Database Management System

Lecture 7

Index - Hashing

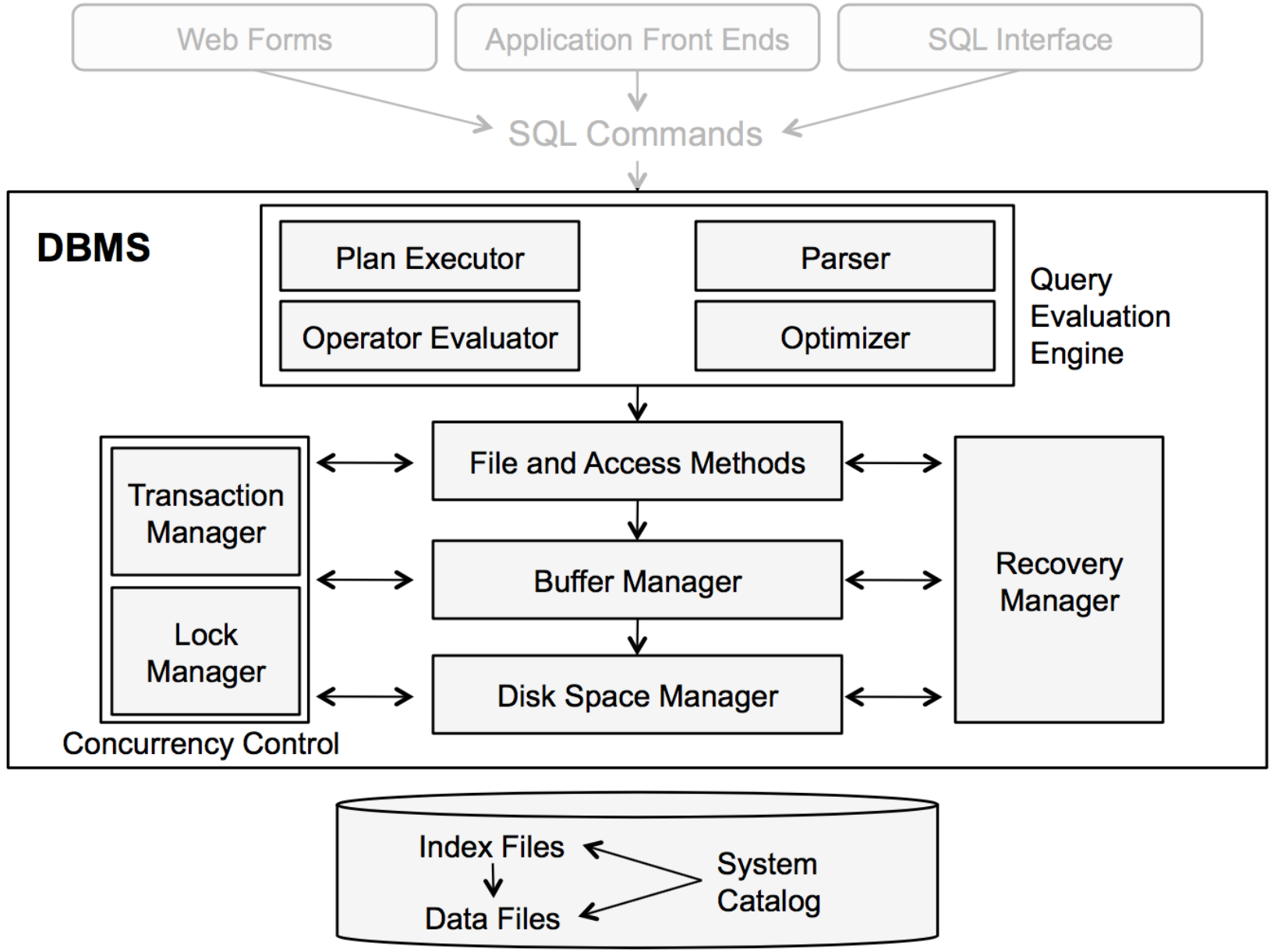
* Some materials adapted from R. Ramakrishnan, J. Gehrke and Shawn Bowers

Today's Agenda

- Index
  - Hash

Hash Indexex (Hashing)

Basic Database Architecture



Hash Indexes (Hashing)

