

数据库系统原理与设计

(第3版)

传统教学

学生：

外部控制

外驱动机

独自学习

被动接受知识

重温已有知识

回答提问

注重学习结果

现代教学

自我控制

内驱动机

合作互动学习

主动构建知识

应用已有知识

提出问题、探究和解决问题

在乎学习过程

数据库系统原理与设计

(第3版)

第8章 数据库存储结构 与查询处理

目 录

8.1

文件组织与记录组织

8.2

索引与散列

8.3

查询处理

8.4

查询优化

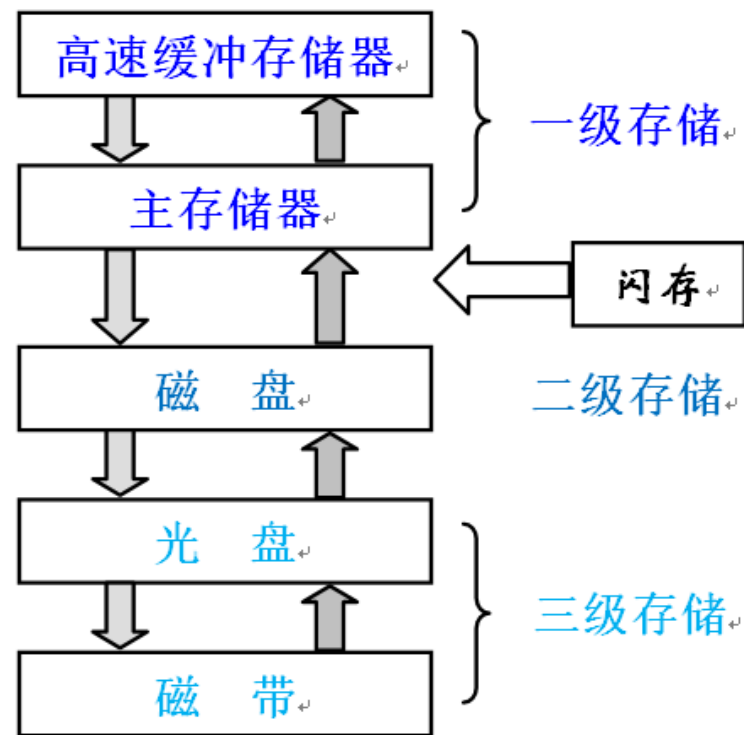
8.5

物理数据库设计

存储介质的

■ 几种有代表性的存储介质：

- 高速缓冲存储器(cache)
- 主存储器(main memory)
- 快闪存储器(flash memory)
- 磁盘存储器(magnetic-disk storage)
- 光存储器(optical storage)
- 磁带存储器(tape storage)



■ 计算机的三级存储体系：根据存储介质的速度和成本，形成三级存储：层次越高，价格越贵，速度越快。如图8-1所示。

■ 存储易失性问题：

- 一级存储为易失性存储，二、三级存储都是非易失性存储；
- 易失性存储在设备断电后将丢失所有内容。

磁盘的主要性能指标

- **访问时间(access time)**是从发出读写请求到数据开始传输之间的时间。
 - **访问时间** = **寻道时间** + **旋转等待时间**。
 - 为了访问(即读或写)磁盘上指定扇区的数据, 磁盘臂首先需要移动以定位到正确的磁道, 所需时间称为**寻道时间(seek time)**;
 - 然后等待磁盘旋转直到指定的扇区出现在它下方, 所需的时间称为**旋转等待时间(rotational latency time)**。
- **数据传输率(data-transfer rate)**是从磁盘获得数据(即读操作)或者向磁盘存储数据(即写操作)的速率。
- **磁盘的平均故障时间(mean time to failure, MTTF)**是指磁盘无故障连续运行时间的平均值。
- **磁盘块(block)**是一个**逻辑单元**, 它是包含固定数目的连续扇区。数据在**磁盘**和**主存储器(缓冲区)**之间**以块为单位传输**。

存储访问

- **缓冲区(buffers)**是主存储器中用于存储**磁盘块**的**副本**的区域。
缓冲区中的每个**块**总有一个**副本**存放在**磁盘**上，但是在**磁盘**上的**副本**可能比在**缓冲区**中的**副本**旧。
 - **缓冲区管理器**：负责**缓冲区**空间分配和管理的子系统。
 - 数据库管理系统**通过缓冲区**实现对**磁盘**上数据的存储访问。

■ **缓冲区(buffers)**是主存储
缓冲区中的每个块总有一
上的副本可能比在缓冲区

- **缓冲区管理器**：负责缓

- 数据库管理系统通过缓

■ **DBMS中数据的存取过程**：

■ 具体步骤如下：

- (1) **应用程序**通过DML向**RDBMS**发出存取请求，如Select语句；
- (2) **RDBMS**对命令进行**语法检查**，检查通过后继续检查**语义**和**用户权限**（利用**数据字典DD**），并决定是否接收；
- (3) **RDBMS**执行**查询优化**（利用**数据字典DD**），将命令转换成一串**单记录**的**存取操作序列**；

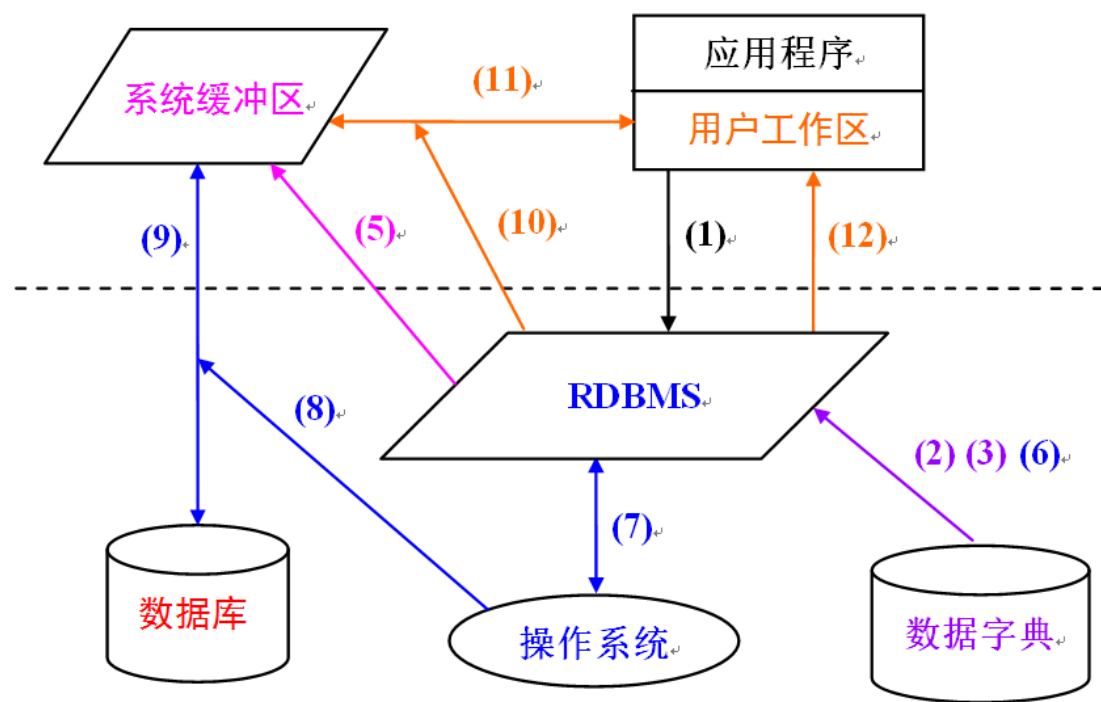


图 8-2 数据存取过程

- **缓冲区(buffers)**是主存储缓冲区中的每个块总有一上的副本可能比在缓冲区
- **缓冲区管理器**：负责缓
- 数据库管理系统通过缓
- **DBMS中数据的存取过程**：

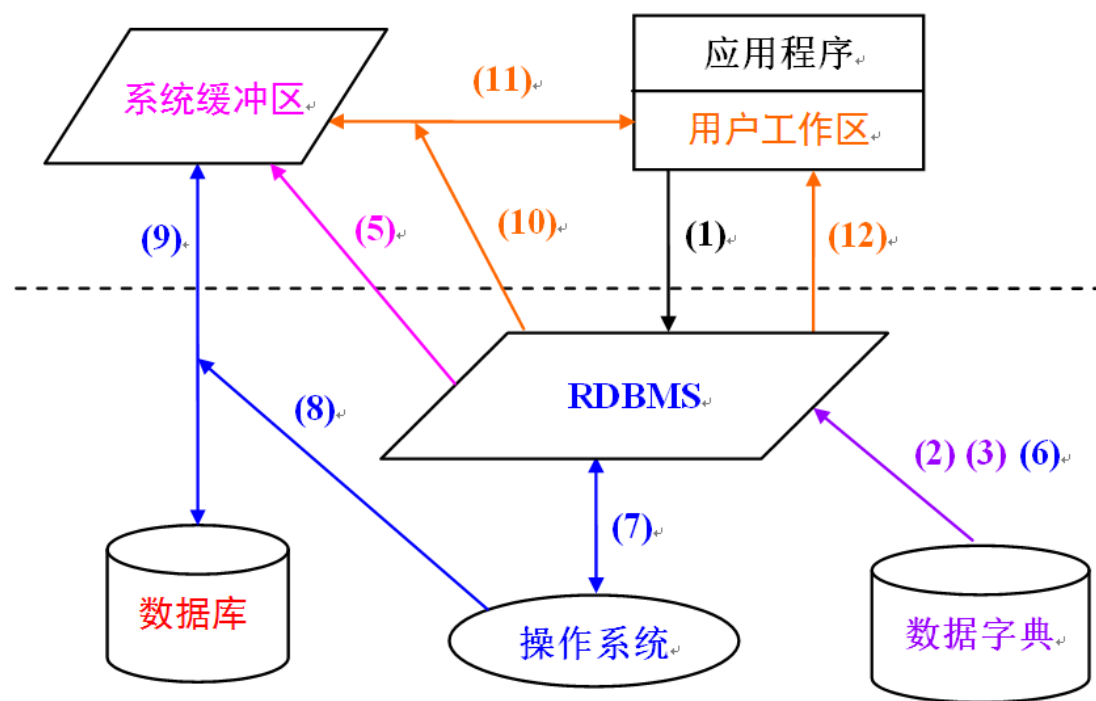


图 8-2 数据存取过程

- (4) 反复执行以下各步，直到结束(执行**存取操作序列**):
- (5) 在**系统缓冲区**中找**记录**，若找到转(10)，否则转(6)从**数据库**中获取；
- (6) 从**DD**查看**存储模式**，决定从**哪个文件**、**用什么方式**读取**物理记录**；
- (7) 根据(6)的结果向**操作系统(OS)**发出读取**记录**的命令；
- (8) **OS**执行该命令，并读取**记录**所在的**文件块(磁盘块)**；
- (9) 在**OS**控制下，将读出的**记录**所在的**文件块**送入**系统缓冲区**；

