2025年春季学期

数据库系统概论

An Introduction to Database Systems

第九章 关系数据库存储管理

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn

9.1 数据组织

9.1.1 数据库的逻辑组织方式与物理组织方式

9.1.2 记录表示

9.1.3 块的组织

9.1.4 关系表的组织

9.1.2 记录表示

- 元组存储
  - 形式一：定长记录存储
  - 形式二：变长记录存储

定长记录

- 元组存储
  - 形式一：定长记录存储
  - 形式二：变长记录存储

定长记录存储

- 定长记录存储
  - 关系表中的每条元组占据相同大小的空间
  - 变长字段以定长形式存储，预留最大长度空间

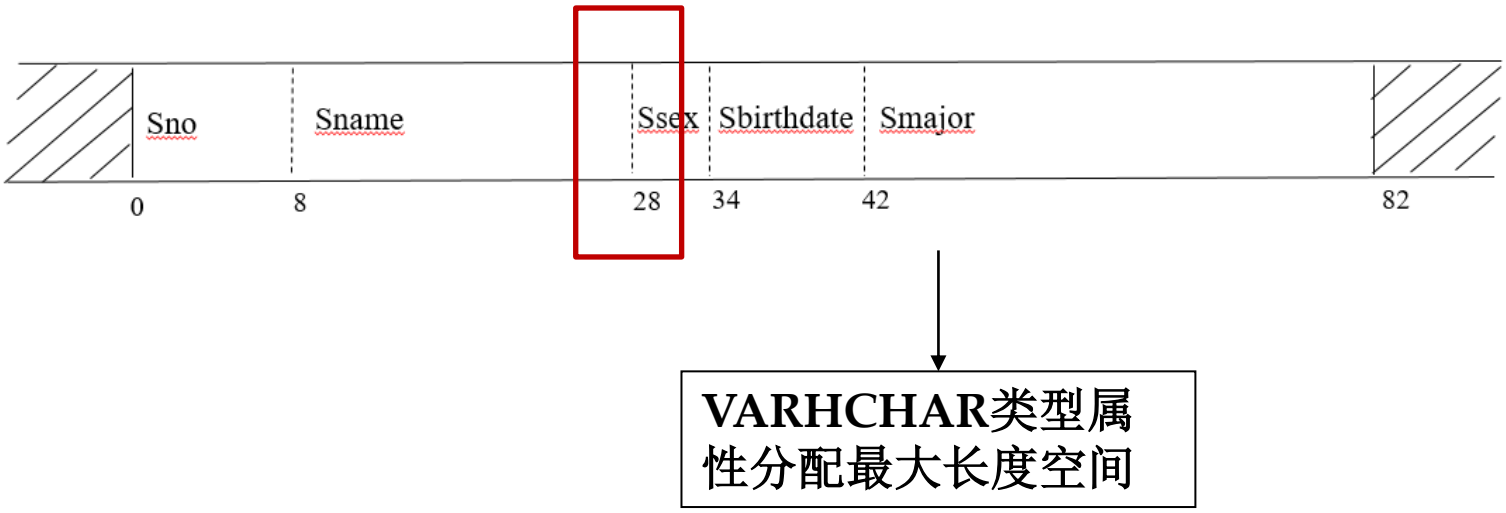
定长记录存储

- 定长记录存储
  - 例：

```
CREATE  TABLE  Student (
   Sno  CHAR(8) PRIMARY KEY,
   Sname CHAR(20) UNIQUE,
   Ssex  CHAR(6),
   Sbirthdate  Date,
   Smajor  VARCHAR(40)
 );
```