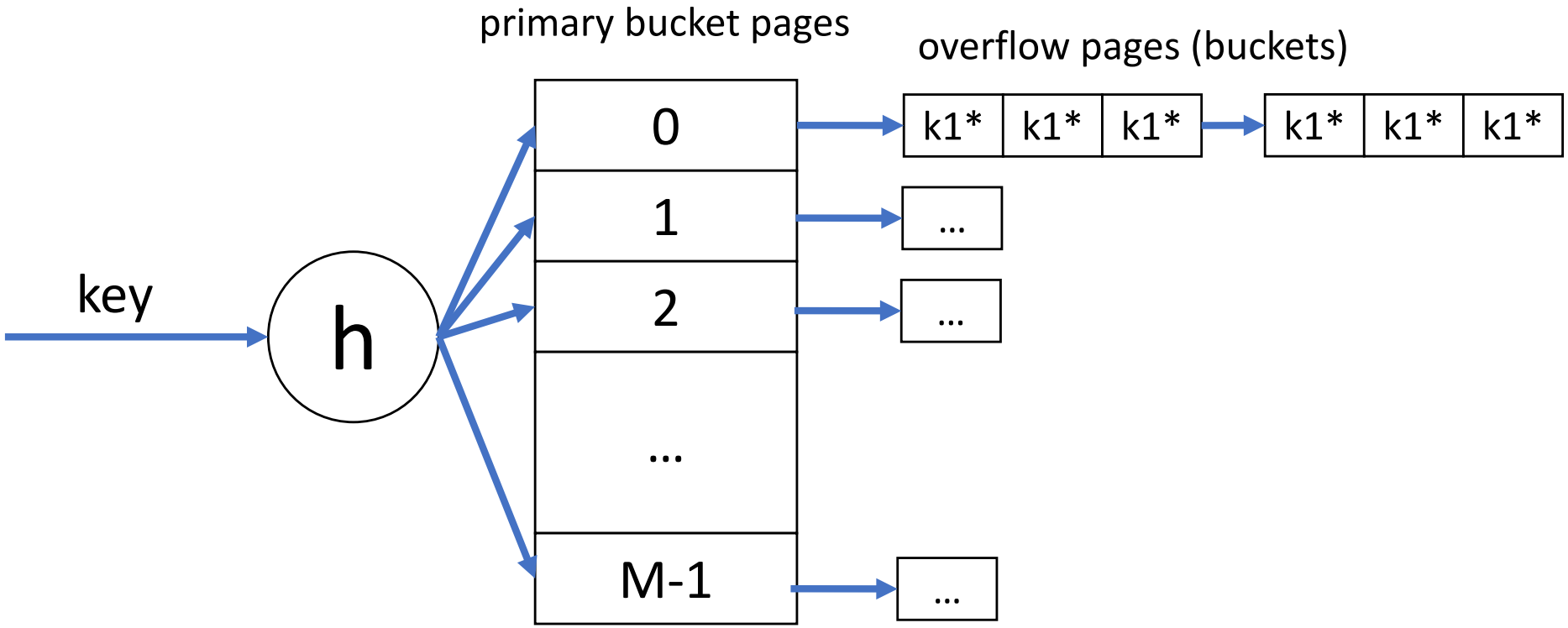
- Again, 3 alternatives for data entry *k**
  - Data record with key value *k*
  - <*k*, RID of record having k>
  - <*k*, list of RIDs of records having k>
- Hash-based indexes are best for equality selections
  - Cannot support range searches
- Static and dynamic hashing techniques
  - Similar trade-offs for ISAM vs. B+ trees

Static Hashing

- Number of primary pages fixed
  - Allocated sequentially
  - **Never de-allocated**
  - ***Overflow*** pages if needed
- A hash function ***h*** : *K* → Int
- ***h***(*k*) mod *M* = bucket data entry for key k

  (*M* = # of buckets)

