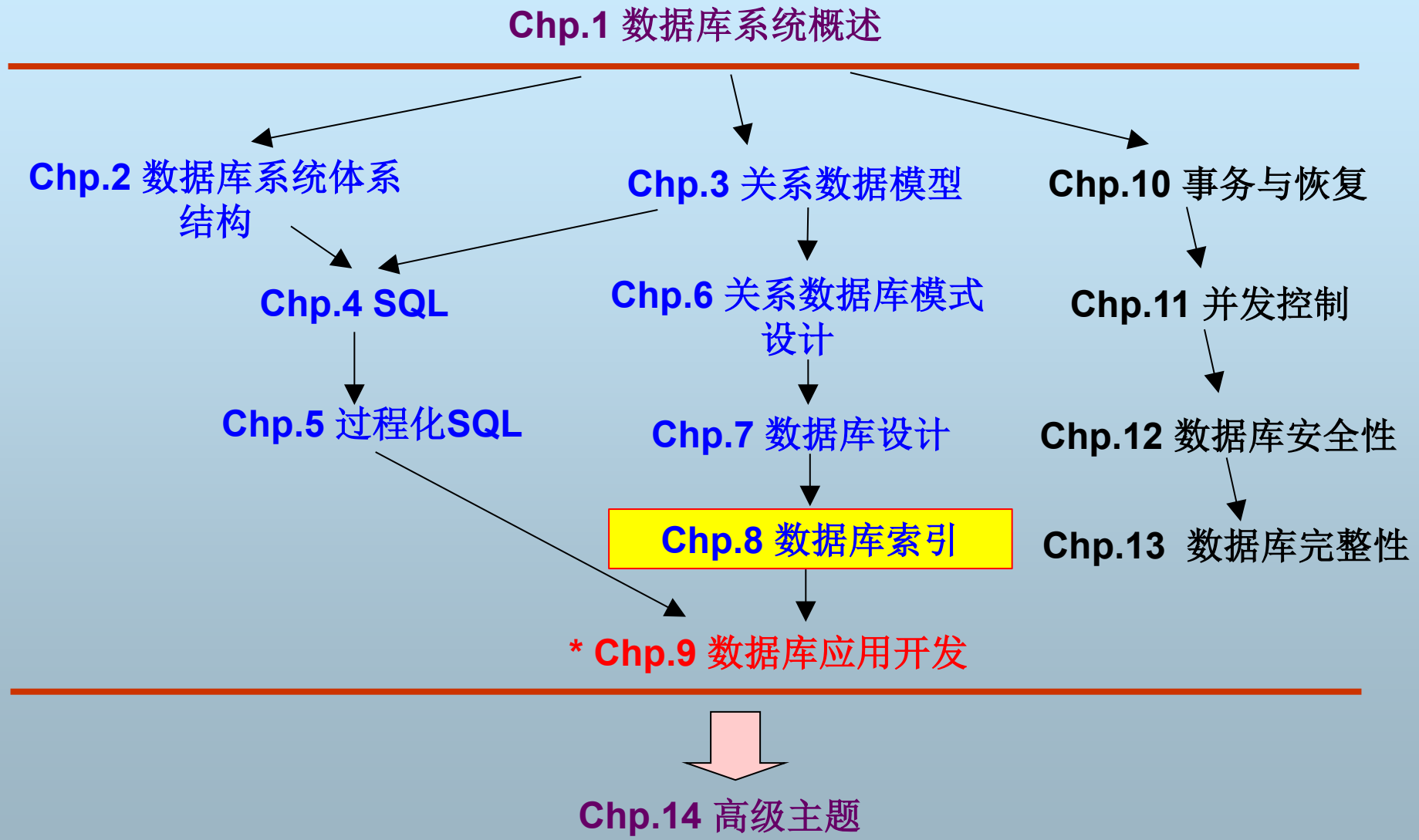
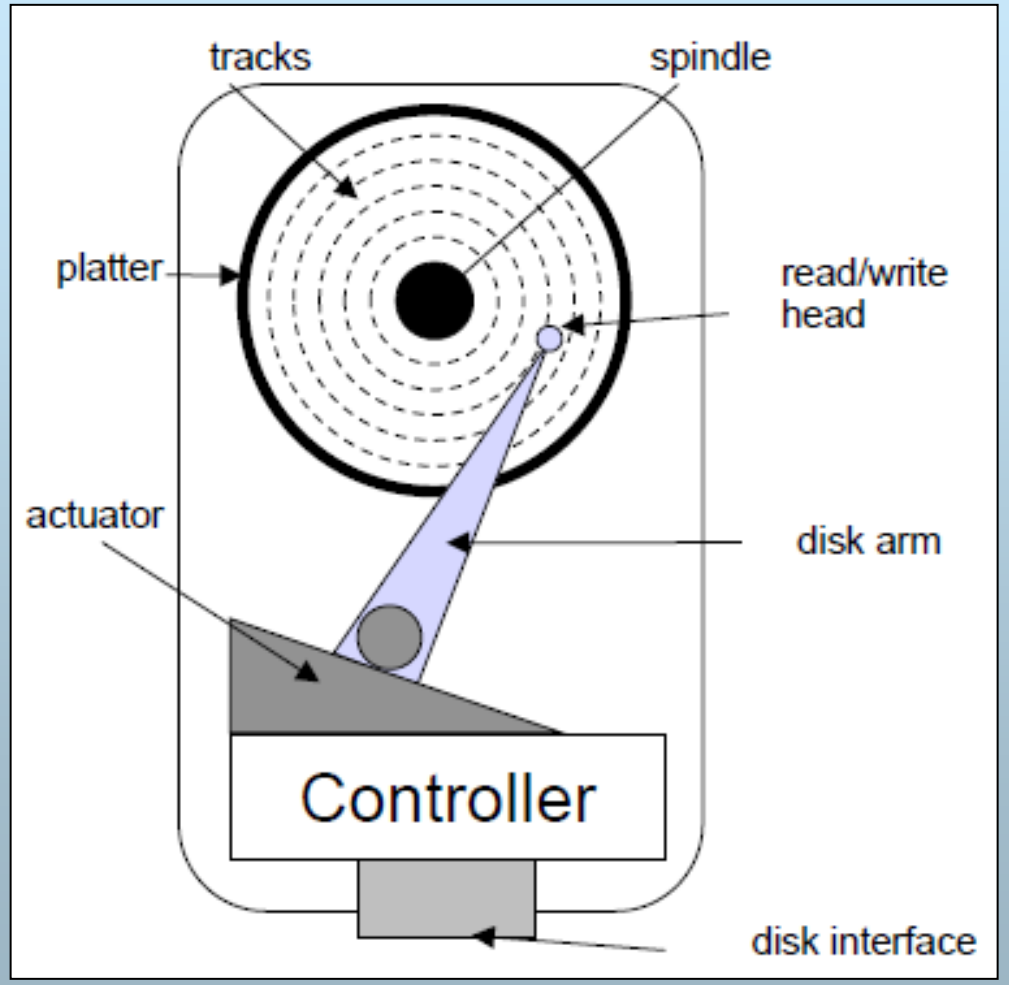
第8章 数据库索引

