6. Storage

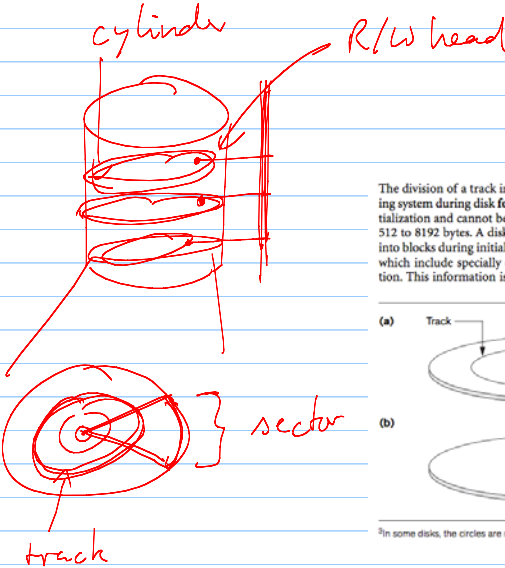
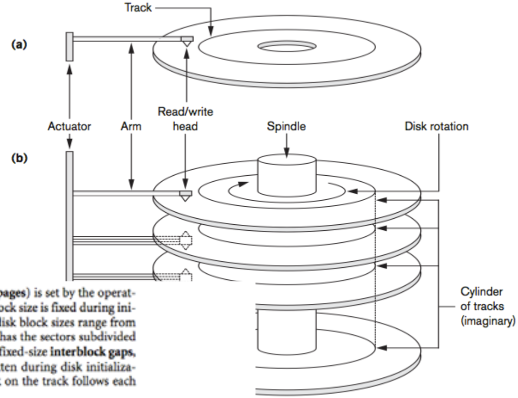


Figure 17.1
(a) A single-sided disk with read/write hardware.
(b) A disk pack with read/write hardware.



The division of a track into equal-sized **disk blocks** (or **pages**) is set by the operating system during disk **formatting** (or **initialization**). Block size is fixed during initialization and cannot be changed dynamically. Typical disk block sizes range from 512 to 8192 bytes. A disk with hard-coded sectors often has the sectors subdivided into blocks during initialization. Blocks are separated by fixed-size **interblock gaps**, which include specially coded control information written during disk initialization. This information is used to determine which block on the track follows each

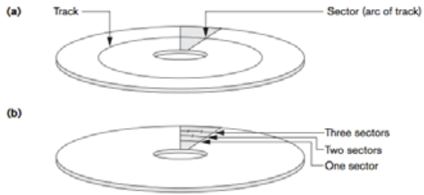
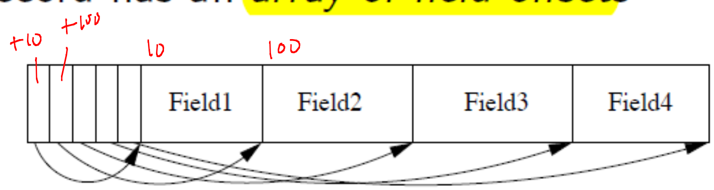


Figure 17.2
Different sector organizations on disk. (a) Sectors subtending a fixed angle. (b) Sectors maintaining a uniform recording density.

[3]In some disks, the circles are now connected into a kind of continuous spiral.