Static Hashing



Static Hashing

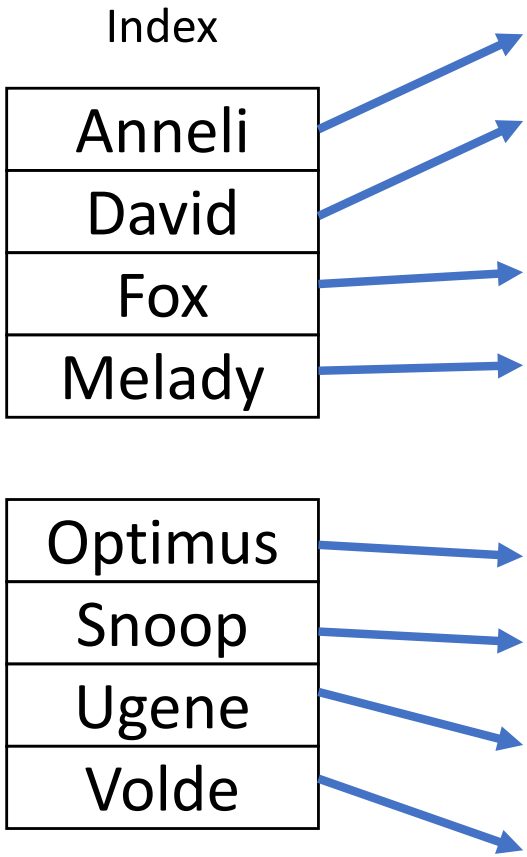
- Buckets contain *data entries*
- Hash function on search key field of records
  - Distributes values over range 0 ... *M* 1

    (**No guarantee about the distribution**)
- Overflow chains can develop & degrade performance
  - *Extendible and Linear hashing*: dynamic techniques to fix this

Terminology: Dense Index

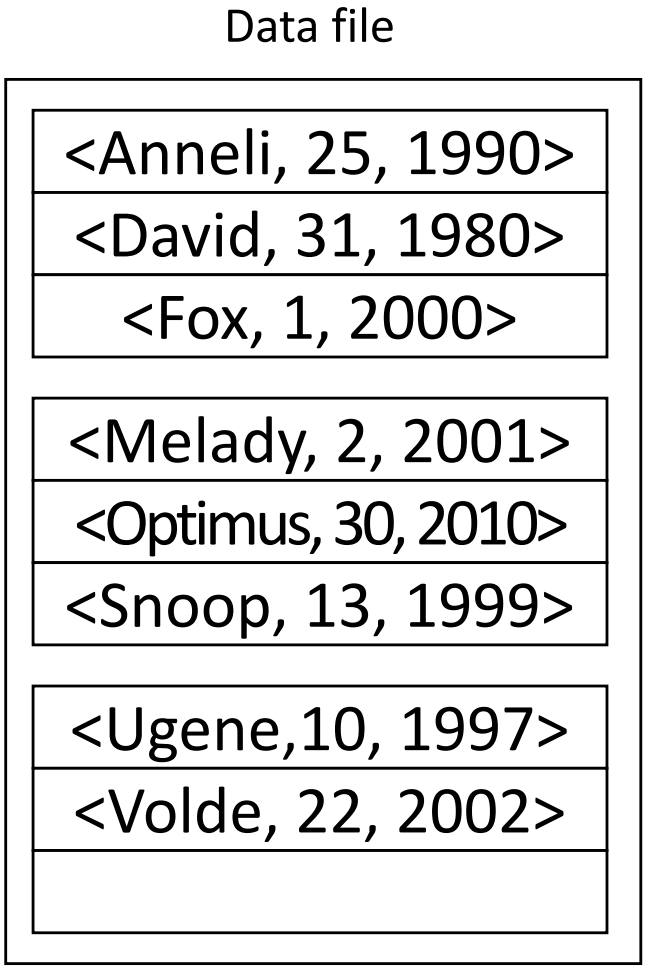
- One index entry for each data record

Search key is “name”