金培权（jpq@ustc.edu.cn）

课程知识结构


Chp.1 数据库系统概述
Chp.2 数据库系统体系结构
Chp.3 关系数据模型
Chp.10 事务与恢复
Chp.4 SQL
Chp.6 关系数据库模式设计
Chp.11 并发控制
Chp.5 过程化SQL
Chp.7 数据库设计
Chp.12 数据库安全性
Chp.8 数据库索引
Chp.13 数据库完整性
* Chp.9 数据库应用开发
Chp.14 高级主题

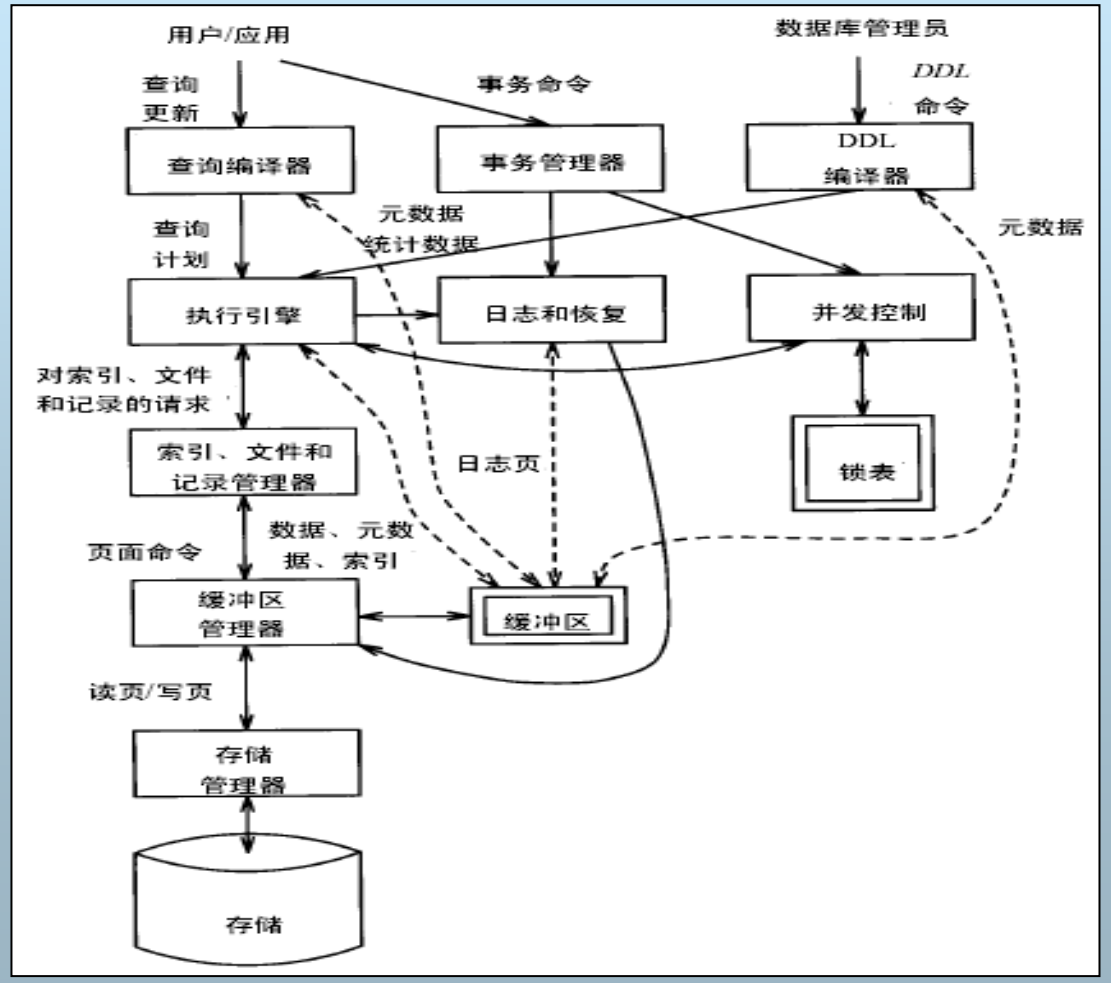
为什么DBMS底层需要按页存取？



- 一次磁盘访问的延迟包括：
  - 寻道时间 (10-40ms)
  - 旋转延迟 (~4ms，7200RPM)
  - 传输时间 (<1ms)
  - 其它延迟 (忽略不计)
- 磁盘最小物理存取单位
  - 扇区（512B）
- DBMS和OS的逻辑存取单位
  - 页（磁盘块）——若干连续扇区
  - 16KB——MySQL
    8KB——MS SQL Server
  - 减少数据存取时的寻道时间