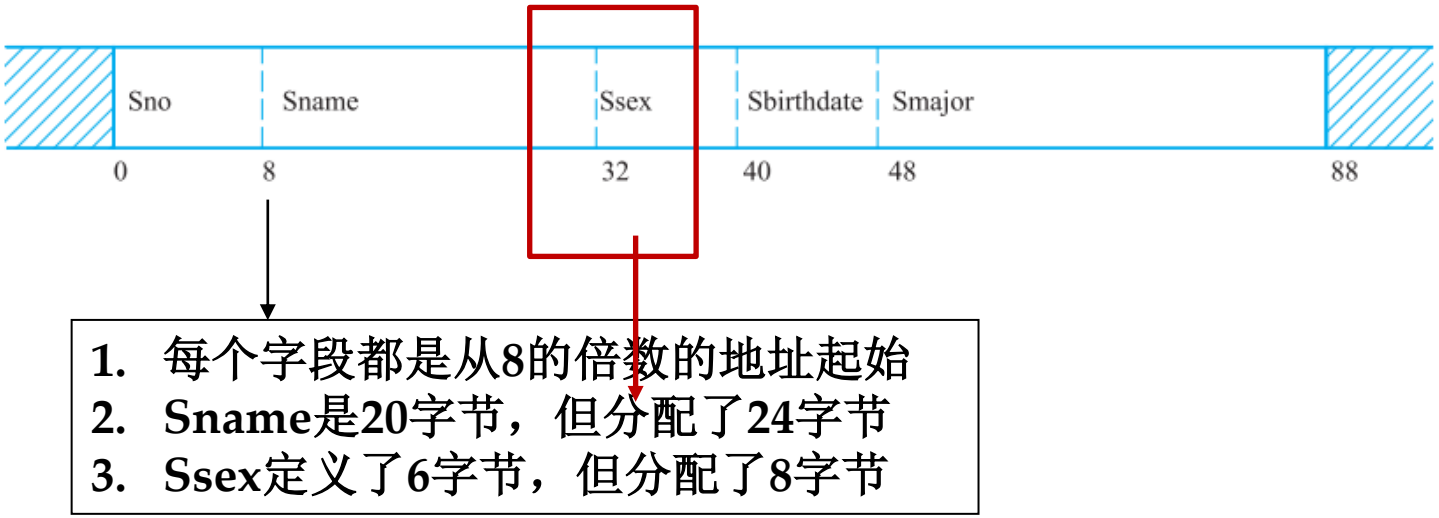
定长记录存储

- 定长记录存储
  - 例：



定长记录存储

- 有些系统要求数据在内存中的存取地址
  - 在外存中保证各字段的起始地址是4或8的倍数
  - 例：



定长记录存储

- 定长存储的优劣
  - 优势
    - 快速定位到记录及其属性的物理位置
    - 增删改比较方便快捷
  - 劣势
    - 浪费存储空间

变长记录存储

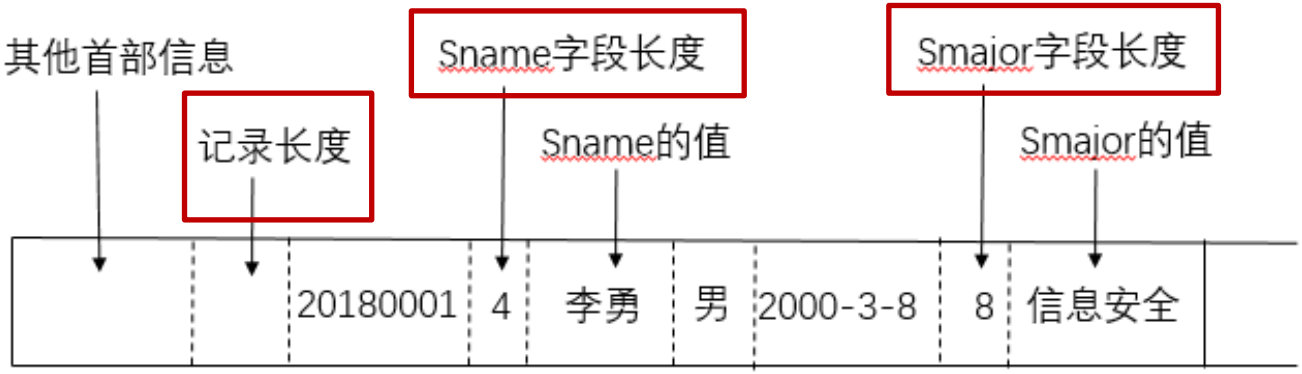
- 元组存储
  - 形式一：定长记录存储
  - 形式二：变长记录存储

变长记录存储

- 变长记录
  - 存储要求
    - 能快速访问一条记录
    - 能快速访问记录中所有属性（定长个变长属性）
  - 变长记录存放的三种方式

变长记录存储

- 方式一
  - 在**每条记录**的头部记录该条记录的长度
  - 在记录的**每个变长字段前**记录该字段的长度
  - 例：

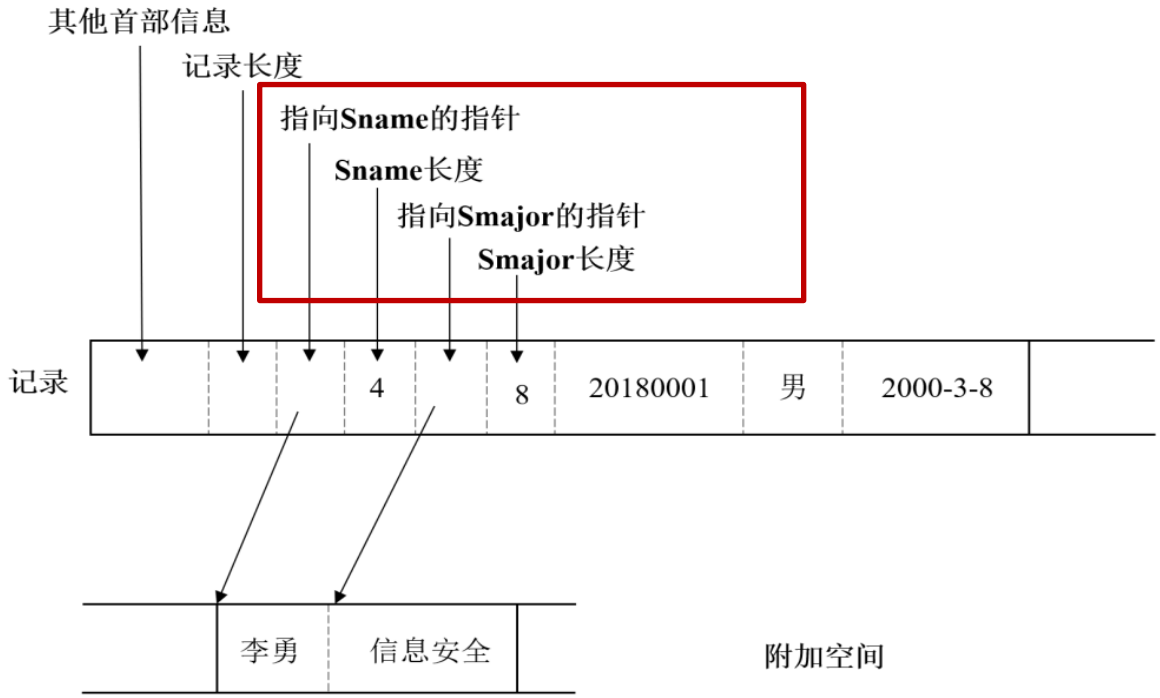


变长记录存储

- 方式二
  - 先存放定长字段，再存放变长字段
  - 第一个变长字段紧随定长字段，从第二个变长字段开始，在**记录首部用指针（偏移量）指向变长字段**
  - 例：