- **缓冲区(buffers)**是主存储缓冲区中的每个块总有一上的副本可能比在缓冲区
- **缓冲区管理器**：负责缓
- 数据库管理系统通过缓
- **DBMS中数据的存取过程**：

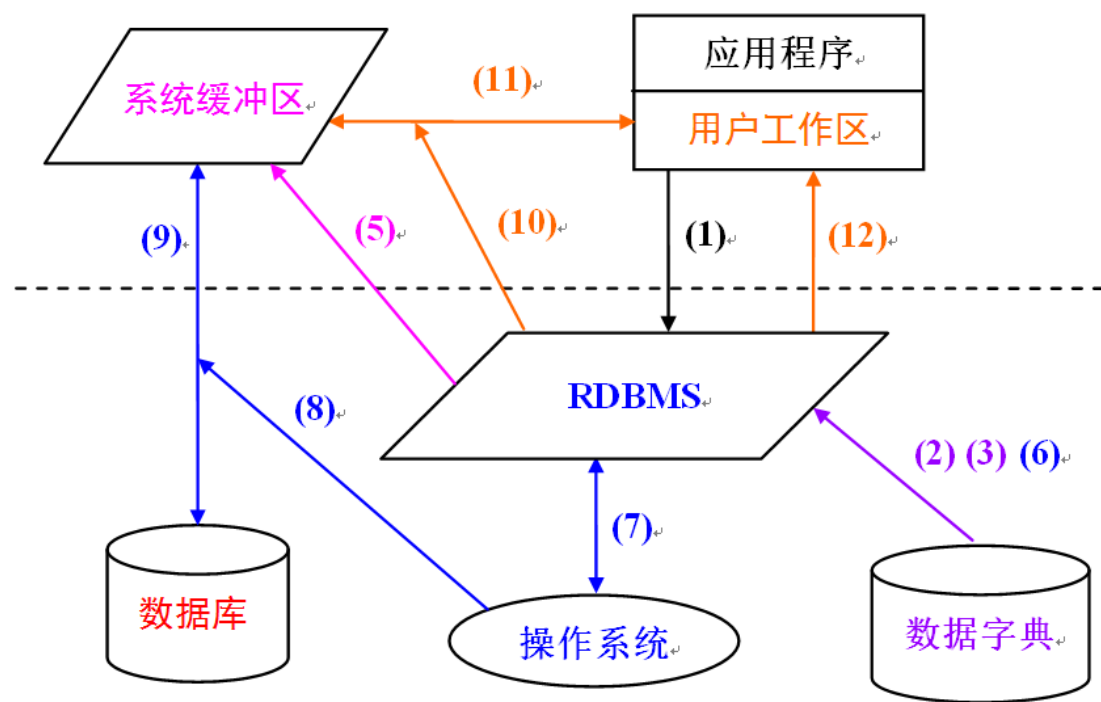
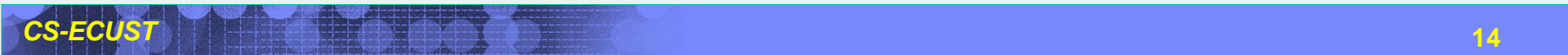
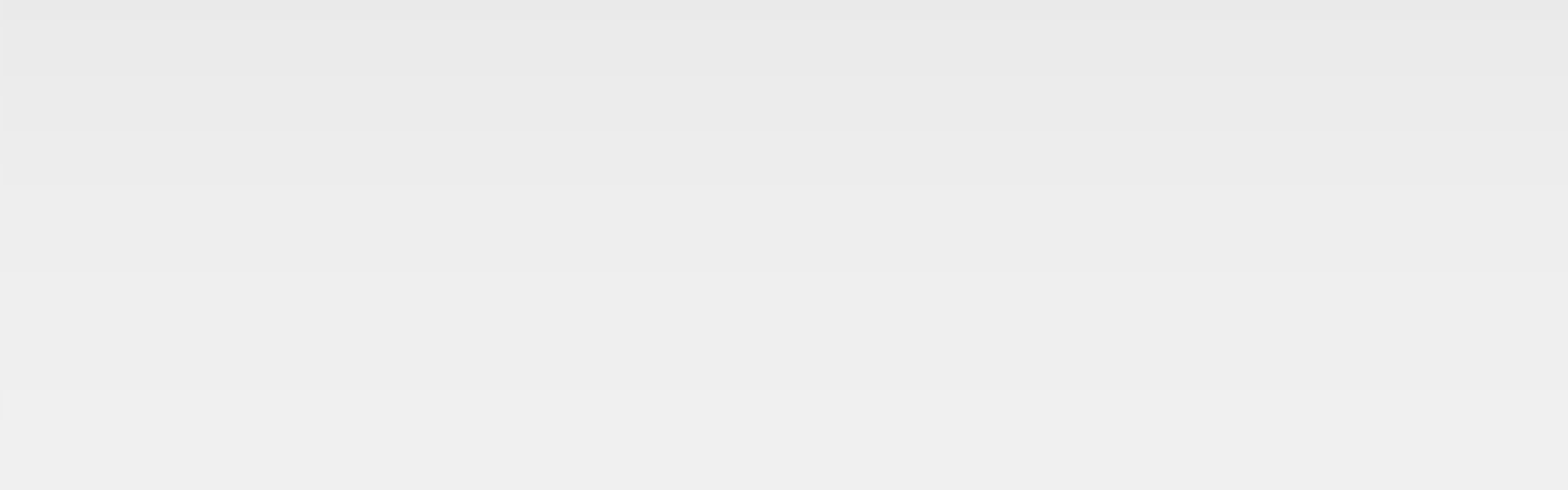
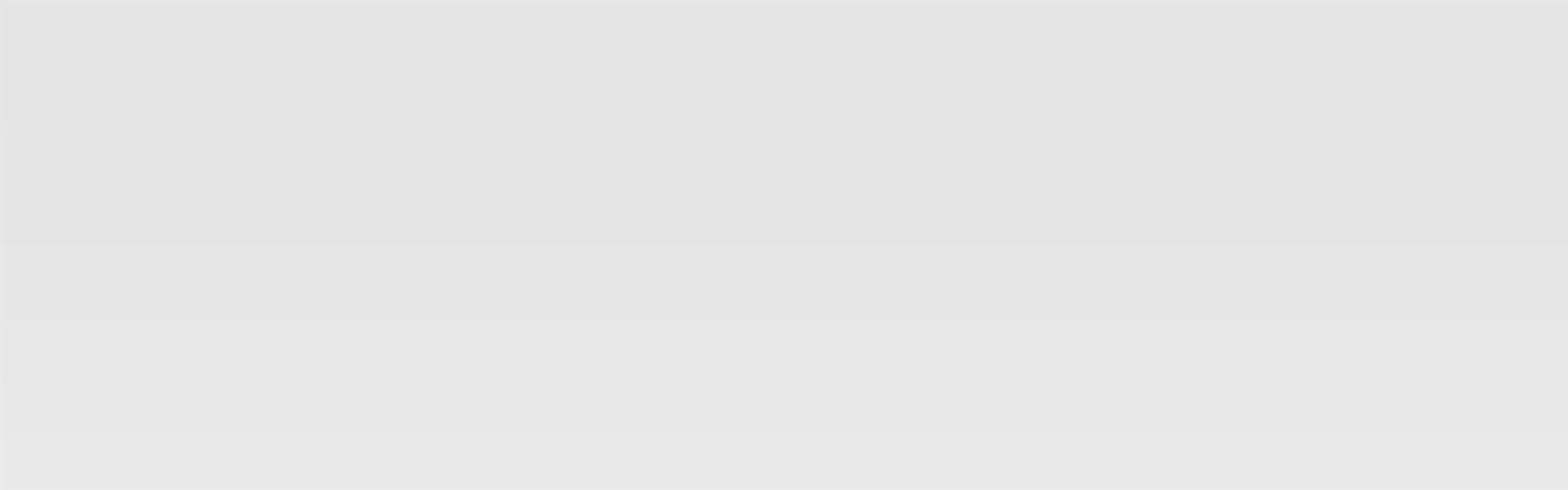
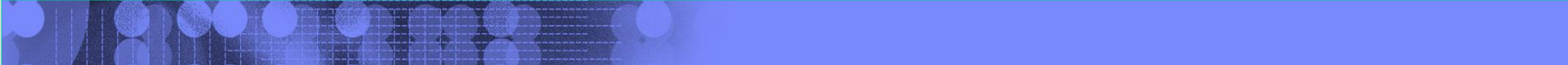


图 8-2 数据存取过程

- (10) **RDBMS**根据查询命令和**DD**的内容导出用户所要读取的**记录格式**；
- (11) **RDBMS**将数据从**系统缓冲区**中送入**用户工作区**；
- (12) **RDBMS**将执行状态信息(成功或不成功等)返回给**应用程序**；
- (13) **应用程序**对**用户工作区**中读入的数据进行处理。数据处理后，可能还要从**用户工作区**写回**系统缓冲区**；最后从**系统缓冲区**存入**数据库**中。



定长记录与变长记录

- **文件**在逻辑上可看作**记录的序列**，这些**记录**被映射到**磁盘**的**物理块**上。
- 用**文件**表示逻辑数据模型的不同方式：**定长记录**和**变长记录**
- 所谓**定长记录**指**文件**中所有**记录**均具有同样的字节长度，如图8-3所示：

记录 0	1501001	151	CS005	92
记录 1	1501001	152	CS012	88
记录 2	1501008	151	CS005	86
记录 3	1501008	152	CS012	93
记录 4	1501008	161	CP001	78
记录 5	1602002	161	CP001	95
记录 6	1602002	161	CS008	85
记录 7	1602005	162	CS012	72

图 8-3 存储 Score 记录的文件

有什么问题？

■ 这种简单的方法明显地有两个问题：

- 删除一条记录比较困难。要么填充被删空间，要么标记被删记录；
- 除非块的大小恰好是记录大小的倍数，否则有的记录会跨块存储。
对于跨块存储的记录的访问需要涉及两次磁盘I/O操作。