数据库为什么需要索引？

- 没有索引，数据查询效率低

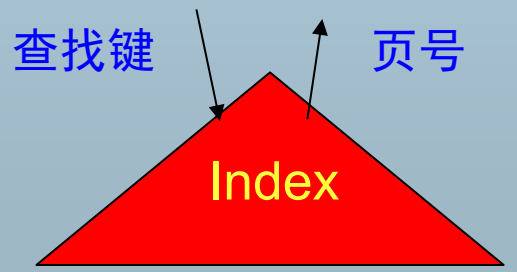


若 page size = 8 KB，page I/O 10ms

1 MB (128 pages)：1.28 s
128 MB (16384 pages)：163.8 s
1 GB (131072 pages)：1310.7 s ≈21.8 min

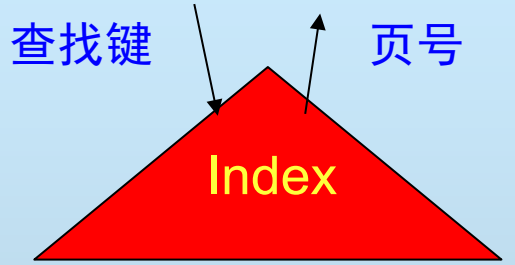
索引的动机：

提高按查找键（Search Key）查找的性能，将记录请求快速定位到页地址

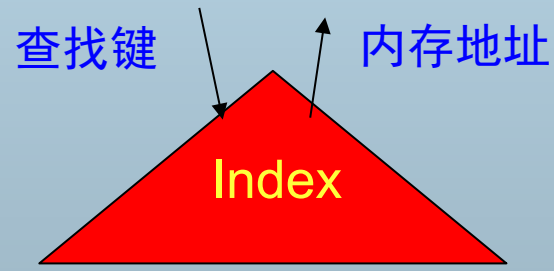


外存索引 vs. 内存索引

- 外存索引
  - 数据在外存，e.g. SSD or HDD
  - 目标是减少I/O代价
  - 数据库索引通常指外存索引



- 内存索引
  - 数据在内存，e.g. DRAM or NVM
  - 目标是减少内存cacheline访问次数

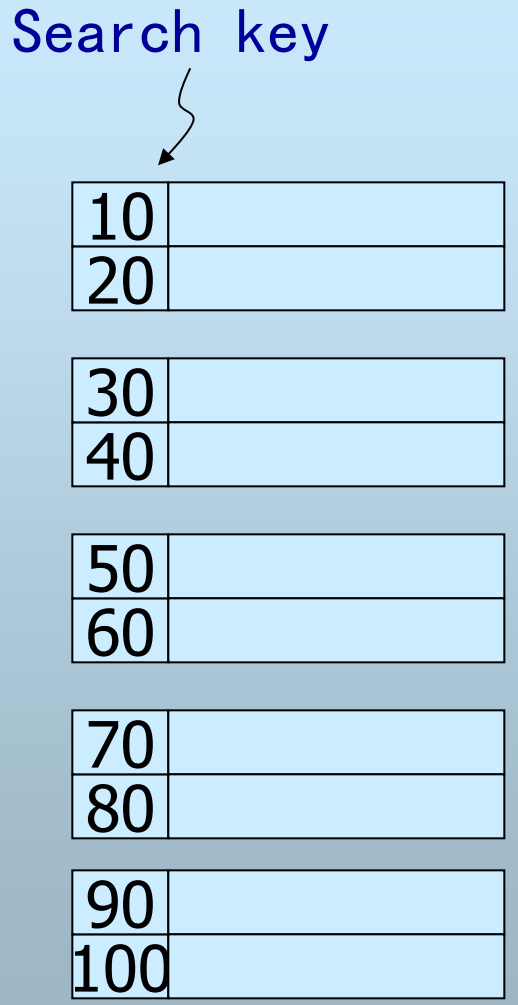


主要内容

- 顺序文件上的索引
- 辅助索引
- B+树
- 散列表 (Hash Tables)

一、顺序文件上的索引

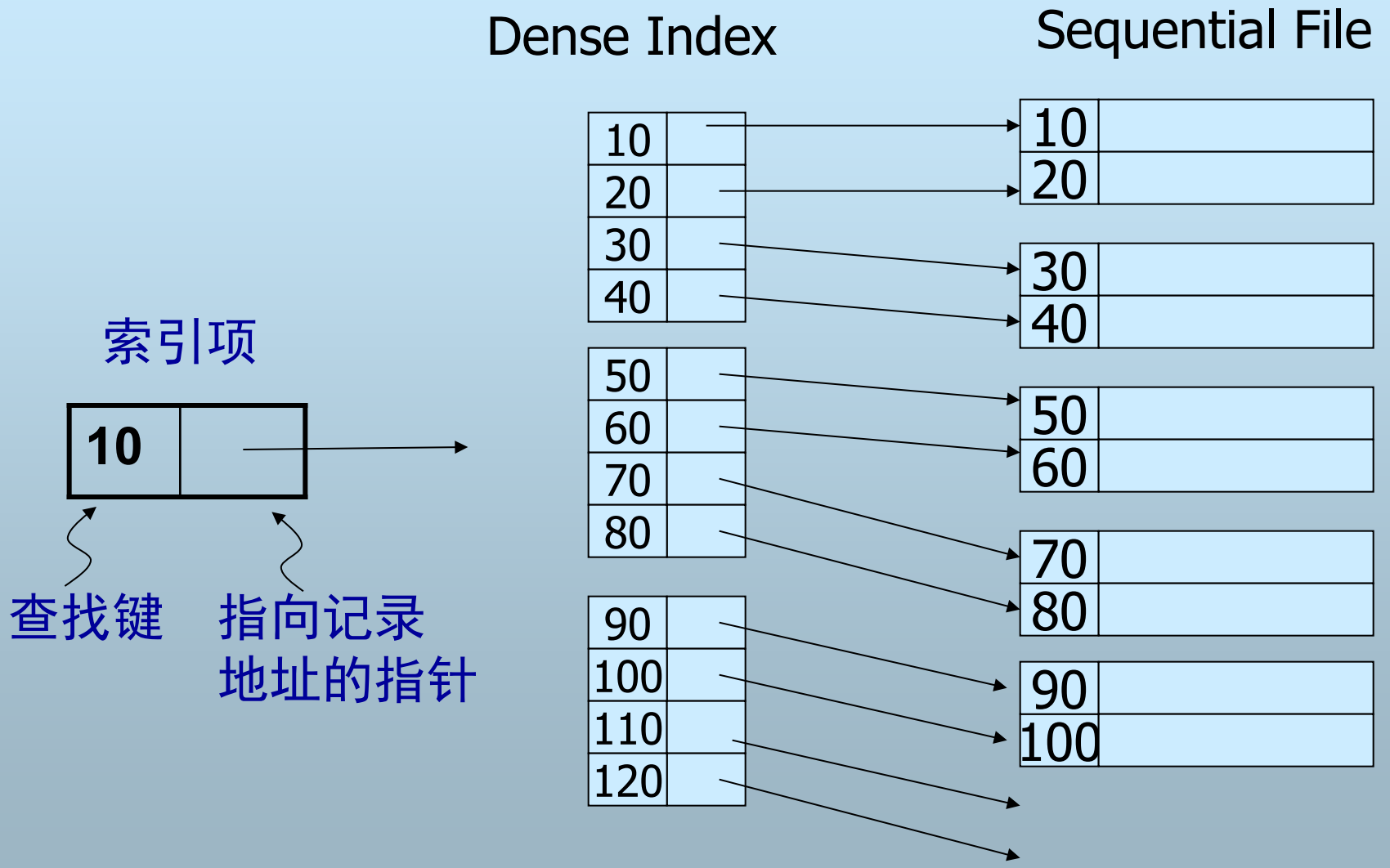
- 顺序文件
  - 记录按查找键排序



1、密集索引(Dense Index)

