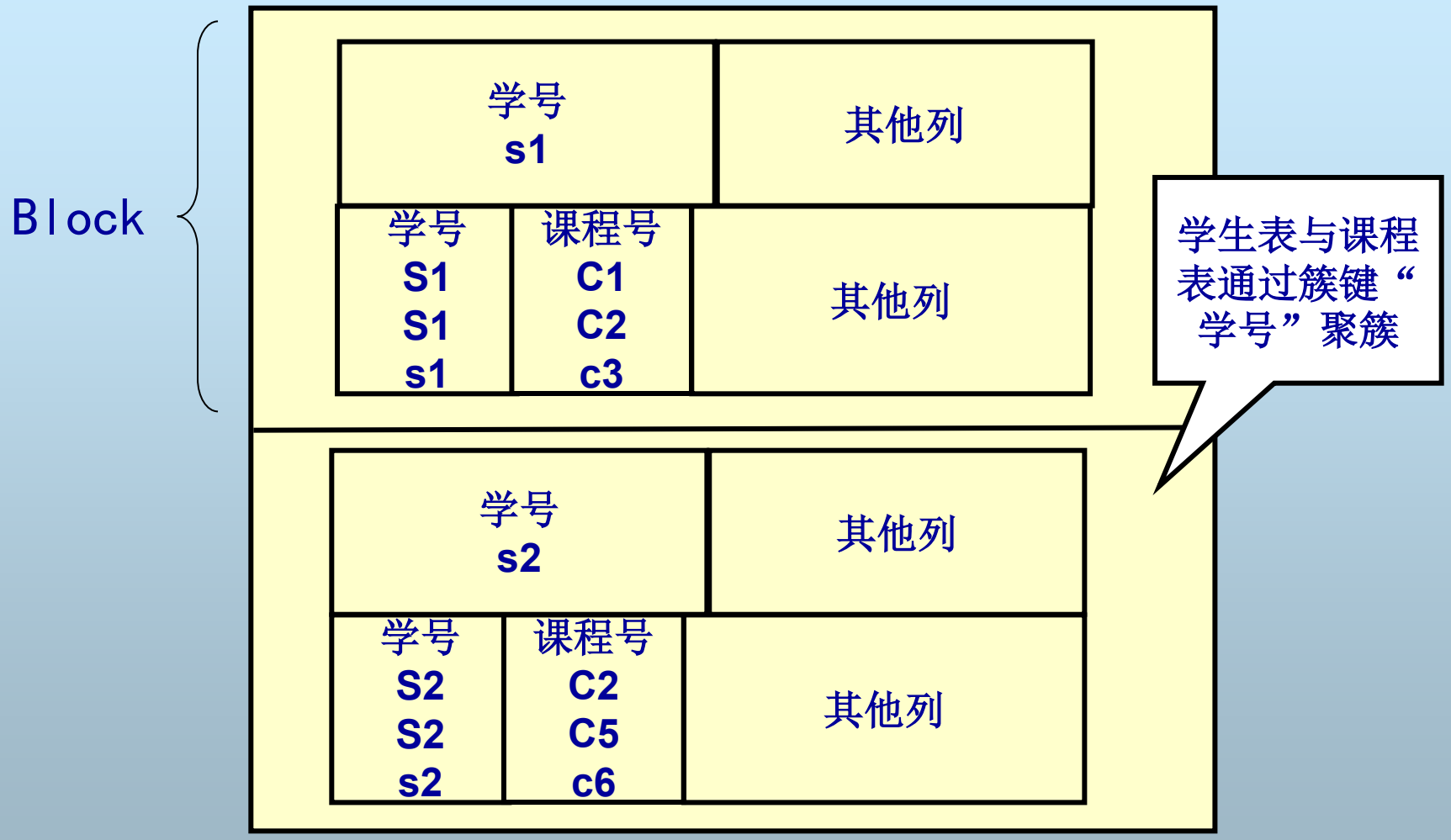
3、不同类型的记录聚簇

- 一个块中存储不同类型的记录
  （对于RDB：多关系上的聚簇）



- 好处——聚簇 (clustering)
  - 经常一起访问的记录存储在同一块或连续块中

3、不同类型的记录聚簇


Block
学号
s1
其他列
学号
S1
S1
s1
课程号
C1
C2
c3
其他列
学生表与课程表通过簇键“学号”聚簇
学号
s2
其他列
学号
S2
S2
s2
课程号
C2
C5
c6
其他列