■ 一般对被删除记录做标记，且使用空闲记录链表来管理记录的插入和删除，如图8-4所示：

文件头					
记录 0	1501001	151	CS005	92	
记录 1					
记录 2	1501008	151	CS005	86	
记录 3	1501008	152	CS012	93	
记录 4					
记录 5	1602002	161	CP001	95	
记录 6					
记录 7	1602005	162	CS012	72	

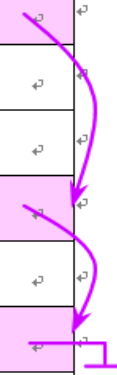


图 8-4 存储 Score 记录的文件

- 在**文件**开始处，分配一定数量的字节作为**文件头**(file header)，**文件头**中存储有关**文件**的各种信息。
- 到目前为止，需要在**文件头**中存储的信息只有一个，即第一条**被删除记录**(即第一条可用**记录**)的地址。
- 一般对**被删除记录**做标记，且使用**空闲记录链表**来管理**记录**的插入和删除，如图8-4所示：

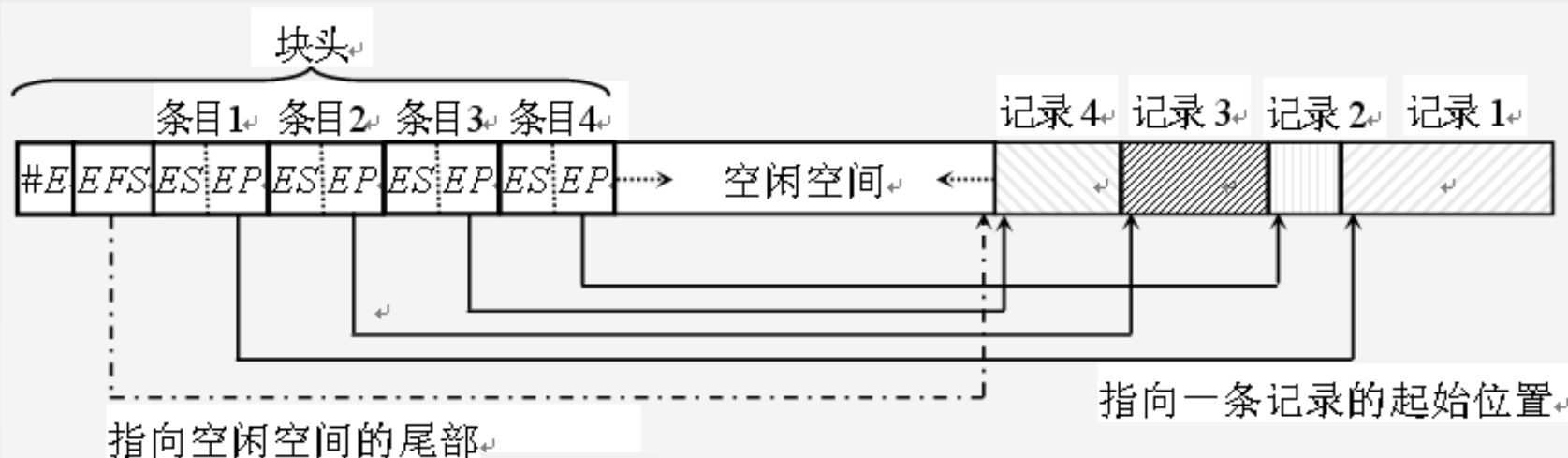
文件头					
记录 0	1501001	151	CS005	92	
记录 1					
记录 2	1501008	151	CS005	86	
记录 3	1501008	152	CS012	93	
记录 4					
记录 5	1602002	161	CP001	95	
记录 6					
记录 7	1602005	162	CS012	72	

图 8-4 存储 Score 记录的文件

定长记录与变长记录

- 变长记录指文件中的记录具有不同的存储字节数。
- 在数据库管理系统中，以下几种情况会导致使用变长记录：
 - 多种记录类型(即多个关系表)在一个文件中存储；
 - 允许记录类型中包含一个或多个变长字段；
 - 允许记录类型中包含重复字段(属性)，如数组等。
- 目前已有多种变长记录的存储管理技术。

- 有多种变长记录的存储管理技术，这里仅介绍**分槽页结构 (slotted-page structure)**。分槽页结构一般用于在块中组织记录，如下图所示。
- 每个块的开始处有一个块头，块头中包含的信息有：
 - 块头中已存储的**条目(entry)个数 $\#E$** (number of entries);
 - 块中空闲空间的**末尾地址 EFS** (end of free space);
 - 条目数组，每个条目中存储了该条目所对应**变长记录的大小 ES** (entry size)和**地址 EP** (entry pointer)。

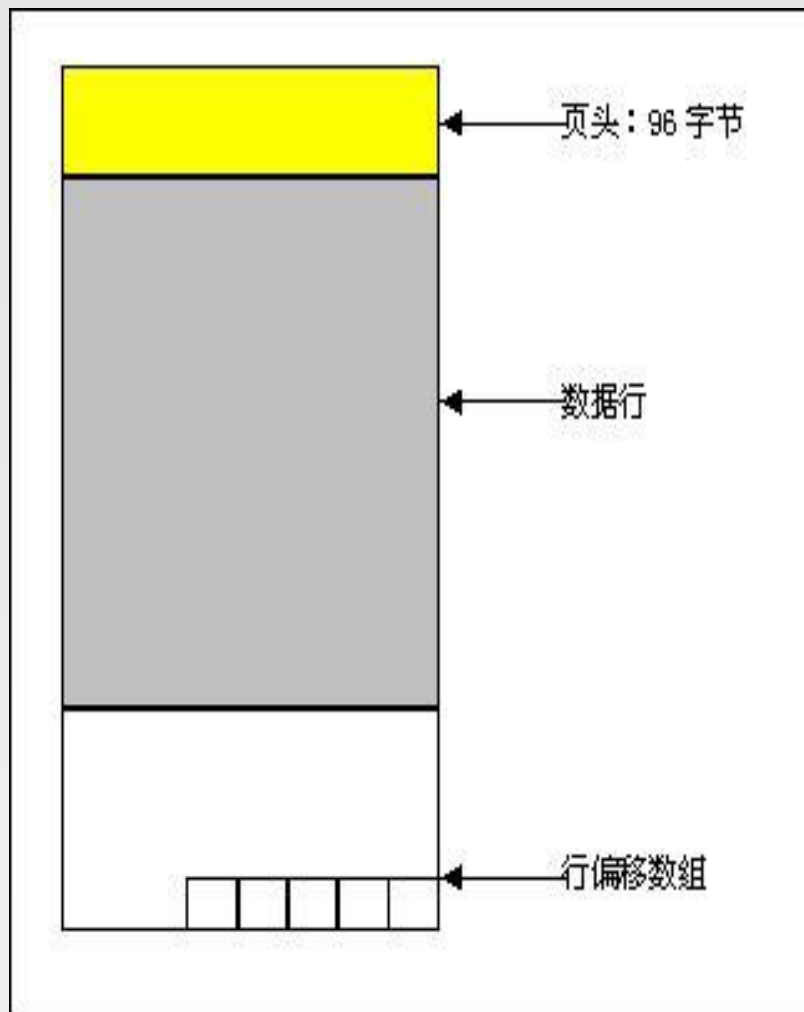


分槽页结构示意图

➤ SQL SERVER 的页

- ✓ 数据页是表的数据的存储结构;
- ✓ 数据页的固定大小为8KB，即8192字节，其中8096字节用于存储数据，其余的96字节用于存储页结构信息;
- ✓ 数据页由3个主要的部分组成：它们是……

记录不能跨页存储，它的最大长度是8060字节！



➤Oracle的页格式:

Oracle的页大小可以设定, 如2048或4096等

头部含通用的块信息,
如块地址和段类型 (数
据段或索引段)

Command and Variable Header

记录在本块有数据的表的信息

Table Directory

记录块中实际行的信息
(如每个行的地址等)

Row Directory

Free Space

ROW DATA

Oracle允许记录跨页存
储, 一个页中也可以存
储多张表的数据

记录组织

■ 文件中组织记录的常用方法有：

- 堆文件组织、顺序文件组织、多表聚集文件组织。本节介绍。
- B⁺树文件组织、散列(hashing)文件组织。将在8.2.3, 8.2.4节介绍。

■ 堆文件组织：

- 一条记录可以放在文件中的任何地方，只要那个地方有空间。
- 也就是说，文件中的记录是没有顺序的，是堆积起来的。
- 通常每个关系使用一个单独的文件。

记录

■ 文件中组织记录的常用方法有

- 堆文件组织、顺序文件组织、
- B+树文件组织、散列(hashing)

■ 堆文件组织：

- 一条记录可以放在文件中的任意位置
- 也就是说，文件中的记录是没有顺序的
- 通常每个关系使用一个单独的堆文件

■ 顺序文件组织：

- 为了高效地按某搜索码(如课程号)值的顺序有序处理记录而设计。
- 通过指针将磁盘块逻辑上有序地链接起来。每个磁盘块的指针指向搜索码值顺序的下一个磁盘块 (图8-5中，假设一个磁盘块存储3条记录，其中第1个磁盘块目前只存储了2条记录)。
- 同时，为了减少顺序文件处理中磁盘块的访问数量，在物理上按搜索码值的顺序或者尽可能地接近搜索码值的顺序存储磁盘块。

搜索码 ↙

1501008	161	CP001	78	↓
1602002	161	CP001	95	
1501001	151	CS005	92	↓
1501008	151	CS005	86	
1602002	161	CS008	85	
1501001	152	CS012	88	↓
1501008	152	CS012	93	
1602005	162	CS012	72	

图 8-5 存储 Score 记录的顺序文件

■ 顺序文件中插入操作的处理：

- 在文件中按搜索码值定位待插入记录的磁盘块(记为磁盘块A)，假设在一个磁盘块内记录是按搜索码值顺序存储)。
- 如果磁盘块A中有空间(可能是删除后留下来的空间)，就在磁盘块A中插入新的记录；

■ 顺序文件中插入操作的处理：

- 在文件中按搜索码值定位待插入记录A)，假设在一个磁盘块内记录是
- 如果磁盘块A中有空间(可能是删除记录)，假设在一个磁盘块内记录是

- 否则申请一个溢出磁盘块，将磁盘块A中的记录平分一半到

溢出块中，并将待插入记录插入到磁盘块A或溢出块中去。

(基于图8-5，插入记录(1602002, CS006, 161, 68)后如图8-6所示)