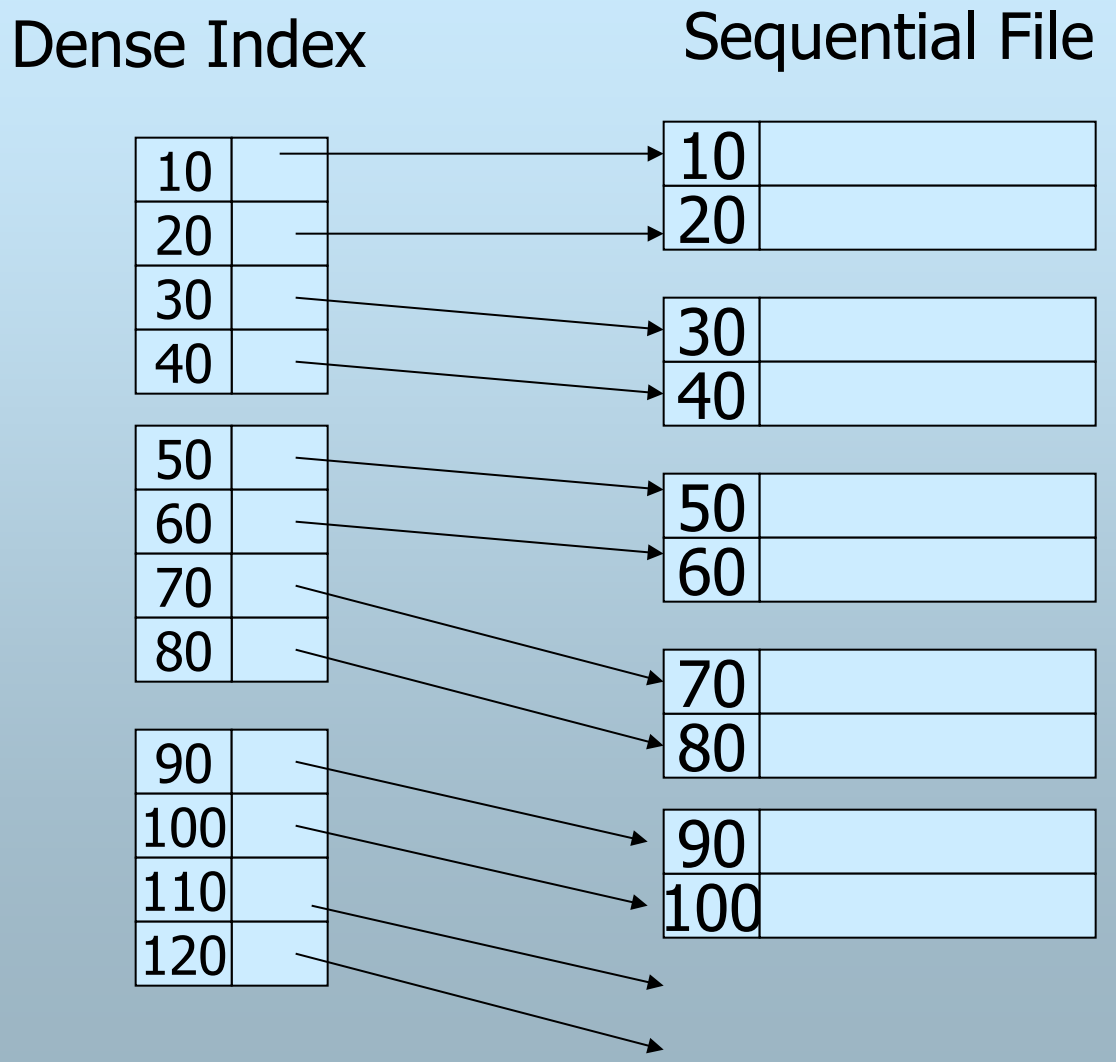
- 每个记录都有一个索引项
- 索引项按查找键排序

1、密集索引(Dense Index)



1、密集索引(Dense Index)

查找：
查找索引项，
跟踪指针即可



1、密集索引(Dense Index)

