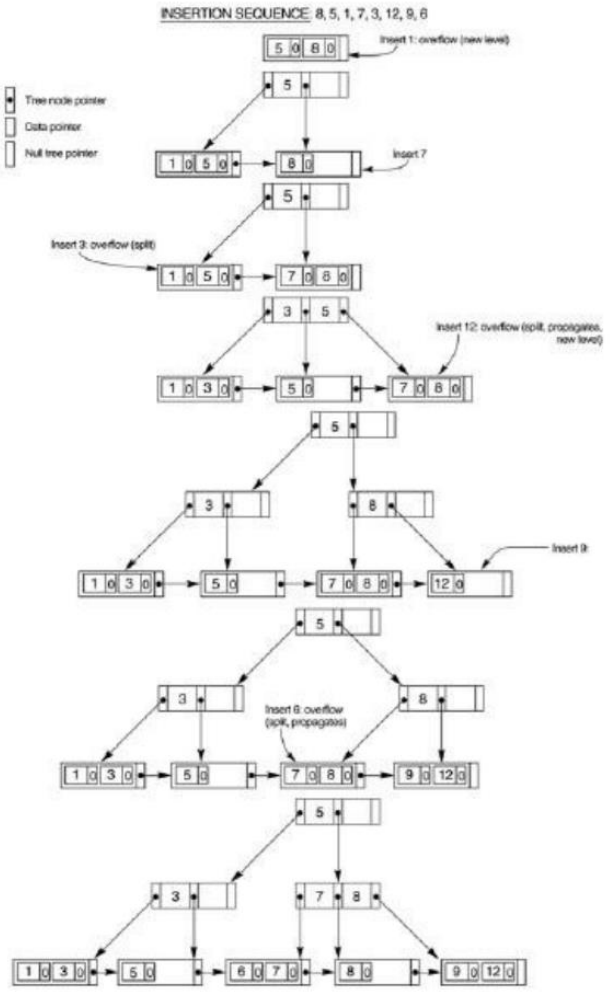


SQL clause	Relational operation	Meaning
FROM a single table	(none)	Input table
FROM table1, table2	table1 X table2	Cartesian product
FROM table1 JOIN table2 ON conditions	table1 $\bowtie_{\text{conditions}}$ table2	Theta join
WHERE conditions	$\sigma_{\text{conditions}}$	Selection
SELECT an attribute list	$\pi_{\text{an attribute list}}$	Projection
SELECT a function list ... [GROUP BY a grouping attribute list]	<a grouping attribute list> $\Im$ <function list>	Aggregation

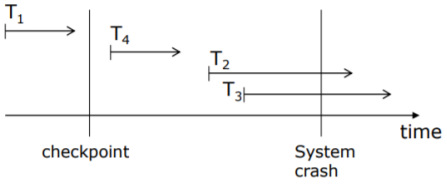
Deferred Update with concurrent users

(a)

T_1	T_2	T_3	T_4
read_item (A)	read_item (B)	read_item (A)	read_item (B)
read_item (D)	write_item (B)	write_item (A)	write_item (B)
write_item (D)	read_item (D)	read_item (C)	read_item (A)
	write_item (D)	write_item (C)	write_item (A)

(b)

[start_transaction, T_1]
[write_item, T_1, D, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[write_item, T_4, A, 20]
[commit, T_4]
[start_transaction T_2]
[write_item, T_2, B, 12]
[start_transaction, T_3]
[write_item, T_3, A, 30]
[write_item, T_2, D, 25] ← system crash



T_2 and T_3 are ignored because they did not reach their commit points.
T_4 is redone because its commit point is after the last checkpoint.

26

Câu 12. Sự khác biệt giữa tối ưu hóa dựa trên quy tắc kinh nghiệm và tối ưu hóa dựa trên chi phí là …

A. Đường truy đạt cụ thể trên mỗi tập tin trong cây truy vấn tối ưu không được xác định tường minh trong tối ưu hóa dựa trên quy tắc kinh nghiệm, nhưng được chỉ rõ với chi phí bao nhiêu trong tối ưu hóa dựa trên chi phí.

