



第 11 章: 文件组织和索引 and Indexing

数据库系统概念,第 7 版。

©Silberschatz、Korth 和 Sudarshan
见www.db-book.com再利用条件



文件组织 File Organization

- 数据库存储为操作系统文件的集合
 - 每个文件是一个记录序列 ▪ 一个记录是一个字段序列
 - 从 O/S 的角度来看 ,每个文件都由固定长度的块组成 (也称为物理记录)
 - 块是存储分配和数据传输的单位 ▪ 一种方法
 - 假设记录大小是固定的
 - 每个文件只有一种特定类型的记录
 - 不同的文件用于不同的关系



固定长度记录

- 简单的方法：

- 记录访问很简单,但记录可能会跨块

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000



固定长度记录

■删除记录 i :不同的替代方案:

- 移动记录 $i+1, \dots, n$ 到 $i, \dots, n-1$
- 将记录 n 移动到 i
- 不移动记录,而是链接空闲列表中的所有空闲记录

记录 3 已删除

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000



固定长度记录

▪ 删除记录 i :不同的替代方案:

- 移动记录 $i+1, \dots, n$ 到 $i, \dots, n-1$
- 将记录 n 移动到 i
- 不移动记录,而是链接空闲列表中的所有空闲记录

记录 3 被删除并被记录 11 替换

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000



固定长度记录

▪ 删除记录 i :不同的替代方案:

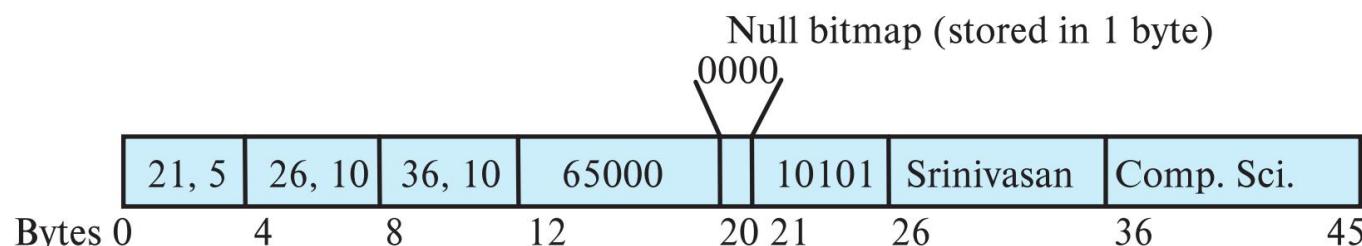
- 移动记录 $i+1, \dots, n$ 到 $i, \dots, n-1$
- 将记录 n 移动到 i
- 不移动记录,而是链接空闲列表中的所有空闲记录

header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000



可变长度记录 Variable Length Records

- 可变长度记录在数据库系统中以多种方式出现：
 - 在一个文件中存储多种记录类型
 - 允许一个或多个字段长度可变的记录类型,例如字符串(varchar)
 - 允许重复字段的记录类型 (用于非规范化数据模式)
- 属性按顺序存储
- 由固定大小 (偏移量、长度)表示的可变长度属性,具有在所有固定长度属性之后存储的实际数据
- 由空值位图表示的空值





文件中记录的组织 Records in Files

- **堆** 记录可以放在文件中任何有空间的地方
- **Sequential** – 按顺序存储记录,基于每条记录的搜索关键字
- 在一个**多表集群文件**中组织几个不同的记录关系可以存储在同一个文件中
 - 动机:将相关记录存储在同一块上以最小化 I/O
- **B+ -tree 文件组织**
 - 即使插入/删除也可以有序存储
- **散列** 根据搜索键计算的散列函数;结果指定在记录应该放在文件的哪个块



堆文件组织

- 记录可以放在文件中任何有空闲空间的地方
- 记录通常一旦分配就不会移动
- 能够有效地找到文件中的可用空间很重要
- 自由空间地图
 - 每个块 1 个条目的数组。每个条目是几位到一个字节，并且记录空闲块的一部分
 - 在下面的示例中，每块 3 位，值除以 8 表示空闲块的分数

4	2	1	4	7	3	6	5	1	2	0	1	1	0	5	6
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- 可以有二级自由空间地图
- 在下面的示例中，每个条目最多存储来自第一级自由空间映射的 4 个条目

4	7	2	6
---	---	---	---

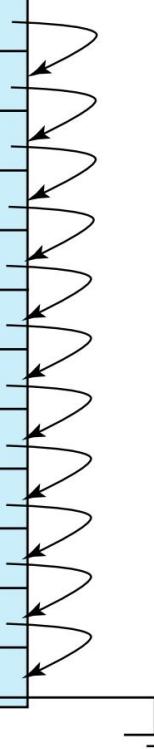
- 可用空间映射存储在一个文件中，其块被提取到内存中按要求



顺序文件组织 Sequential File Organization

- 适用于需要对整个文件进行顺序处理的应用程序
- 文件中的记录按搜索键排序

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	





顺序搜索和二分搜索

- 对于N个项目的顺序搜索,平均比较次数E(X)为 $\approx N/2$

假设每个项目成为所需项目的概率相同 (即 $1/N$) ,那么第一个项目成为所需项目的概率为 $1/N$,第二个项目成为所需项目的概率为 $1/N$,依此类推,搜索将在找到所需项目后终止:

$$\begin{aligned}
 E(X) &= \frac{1}{N} + \frac{2}{N} + \frac{3}{N} + \dots + \frac{N}{N} \\
 &= \frac{1}{N} \sum_{i=1}^N i = \frac{1}{N} \times \frac{(N+1)}{2} = \frac{(N+1)}{2N} \approx \frac{1}{2}
 \end{aligned}$$