- 为什么使用密集索引？
  - 记录通常比索引项要大
  - 索引可以常驻内存
  - 要查找键值为K的记录是否存在，不需要访问磁盘数据块

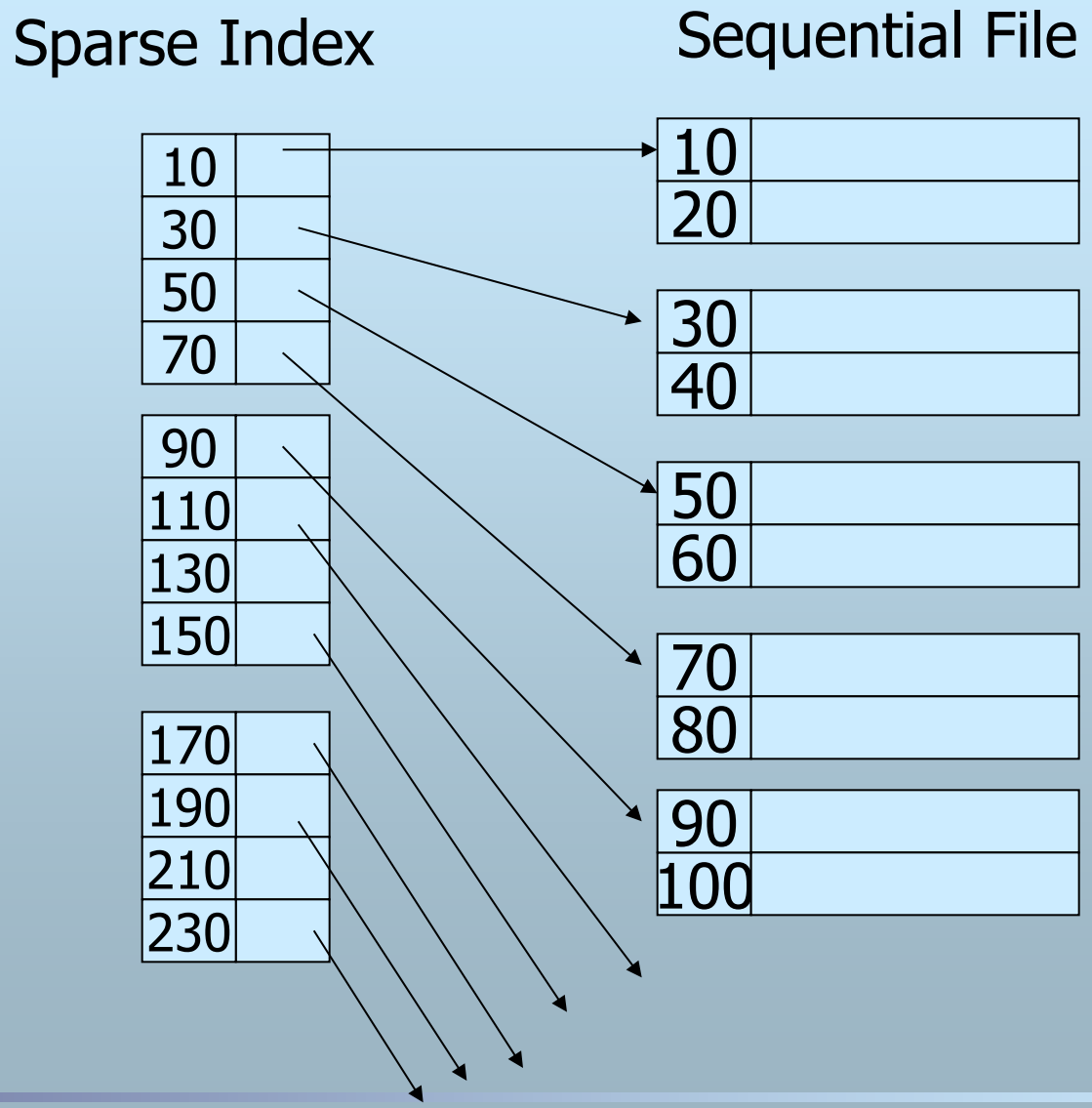
- 密集索引缺点？
  - 索引占用太多空间

用稀疏索引改进

2、稀疏索引(Sparse Index)

- 仅部分记录有索引项
- 一般情况：为每个数据块的第一个记录建立索引

2、稀疏索引(Sparse Index)



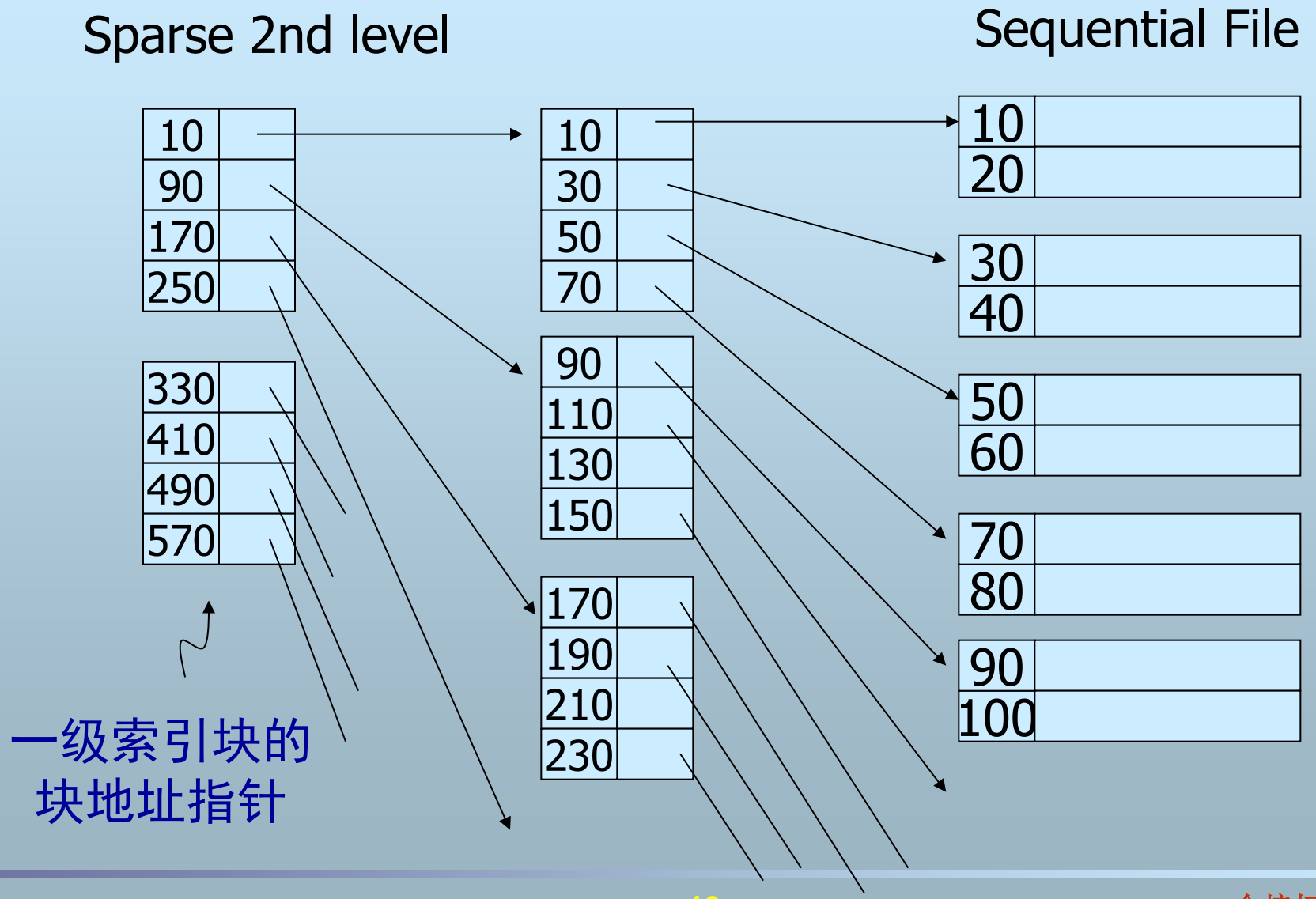
2、稀疏索引(Sparse Index)

- 有何优点？
  - 节省了索引空间
  - 对同样的记录，稀疏索引可以使用更少的索引项
- 有何缺点？
  - 对于“是否存在键值为K的记录？”，需要访问磁盘数据块

3、多级索引(Multi-level Index)

- 索引上再建索引
  - 二级索引、三级索引……

3、多级索引(Multi-level Index)



3、多级索引(Multi-level Index)

- 多级索引的好处？
  - 一级索引可能还太大而不能常驻内存
  - 二级索引更小，可以常驻内存
  - 减少磁盘I/O次数

3、多级索引(Multi-level Index)

例：一块＝4KB。一级索引10,000个块，每个块可存100个索引项，共40MB。二级稀疏索引100个块，共400KB。

按一级索引查找(二分查找)：平均 lg10000≈13 次I/O定位索引块，加一次数据块I/O，共约14次I/O

按二级索引查找：定位二级索引块0次I/O，读入一级索引块1次I/O，读入数据块1次I/O，共2次I/O

3、多级索引(Multi-level Index)

- 当一级索引过大而二级索引可常驻内存时有效
- 二级索引仅可用稀疏索引
  - 思考：二级密集索引有用吗？
- 一般不考虑三级以上索引
  - 维护多级索引结构
  - 有更好的索引结构——B^+树

二、辅助索引

- 主索引（Primary Index）
  - 顺序文件上的索引
  - 记录按索引属性值有序
  - 根据索引值可以确定记录的位置
- 辅助索引（Secondary Index）
  - 数据文件不需要按查找键有序
  - 根据索引值不能确定记录在文件中的顺序