Câu 23. Phân trang bóng âm (*Shadow paging*) có ưu điểm gì so với kỹ thuật phục hồi ARIES?

A. Không gian lưu trữ và quản lý lưu trữ được chuẩn bị dễ dàng so với sổ ghi nhật ký với các bảng Transaction và Dirty Page của ARIES.

B. Quá trình phục hồi đơn giản vì sau khi hệ thống gặp sự cố, không có tác vụ của bất kỳ giao tác nào cần được tái thực thi hay tháo gỡ so với quá trình tái hiện lịch sử cập nhật dữ liệu của ARIES.

Câu 22. Giao thức ghi nhật ký trước (*Write-Ahead Logging*) đảm bảo phục hồi cho …

A. các giao tác đã commit với các nội dung AFIM và các giao tác chưa commit với các nội dung BFIM trong các kỹ thuật phục hồi tại chỗ.

B. các giao tác đã commit với các nội dung BFIM và các giao tác chưa commit với các nội dung AFIM trong các kỹ thuật phục hồi tại chỗ.

C. các giao tác đã commit với các nội dung AFIM và các giao tác chưa commit với các nội dung BFIM trong tất cả các kỹ thuật phục hồi.

D. các giao tác đã commit với các nội dung BFIM và các giao tác chưa commit với các nội dung AFIM trong tất cả các kỹ thuật phục hồi.

E. Ý kiến khác.

Câu 24. ARIES là kỹ thuật phục hồi khác với phân trang bóng âm (*shadow paging*) ở điểm nào sau đây?

A. ARIES dạng phục hồi tại chỗ và phân trang bóng âm không phải.

B. ARIES tái hiện lịch sử cập nhật dữ liệu và phân trang bóng âm không phải.

C. ARIES thuộc dạng UNDO/REDO và phân trang bóng âm không phải.

D. Cả ba đặc điểm trên.

Câu 23. Phân trang bóng âm (*Shadow paging*) có ưu-khuyết điểm gì?

A. Không gian lưu trữ và quản lý lưu trữ cần được chuẩn bị.

B. Cơ chế thu rác cần được chuẩn bị.

C. Quá trình phục hồi đơn giản vì sau khi hệ thống gặp sự cố, không có tác vụ của bất kỳ giao tác nào cần được tái thực thi hay tháo gỡ.

D. Cả ba đặc điểm trên.

E. Ý kiến khác.

Câu 8. Sự khác biệt giữa chỉ mục thứ tự (*ordered index*) và chỉ mục băm (*hash index*) là …

A. Tập tin dữ liệu là tập tin có thứ tự đối với chỉ mục thứ tự và tập tin dữ liệu là tập tin băm đối với chỉ mục băm.

B. Các giá trị được chỉ mục sẽ được sắp thứ tự trước trong tập tin dữ liệu rồi sau đó được lưu vào tập tin chỉ mục thứ tự đối với chỉ mục thứ tự và các giá trị được chỉ mục sẽ được băm trước trong tập tin dữ liệu rồi sau đó được lưu vào tập tin chỉ mục băm đối với chỉ mục băm.

C. Tập tin chỉ mục là tập tin có thứ tự đối với chỉ mục thứ tự và tập tin chỉ mục là tập tin băm đối với chỉ mục băm.

D. Các giá trị được chỉ mục sẽ được sắp thứ tự và được lưu vào tập tin chỉ mục thứ tự đối với chỉ mục thứ tự và các giá trị được chỉ mục sẽ được băm và lưu vào các thùng băm trong vùng đệm của tập tin chỉ mục băm đối với chỉ mục băm.