变长记录存储

- 方式三
  - 将定长字段与变长字段分开存储在不同的块中
  - 多用于BLOB等类型数据存储
  - 若经常访问定长字段，可减少数据存取时的I/O数量



9.1 数据组织

9.1.1 数据库的逻辑组织方式与物理组织方式

9.1.2 记录表示

9.1.3 块的组织

9.1.4 关系表的组织

9.1.3 块的组织

- 定长记录存储的块组织
- 变长记录存储的块组织

定长记录存储的块组织

- 定长记录存储的块组织
- 变长记录存储的块组织

定长记录存储的块组织

- 定长记录存储的块组织
  - 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

定长记录存储的块组织

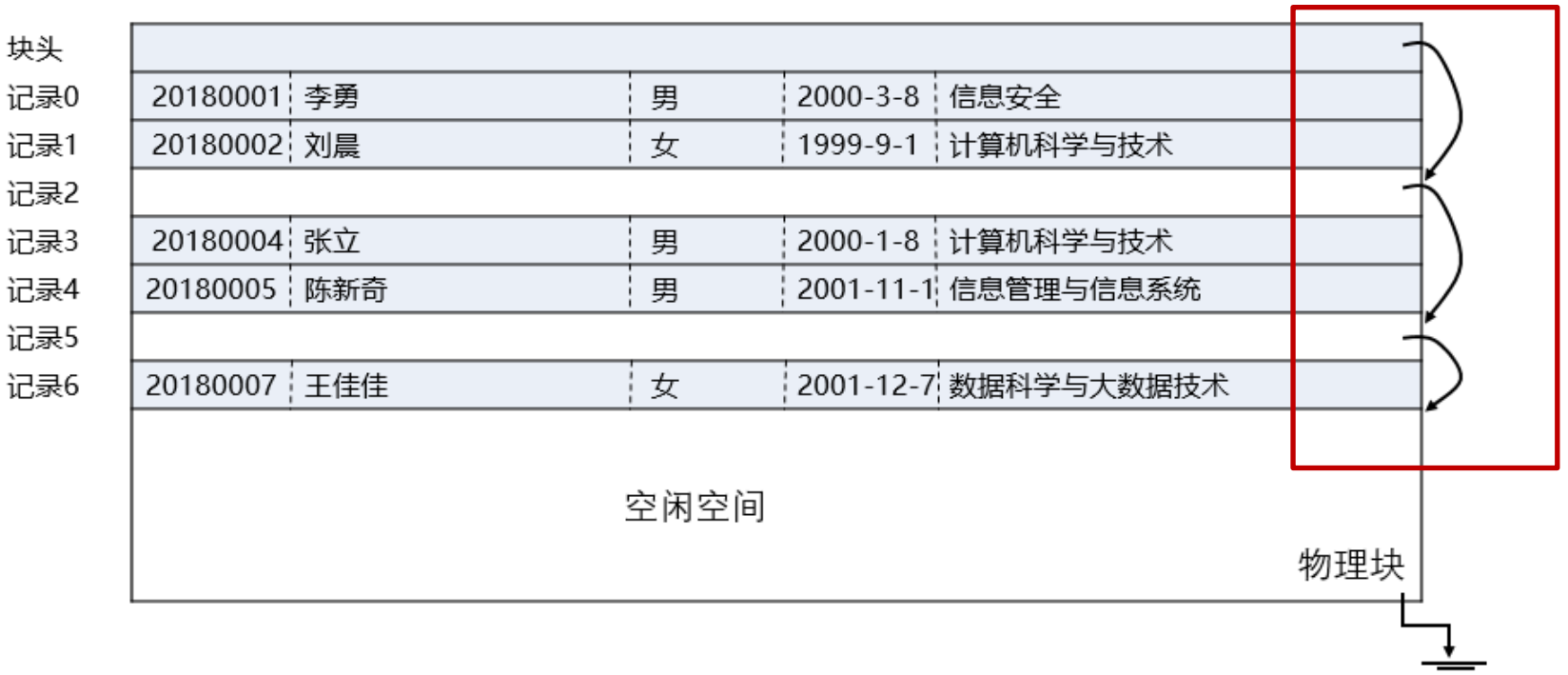
- 定长记录存储的块组织
  - 例：

定长记录存储的块组织

- 定长记录存储的块维护
  - 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
    - 例：在Student表中删除2018003和2018006两条元组