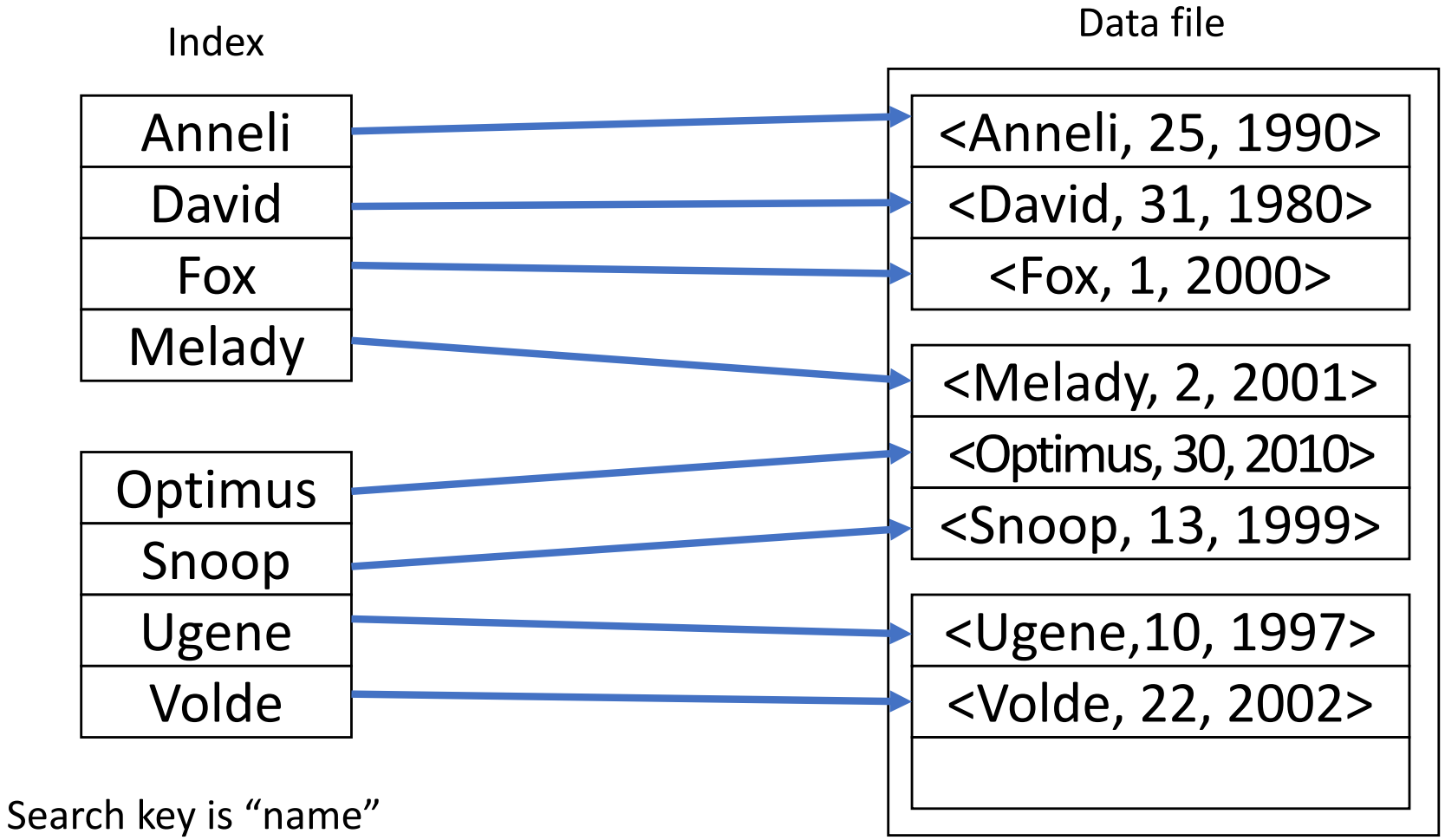
Terminology: Clustered Index

- Records are sorted based on search key



Terminology: Dense Clustered Index

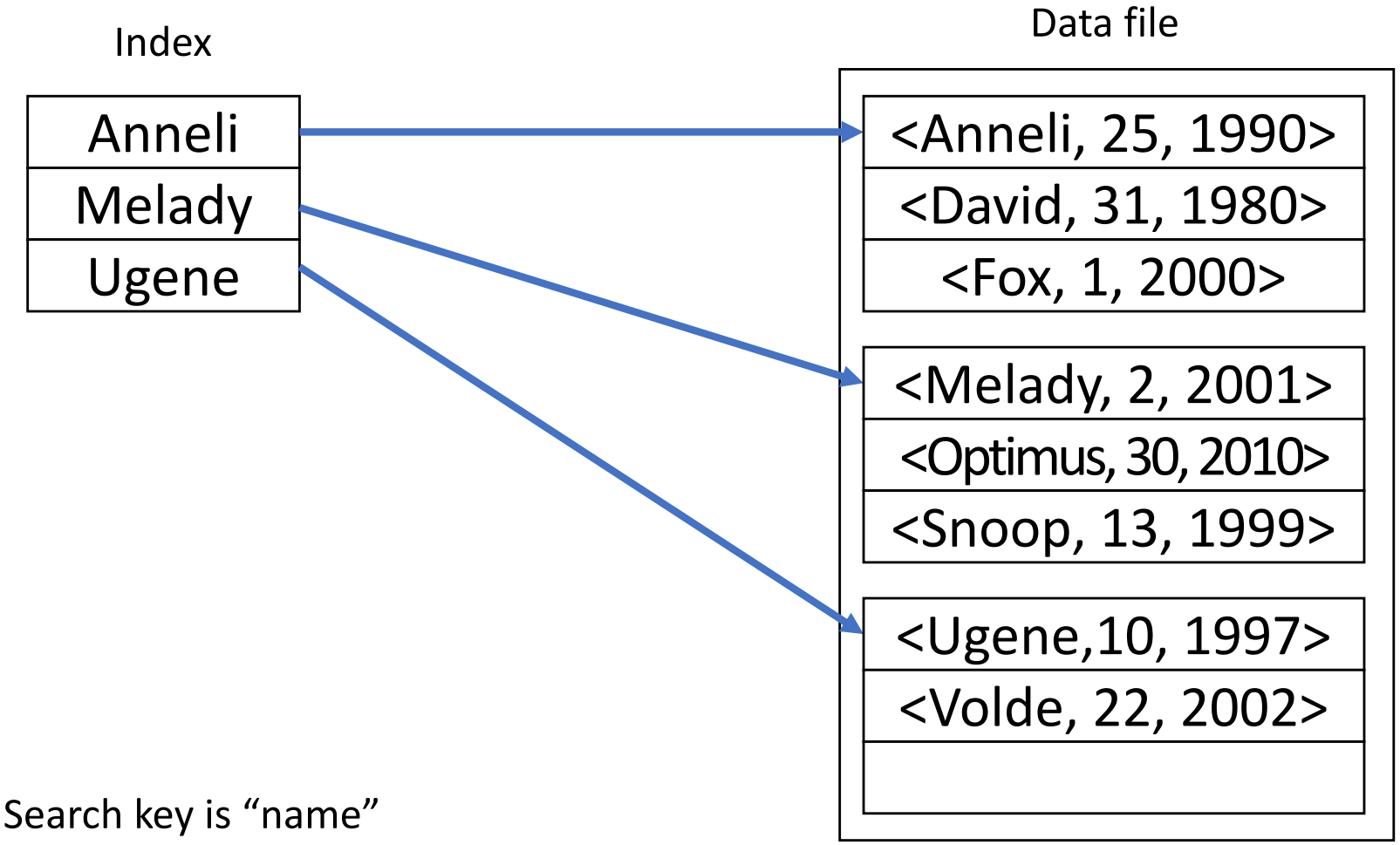
- One index entry for each data record
- Records are sorted based on search key



Search key is “name”

Terminology: Sparse Clustered Index