E. Ý kiến khác.

4.4. Algorithms for SELECT and JOIN Operations

- J1. **Nested-loop join** (brute force):

OP8: EMPLOYEE $\bowtie_{DNO=DNUMBER}$ DEPARTMENT

b_E = 2000 blocks, b_D = 10 blocks, n_B = 7 blocks

J1.1. EMPLOYEE on the outer loop

Cost = $b_E + b_D * \lceil b_E/(n_B-2) \rceil = 2000 + 10 * \lceil 2000/(7-2) \rceil$

Cost = 6000 block accesses

J1.2. DEPARTMENT on the outer loop

Cost = $b_D + b_E * \lceil b_D/(n_B-2) \rceil = 10 + 2000 * \lceil 10/(7-2) \rceil$

Cost = 4010 block accesses Smaller file on the outer loop!!!

Cost Functions for SELECT:

- **S1. Linear search (brute force) approach**
  To retrieve all records satisfying the selection condition:
  $C_{S1a} = b$
  To retrieve a single record for an equality condition on a key:
  if the record is found, $C_{S1b} = \lceil b/2 \rceil$
  otherwise, $C_{S1a} = b$
- **S2. Binary search**
  $C_{S2} = \lceil \log_2 b \rceil + \lceil s/bfr \rceil - 1$
  For an equality condition on a unique (key) attribute (s=1),
  $C_{S2} = \lceil \log_2 b \rceil$
- **S3. Using a primary index (S3a) or hash key (S3b) to retrieve a single record**
  $C_{S3a} = x + 1$
  C_{S3b} = 1 for static or linear hashing
  C_{S3b} = 2 for extendible hashing

103

4.9. Using Selectivity and Cost Estimates in Query Optimization

Cost Functions for SELECT:

- **S4. Using an ordering index to retrieve multiple records**
  For the comparison condition (>, >=, <, or <=) on a key field with an ordering index
  $C_{S4} = x + \lceil b/2 \rceil$
- **S5. Using a clustering index to retrieve multiple records for an equality condition**
  $C_{S5} = x + \lceil s/bfr \rceil$
- **S6. Using a secondary (B^+-tree) index**
  For an equality comparison, $C_{S6a} = x + 1 + s$
  (The additional 1 is to account for the disk block that contains the record pointers after the index is searched.)
  For a comparison condition (>, <, >=, or <=),
  $C_{S6b} = x + \lceil b_{I1}/2 \rceil + \lceil r/2 \rceil$
  (Half the file records are assumed to satisfy the condition.)

104

OP8: EMPLOYEE $\bowtie_{DNO=DNUMBER}$ DEPARTMENT

- Method J1 (Nested loop) with Employee as outer:
  - $C_{J1} = b_E + b_E * b_D + \lceil (js_{OP6} * r_E * r_D)/bfr_{ED} \rceil$
    $= 2000 + 2000 * 13 + \lceil ((1/125) * 10{,}000 * 125)/4 \rceil = 30{,}500$
- Method J1 (Nested loop) with Department as outer:
  - $C_{J1} = b_D + b_E * b_D + \lceil (js_{OP6} * r_E * r_D)/bfr_{ED} \rceil$
    $= 13 + 13 * 2000 + \lceil ((1/125) * 10{,}000 * 125)/4 \rceil = 28{,}513$
- Method J2 (Single loop) with EMPLOYEE as outer loop:
  - $C_{J2c} = b_E + r_E * (x_{DNUMBER} + 1) + \lceil (js_{OP6} * r_E * r_D)/bfr_{ED} \rceil$
    $= 2000 + 10{,}000 * 2 + \lceil ((1/125) * 10{,}000 * 125)/4 \rceil = 24{,}500$
- Method J2 (Single loop) with DEPARTMENT as outer loop:
  - $C_{J2a} = b_D + r_D * (x_{DNO} + s_{DNO} + 1) + \lceil (js_{OP6} * r_E * r_D)/bfr_{ED} \rceil$
    $= 13 + 125 * (2 + 80 + 1) + \lceil ((1/125) * 10{,}000 * 125)/4 \rceil = 12{,}888$
- Method J4 (Hash join) with DEPARTMENT as a hashed file:
  - $C_{J4} = 3*(b_D + b_E) + \lceil (js_{OP6} * r_E * r_D)/bfr_{ED} \rceil$ Chosen!!!
    $= 3*(13+2000) + \lceil ((1/125) * 10{,}000 * 125)/4 \rceil = 8{,}539$