- 不管哪种情况，都要调整指针，使其能按搜索码值的顺序把磁盘块链接起来。

1501008	161	CP001	78
1602002	161	CP001	95
1501001	151	CS005	92
1501008	151	CS005	86
1501001	152	CS012	88
1501008	152	CS012	93
1602005	162	CS012	72

1602002	161	CS006	68
1602002	161	CS008	85

图 8-6 执行插入后的顺序文件。

多表聚集文件组织

多表聚集文件组织

- **问题的提出**：两个关系中作**连接运算**时，最坏的情况下，每个相匹配的**记录**都处在不同的**磁盘块**中，这将导致**为获取所需的每一条记录都要读取一个磁盘块**。
- **问题的解决**：将**两个关系**的**元组**混合在一起聚集存储，从而支持高效的连接运算。如图8-7所示的**两个关系**，为了支持**高效连接运算**，可以采用**多表聚集文件结构**。

Student 关系			
studentNo	studentName	sex	birthday
1501001	李小勇	男	1998-12-21
1602002	刘方晨	女	1998-11-11

Score 关系			
studentNo	termNo	courseNo	score
1501001	151	CS005	92
1501001	152	CS012	88
1501001	161	CP001	86
1602002	161	CP001	95
1602002	161	CS008	85

图 8-7 Student 关系与 Score 关系

多表聚集文件组织

1501001	李小勇			男	1998-12-21
1501001	151	CS005	92		
1501001	152	CS012	88		
1501001	161	CP001	86		
1602002	刘方晨			女	1998-11-11
1602002	161	CP001	95		
1602002	161	CS005	85		

图 8-8 多表聚集文件组织结构

1501001	李小勇			男	1998-12-21
1501001	151	CS005	92		
1501001	152	CS012	88		
1501001	161	CP001	86		
1602002	刘方晨			女	1998-11-11
1602002	161	CP001	95		
1602002	161	CS005	85		

图 8-9 多表聚集文件组织结构

- **多表聚集文件组织(multitable clustering file organization)**是一种在每一个块中存储两个或多个关系的相关记录的文件结构。
- 多表聚集文件组织，可加速特定查询(如查学生-选课信息)；但是，它将导致其它类型查询的处理变慢(如查学生信息)。
- 为此，在图8-9中，通过指针将一个关系中的所有记录链接起来以方便查找。

目 录

8.1

文件组织与记录组织

8.2

索引与散列

8.3

查询处理

8.4

查询优化

8.5

物理数据库设计

索引基本概念

■ 两种基本的索引类型:

- 顺序索引(ordered index): 索引中的记录(索引项)基于搜索码值顺序排列。

- 搜索码(search key): 用于在文件(指数据文件, 如存放学生表的文件)中查找记录的属性或属性集。经常需要在在一个文件上建立多个索引, 此时该文件就有多个搜索码。

索引基本概念

■ 两种基本的索引类型:

● **顺序索引(ordered index):** 索引中的记录(索引项)基于搜索码值顺序排列。

- **组织结构:** 在索引中按搜索码值的顺序存储索引项, 并将索引项与包含该索引项中搜索码值的文件记录关联起来(通过指针)。
- 用于支持快速地对文件中的记录进行顺序或随机地访问。



索引基本概念

■ 两种基本的索引类型:

- **顺序索引(ordered index):** 索引中的记录(索引项)基于搜索码值顺序排列。
 - **组织结构:** 在索引中按搜索码值的顺序存储索引项, 并将索引项与包含该索引项中搜索码值的文件记录关联起来(通过指针).
 - 用于支持快速地对文件中的记录进行顺序或随机地访问。
- **散列索引(hash index):** 索引中的记录(索引项)基于搜索码值的散列函数(即哈希函数)的值平均、随机地分布到若干个散列桶中。

索引基本概念

- 建立了索引的文件称为索引文件(指数据文件)。索引文件中的记录自身可以按照某种排序顺序存储。一个索引文件可以有多个索引，分别对应于不同的搜索码。
- 如果索引文件中的记录按照某个搜索码值指定的顺序物理存储，那么该搜索码对应的索引就称为主索引(primary index)，也叫聚集索引(clustering index)。
- 与此相反，搜索码值顺序与索引文件中记录的物理顺序不同的那些索引称为辅助索引(secondary index)或非聚集索引(nonclustering index)。

顺序索引：索引顺序文件

- 建立了主索引的索引文件称为索引顺序文件(index-sequential file)。也就是说，索引顺序文件是按主索引的搜索码值物理有序存储。
- 对于索引顺序文件，顺序索引有两类：稠密索引和稀疏索引。



- **稠密索引**。对应索引顺序文件中搜索码的每个值在索引中都有一个索引记录(或称为索引项)。每一个索引项包含搜索码值和指向具有该搜索码值的第一个数据记录的指针，如图8-10所示，其中studentName是搜索码。



- **稠密索引**。对应**索引顺序文件**中**搜索码**的每个**值**在**索引**中都有一个**索引记录**(或称为**索引项**)。每一个**索引项**包含**搜索码值**和指向具有该**搜索码值**的第一个数据记录的**指针**，如图8-10所示，其中studentName是**搜索码**。
- **稀疏索引**。**稀疏索引**只为**索引顺序文件**中**搜索码**的某些**值**建立**索引记录**(或称为**索引项**)。每一个**索引项**包含**搜索码值**和指向具有该**搜索码值**的第一个数据记录的**指针**，如图8-11所示。

索引顺序文件

搜索码

顺序索引

李宏冰	●
李小勇	●
刘方晨	●
王 红	●
王红敏	●

稠密顺序索引

1503010	李宏冰	女	2000-03-09	会计学	1
1501001	李小勇	男	1998-12-21	计算机	2
1603025	李小勇	女	1999-03-18	会计学	3
1602002	刘方晨	女	1998-11-11	信息系统	4
1501008	王 红	男	2000-04-26	计算机	5
1503045	王 红	男	2000-04-26	会计学	6
1602005	王红敏	女	1998-10-01	信息系统	7

索引顺序文件

搜索码

顺序索引

李宏冰	●
刘方晨	●
王 红	●

稀疏顺序索引

1503010	李宏冰	女	2000-03-09	会计学	1
1501001	李小勇	男	1998-12-21	计算机	2
1603025	李小勇	女	1999-03-18	会计学	3
1602002	刘方晨	女	1998-11-11	信息系统	4
1501008	王 红	男	2000-04-26	计算机	5
1503045	王 红	男	2000-04-26	会计学	6
1602005	王红敏	女	1998-10-01	信息系统	7

顺序索引：多级索引

- 即使采用稀疏顺序索引，对于一个大型数据库而言，顺序索引本身也可能变得很大。
- 如果索引过大，主存中不可能读入所有的索引块，也就是大部分索引块只能存储在磁盘上，这样在查询处理过程中，搜索索引就必须读大量的磁盘块。
- 通过多级索引技术能够较好地解决上述问题。所谓多级索引就是在索引之上再建立索引。

- 即使采用索引本身
- 如果索引是大部分程中，指
- 通过多级索引就是

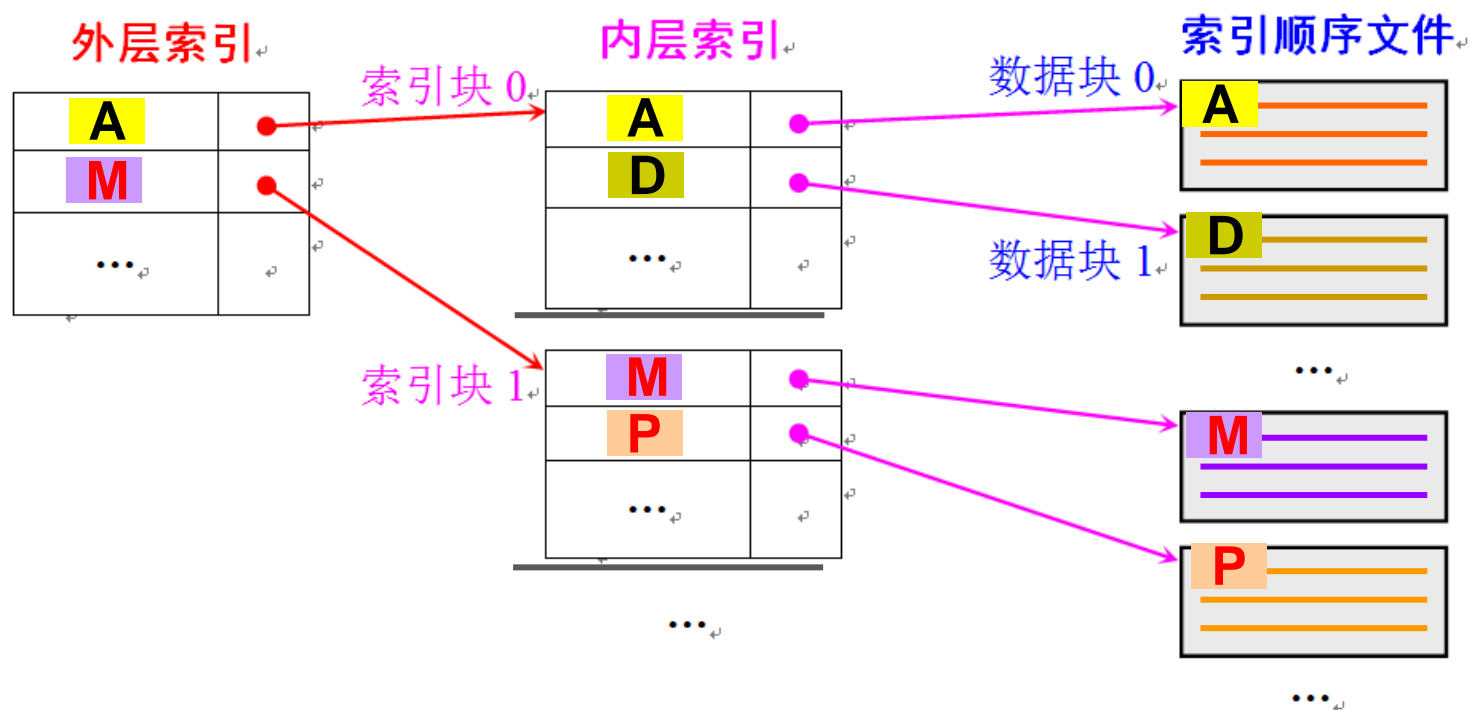


图 8-12 二级稀疏索引结构

- 像对待其他顺序文件那样对待顺序索引，在顺序索引之上再构造一个稀疏顺序索引，如图8-12所示（假设对每个磁盘块建立一个索引项——稀疏顺序索引）。

小 结

- **三级存储体系**：一级存储为**易失性存储**，二、三级存储都是**非易失性存储**。
- **数据访问**：访问时间(寻道时间+旋转等待时间)+数据传输时间。
- **磁盘块**：一个**逻辑单元**，存与内存之间交换数据的**单位**。
- **缓冲区**：**RDBMS**通过**缓冲区**实现对磁盘上数据的存储访问，即**缓冲区**是**程序工作区**与**数据库**之间交换数据的桥梁。
- **记录格式**：定长记录和变长记录。
- **记录组织**：—— **B+树文件组织**、**散列文件组织**(还未介绍)。
 - **堆文件组织**：**文件**中的**记录**没有顺序，是**堆积**起来的。
 - **顺序文件组织**：**文件**中的**记录**按**搜索码值**有序。**磁盘块**内的**记录**有序，且通过**指针**将**磁盘块****逻辑上**有序地链接起来。
 - **多表聚集文件组织**：在一个**块**中**存储多个关系的相关记录**。