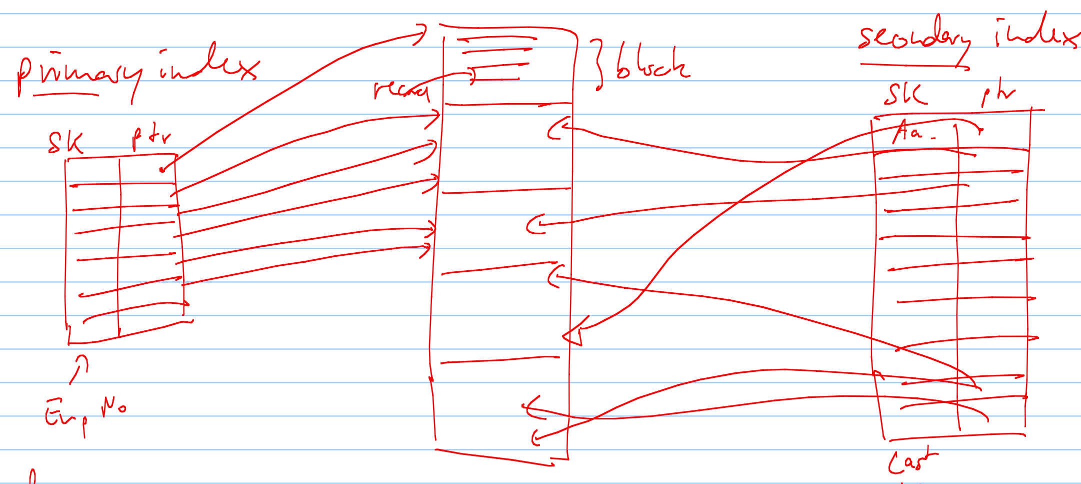
3. Each record has an *array of field offsets*



+ For the overhead of the offset, we get direct access to any field
+ Null values: begin/end pointer of a field point to the same address

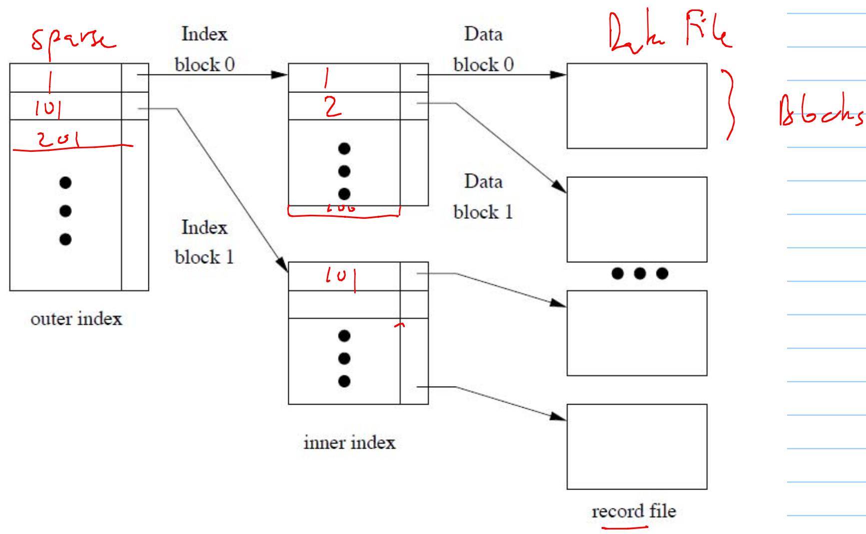
7. Indexing

EMPLOYEE file (sorted by EmpNo)