定长记录存储的块组织

- 删除后：

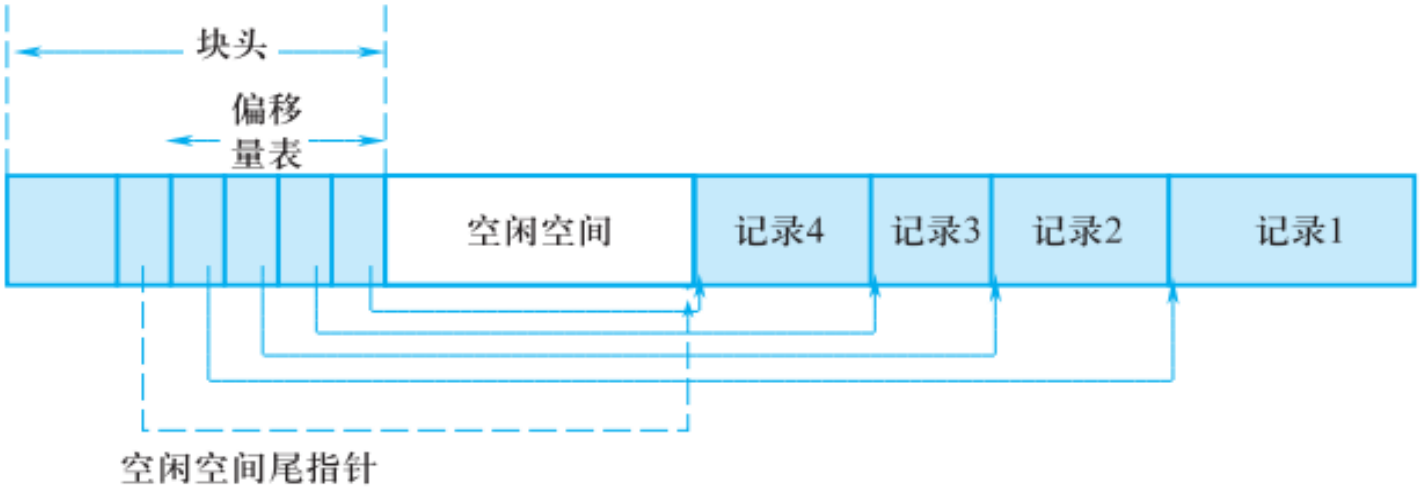


变长块记录存储的块组织

- 定长记录存储的块组织
- 变长记录存储的块组织

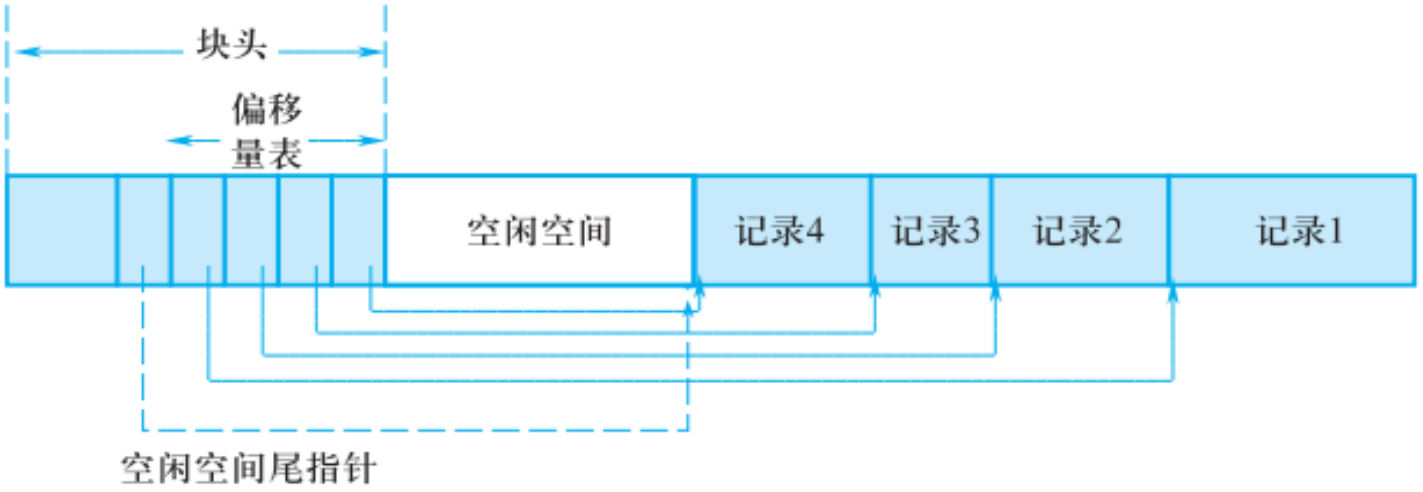
变长块记录存储的块组织

- 变长记录存储的块组织
  - 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）



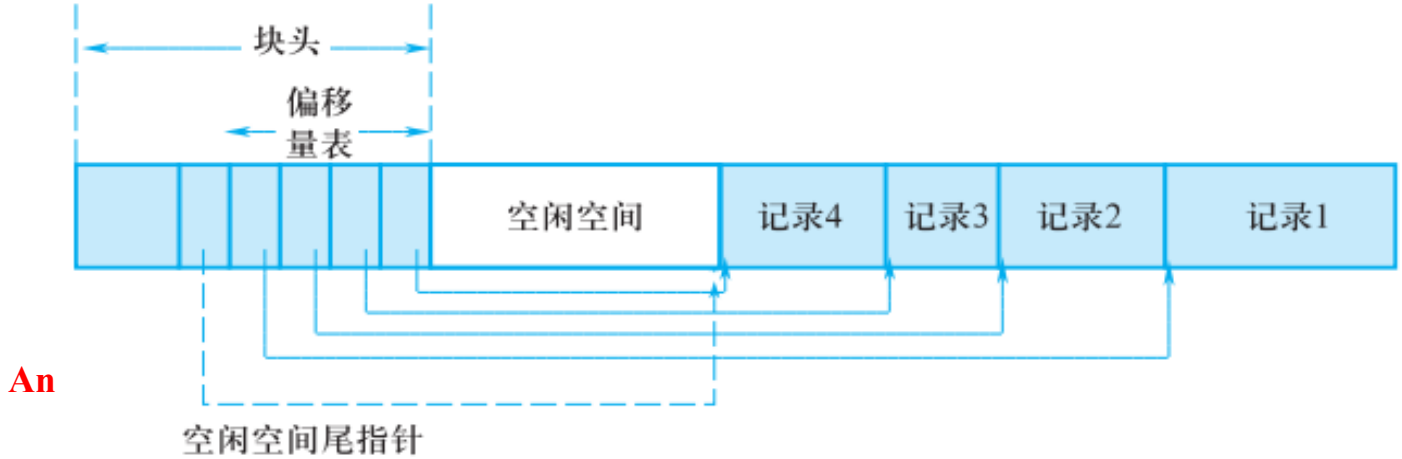
变长块记录存储的块组织

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



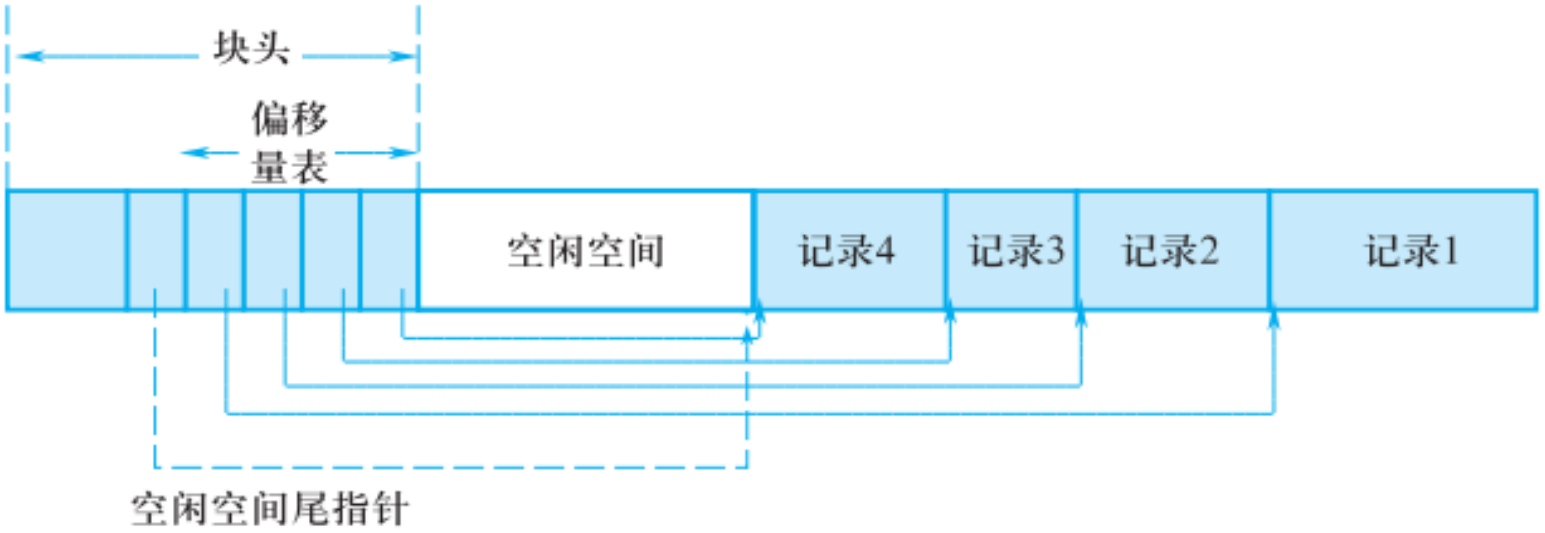
变长块记录存储的块组织

- 变长记录存储块的维护
  - 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



变长块记录存储的块组织

- 变长记录存储块的维护
  - 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



9.1 数据组织

9.1.1 数据库的逻辑组织方式与物理组织方式

9.1.2 记录表示

9.1.3 块的组织

9.1.4 关系表的组织

9.1.4 关系表的组织

- 关系表的5种存放方式
  - 堆存储
  - 顺序存储
  - 多表聚簇存储
  - B+树存储
  - 哈希存储

堆存储

- 关系表的5种存放方式
  - 堆存储
  - 顺序存储
  - 多表聚簇存储
  - B+树存储
  - 哈希存储

堆存储

- 堆存储
  - 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

顺序存储

- 关系表的5种存放方式
  - 堆存储
  - 顺序存储
  - 多表聚簇存储
  - B+树存储
  - 哈希存储

顺序存储

- 顺序存储
  - 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

顺序存储

- 顺序存储的优劣
  - 优势：可以高效地处理按排序属性（组）进行查询的请求
    - 例：如果Student表按专业Smajor的升序存放，则可以快速回答如下SQL语句

```
SELECT Sno, Sname, Ssex
FROM Student
WHERE Smajor='计算机科学与技术'
```