- One index entry per page of data
- Records sorted on search key

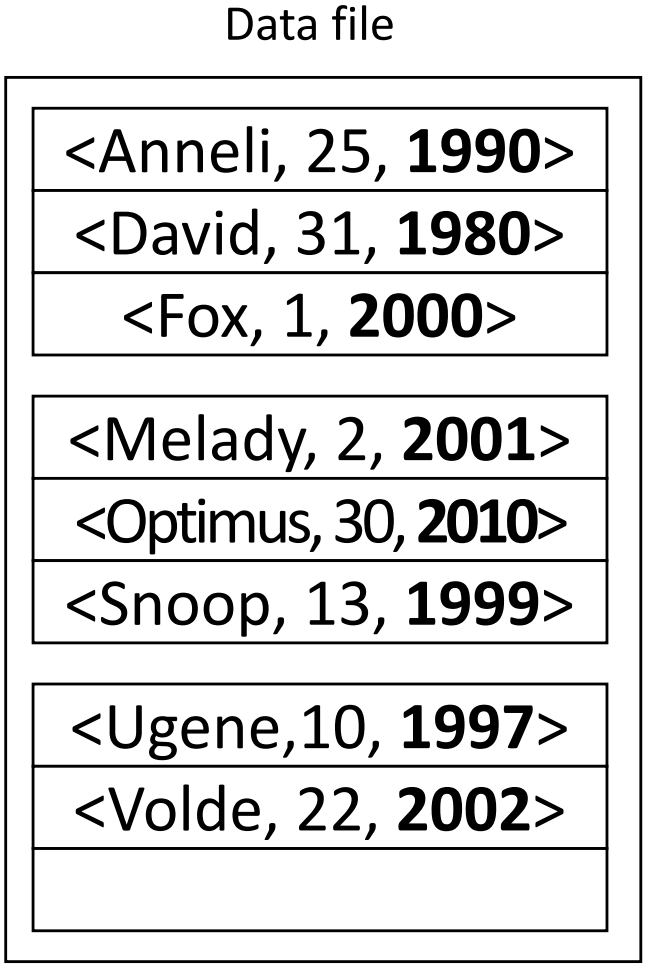


Search key is “name”

Terminology: Unclustered Index

- Records NOT sorted on search key

if the search key is year
This becomes
an unclustered index of year



Combinations

Can we define

- A sparse unclustered index?