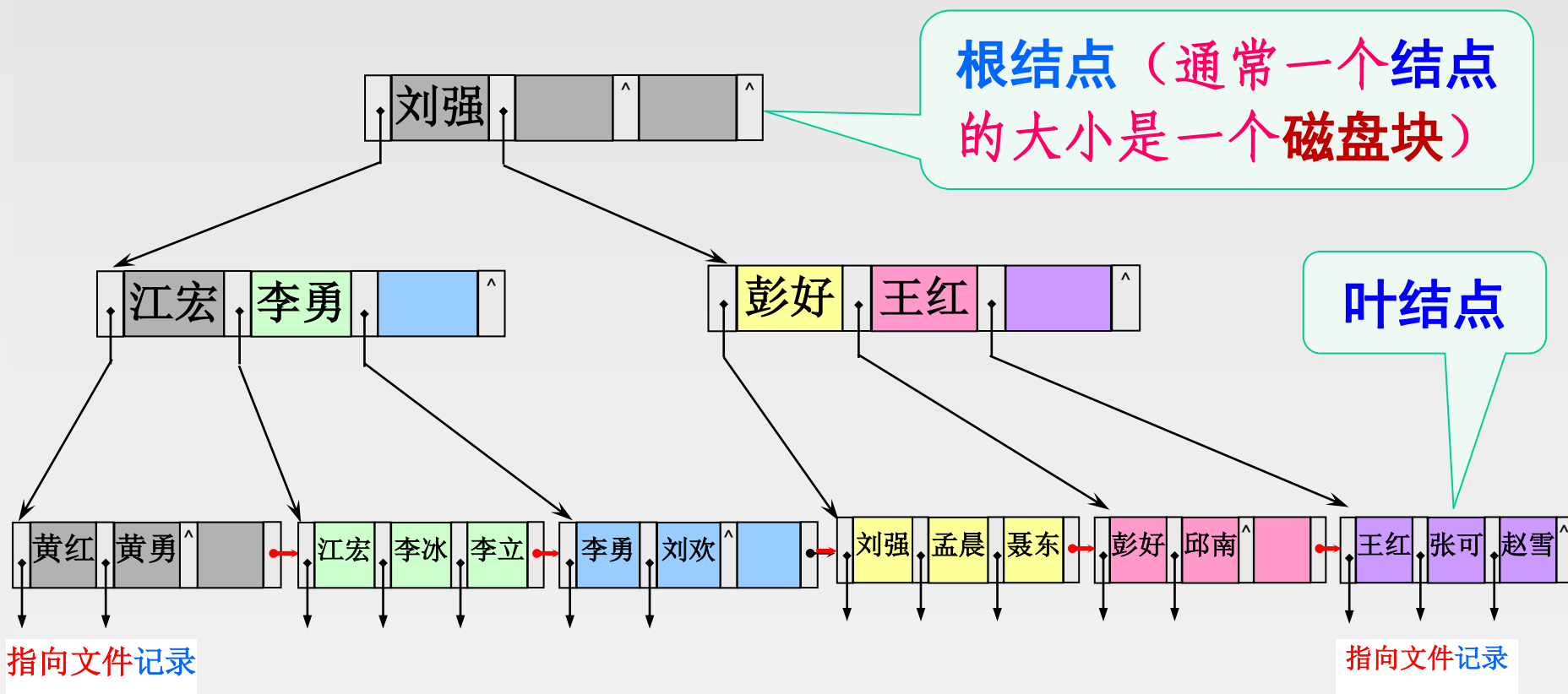
小 结

- 顺序索引：索引中的记录(索引项)基于搜索码值的顺序排列。
- 散列索引：索引中的记录(索引项)基于搜索码值的散列函数(即哈希函数)的值平均、随机地分布到若干个散列桶中。
- 顺序索引：稠密索引、稀疏索引。
- 索引文件：建立了索引的文件。
- 主索引 或 聚集索引 —— 索引顺序文件。
- 辅助索引 或 非聚集索引 —— 只能是稠密索引。
- 多级索引：在索引之上再建立索引。

B⁺树索引的结构

■ B⁺树索引的结构满足：

- B⁺树索引是一个多级索引，但其结构不同于多级顺序索引
- B⁺树索引采用平衡树结构，即每个叶结点到根结点的路径长度相同



B⁺树索引的结构

叶结点链表： 搜索码值在逻辑上按顺序存储

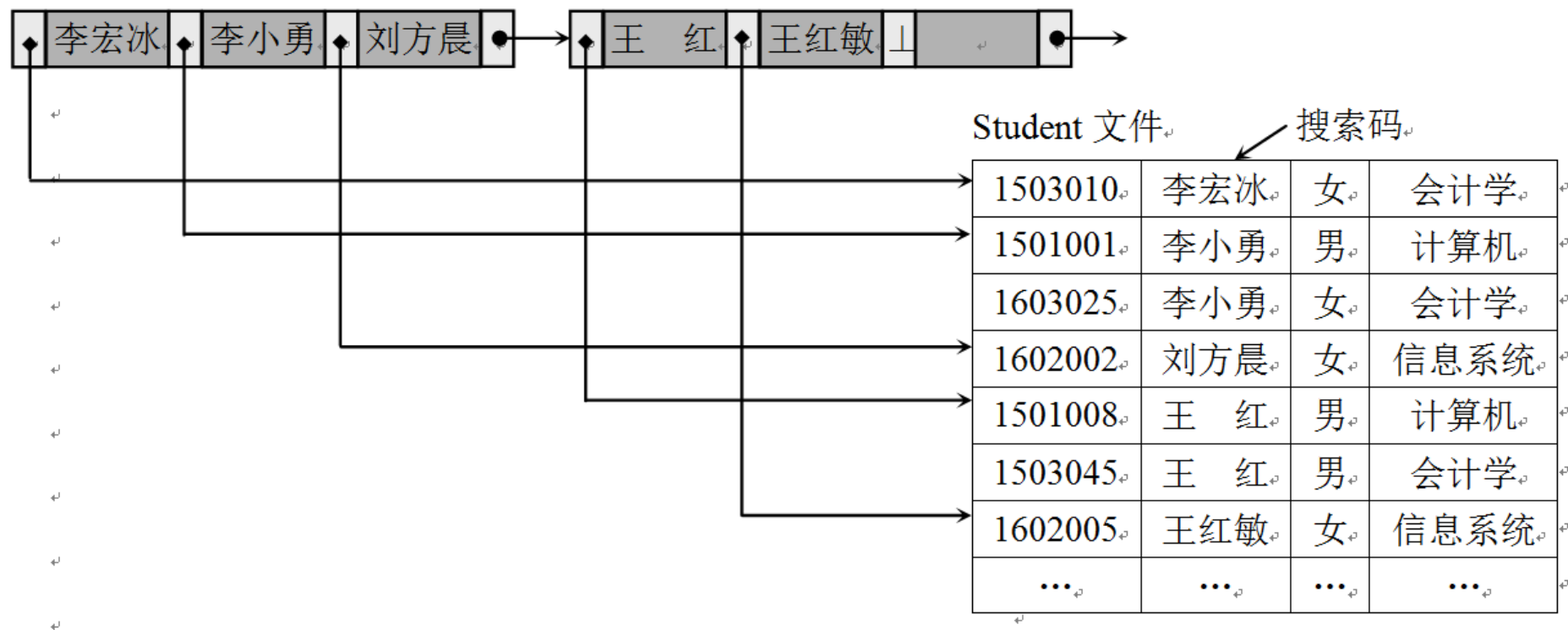
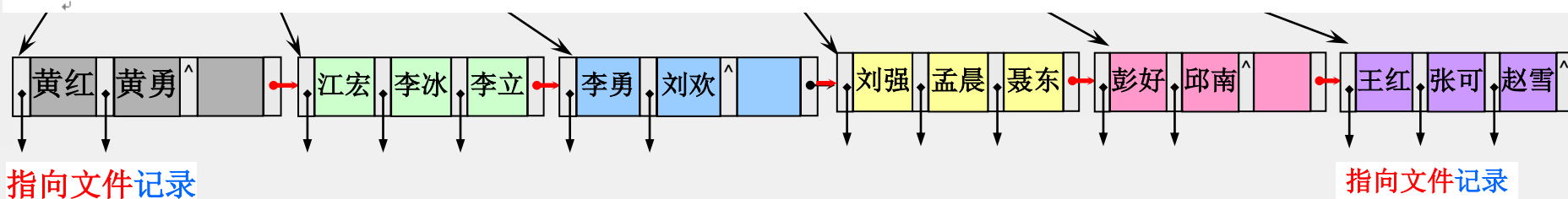


图 8-15 Student 文件的 B⁺树索引($n=4$)的叶结点结构



查询成功与失败的示意图

查找“裘北”

查找“李冰”

查找“张可”