- 对N个有序项进行二分查找,最大比较次数k $\approx \log_2 N$

考虑一条长度为N 的线,并将长度一分为二,直到只剩下个项目

• $\frac{N}{2} = 1 \Rightarrow N = 2 \Rightarrow \log_2 N$

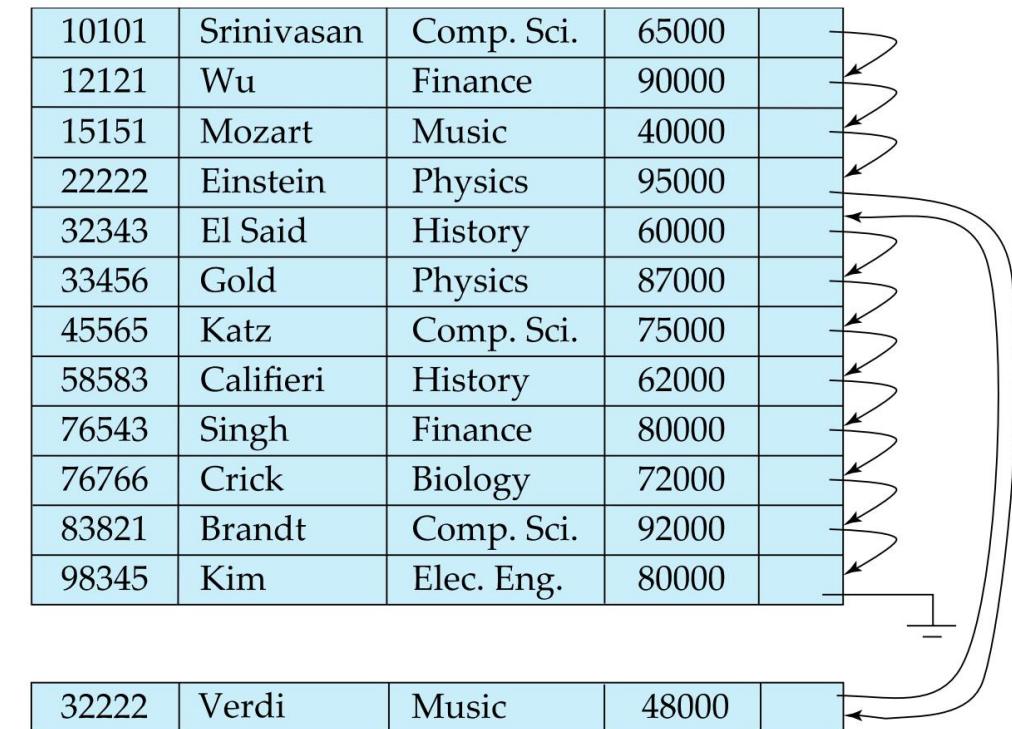


顺序文件组织

- 删除 使用指针链
- 插入 – 定位要插入记录的位置
 - 如果有空闲空间插入那里如果没有
 - 空闲空间,将记录插入溢出块
 - 无论哪种情况,都必须更新指针链

■ 需要重新组织文件

不时恢复
顺序





多表集群文件组织 Multi-table Clustering File Organization

使用多表集群文件组织将多个关系存储在一个文件中

部

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Physics	Watson	70000

讲师

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

部门和讲师的多表聚类

Comp. Sci.	Taylor	100000	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000



多表集群文件组织 Multi-table Clustering File Organization

- 适用于涉及部门讲师的查询,以及涉及单个部门及其讲师的查询
- 不适合只涉及部门的查询
- 产生可变大小的记录
- 可以添加指针链来链接特定关系的记录



分区 Partitioning

- 表分区：关系中的记录可以划分为较小的关系，分别存储
- 例如，事务关系可以划分为transaction_2018、transaction_2019等。
- 写在事务上的查询必须访问所有分区中的记录
 - 除非查询有选择，例如year=2019，在这种情况下仅需要一个分区
- 分区
 - 降低一些操作的成本，例如可用空间管理
 - 允许不同的分区存储在不同的存储设备上
 - 例如，SSD上当前年份的事务分区，旧年份的事务分区在磁盘上



数据字典存储 Dictionary Storage

数据字典 (也称为系统目录) 存储元数据; 即关于数据的数据, 例如

- 有关关系的信息
 - 关系名称
 - 每个关系的属性名称、类型和长度
 - 视图的名称和定义
 - 完整性约束
- 用户和帐户信息, 包括密码
- 统计和描述性数据