3、不同类型的记录聚簇

STUDENT(s#,sname,age)

SC(s#,cname,score)

Q1: select student.s#,sc.cname from student s,sc where s.s# = sc.s#

Q2: select * from student

- **如果Q1经常被查询，则聚簇非常有效**
- **若Q2经常被查询，则聚簇反而降低了效率**

4、在块中按序存储记录

- 另一种聚簇（对于RDB：单关系上的聚簇）
  - 将记录按某个字段顺序排列在块中
- 好处
  - 加快按排序字段查询记录时的效率
  - 利于归并联接 （will be discussed later）

4、在块中按序存储记录

按Dept顺序组织的Student记录

化学系	……
化学系	……
化学系	……
化学系	……
物理系	……
物理系	……
物理系	……
中文系	……

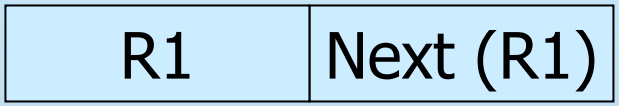
无序组织的Student记录

化学系	……
物理系	……
物理系	……
化学系	……
中文系	……
化学系	……
物理系	……
化学系	……

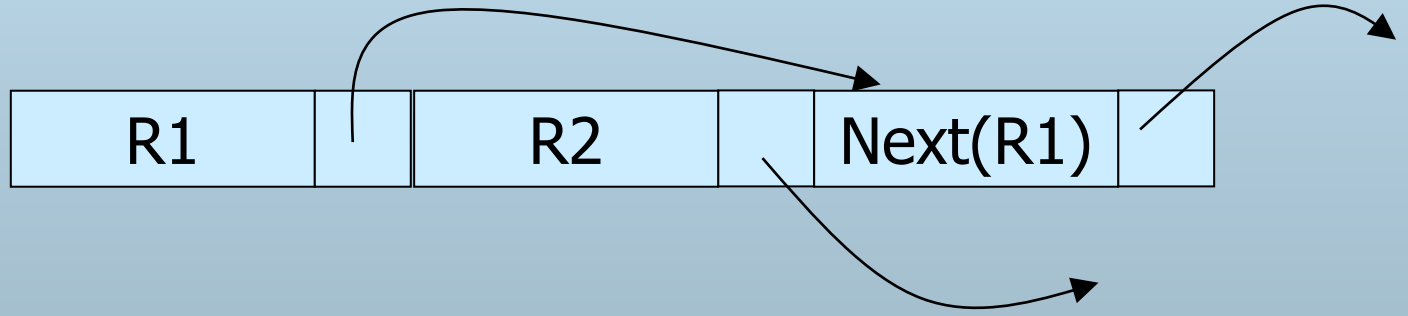
假设一个磁盘块2条定长记录

4、在块中按序存储记录

- 物理连续



- 指针连接



5、记录的分裂

- 适合于变长记录的混合格式表示
  - 定长部分存储于某个块中
  - 变长部分存储于另一个块中
  - 与spanned存储类似

6、记录地址

- 物理地址
- 逻辑地址（间接地址）

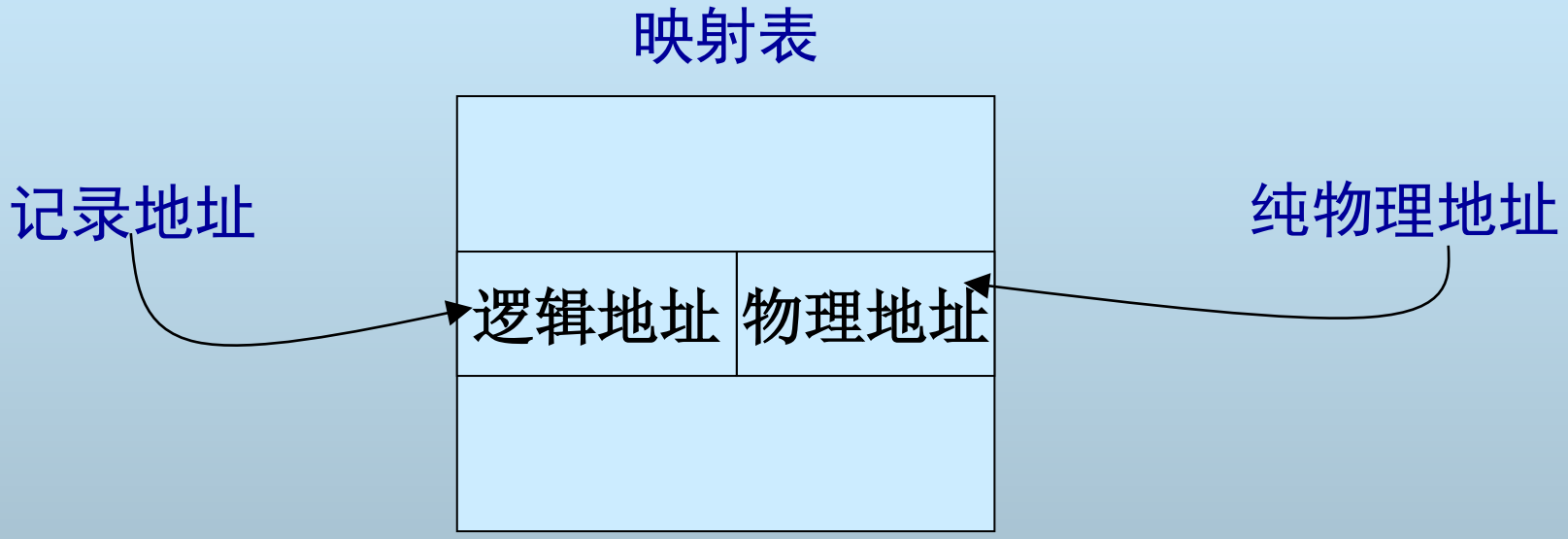