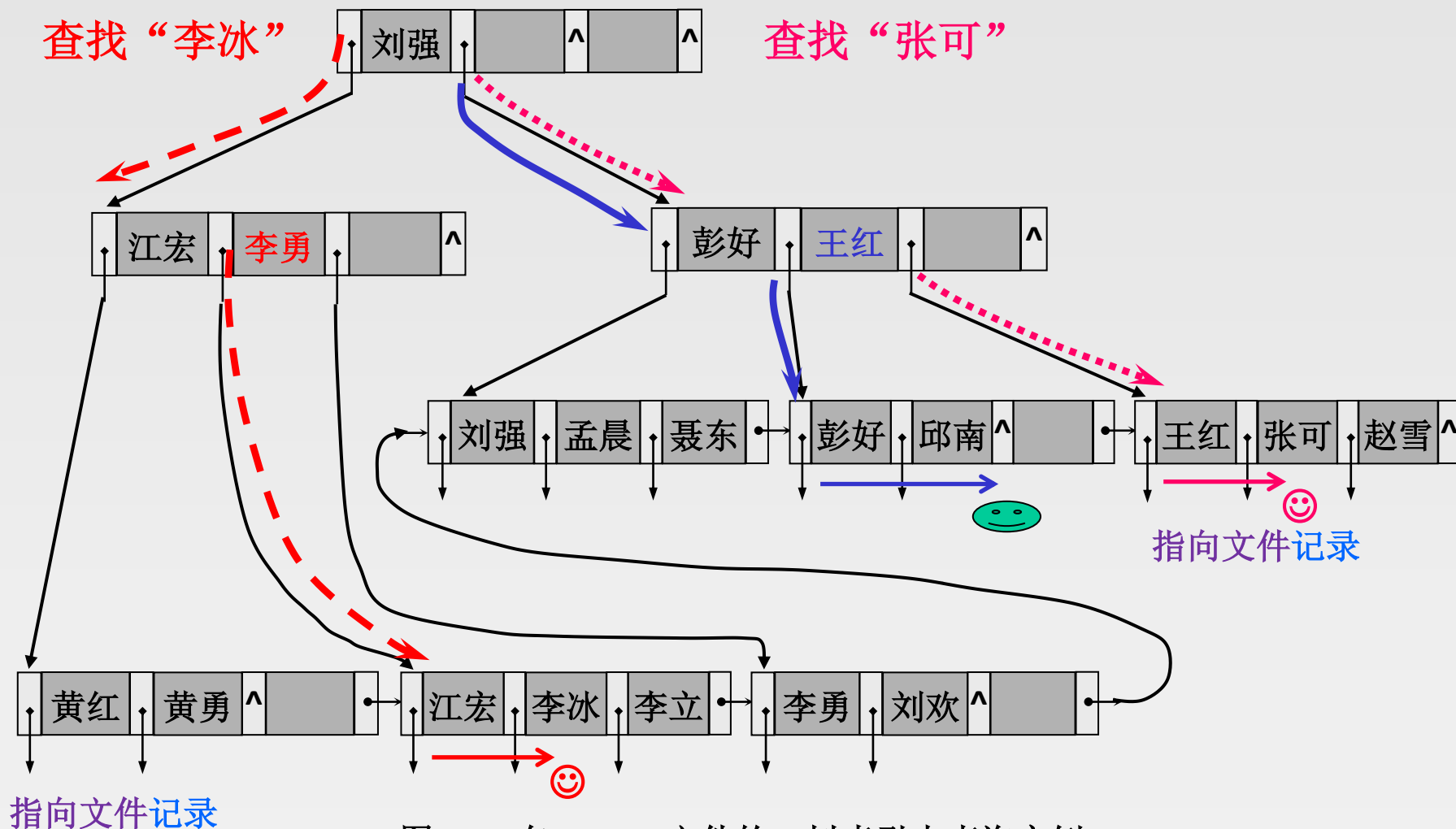


图8-18 在Student文件的B+树索引中查询实例

B⁺树文件组织

- B⁺树文件组织是通过在B⁺树的叶结点层直接包含真实的数据记录(即每个叶结点中直接存放若干条数据记录, 而不是存放搜索码值和指向记录的指针), 以解决索引顺序文件中随着文件的增大而性能下降的缺点。
- 在B⁺树文件组织中, B⁺树结构不仅用做索引, 同时也是文件中记录的组织者, 树叶结点中存储的是记录。
- 如图8-19是Student文件的B⁺树文件组织结构。

B⁺树文件组织

文件进行重建

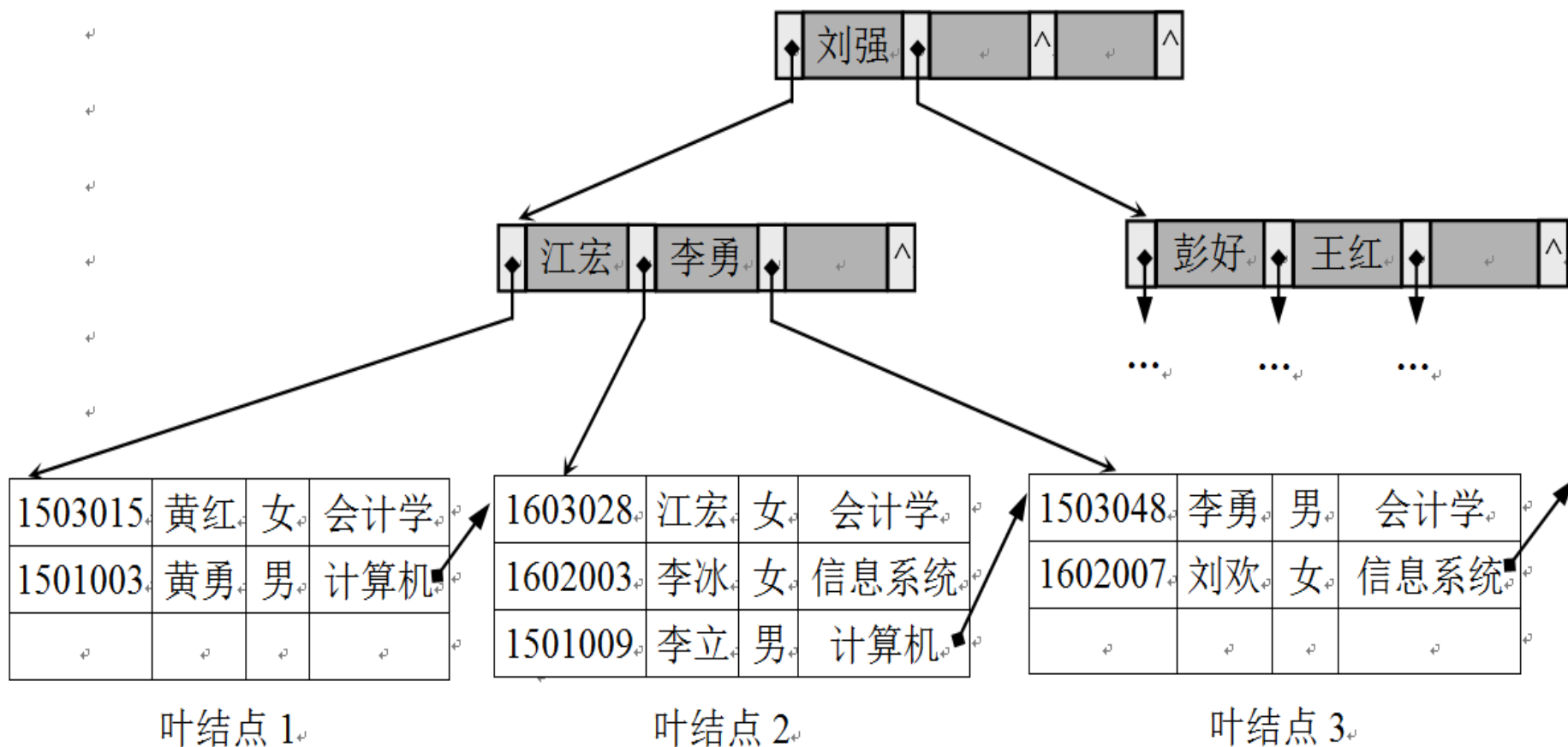


图 8-19 Student 文件的 B⁺树文件组织

散列 一个例子

某一文件有16个记录，其关键字分别为：23，05，26，01，18，02，27，12，07，09，04，19，06，16，33，24。桶的容量 $m=3$ ，桶数 $b=7$ 。用除余法作散列函数 $H(\text{key})=\text{key}\%7$ 。由此得到的散列文件如下图所示

桶编号	基桶			溢出桶
0	07			Λ
1	01			Λ
2	23	02	09	→ 16 Λ
3	24			Λ
4	18	04		Λ
5	05	26	12	→ 19 33 Λ
6	27	06		Λ

散列文件示例

散列文件组织

桶编号	基桶				溢出桶
0	07			A	
1	01			A	
2	23	02	09		16
3	24			A	
4	18	04		A	
5	05	26	12		19 33
6	27	06		A	

散列文件示例

- 在**散列**的描述中，用**散列桶**(即存储桶，简称为桶)来表示可以存储一条或多条**记录**的一个存储单位，通常一个桶就是一个**磁盘块**。
- 通过**散列函数**计算**搜索码值**的**散列值**，并根据该**散列值**来决定包含该**搜索码值**的**记录**应该存储在哪个**桶**中。
- **散列**可以用于两个不同的目的：
 - 在**散列文件**组织中，通过计算一条数据**记录**的**搜索码值**的**散列函数值**，可以直接获得包含该**记录**的**磁盘块(桶)**的**地址**；
 - 在**散列索引**组织中，把**搜索码值**以及与它们相关联的**记录指针**组织成一个**散列索引项**，通过计算一个**散列索引项**中**搜索码值**的**散列函数值**，可以直接获得该**散列索引项**的**磁盘块(桶)**的**地址**。

散列与顺序索引的比较

- **散列**其实就是一种**不通过值的比较**，而**通过值的含义**来确定存储位置的方法，它是为有效地实现**等值查询**而设计的。
- 不幸的是，基于**散列**技术**不支持范围检索**。
- 而基于**B+树**的**索引**技术能**有效地支持范围检索**，并且它的**等值检索**效果也很好。
- 但是，**散列**技术在**等值连接**等操作中是很有用的，尤其是在**索引嵌套循环连接**方法中，基于**散列**的**索引**和基于**B+树**的**索引**在代价上的差别会很大。

散列与顺序索引的选择

- 在实际的数据库设计中，到底是用索引还是散列，要充分考虑以下几个问题：
 - 索引或散列的周期性重组的代价如何？
 - 在文件中插入和删除记录的频率如何？
 - 是否愿意以增加最坏情况下的访问时间为代价优化平均访问时间？
 - 用户可能提出哪些类型的查询？