· 每个关系中的元组数

· 物理文件组织信息

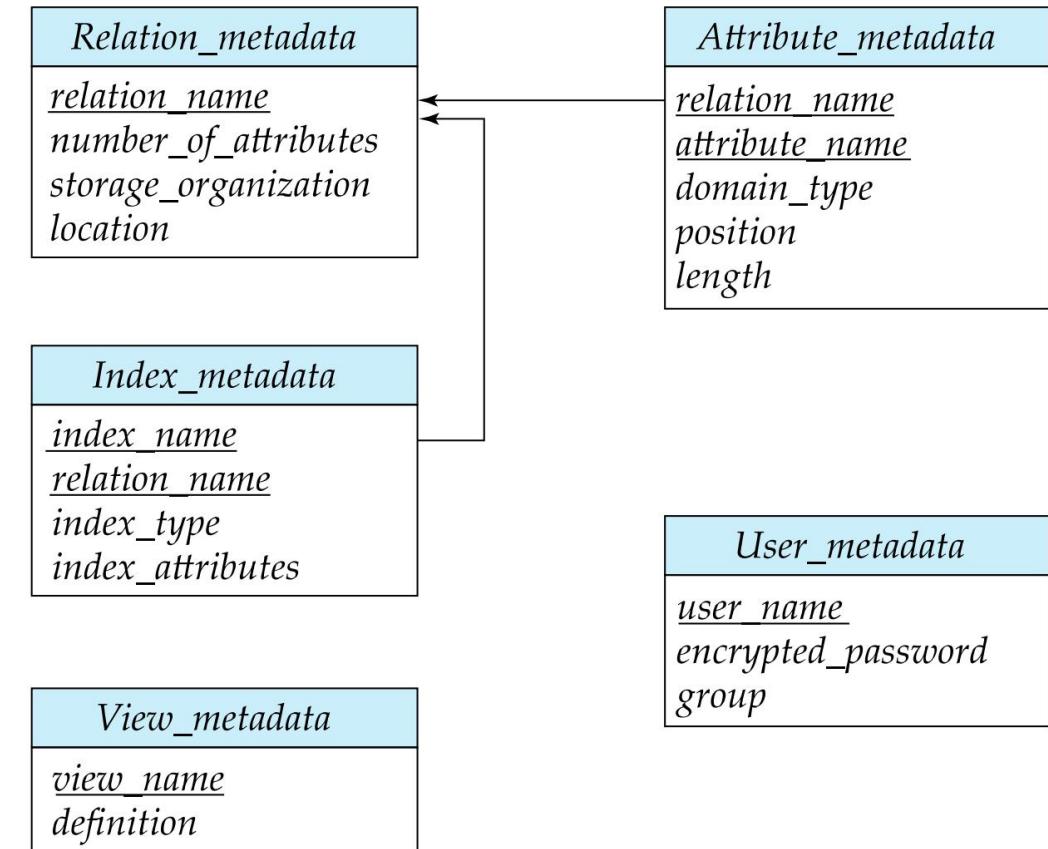
· 关系

如何存储 (顺序/散列/...) · 关系的物理位置



系统元数据的关系表示

- 磁盘上的关系表示
- 专业数据为高效访问而设计的结构，在内存中





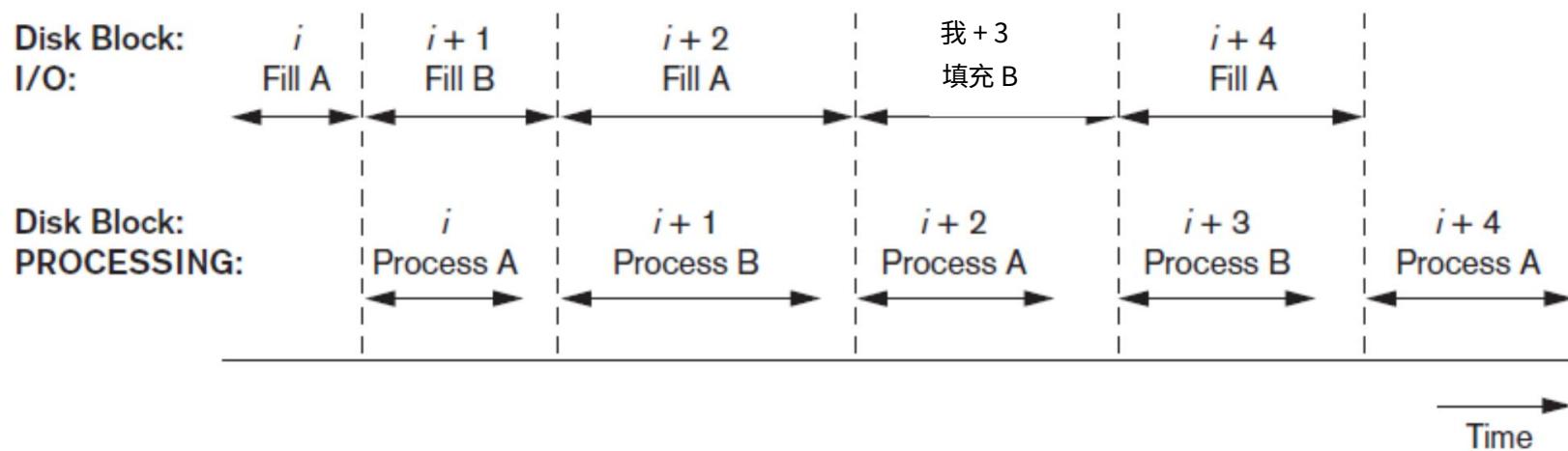
存储访问 Access

- 块是存储分配和数据传输的单位
- 数据库系统力求最小化磁盘和内存之间的块传输次数。我们可以通过在主存中保留尽可能多的块来减少磁盘访问次数
- 缓冲区 可用于存储磁盘块副本的主内存部分
- 缓冲区管理器 负责在主存中分配缓冲区空间的子系统