6、记录地址

- 记录的纯物理地址
  - 主机标识
  - 磁盘或其他设备标识
  - 柱面号
  - 磁头号（盘面号）
  - 块号
  - 块内的偏移量

块地址

6、记录地址

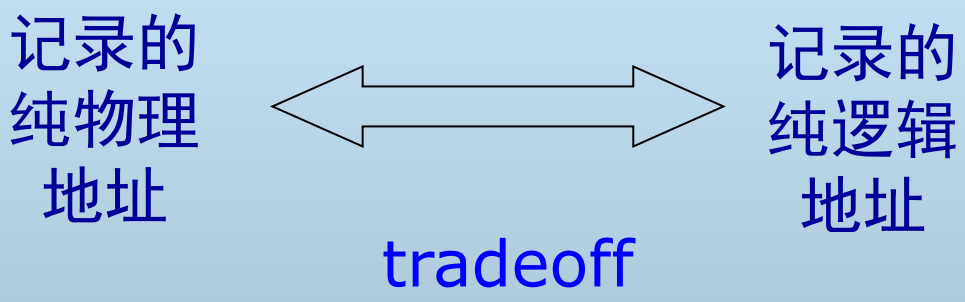
- 记录的纯逻辑地址



缺点——访问代价增加：映射表占存储空间；需要地址转换

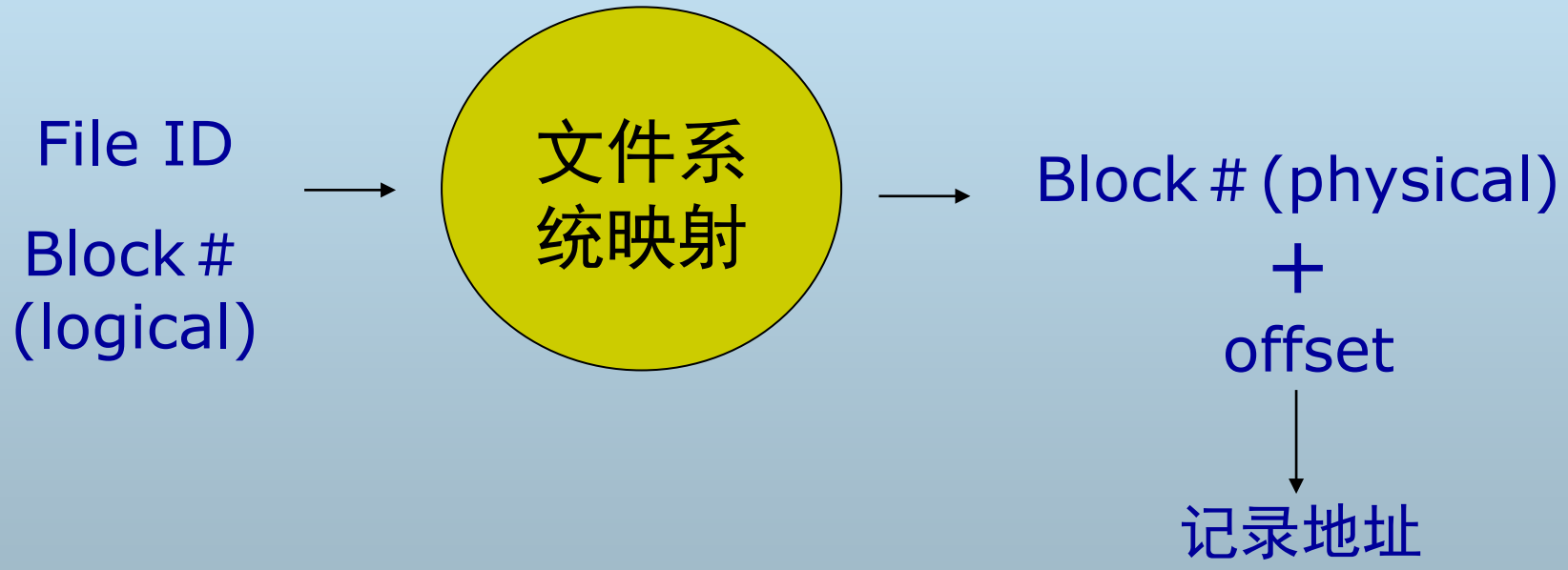
好处——灵活性：删除或移动记录时只要改变映射表项

6、记录地址



6、记录地址

- 借助文件系统的逻辑块地址
  - 文件号＋逻辑块地址＋块内偏移

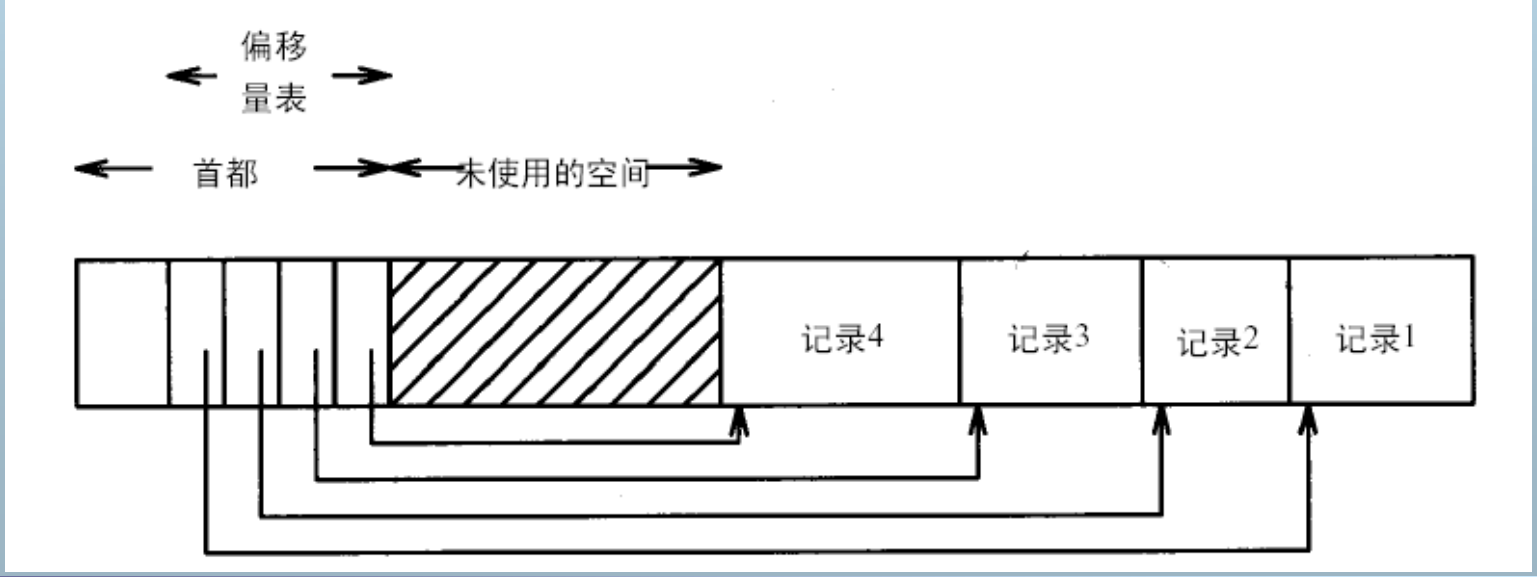


四、记录的修改

- 插入
- 删除