117

Catalog Information Used in Cost Functions

- Information about the size of a file
  - **number of records (tuples) (r)**
  - **record size (R)**
  - **number of blocks (b)**
  - **blocking factor (bfr)**
- Information about indexes and indexing attributes of a file
  - **Number of levels (x)** of each multilevel index
  - **Number of first-level index blocks (b_{I1})**
  - **Number of distinct values (d)** of an attribute
  - **Selectivity (sl)** of an attribute (the fraction of records satisfying an equality condition on the attribute)
  - **Selection cardinality (s)** of an attribute (s = sl * r)

s = số bán ghi / giá trị pb

1

Cost Functions for JOIN:

- **J1. Nested-loop join**
  $C_{J1} = b_R + b_R * b_S + \lceil (js * |R| * |S|)/bfr_{RS} \rceil$
  (Use *R* for outer loop. Assume a three-block buffer.)
- **J2. Single-loop join**: using an access structure to retrieve the matching record(s)
  If an index exists for the join attribute *B* of *S* with index levels
  x_B, we can retrieve each record *s* in *R* and then use the index to retrieve all the matching records *t* from *S* that satisfy t[B] = s[A].
  The cost depends on the type of index.

1

4.9. Using Selectivity and Cost Estimates in Query Optimization

Cost Functions for JOIN:

- **J2. Single-loop join**
  For a secondary index,
  $C_{J2a} = b_R + |R| * (x_B + s_B + 1) + \lceil (js * |R| * |S|)/bfr_{RS} \rceil$
  For a clustering index,
  $C_{J2b} = b_R + |R| * (x_B + \lceil s_B/bfr_B \rceil) + \lceil (js * |R| * |S|)/bfr_{RS} \rceil$
  For a primary index,
  $C_{J2c} = b_R + |R| * (x_B + 1) + \lceil (js * |R| * |S|)/bfr_{RS} \rceil$
  If a hash key exists for one of the two join attributes — B of S
  $C_{J2d} = b_R + |R| * h + \lceil (js * |R| * |S|)/bfr_{RS} \rceil$
  h: the average number of block accesses to retrieve a record, given its hash key value, h>=1

1

Assume that the DEPARTMENT file of $r_D = 125$ and $b_D = 13$, $x_{DNUMBER} = 1$, secondary index on MGRSSN of DEPARTMENT, $s_{MGRSSN} = 1$, $x_{MGRSSN} = 2$, $js_{OP9} = (1/|EMPLOYEE|) = 1/r_E = 1/10{,}000$, $bfr_{ED} = 4$
A secondary index on the key attribute SSN of EMPLOYEE, with $x_{SSN} = 4$ ($S_{SSN} = 1$).