小结

■ B⁺树索引：高效支持范围查询，较高效支持等值查询

- 一个多级索引，采用平衡树结构，特点是胖而矮
- 所有结点的结构都相同，每个结点最多有 $n-1$ 个搜索码值 K_1, K_2, \dots, K_{n-1} ，以及 n 个指针 P_1, P_2, \dots, P_n ，每个结点中的搜索码值升序存放
- 每个非叶结点有 $\lfloor n/2 \rfloor$ 到 n 个孩子结点(根结点除外)，叶结点依次链接起来

■ B⁺树文件组织：叶结点组织文件记录，非叶结点起索引作用

■ 散列结构：高效支持等值查询，不支持范围查询

- 散列文件组织、散列索引(辅助索引)
- 散列函数，桶溢出 —— 闭散列、开散列，静态散列、动态散列

第九章 关系查询处理和查询优化

王占全

zhqwang@ecust.edu.cn

9.2 关系系统的查询优化

9.2.1 查询优化概述

9.2.2 查询优化的必要性

9.2.3 查询优化的一般准则

9.2.4 优化的策略：等价变换

9.2.5 关系代数表达式的优化算法

9.2.6 优化的一般步骤

9.2.1 查询优化概述

■为什么要进行查询优化

- 查询优化极大地影响RDBMS的性能。

■查询优化的可能性

- 关系数据语言为查询的优化提供了可能性。

由DBMS进行查询优化的好处

- 用户不必考虑如何最好地表达查询以获得较好的效率
- 2. 系统可以比用户程序的优化做得更好
 - (1) 优化器可以从数据字典中获取许多统计信息，而用户程序则难以获得这些信息

由DBMS进行查询优化的好处

- (2)如果数据库的物理统计信息改变了，系统可以自动对查询重新优化以选择相适应的执行计划。
- (3)优化器可以考虑数百种不同的执行计划，而程序员一般只能考虑有限的几种可能性。
- (4)优化器中包括了很多复杂的优化技术

查询优化目标

■ 查询优化的总目标

选择有效策略，求得给定关系表达式的值

查询优化的途径？

查询优化的途径包括

■ 查询优化分3步进行

- 1 逻辑优化，即产生逻辑上与给定关系代数表达式等价的表达式；
- 2 代价估计，即估计每个执行计划的代价；
- 3 物理优化，即对所产生的表达式以不同方式作注释，产生不同的查询执行计划。

• 查询优化的途径包括（5点）

3. 查询优化的目标和途径

• 查询优化的途径包括（续）

- 3) 有些查询可根据启发式规则选择执行策略，这称为规则优化。
- 4) 根据可供选择的执行策略进行代价估算，并从中选择代价最小的执行策略，这称为代价估算优化。
- 5) 此外，还可以通过应用数据库的语义信息对查询进行优化，这称为语义优化。

实际系统的查询优化步骤

1. 将查询转换成某种内部表示，通常是语法树
2. 根据一定的等价变换规则把语法树转换成标准（优化）形式
3. 选择低层的操作算法

对于语法树中的每一个操作：计算各种执行算法的执行代价；选择代价小的执行算法

4. 生成查询计划(查询执行方案)

查询优化
依靠的体
系结构？

■ 集中式数据库

➤ 单用户系统

总代价 = I/O代价 + CPU代价

➤ 多用户系统

总代价 = I/O代价 + CPU代价 + 内存代价

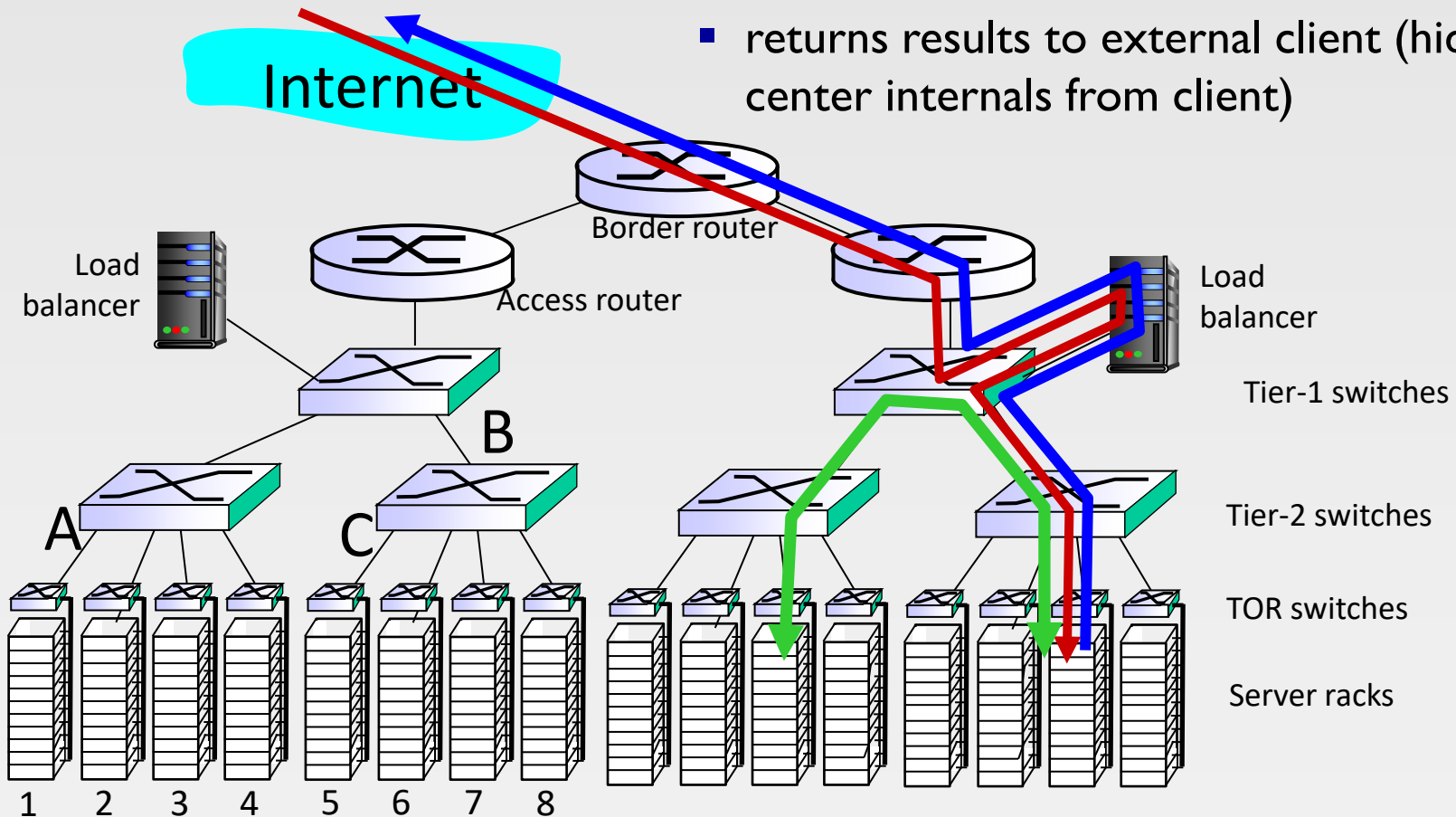
■ 分布式数据库

总代价 = I/O代价 + CPU代价[+ 内存代价] + 通信代价

Data center networks

load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



优化优点
优化途径
优化一般过程



9.2 关系系统的查询优化

- | 9.2.1 查询优化概述
- | 9.2.2 查询优化的必要性
- | **9.2.3 查询优化的一般准则**
- | 9.2.4 优化的策略：等价变换
- | 9.2.5 关系代数表达式的优化算法
- | 9.2.6 优化的一般步骤

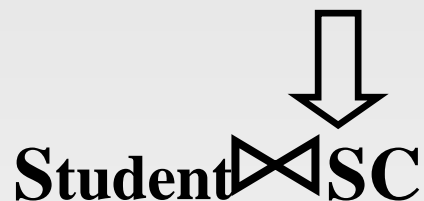
9.2.3 查询优化的一般准则

- 1选择运算应尽可能先做
 - 目的：减小中间关系
- 2在执行连接操作前对关系适当进行预处理
 - 按连接属性排序
 - 在连接属性上建立索引
- 3投影运算和选择运算同时做
 - 目的：避免重复扫描关系
- 4将投影运算与其前面或后面的双目运算结合
 - 目的：减少扫描关系的遍数

查询优化的一般准则（续）

- 5 某些选择运算 + 在其前面执行的笛卡尔积
====> 连接运算

例： $\sigma_{\text{Student.Sno}=\text{SC.Sno}} (\text{Student} \times \text{SC})$



9.2 关系系统的查询优化

- | 9.2.1 查询优化概述
- | 9.2.2 查询优化的必要性
- | 9.2.3 查询优化的一般准则
- | **9.2.4 优化的策略：等价变换**
- | 9.2.5 关系代数表达式的优化算法
- | 9.2.6 优化的一般步骤

9.2.4 优化的策略：等价变换

■ 关系代数表达式等价

- 指两个来自同一个关系的不同表达式所得到的结果是相同的
- 上面的优化策略大部分都涉及到关系代数表达式的等价变换

9.2 关系系统的查询优化

- 9.2.1 查询优化概述
- 9.2.2 查询优化的必要性
- 9.2.3 查询优化的一般准则
- 9.2.4 优化的策略：等价变换
- 9.2.5 关系代数表达式的优化算法
- 9.2.6 优化的一般步骤

9.2 关系系统的查询优化

9.2.1 查询优化概述

9.2.2 查询优化的必要性

9.2.3 查询优化的一般准则

9.2.4 优化的策略：等价变换

9.2.5 关系代数表达式的优化算法

9.2.6 优化的一般步骤

9.2.6 优化的一般步骤

1. 把查询转换成某种内部表示
2. 代数优化：把语法树转换成标准（优化）形式
3. 物理优化：选择低层的存取路径
4. 生成查询计划，选择代价最小的

优化的一般步骤 (续)