块缓冲块的读取

- 双缓冲可用于读取连续的块流
- 使用两个缓冲区 A 和 B 从磁盘读取
- 可以使用超过 2 个缓冲区





缓冲区管理器

程序在需要磁盘块时调用缓冲区管理器

- 如果块已经在缓冲区中,缓冲区管理器返回块在主存中的地址
- 如果块不在缓冲区中,则缓冲区管理器
 - 在缓冲区中为块分配空间
 - 如果需要,更换(扔掉)一些其他块,以使新块的空间
 - 仅在修改后才将替换块写回磁盘
自最近一次将其写入磁盘/从磁盘读取
 - 从磁盘读取块到缓冲区,并将块在主存中的地址返回给请求者。



缓冲区管理器

- 缓冲区替换策略

- 例如, LRU、FIFO、MRU

- Pinned block: 不允许写回磁盘的内存块

- 在从块读取/写入数据之前完成固定
 - 读取/写入完成时取消固定完成

- 缓冲区上的共享锁和排它锁

- 需要防止并发操作
 - 锁定规则:
 - 一次只有一个进程可以获得排他锁
 - 共享锁不能与排他锁同时持有
 - 多个进程可以同时获得共享锁



缓冲区替换策略 Replacement Policies

- 大多数操作系统会替换最近最少使用的块 (LRU 战略)
 - LRU 背后的理念 使用过去的块引用模式作为未来引用的预测器
- 查询具有明确定义的访问模式 (例如顺序扫描) , 并且数据库系统可以使用用户查询中的信息来预测未来的引用



缓冲区替换策略

- Toss-immediate策略 一旦该块的最终元组被处理,就释放该块占用的空间
- FIFO策略 先进先出
- 最近使用 (MRU) 策略- 系统必须固定块

目前正在处理中。在处理完该块的最终元组后,该块被取消固定,它成为最近使用的块
- 缓冲区管理器可以使用有关请求将引用特定关系的概率的统计信息
 - 例如,经常访问数据字典。启发式:保持主内存缓冲区中的数据字典块



面向列的存储 Column-oriented Storage

- 也称为列表表示
- 分别存储关系的每个属性
- 示例

10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000



面向列的表示

Column-oriented Representation

- 好处:

- 如果只访问某些属性,则减少 I/O
- 改进的 CPU 缓存性能 ·改进的压缩
- 现代 CPU 架构上的矢量处理,允许在数组的多个元素上并行应用 CPU 操作 (例如,将属性与常数进行比较)

- 缺点

- 从柱状表示重建元组的成本
- 元组删除和更新的成本
- 减压成本
- 在决策支持方面,列表示通常比面向行的表示更有效
- 传统的面向行的表示更适合事务处理
- 一些数据库同时支持这两种表示
·称为混合行/列存储



索引概念 Indexing Concepts

数据库系统概念,第 7 版。

©Silberschatz、Korth 和 Sudarshan
见www.db-book.com再利用条件



基本概念

- 索引机制用于加快对所需数据的访问。
 - 例如,图书馆中的作者目录
- 搜索关键字 - 一组属性的属性,用于在数据库中查找记录文件。
- 索引是由记录 (称为索引条目) 组成的文件,形式为