1、辅助索引概念

MovieStar(name char(10) PRIMARY KEY, address char(20))

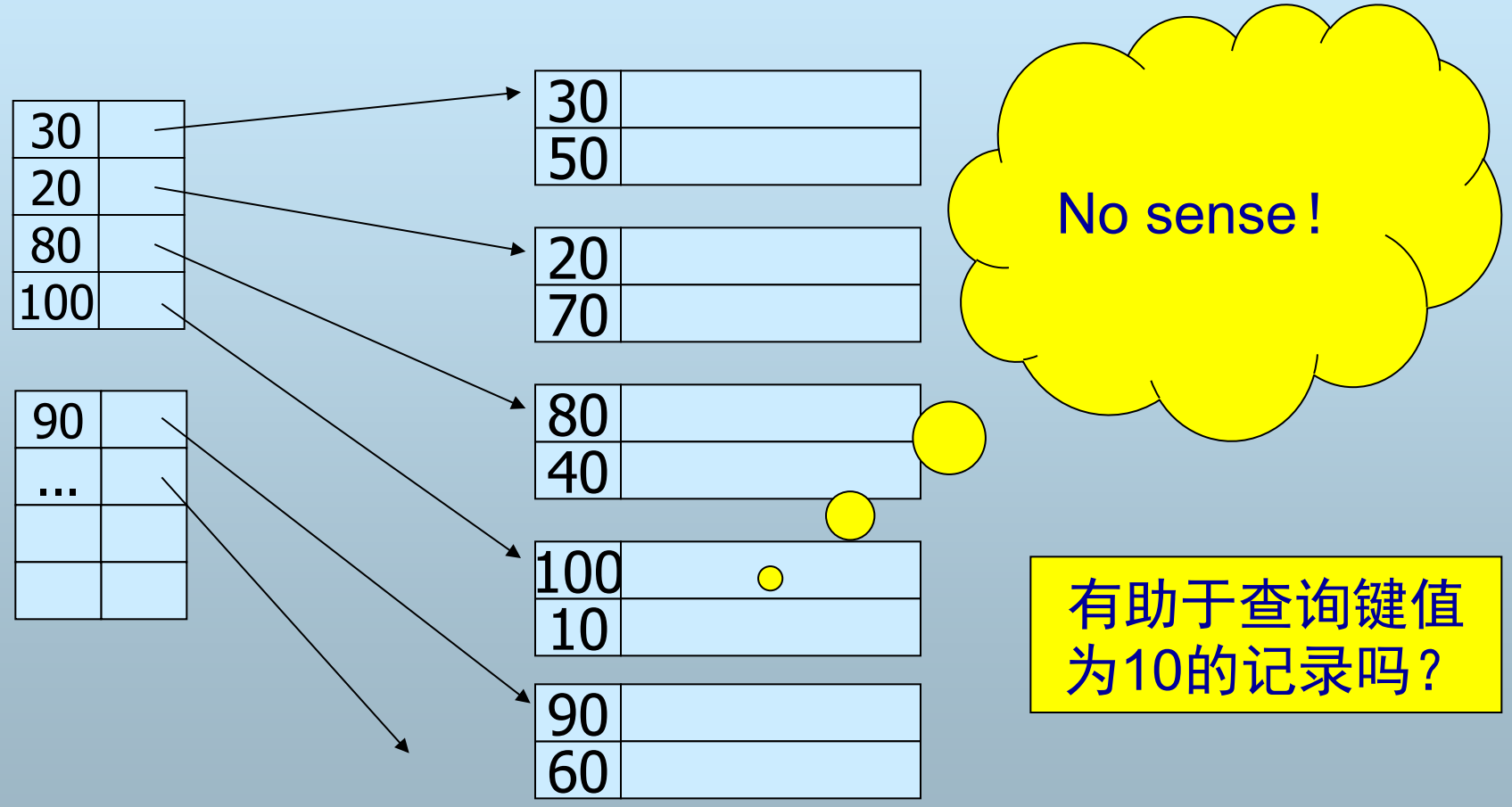
- **Name上创建了主索引，记录按name有序**
- **Address上创建辅助索引**

Create Index adIndex On MovieStar(address)