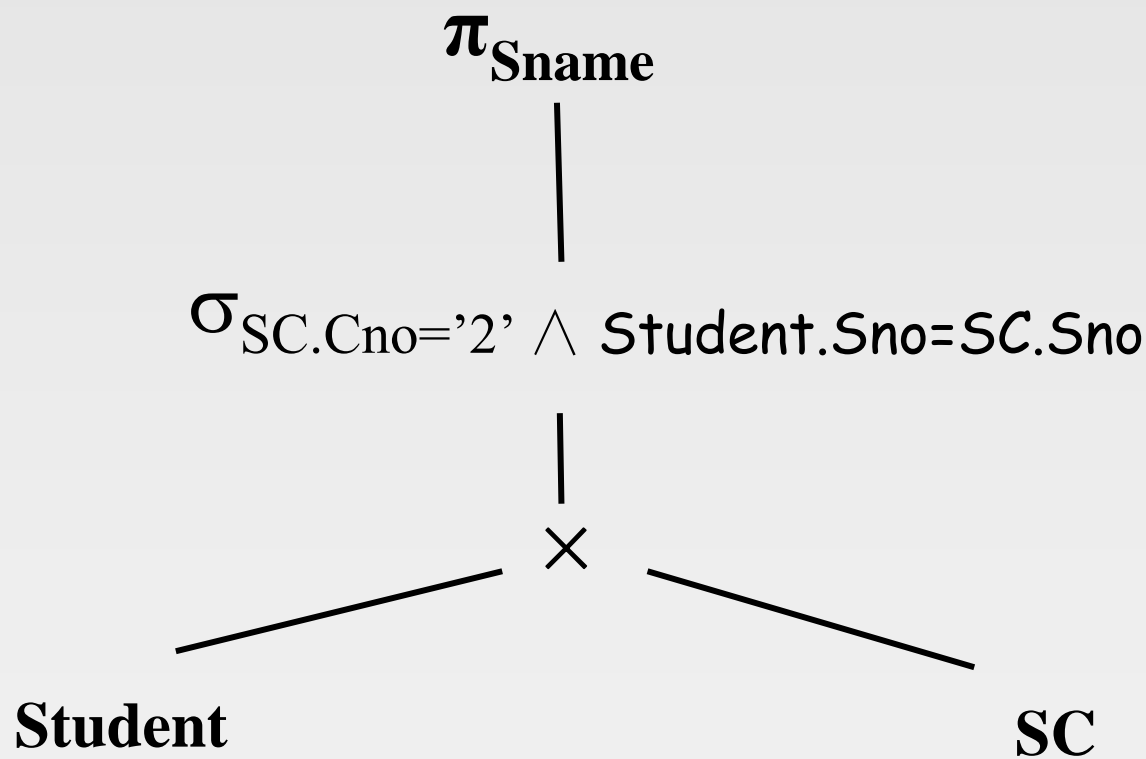
(1) 把查询转换成某种内部表示 (语法树)

建立语法树的规则:

对一个关系表达式进行语法分析, 将关系作为叶子节点, 而对关系的操作作为非叶子节点。

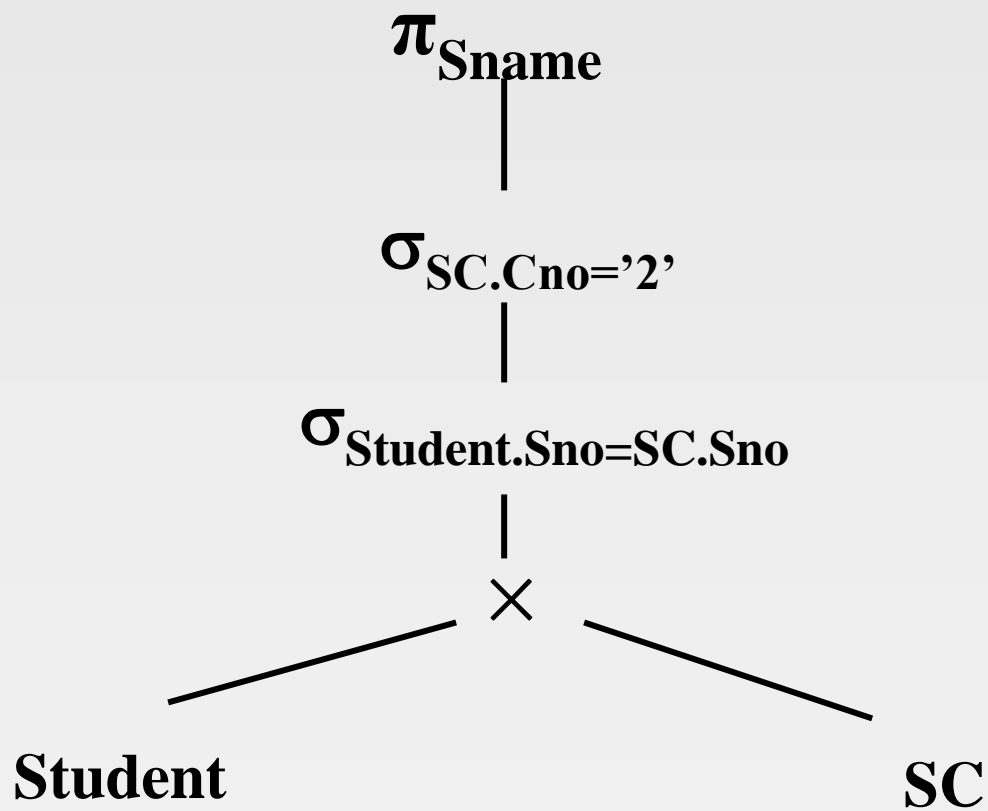
例: 求选修了课程号为2的学生姓名

(1) 把查询转换成某种内部表示

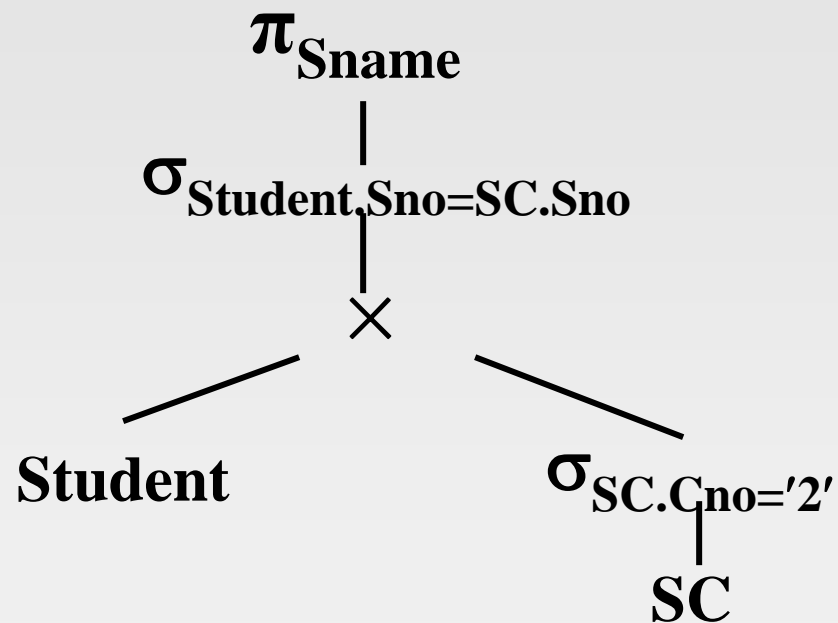


(2) 代数优化

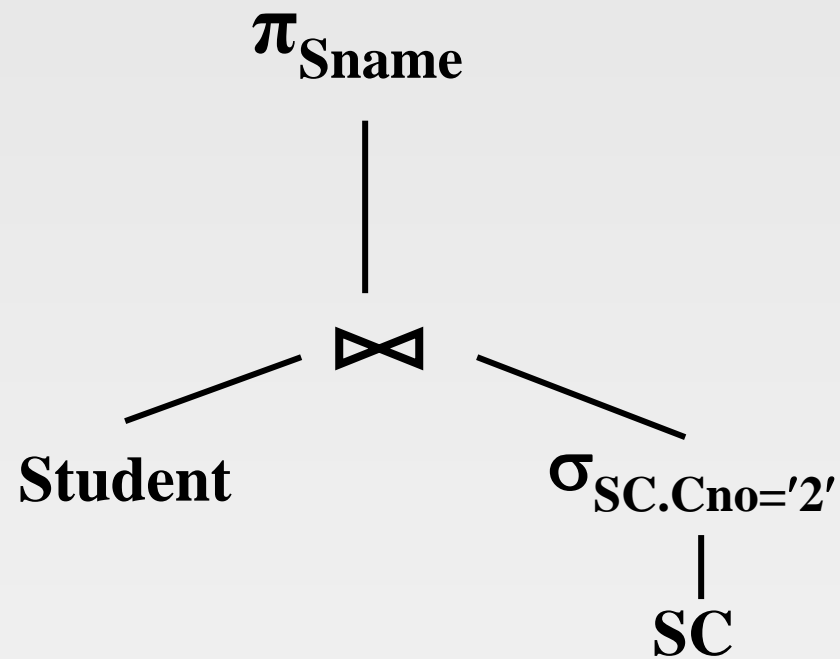
利用优化算法把语法树转换成标准（优化）形式



(2) 代数优化



(2) 代数优化



(3) 物理优化：选择低层的存取路径

所谓选择低层存取路径，指的就是要充分利数据库中已有的索引等信息。假如选择条件或连接条件所涉及的属性上有索引，那么利用该索引进行存取就可以节省很多时间，这也能提高查询的效率。

(3) 物理优化：选择低层的存取路径

- 优化器查找数据字典获得当前数据库状态信息

- 选择字段上是否有索引
- 连接的两个表是否有序
- 连接字段上是否有索引

- 然后根据一定的优化规则选择存取路径

如本例中若SC表上建有Cno的索引，则应该利用这个索引，而不必顺序扫描SC表。

(4) 生成查询计划，选择代价最小的

在作连接运算时，若两个表(设为R1, R2)均无序，连接属性上也没有索引，则可以有下面几种查询计划：

- 对两个表作排序预处理
 - 对R1在连接属性上建索引
 - 对R2在连接属性上建索引
 - 在R1, R2的连接属性上均建索引
- 对不同的查询计划计算代价，选择代价最小的一个。
- 在计算代价时主要考虑磁盘读写的I/O数，内存CPU处理时间在粗略计算时可不考虑。

第9章 关系系统及其查询优化

9.1 关系系统

9.2 关系系统的查询优化

9.3 小结

小结

■ 关系系统的查询优化

- 代数优化：关系代数表达式的优化
 - 关系代数等价变换规则
 - 关系代数表达式的优化算法
- 物理优化：存取路径和低层操作算法的选择