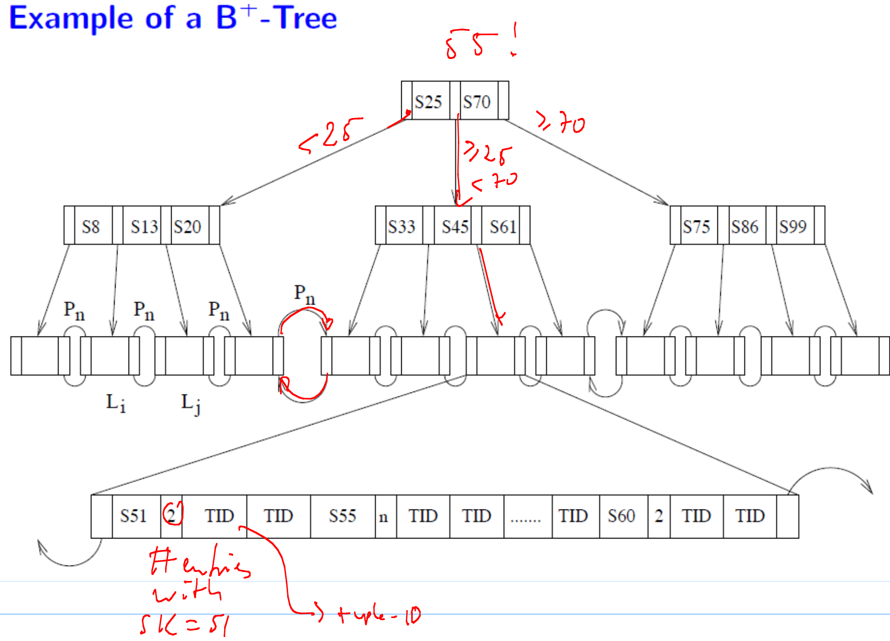


dense or sparse

must be dense



Example of a B^+-Tree



9. Query Processing

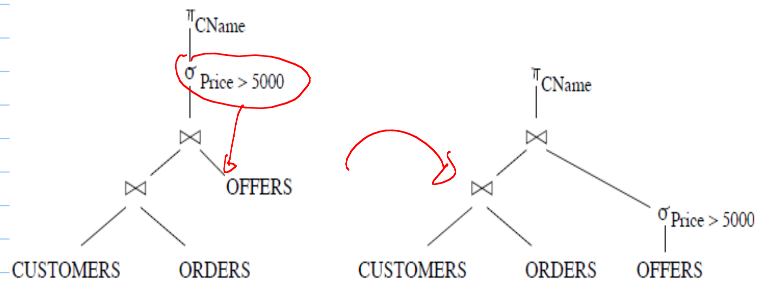
$\pi_{\texttt{CName}}(\sigma_{\texttt{Price}>5000}((\texttt{CUSTOMERS} \bowtie \texttt{ORDERS}) \bowtie \texttt{OFFERS}))$

$\pi_{\texttt{CName}}((\texttt{CUSTOMERS} \bowtie \texttt{ORDERS}) \bowtie (\sigma_{\texttt{Price}>5000}(\texttt{OFFERS})))$

Representation as *logical query plan* (a tree):

$(A \bowtie B) \bowtie C$

$= A \bowtie (B \bowtie C)$



- N_R: number of tuples in the relation R.
- B_R: number of blocks that contain tuples of the relation R.
- S_R: size of a tuple of R.
- F_R: blocking factor; number of tuples from R that fit into one block ($F_R = \lceil N_R/B_R \rceil$)
- $V(A, R)$: number of distinct values for attribute A in R. $|\pi_A(R)|$
  if A is a key $\rightsquigarrow V(A,R) = N_R$
- $SC(A, R)$: selectivity of attribute A
  $\equiv$ average number of tuples of R that satisfy an equality condition on A. $\sim \sigma_{A=c}(R)$
  $SC(A, R) = N_R / V(A, R)$.
  1 ... NR

- big value for V $\rightsquigarrow SC \sim 1$
  $\Rightarrow$ very selective
- small value for V $\rightsquigarrow SC \sim N_R$
  $\Rightarrow$ not selective

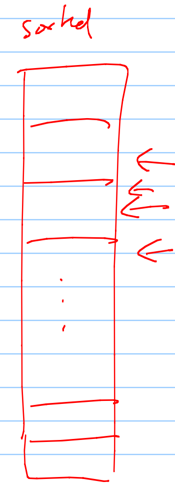
$\sigma_{A='c'}(R)$

- **S2 – Binary search**, i.e., the file ordered based on attribute A (primary index)

  # block with result tuples

  $$\text{cost(S2)} = \lceil \log_2(B_R) \rceil + \left\lceil \frac{SC(A,R)}{F_R} \right\rceil - 1$$

  - $\lceil \log_2(B_R) \rceil \equiv$ cost to locate the first tuple using binary search
  - Second term $\equiv$ blocks that contain records satisfying the selection.
  - If A is primary key, then $SC(A,R) = 1$, hence $\text{cost(S2)} = \lceil \log_2(B_R) \rceil$.