顺序存储

- 顺序存储
  - 劣势：维护代价较高（在增改时需要移动已有的记录以保序）

多表聚簇存储

- 关系表的5种存放方式
  - 堆存储
  - 顺序存储
  - 多表聚簇存储
  - B+树存储
  - 哈希存储

多表聚簇存储

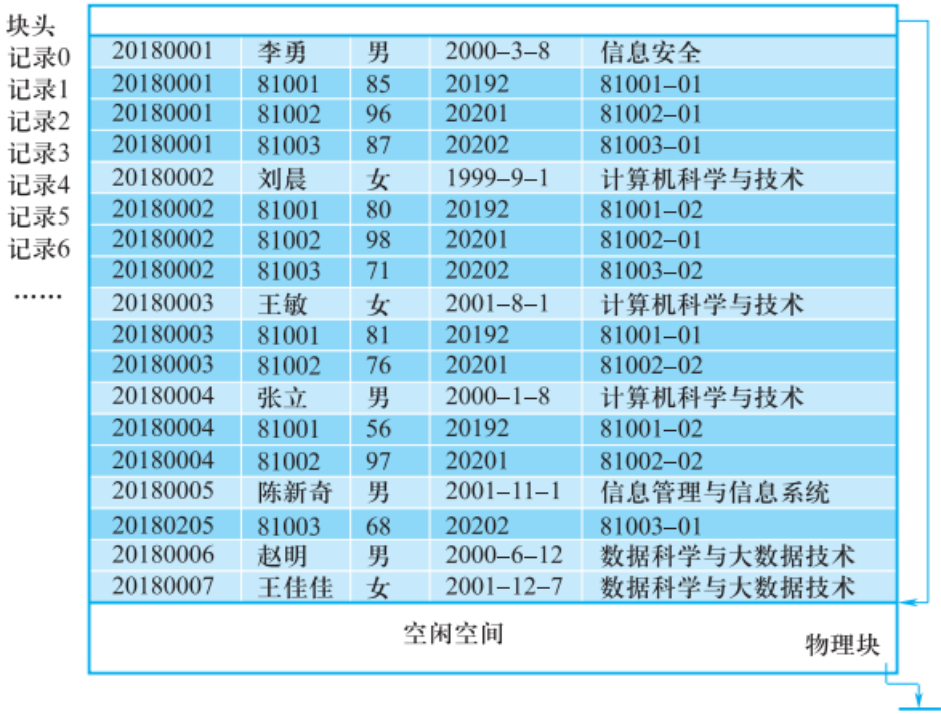
- 多表聚簇存储
  - 不同表的元组聚簇存放在同一组块中
  - 思考
    - How：如何聚簇？
    - Why：为什么聚簇？

多表聚簇存储

- 多表聚簇存储
  - 例：两个具有主外码的参照关系表Student和SC，按照学号Sno相等聚簇存放

多表聚簇存储

- 多表聚簇存储的优劣
  - 优势：
    - 减少连接操作带来的开销
      - 可快速回答如下查询：




```
SELECT SC.Sno, Sname, count(*)
FROM Student, SC
WHERE Student.Sno=SC.sno AND
SC.Sno='20180001';
```

```
SELECT Sdept, avg(grade)
FROM Student, SC
WHERE Student.Sno=SC.sno
GROUP BY Sdept
```