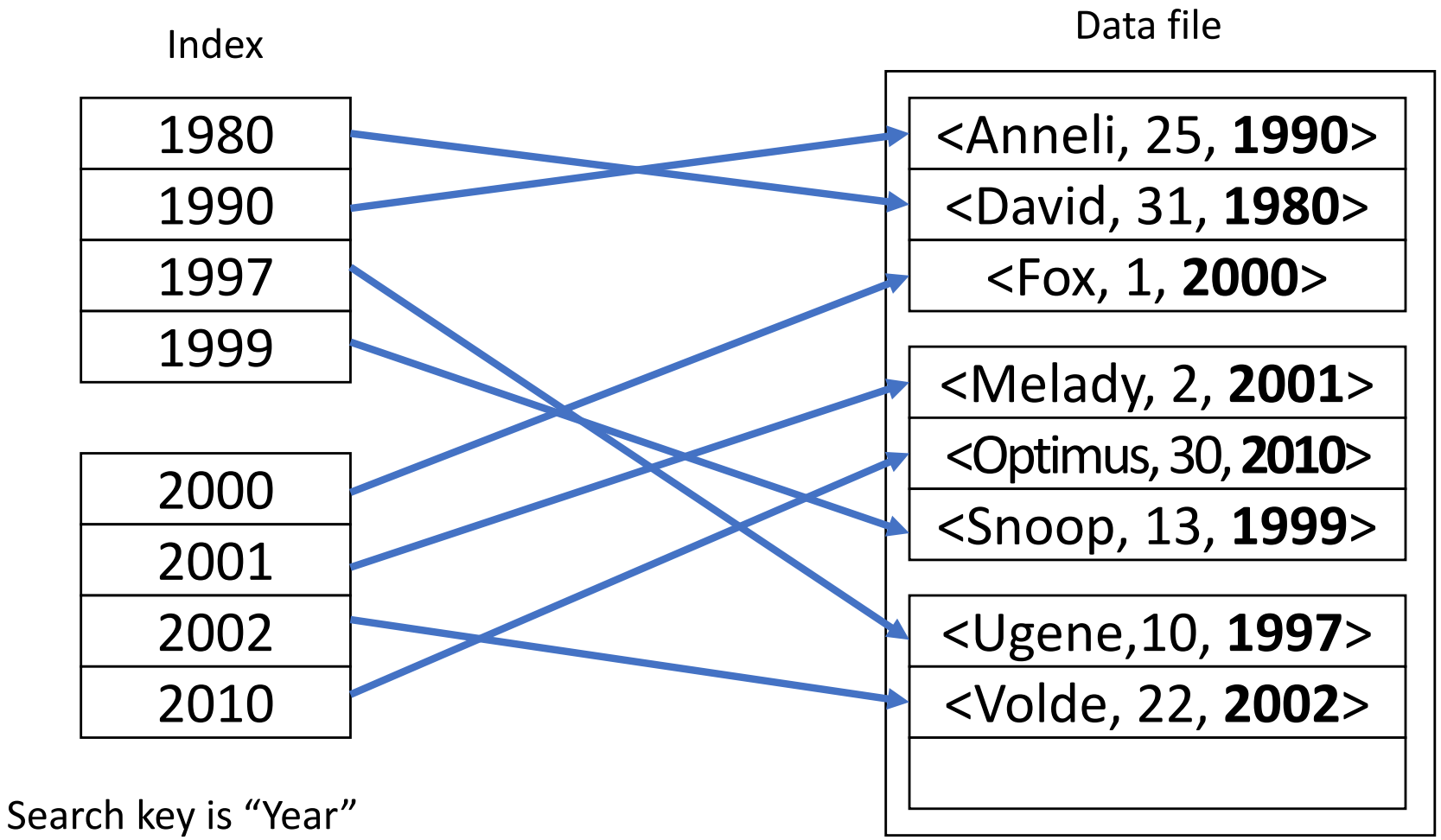
- A dense unclustered index?