- 索引文件通常比原始文件小很多



指数评估指标

- 访问时间
- 插入时间
- 删除时间
- 空间开销



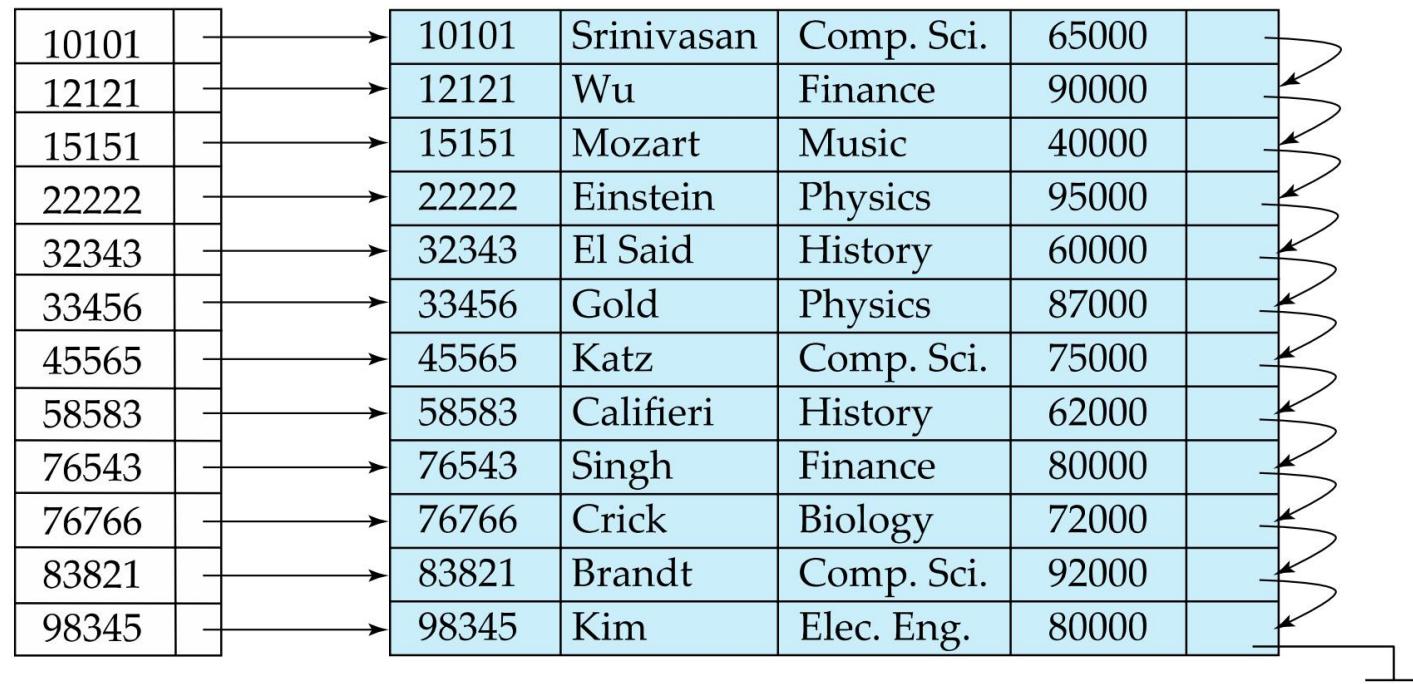
有序指数 Ordered Indices

- 在有序索引中,索引条目按搜索键排序存储
价值
- 主索引:在顺序排序的文件中,其搜索关键字指定文件顺序的索引
 - 主索引的搜索键通常但不一定是首要的关键
- 二级索引:搜索关键字指定顺序的索引
不同于文件的顺序
 - 键值可能是唯一的,也可能不是唯一的
- 索引顺序文件 (ISAM – 索引顺序访问方法) :按搜索键排序的顺序文件,在搜索键上有索引



密集索引文件 Index Files

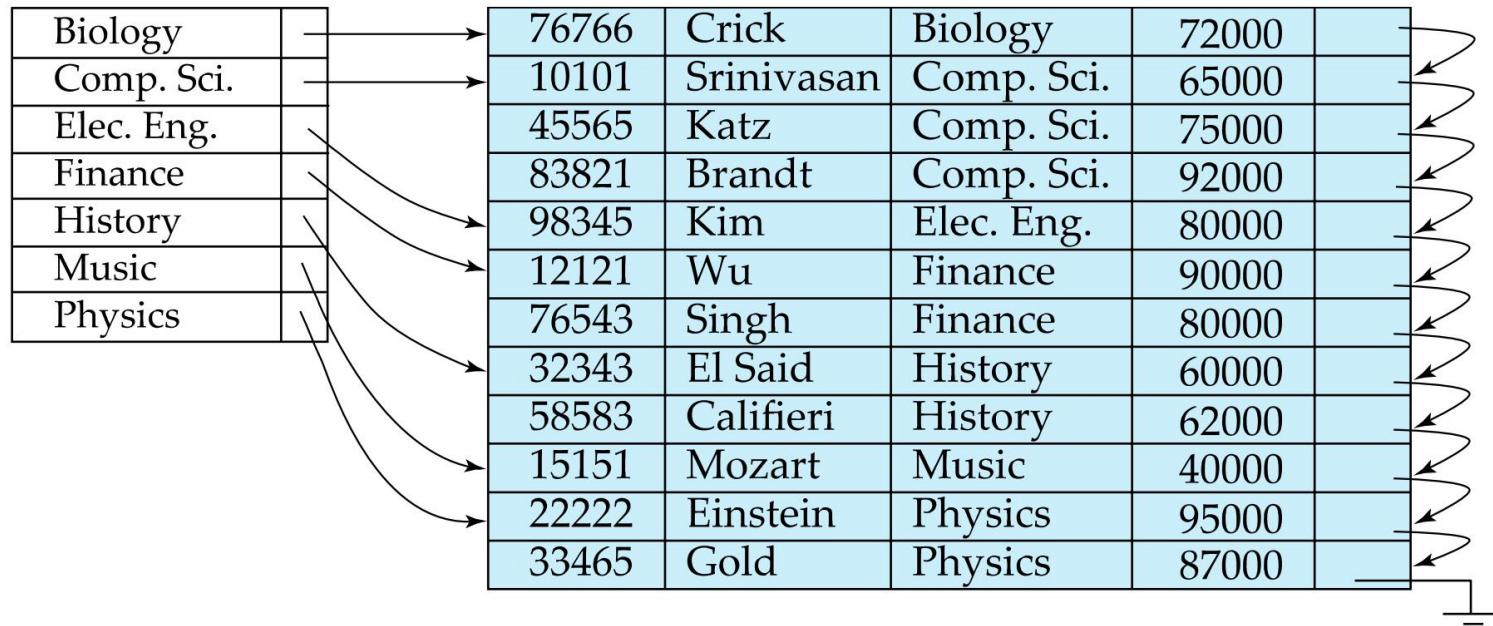
- **密集索引** 索引记录出现在每个搜索键值的文件
- 例如讲师关系ID属性的索引