- OP9: DEPARTMENT $\bowtie_{MGRSSN=SSN}$ EMPLOYEE
  - Method J1 (Nested loop) with Employee as outer:
    - $C_{J1} = b_E + b_E * b_D + \lceil (js_{OP7} * r_E * r_D)/bfr_{ED} \rceil$
    = 2000 + 2000 * 13 + $\lceil ((1/10{,}000) * 10{,}000 * 125)/4 \rceil$
    = $\lceil 28{,}031.25 \rceil$ = 28,032
  - Method J1 (Nested loop) with Department as outer:
    - $C_{J1} = b_D + b_E * b_D + \lceil (js_{OP7} * r_E * r_D)/bfr_{ED} \rceil$
    = 13 + 13 * 2000 + $\lceil ((1/10{,}000) * 10{,}000 * 125)/4 \rceil$
    = $\lceil 26{,}044.25 \rceil$ = 26,045
  - Method J2 (Single loop) with EMPLOYEE as outer loop:
    - $C_{J2c} = b_E + r_E * (x_{MGRSSN} + s_{MGRSSN} + 1) + \lceil (js_{OP7} * r_E * r_D)/bfr_{ED} \rceil$
    = 2000 + 10,000 * (2 + 1 + 1) + $\lceil ((1/10{,}000) * 10{,}000 * 125)/4 \rceil$
    = $\lceil 42{,}031.25 \rceil$ = 42,032
  - Method J2 (Single loop) with DEPARTMENT as outer loop:
    - $C_{J2a} = b_D + r_D * (x_{SSN} + s_{SSN} + 1) + \lceil (js_{OP7} * r_E * r_D)/bfr_{ED} \rceil$
    = 13 + 125 * (4 + 1 + 1) + $\lceil ((1/10{,}000) * 10{,}000 * 125)/4 \rceil$
    = $\lceil 794.25 \rceil$ = 795 Chosen!!!

118

Examples for SELECT

- EMPLOYEE: $r_E = 10{,}000$, $b_E = 2000$, $bfr_E = 5$
- Access paths:
  - 1. A clustering index on SALARY, with levels $x_{SALARY} = 3$ and average selection cardinality $S_{SALARY} = 20$.
  - 2. A secondary index on the key attribute SSN, with $x_{SSN} = 4$ ($S_{SSN} = 1$).
  - 3. A secondary index on the nonkey attribute DNO, with $x_{DNO} = 2$ and first-level index blocks $b_{I1DNO} = 4$.
    There are $d_{DNO} = 125$ distinct values for DNO, so the selection cardinality of DNO is $S_{DNO} = \lceil r_E/d_{DNO} \rceil = 80$.
  - 4. A secondary index on SEX, with $x_{SEX} = 1$.
    There are $d_{SEX} = 2$ values for the sex attribute, so the average selection cardinality is $S_{SEX} = \lceil r_E/d_{SEX} \rceil = 5000$.

106

Examples for SELECT

- OP1: $\sigma_{SSN='123456789'}$ (EMPLOYEE)
  - $C_{S1b} = 1000$
  - $C_{S6a} = x_{SSN} + 1 = 4 + 1 = 5$
- OP2: $\sigma_{DNO>5}$ (EMPLOYEE)
  - $C_{S1a} = 2000$
  - $C_{S6b} = x_{DNO} + \lceil b_{I1DNO}/2 \rceil + \lceil r_E/2 \rceil = 2 + \lceil 4/2 \rceil + \lceil 10000/2 \rceil = 5004$

107

Examples for SELECT

- OP3: $\sigma_{DNO=5}$ (EMPLOYEE)
  - $C_{S1a} = 2000$
  - $C_{S6a} = x_{DNO} + s_{DNO} + 1 = 2 + 80 + 1 = 83$
- OP4: $\sigma_{DNO=5 \text{ AND } SALARY>30000 \text{ AND } SEX='F'}$ (EMPLOYEE)
  - $C_{S6a\text{-}DNO} = 83$
  - $C_{S4\text{-}SALARY} = x_{SALARY} + \lceil b/2 \rceil = 3 + \lceil 2000/2 \rceil = 1003$
  - $C_{S6a\text{-}SEX} = x_{SEX} + s_{SEX} + 1 = 1 + 5000 + 1 = 5002$

  => chose DNO=5 first and check the other conditions

108

4.9. Using Selectivity and Cost Estimates in Query Optimization

Cost Functions for JOIN:

- **J3. Sort-merge join**

  $C_{J3a} = C_S + b_R + b_S + \lceil (js^* |R|^* |S|)/bfr_{RS} \rceil$

  (C_S: Cost for sorting files. If both files are sorted, $C_S = 0$.)
- **J4. Hash join**

  The records of files *R* and *S* are partitioned into smaller files. The partitioning of each file is done using the same hashing function *h* on the join attribute *A* of *R* (for partitioning file R) and *B* of *S* (for partitioning file S).