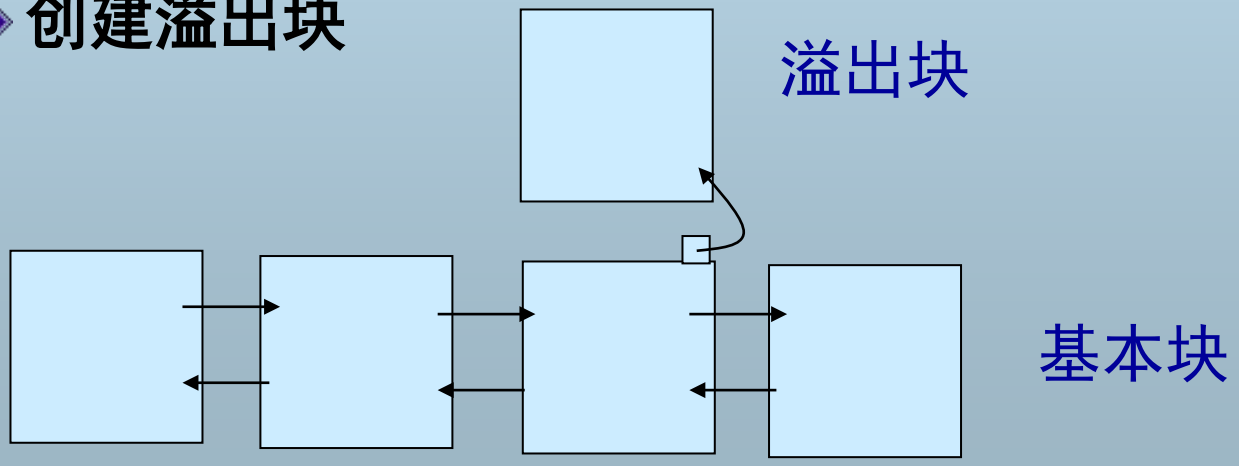
1、插入

- 记录无序
  - 插入到任意块的空闲空间中
  - 或申请一个新块（当所有块都已满时）
  - 记录变长时，可使用偏移量表



1、插入

- 记录有序
  - 找到记录应该放置的块
  - 如果有空间，放入并调节记录顺序即可，否则有两种方法：
    - 在“邻近块”中找空间
    - 创建溢出块



2、删除

- 立即回收空间
  - 例如，加到可用空间列表中
- 删除记录时处理溢出块
  - 若删除的记录位于溢出块链上，则删除记录后可对整个链进行重新组织以去除溢出块

2、删除

- 使用删除标记
  - 若使用偏移表，则可以修改偏移表项指针，将其置空
  - 若使用逻辑—物理地址映射表，则可以将物理地址置空
  - 可以在记录首部预留一开始位：0—未删除，1—已删除