- **Example** (for Employee DB)
  - $F_{\text{Employee}} = 10$;
    $V(\text{Deptno}, \text{Employee}) = 50$ (different departments) $\le |\Pi_{DeptNo}(EMP)|$
  - $N_{\text{Employee}} = 10,000$ (Relation Employee has 10,000 tuples)
  - Assume selection $\boxed{\sigma_{\text{Deptno}=20}(\text{Employee})}$ and Employee is sorted on search key Deptno :

    N_R $V(D, Emp)$

    $\Longrightarrow$ 10,000/50 = 200 tuples in Employee belong to Deptno 20;

    (assuming an equal distribution)

    200/10 = 20 blocks for these tuples

    $\Longrightarrow$ A binary search finding the first block would require $\lceil \log_2(1,000) \rceil = 10$ block accesses B+-tree

  Total cost of binary search is 10+20 block accesses (versus 1,000 for linear search and Employee not sorted by Deptno).

  blocks with dept=20 employees

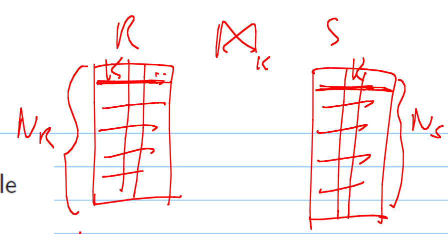
Complex Selections

- General pattern:
  - Conjunction – $\sigma_{\theta_1 \wedge .. \wedge \theta_n}(R)$
  - Disjunction – $\sigma_{\theta_1 \vee .. \vee \theta_n}(R)$
  - Negation – $\sigma_{\neg\theta}(R)$

$p(\theta_1) \cdot p(\theta_2)$

$p(\theta_1) + p(\theta_2) - p(\theta_1 \wedge \theta_2)$

$R \bowtie S$



- If $\mathtt{schema(R)} \cap \mathtt{schema(S)}$ = primary key for R, then a tuple of S will match with at most one tuple from R.
  Therefore, the number of tuples in R⋈S is not greater than N_S $\geq |R \bowtie S|$

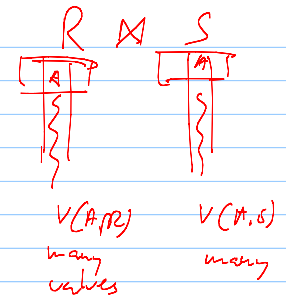
  If $\mathtt{schema(R)} \cap \mathtt{schema(S)}$ = foreign key in S referencing R, then the number of tuples in R⋈S is exactly N_S. not NULL
  Other cases are symmetric.

- In the example query CUSTOMERS ⋈ ORDERS, CName in ORDERS is a foreign key of CUSTOMERS; the result thus has exactly N_{ORDERS} = 10,000 tuples

- If $\mathtt{schema(R)} \cap \mathtt{schema(S)} = \{A\}$ is not a key for R or S; assume that every tuple in R produces tuples in R ⋈ S. Then the number of tuples in R ⋈ S is estimated to be: $\frac{\mathtt{N_R} * \mathtt{N_S}}{\mathtt{V(A, S)}}$ $= SC(A,S)$

  If the reverse is true, the estimate is $\frac{\mathtt{N_R} * \mathtt{N_S}}{\mathtt{V(A, R)}}$ $= SC(A,R)$

  and the lower of the two estimates is probably the more accurate one.



- Evaluate the condition join $R \bowtie_C S$

- **for each** tuple t_R **in** R **do begin**
  - **for each** tuple t_S **in** S **do begin**
    - check whether pair (t_R, t_S) satisfies join condition
    - if they do, add $t_R \circ t_S$ to the result
  - **end**
- **end**

- R is called the *outer* and S the *inner* relation of the join.

- Requires no indexes and can be used with any kind of join condition.

- Worst case: db buffer can only hold one block of each relation
  $\Longrightarrow \mathtt{B_R} + \mathtt{N_R} * \mathtt{B_S}$ disk accesses

- Best case: both relations fit into db buffer
  $\Longrightarrow \mathtt{B_R} + \mathtt{B_S}$ disk accesses.