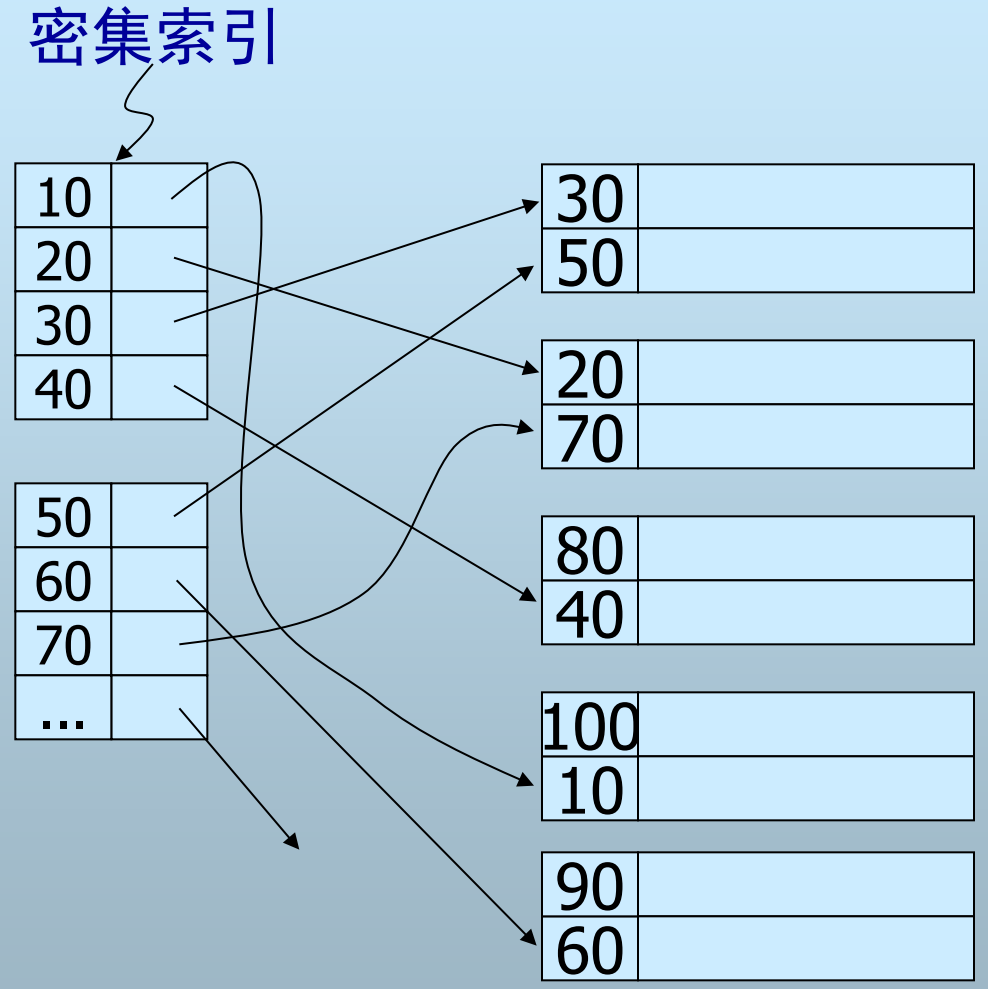
1、辅助索引概念

- 辅助索引只能是密集索引
  - 稀疏的辅助索引有意义吗？

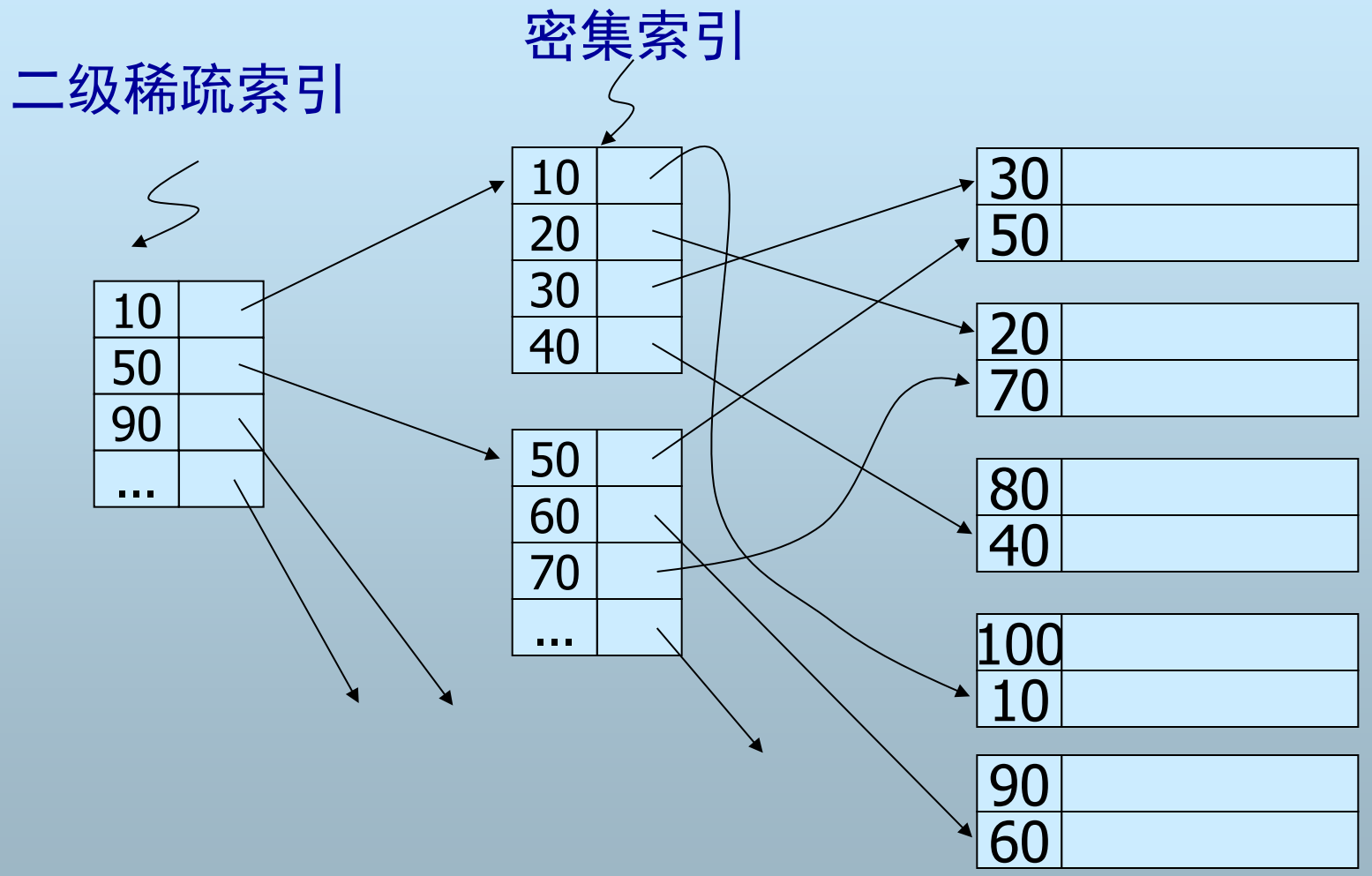
1、辅助索引概念


30
20
80
100
90
...
30
50
20
70
80
40
100
10
90
60
No sense!
有助于查询键值为10的记录吗？

2、辅助索引设计



2、辅助索引设计


二级稀疏索引
密集索引
10
50
90
...
10
20
30
40
50
60
70
...
30
50
20
70
80
40
100
10
90
60

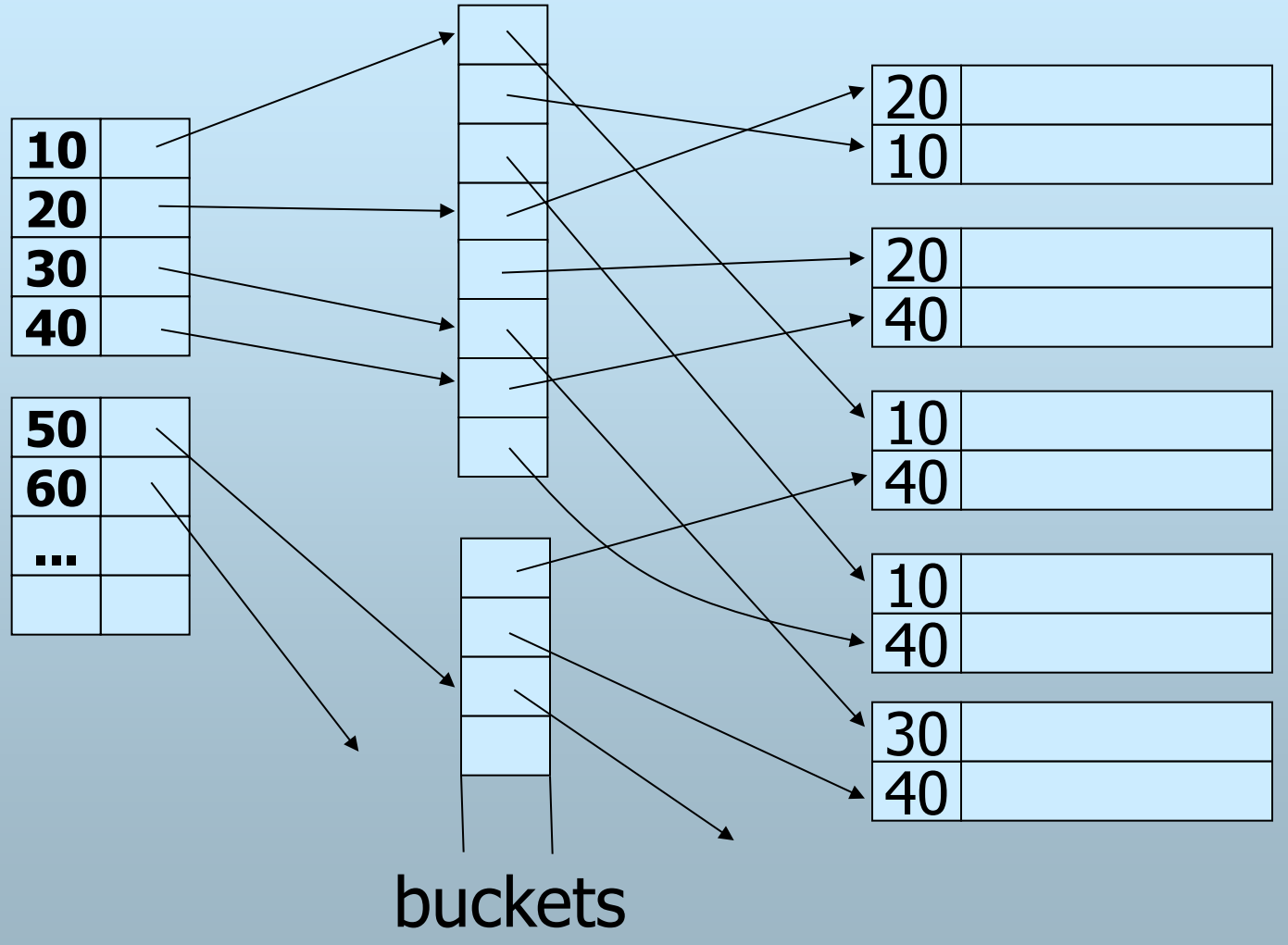