  For the larger files R and S that can not be hashed entirely in the memory: $C_{J4} = 3*(b_R + b_S) + \lceil (js^* |R|^* |S|)/bfr_{RS} \rceil$

  For a smaller file (either R or S) that can be hashed entirely in the memory: $C_{J4} = b_R + b_S + \lceil (js^* |R|^* |S|)/bfr_{RS} \rceil$

115

Examples for SELECT

- OP4': $\sigma_{Dno=5 \text{ OR } Salary > 30000 \text{ OR } Sex='F'}$ (EMPLOYEE)
  - $C_{S1a} = 2000$

  Using a bitmap index
  - $C_{S9} = s = s_{DNO} + s_{Salary} + s_{Sex} = 80 + 20 + 5000 = 5100$

  Using single access paths, a total cost with a disjunctive condition stems from the sum of all paths.
  - $C_{S6a\text{-}DNO} = x_{DNO} + s_{DNO} + 1 = 2 + 80 + 1 = 83$
  - $C_{S4\text{-}SALARY} = x_{SALARY} + \lceil b/2 \rceil = 3 + \lceil 2000/2 \rceil = 1003$
  - $C_{S6a\text{-}SEX} = x_{SEX} + s_{SEX} + 1 = 1 + 5000 + 1 = 5002$

  => Choose linear search with C_{S1a}

109

Examples for SELECT

- OP5: $\sigma_{ESSN='123456789' \text{ AND } PNO=10}$ (WORKS_ON)
  - $C_{S1a} = b_{WORKS\_ON}$

  Using a primary composite index on (ESSN, PNO)
  - $C_{S8} = C_{S3a} = x + 1$

  => choose the access path with a smaller cost: linear search or access via a primary composite index on (ESSN, PNO)

110

Examples for SELECT

- OP6: $\sigma_{DNO \text{ IN } (3, 27, 49)}$ (EMPLOYEE)
  - $C_{S1a} = 2000$

  Using a bitmap index
  - $C_{S9} = s = s_{DNO} + s_{DNO} + s_{DNO} = 80 + 80 + 80 = 240$

  Using a secondary index
  - $C_{S6a} = 3*(x_{DNO} + s_{DNO} + 1) = 3*(2 + 80 + 1) = 249$

  => Choose access via a bitmap index with C_{S9}
- OP7: $\sigma_{((Salary*Commission\_pct) + Salary) > 5000}$ (EMPLOYEE)
  - $C_{S1a} = 2000$

  Using a functional index
  - $C_{S6b} = x + \lceil b_{11}/2 \rceil + \lceil r/2 \rceil > 5000$

  => Choose linear search with CS1a

  How about if > is changed to =?

111

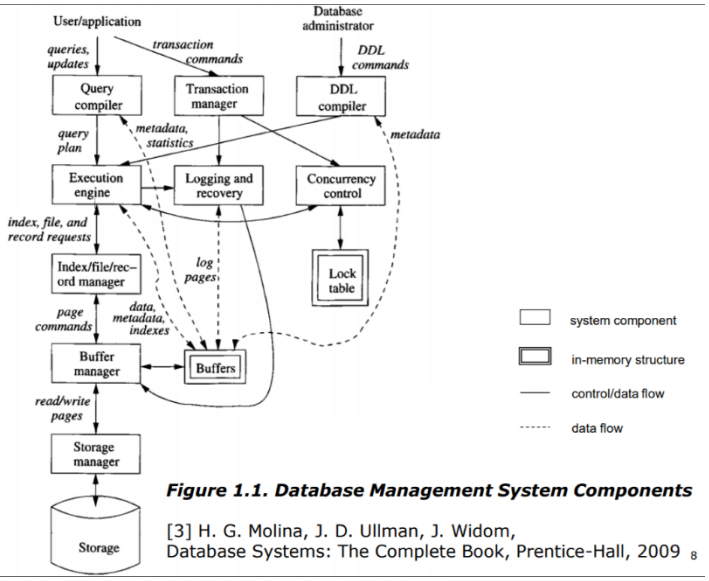
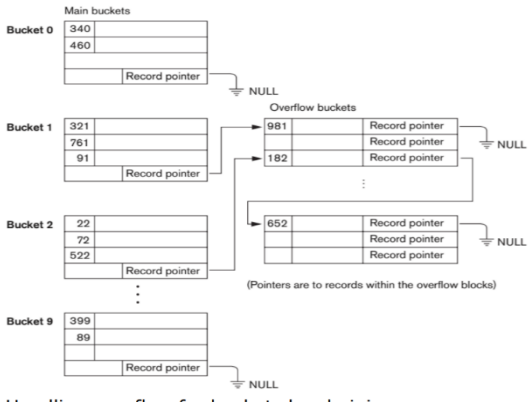