密集索引文件 Index Files

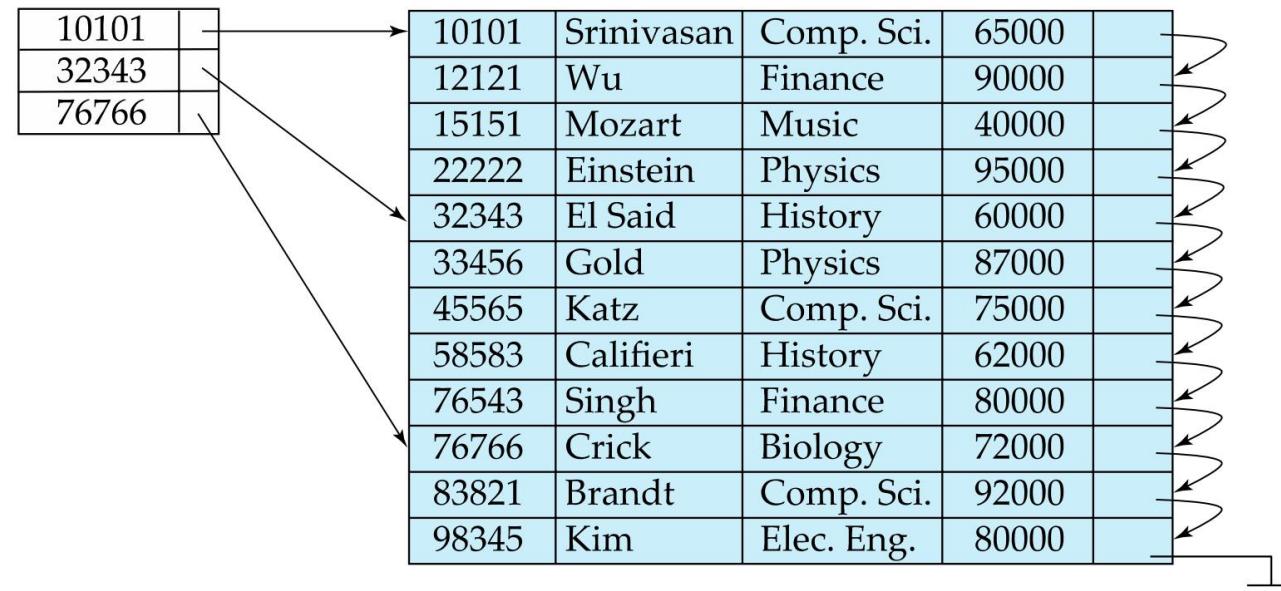
- dept_name上的密集索引,讲师文件按dept_name排序





稀疏索引文件

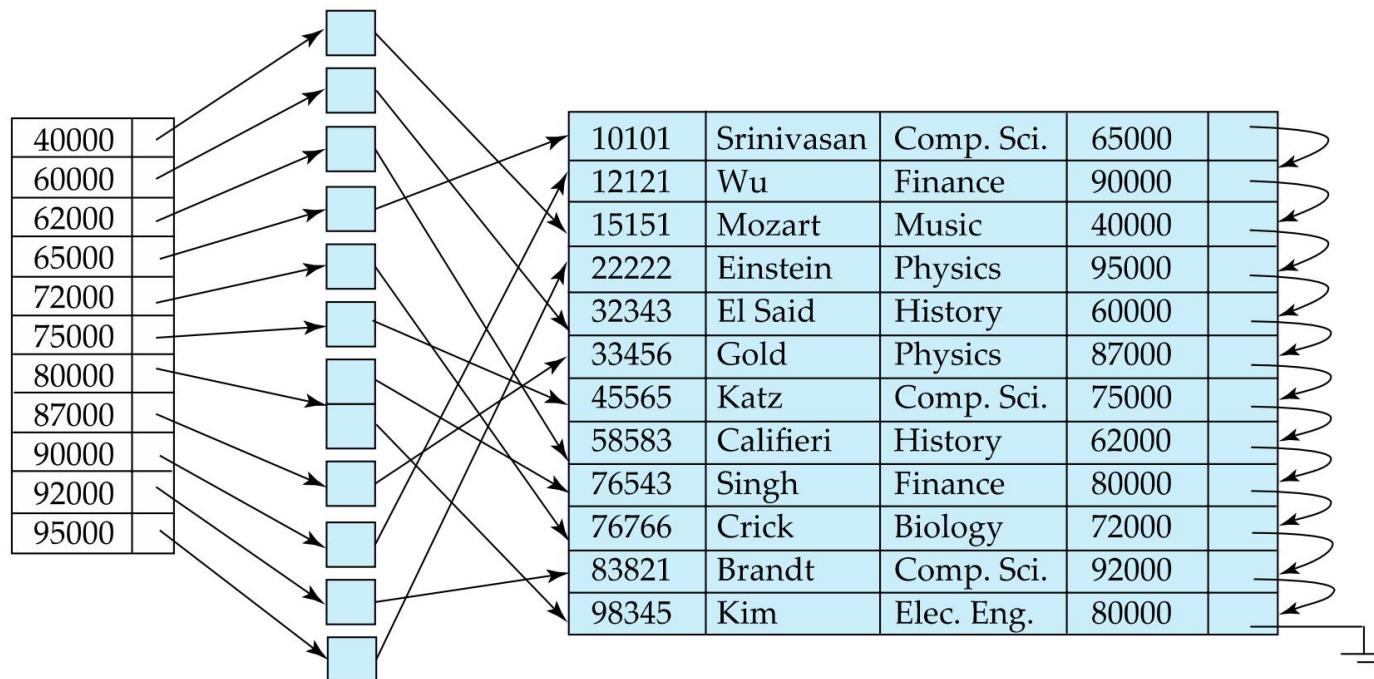
- **稀疏索引**: 仅包含一些搜索键的索引记录
价值观。
 - 适用于记录按搜索键顺序排序的情况
- 要查找具有搜索键值K的记录, 我们:
 - 查找最大搜索键值 $\leq K$ 的索引记录
 - 从索引记录指向的记录开始按顺序搜索文件