多表聚簇存储

- 多表聚簇存储的优劣
  - 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 Student
WHERE Smajor=' 数据科学与大数据技术'
AND Ssex='女';
```

    - 在更新操作时会带来更频繁的数据迁移
      - 如在新学期插入新的学生选课信息

多表聚簇存储

- 适用场景
  - 以聚簇表连接查询为主
  - 较少更新

聚簇存取方法的选择（续）

- 聚簇的用途
  - 1. 大大提高按聚簇码进行查询的效率

    例：学生关系按所在系建有索引，现查询DS系的学生名单。

    - 随机存放：DS系的500名学生分布在500个不同的物理块上时，至少要执行500次I/O操作（注意索引与数据地址）
    - 按照系名聚簇存放：将同一系的学生元组聚簇存放，则每读一个物理块可得到多个满足查询条件的元组，显著减少访问磁盘次数。DS系500名学生聚簇存放50个物理块，只要执行50次I/O
  - 2. 节省存储空间
    - 聚簇以后，聚簇码相同的元组集中在一起了，因而聚簇码值不必在每个元组中重复存储，只要在一组中存一次就行了

聚簇存取方法的选择（续）

- 聚簇的适用范围

1. 既适用于单个关系独立聚簇，也适用于多个关系组合聚簇

例：假设用户经常要按系别查询学生成绩单，

Select sname, cno, grade from student, sc where student.sno=sc.sno

- 查询涉及Student关系和SC关系的连接操作，按学号连接
  - 把具有相同学号值的学生元组和选修元组在物理上聚簇在一起。
  - 相当于把多个关系按“预连接”的形式存放，
  - 从而大大提高连接操作的效率。

聚簇存取方法的选择（续）

- 聚簇的适用范围

2. 当通过聚簇码进行访问或连接是该关系的主要应用，与聚簇码无关的其他访问很少或者是次要的时，可以使用聚簇

- **当SQL语句中包含有与聚簇码有关的ORDER BY，GROUP BY，UNION，DISTINCT等子句或短语时，使用聚簇特别有利，可以省去对结果集的排序操作**

聚簇存取方法的选择（续）

- 聚簇的局限性
  - 一个基表上最多建立一个聚簇索引
  - 聚簇只能提高某些特定应用的性能
  - 建立与维护聚簇的开销相当大
    - 对已有关系建立聚簇，将导致关系中元组移动其物理存储位置，并使此关系上原有的索引无效，必须重建
    - 当一个元组的聚簇码改变时，该元组的存储位置也要做相应移动
      - 例，学生从CS换到DS系

聚簇存取方法的选择（续）

- 聚簇的适用条件：设计候选聚簇
  - 经常在一起进行连接操作的关系可以建立聚簇
  - 一个关系的一组属性经常出现在相等比较条件中，则该单个关系可建立聚簇