Figure 1.1. Database Management System Components

[3] H. G. Molina, J. D. Ullman, J. Widom,
Database Systems: The Complete Book, Prentice-Hall, 2009 8



Address space:
M buckets (=10)

Hash function:
h(*K*) = *K* mod *M*

How to store the following records in this hash file: 32, 179?

Are the following records: 81, 652 in this file?

Secondary indexes

Example 3: given the following data file with the non-ordering key field SSN
EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ...)

record size R=150 bytes
block size B=512 bytes
number of records r=30,000 records

blocking factor bfr= B div R= ⌊B/R⌋ = 512 div 150= 3 records/block
number of file blocks b = ⌈r/bfr⌉= ⌈30,000/3⌉ = 10,000 blocks

For a secondary index on the SSN field, assume the field size V_{SSN}=9 bytes, assume the block pointer size P=6 bytes, the record pointer size P_R=7 bytes.

index entry size R_i=V_{SSN}+ P=9+6=15 bytes
index blocking factor bfr_i= B div R_i= 512 div 15= 34 entries/block
number of index entries r_i = number of file records r = 30,000 entries
number of index blocks b_i= ⌈r_i/ bfr_i⌉ = ⌈30,000/34⌉ = 883 blocks
binary search on the index needs ⌈$\log_2 b_i$⌉ = ⌈$\log_2$883⌉ = 10 block accesses
one extra block access to retrieve the record from the data file
The total search cost via the index is: 10 + 1 = 11 block accesses

Because the data file is not ordered according to the values of SSN, an average linear search is done directly on the data file with the cost:
⌈b/2⌉ = ⌈10,000/2⌉ = 5,000 block accesses

22

- The average time needed to find and transfer *one* block, given its address, is estimated by: (*s* + *rd* + *btt*) msec
- The average time needed to find and transfer any *k* blocks, given the address of each block, is: *k**(*s* + *rd* + *btt*) msec
- The average time needed to find and transfer consecutively *k* *noncontiguous* blocks on the *same cylinder*, given the address of each block, is: (*s* + *k**(*rd* + *btt*)) msec
- The average time needed to find and transfer consecutively *k* *contiguous* blocks on the *same track or cylinder*, given the address of the first block, is: (*s* + *rd* + *k***btt*) msec
- The estimated time to read *k contiguous* blocks consecutively stored on the *same cylinder*, when the *bulk transfer rate* is used to transfer the *useful* data, is: (*s* + *rd* + *k**(*B*/*btr*)) msec

Primary indexes

Example 1: given the following data file with the ordering key field SSN
EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ...)

record size R=150 bytes
block size B=512 bytes
number of records r=30,000 records

blocking factor bfr= B div R= ⌊B/R⌋ = 512 div 150= 3 records/block
number of file blocks b = ⌈r/bfr⌉= ⌈30,000/3⌉ = 10,000 blocks

For a primary index on the SSN field, assume the field size V_{SSN}=9 bytes, assume the block pointer size P_B=6 bytes.