Secondary Index

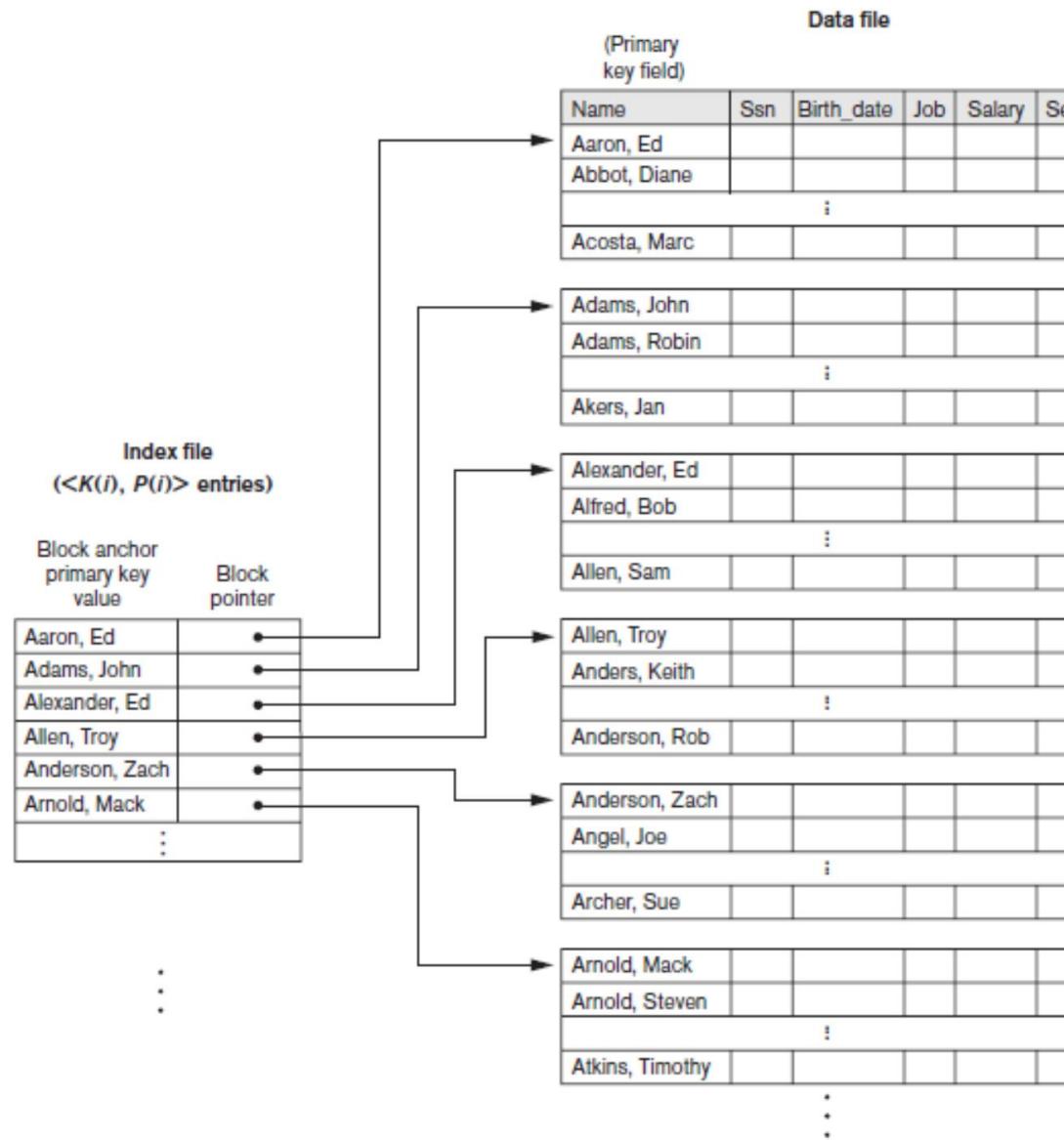
▪ 教师工资字段二级索引



- 索引记录指向一个存储桶，该存储桶包含指向具有该特定搜索键值的所有实际记录的指针。



一级索引 Index



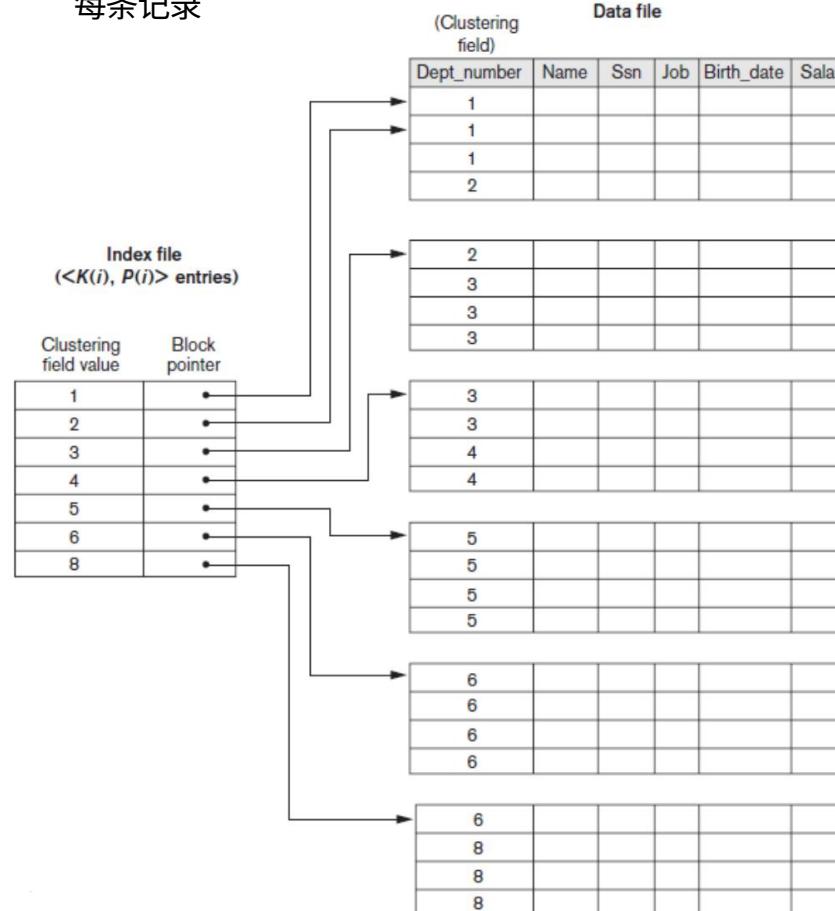


聚类索引 Clustering Indexes

■聚类字段

- 记录在非关键字段上物理排序,没有不同的值

每条记录

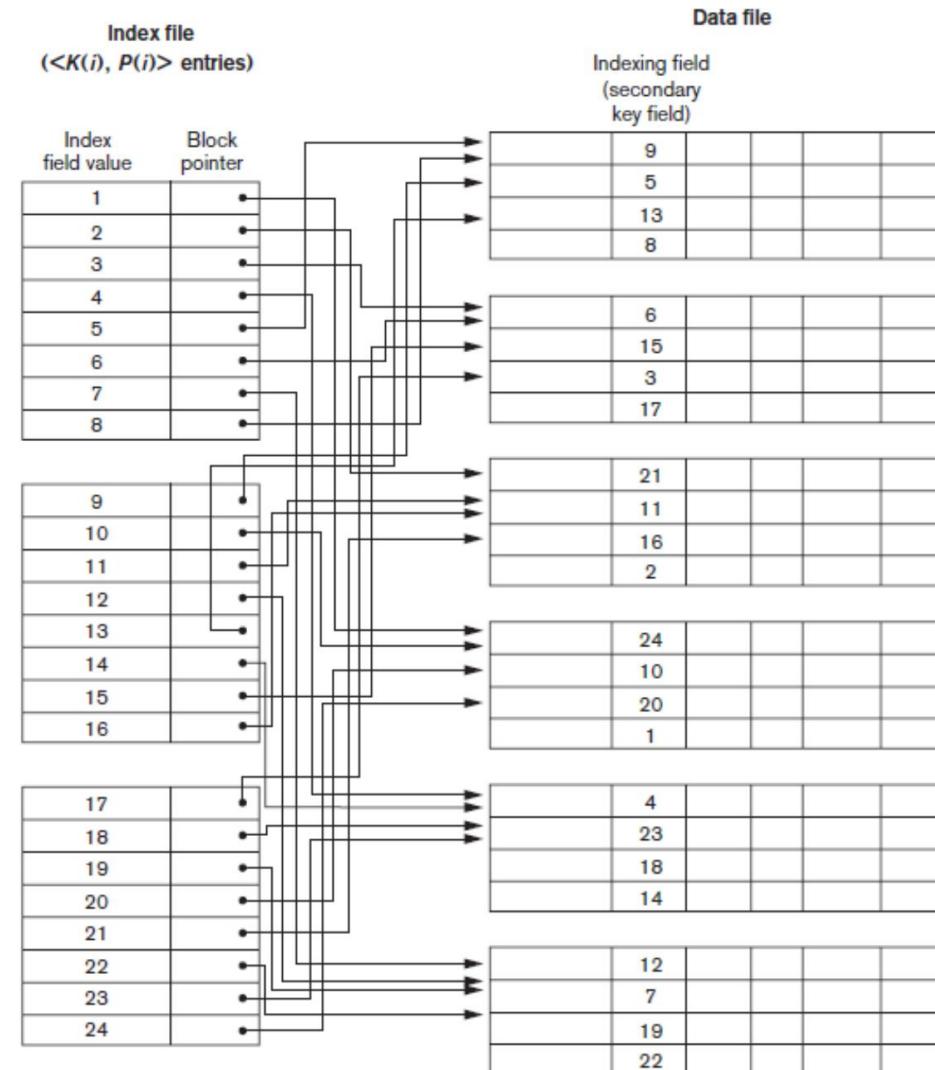


上的聚类索引

Dept_number 是 Employee 文件的排序
非键字段



二级索引 Secondary Index



文件的非排序关键字段上的密集二级索引（带有块指针）。



多级索引 Multi-level Index

- 如果索引不适合内存,访问变得昂贵
- 解决方案:将磁盘上保存的索引视为一个顺序文件,并在其上构造一个稀疏索引
 - 外部索引 基本索引的稀疏索引
 - 内部索引 基本索引文件
- 如果外部索引太大而无法放入主内存,则可以创建另一个级别的索引,依此类推
- 在从文件中插入或删除时,必须更新所有级别的索引



两级 ISAM (索引顺序访问方法)组织