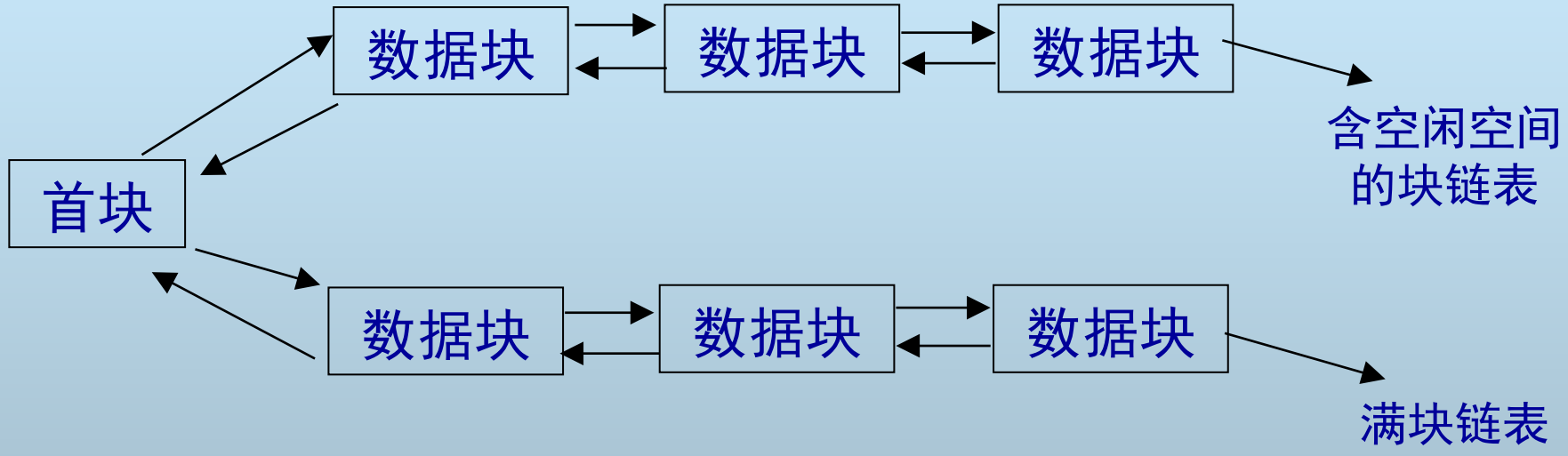
	1	记录1	0	记录2

Where are we?