HW: Find the Answer and
give reasons for your answer

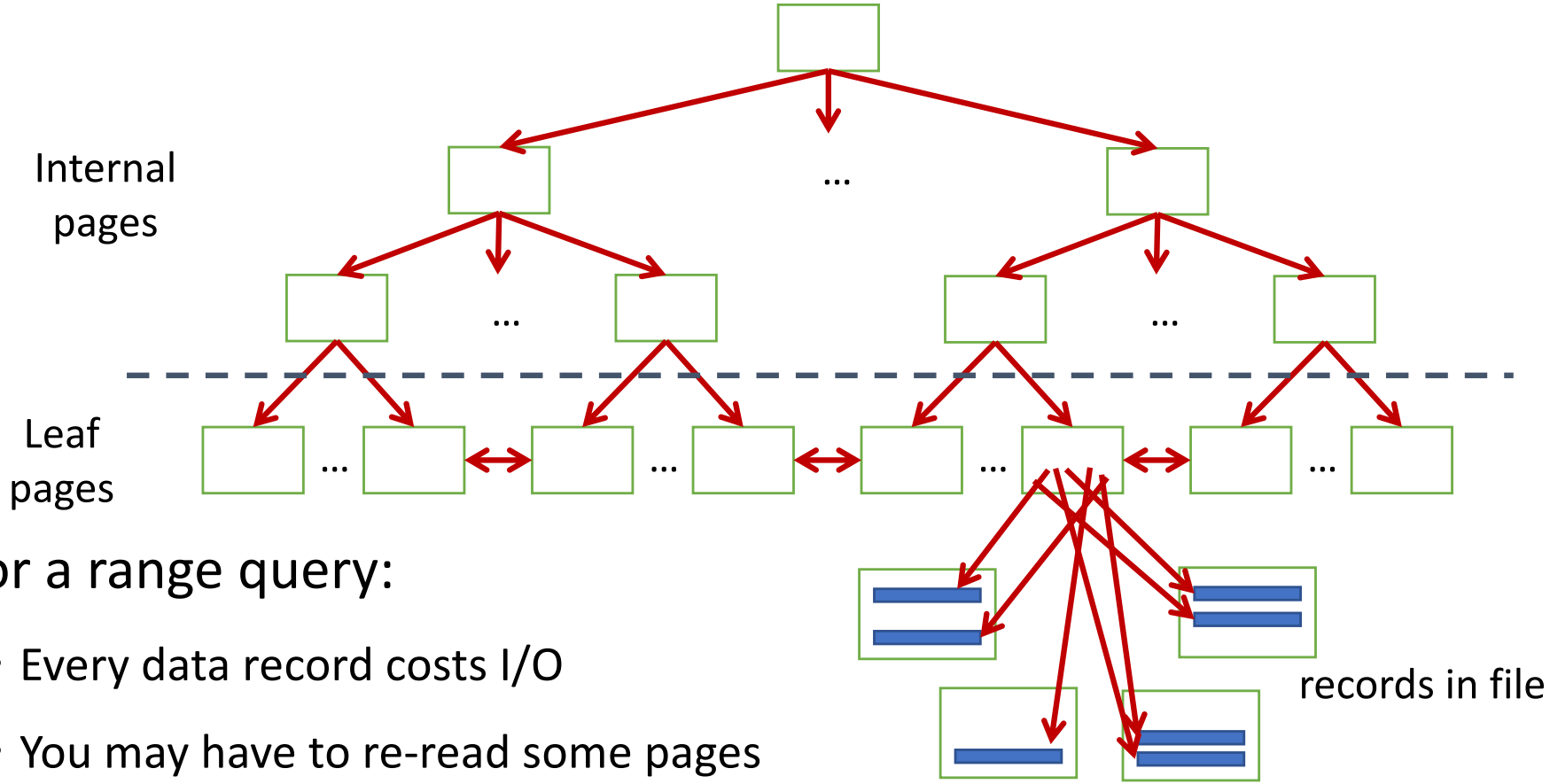
1 Week.