问 题

- 重复键值怎么处理？

3、辅助索引中的间接桶

- Indirect Bucket
- 重复键值
  - 采用密集索引浪费空间
- 间接桶
  - 介于辅助索引和数据文件之间

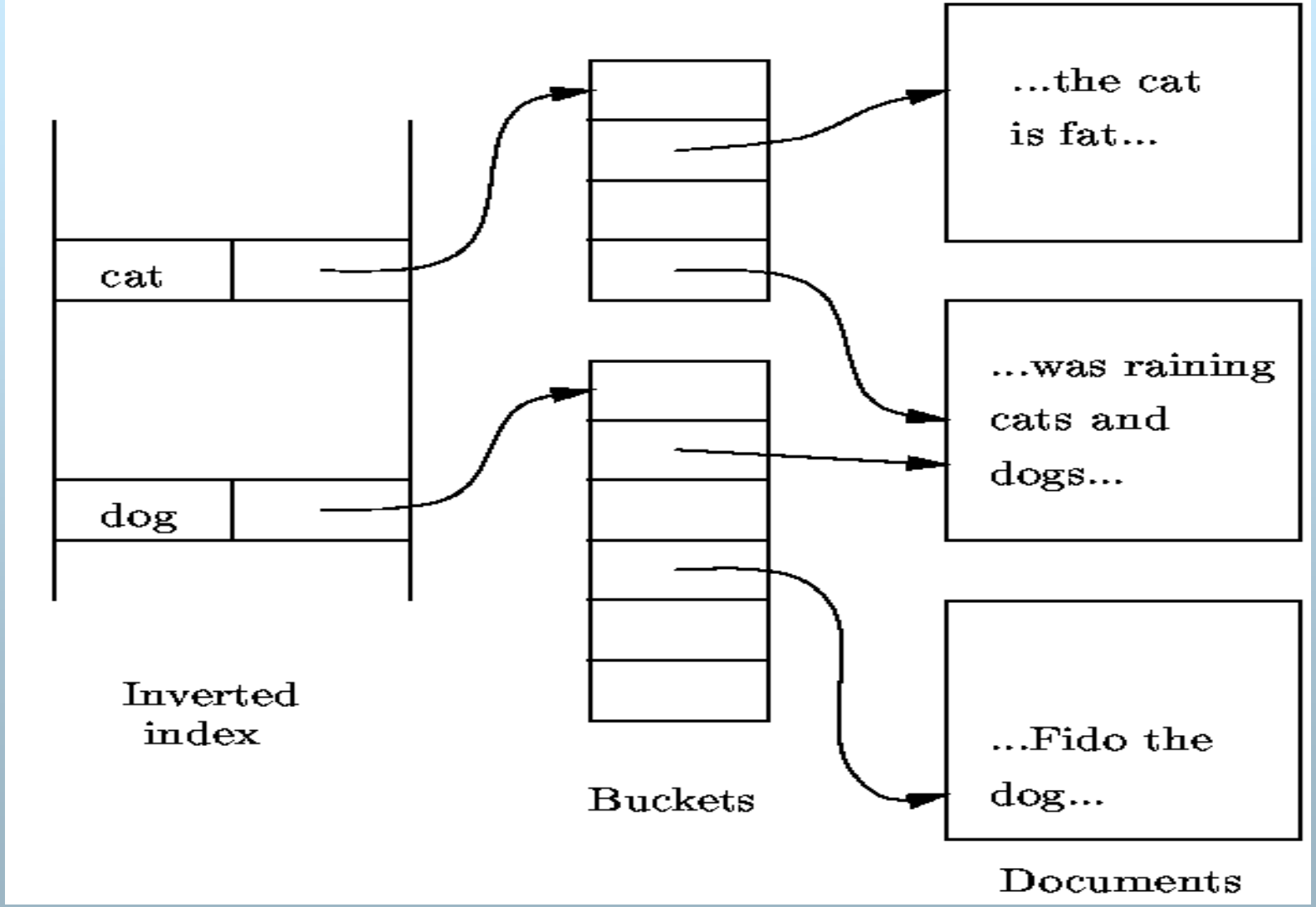
3、辅助索引中的间接桶



4、倒排索引(Inverted Index)

- 应用于文档检索，与辅助索引思想类似
- 不同之处
  - 记录→文档
  - 记录查找→ 文档检索
  - 查找键→ 文档中的词
- 思想
  - 为每个检索词建立间接桶
  - 桶的指针指向检索词所出现的文档

4、倒排索引(Inverted Index)



4、倒排索引(Inverted Index)