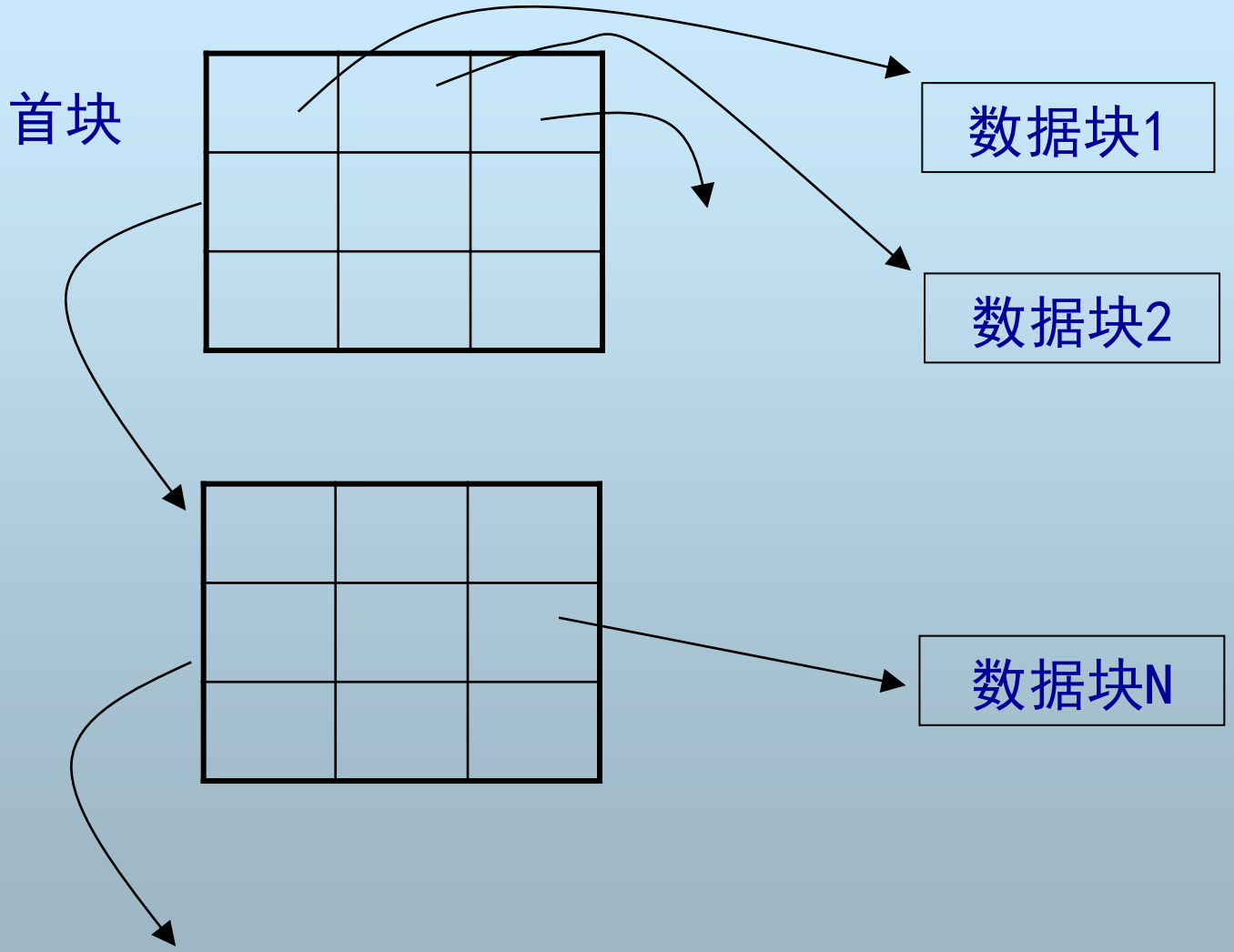
五、块在文件中的组织

- 堆文件（Heap File）
  - 最基本、最简单的文件结构
  - 记录不以任何顺序排序
  - 记录可能存放在物理不邻接的块上
- 插入容易，但查找和删除代价高

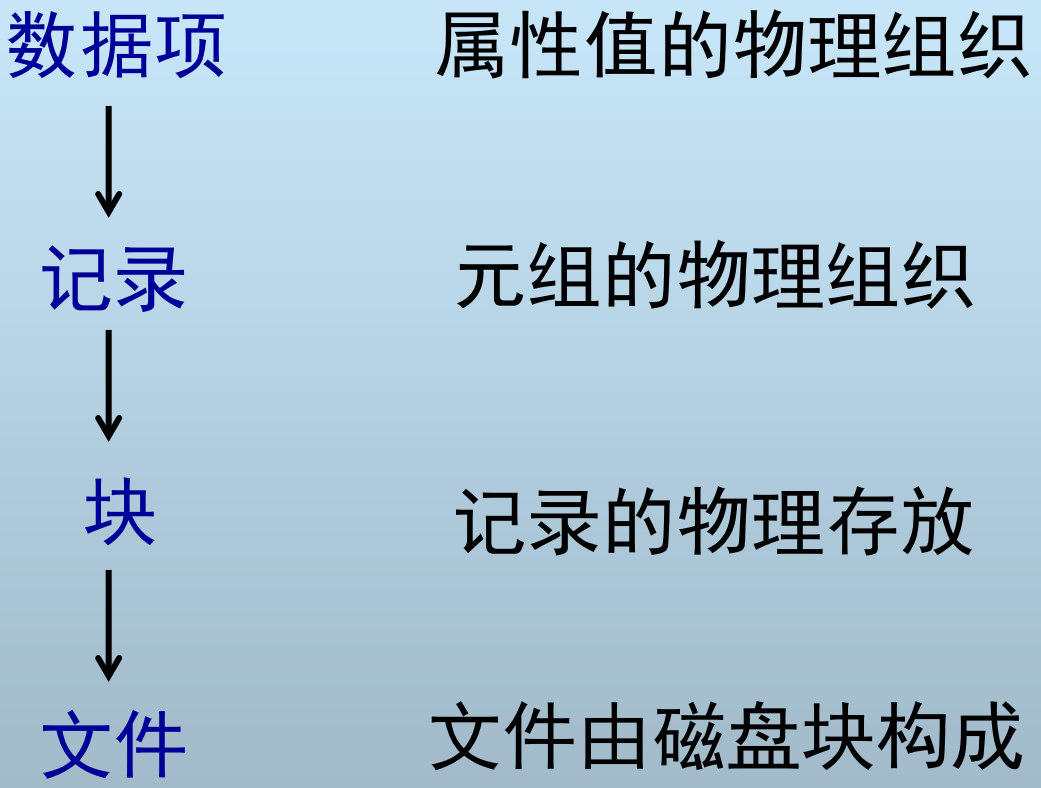
1、链表式堆文件组织



2、目录式堆文件组织



回顾：数据元素的表示层次



考虑

- 如何衡量数据组织方法的好坏？
  - 性能 (performance)
  - 灵活性 (flexibility)
  - 复杂程度 (complexity)
  - 空间利用率 (space utilization)

本章小结

- 数据项的表示
- 记录的表示
- 记录在块中的组织
- 记录的修改
- 块在文件中的组织