Terminology: Dense unclustered Index

- One entry per search key, records NOT sorted on search key



I/O cost of using dense unclustered index

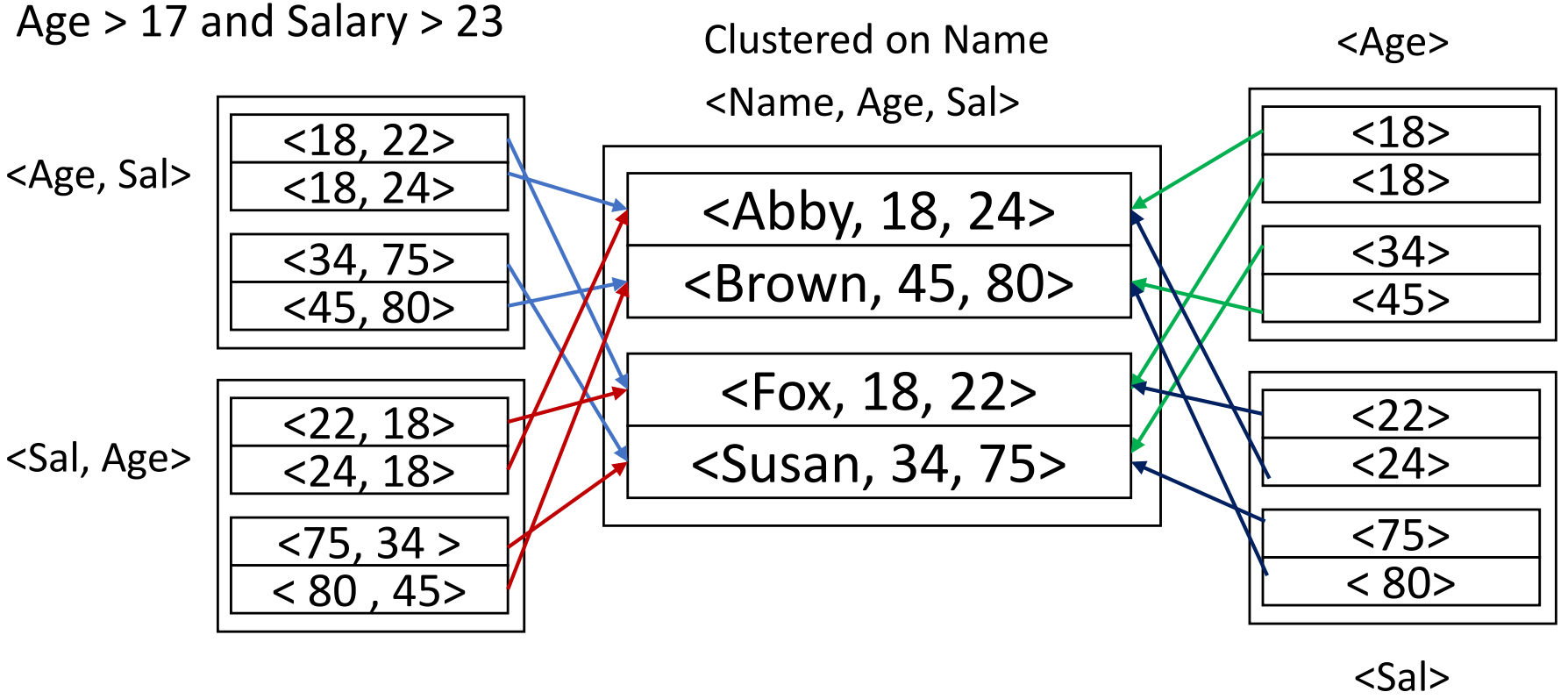


- For a range query:
  - Every data record costs I/O
  - You may have to re-read some pages
  - Cost to scan file in sort order equal to number of records in the file!

    (this is really bad, which is why we'll also talk about external sorting)

Composite Search Keys

- Which indexes should be used to answer these queries? Why?
  - Age = 18
  - Age = 18 and Salary = 22
  - Age = 18 and Salary > 15
  - Age > 17 and Salary > 23



Indexes (almost) final words

- A DBMS often creates *clustered indexes* on primary keys
  - i.e., the file is sorted on the declared keys
  - Since primary keys must be the values used in foreign keys

    ... these indexes used in joins (Employee.cid = Company.id)
- Only one clustered index is create per table ... Why?
- As a DB designer/administrator, you must determine whether you need additional (unclustered) indexes
  - Including composite indexes
  - You have to consider trade-offs between query & update
  - This is one reason why DBAs get paid so well!

For Next Week

- Read
  - Ch. 12: 12.3.3
  - Ch. 14: 14.4 (up to 14.4.2)