  - 一个关系的一个(或一组)属性上的值重复率很高，则此单个关系可建立聚簇
    - 即对应每个聚簇码值的平均元组数不太少。聚簇的效果不明显

聚簇存取方法的选择（续）

- 聚簇的适用条件：优化聚簇设计

  （1）从聚簇中删除经常进行全表扫描的关系

  （2）从聚簇中删除更新操作远多于连接操作的关系

  （3）从聚簇中删除重复出现的关系

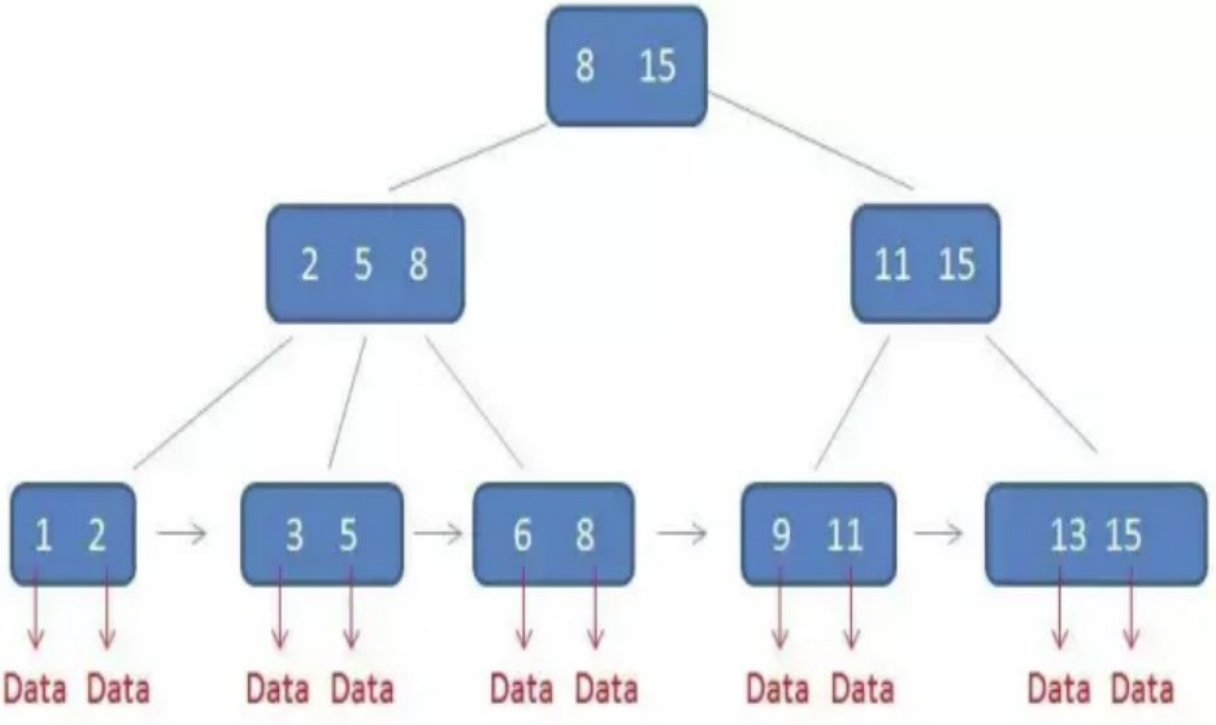
  - 不同的聚簇中可能包含相同的关系，一个关系可以在某一个聚簇中，但不能同时加入多个聚簇
    - 从这多个聚簇方案(包括不建立聚簇)中选择一个较优的，即在这个聚簇上运行各种事务的总代价最小

B+树存储

- 关系表的5种存放方式
  - 堆存储
  - 顺序存储
  - 多表聚簇存储
  - B+树存储
  - 哈希存储

B+树存储

- B+树存储
  - 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



哈希存储

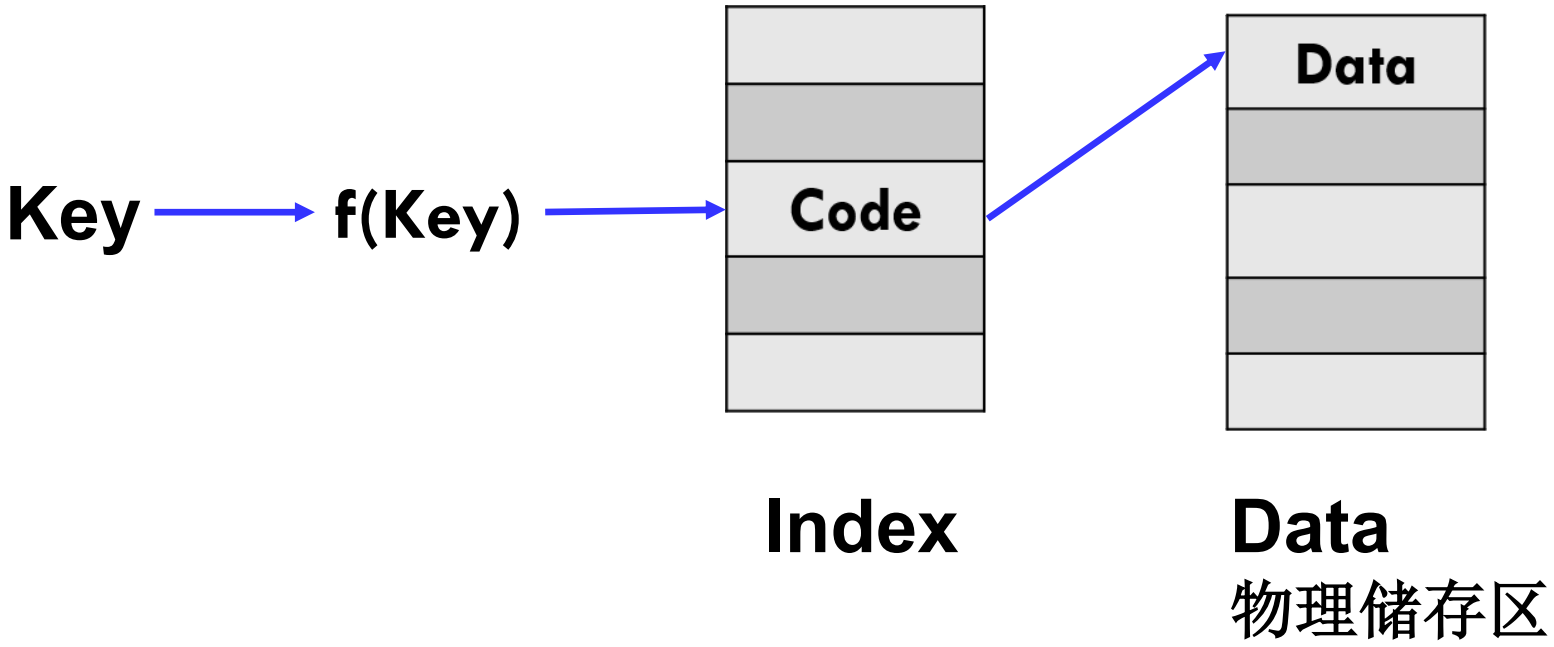
- 关系表的5种存放方式
  - 堆存储
  - 顺序存储
  - 多表聚簇存储
  - B+树存储
  - 哈希存储

哈希存储

- 哈希存储
  - 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

哈希存储

- 哈希存储



哈希存储

- 哈希存储

{Zhao, Qian, Sun, Li, Wu, Chen, Han, Ye, Dai}

设哈希函数 f(key) = ⌊(Ord(关键字首字母) - Ord('A') + 1) / 2⌋

0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Chen	Dai		Han		Li		Qian	Sun		Wu	Ye	Zhao

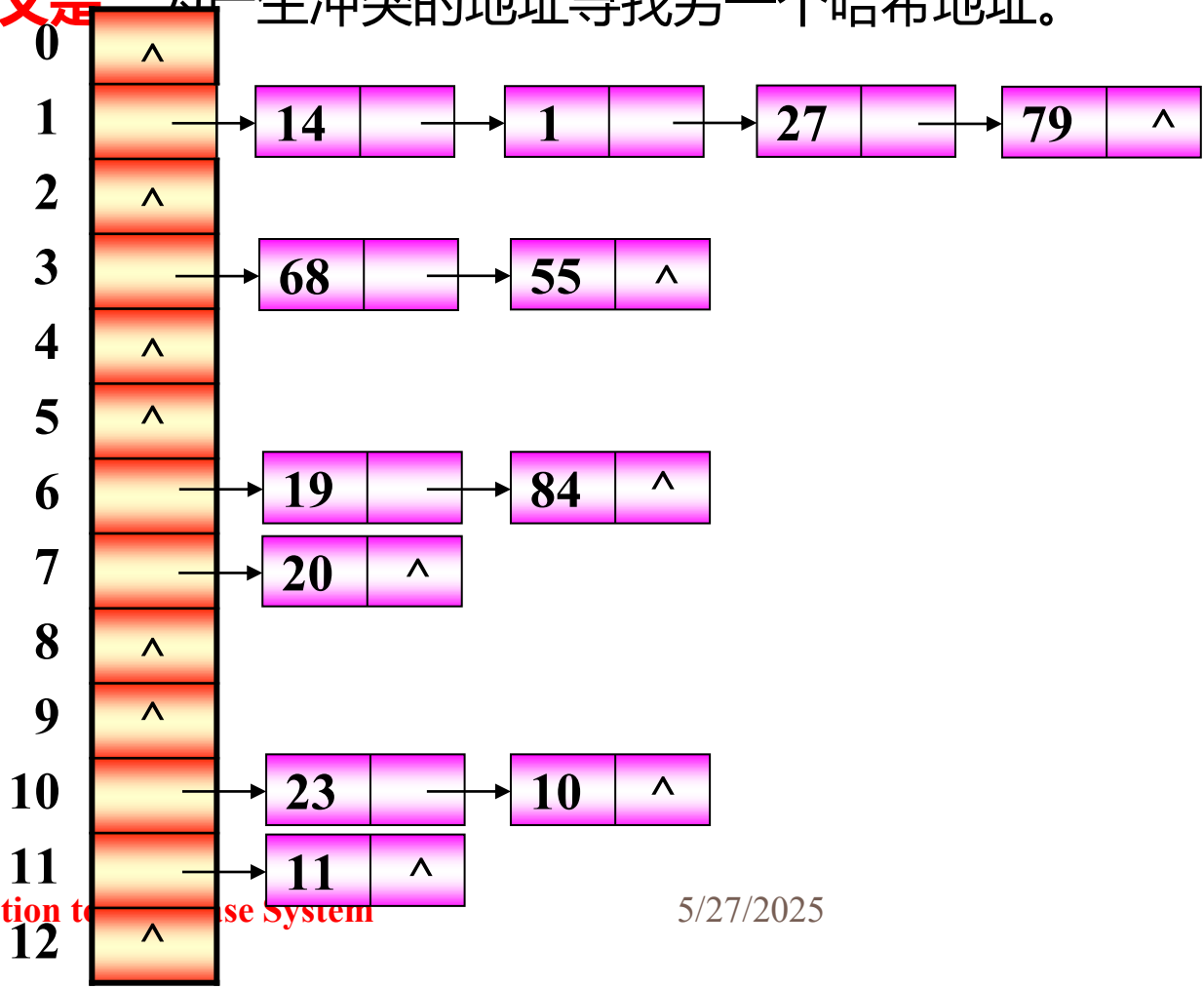
哈希存储

- **“处理冲突”的实际含义是**：为产生冲突的地址寻找另一个哈希地址。

例：已知一组关键字 (19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79)

哈希函数为：

H(key)=key MOD 13，用链地址法处理冲突。



哈希存储

- 哈希存储的优势
  - 将记录的存储位置与其指定属性的取值之间建立一个对应关系，查找记录时可快速定位这条记录

第九章 关系数据库存储管理

9.3 小结

- 本章主要内容
  - 关系数据库的数据组织方式
    - 逻辑组织方式
    - 物理组织方式：记录、块和关系表
  - 索引结构：顺序表索引、辅助索引、B+树索引、哈希索引和位图索引