index entry size R_i=V_{SSN}+ P_B=9+6=15 bytes
index blocking factor bfr_i= B div R_i= 512 div 15= 34 entries/block
number of index entries r_i = number of file blocks b = 10,000 entries
number of index blocks b_i= ⌈r_i/ bfr_i⌉ = ⌈10,000/34⌉ = 295 blocks
binary search on the index needs ⌈$\log_2 b_i$⌉ = ⌈$\log_2$295⌉ = 9 block accesses
one extra block access to retrieve the record from the data file
The total search cost via the index is: 9 + 1 = 10 block accesses

This is compared to an average linear search cost directly on the data file:
⌈b/2⌉ = ⌈10,000/2⌉ = 5,000 block accesses
Because the file records are ordered, the binary search cost would be:
⌈$\log_2 b$⌉ = ⌈$\log_2$10,000⌉ = 14 block accesses

15

Secondary indexes using implementation with option (3)

Example 4: given the file with the non-ordering non-key field DEPT_NUMBER
EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ..., DEPT_NUMBER)

record size R=150 bytes; block size B=512 bytes
number of records r=30,000 records
It is assumed that there are *125 distinct values of the DEPT_NUMBER field* and *even distribution* across DEPT_NUMBER values.

blocking factor bfr= B div R= ⌊B/R⌋ = 512 div 150= 3 records/block
number of file blocks b = ⌈r/bfr⌉= ⌈30,000/3⌉ = 10,000 blocks

For a secondary index on DEPT_NUMBER, the field size V_{DEPT_NUMBER}=4 bytes, assume the block pointer size P_B=6 bytes, the record pointer size P_R=7 bytes.

index entry size R_i=V_{DEPT_NUMBER}+ P_B=4+6=10 bytes
index blocking factor bfr_i= B div R_i= 512 div 10= 51 entries/block
number of index entries r_i = number of distinct values = 125 entries
number of index blocks b_i= ⌈r_i/ bfr_i⌉ = ⌈125/51⌉ = 3 blocks

index blocking factor at the indirection level bfr_{ii} = ⌊(B-P_B)/P_R⌋ = ⌊(512-6)/7⌋ = 72 pointers/block (*it is supposed that linked allocation is used at this level*)
number of index entries r_{ii} per distinct value at the indirection level = number of record pointers per distinct value of DEPT_NUMBER = ⌈30,000/125⌉ = 240 pointers
number of index blocks per distinct value at the indirection level b_{ii} = ⌈r_{ii}/bfr_{ii}⌉ = ⌈240/72⌉ = 4 blocks

Secondary indexes using implementation with option (3)

Example 4: given the file with the non-ordering non-key field DEPT_NUMBER
EMPLOYEE(NAME, SSN, ADDRESS, JOB, SAL, ..., DEPT_NUMBER)

record size R=150 bytes; block size B=512 bytes
number of records r=30,000 records
It is assumed that there are *125 distinct values of the DEPT_NUMBER field* and *even distribution* across DEPT_NUMBER values.

To retrieve the ***first record*** from the data file, given a DEPT_NUMBER value:
binary search on the index needs ⌈$\log_2 b_i$⌉ = ⌈$\log_2 3$⌉ = 2 block accesses
one extra block access to have access to the ***indirection level***
one extra block access to retrieve the ***first*** *record from the data file*
The total search cost via the index is: 2 + 1 + 1 = 4 block accesses

To retrieve ***all*** the records with the indirection level and *even distribution*, given a DEPT_NUMBER value:
binary search on the index needs ⌈$\log_2 b_i$⌉ = ⌈$\log_2 3$⌉ = 2 block accesses
number of block accesses to the blocks in the indirection level: 4 block accesses,
number of block accesses to the data file: ⌈30,000/125⌉ = 240 block accesses
The total cost to retrieve all the records = 2 + 4 + 240 = 246 block accesses

26