

Chapter 15

Indexed File Processing

Chapter Objectives

- To familiarize you with
- Methods of disk file organization
- Random processing of disk files
- How to create, update, and access indexed disk files

Disk File Organization

- File is collection of records
- Three major ways records stored or organized on disk
 - Sequential File Organization
 - Indexed File Organization
 - Relative File Organization

Sequential File Organization

- Records stored in order they are written to file
- Must be accessed in sequence - to access 50th record in file, must read past first 49
- Typically sorted into sequence by a key field

Indexed File Organization

- Consists of two files
 - Data file - records in sequence
 - Index file - contains value of
 - Each key field
 - Disk address of record with that corresponding key field
- For random access, look up key field in index file to find address
- Then access record in data file directly

Relative File Organization

- When records created, key field used to compute a disk address where record is written
- To randomly access records
 - User enters key field
 - Disk address computed from key field
 - Record then accessed directly
- No index needed

Creating an Indexed File

- Records written in sequence by key field as for sequential disk file
- Once index file created, records can be accessed randomly

SELECT Statement

Format

SELECT file-name-1

ASSIGN TO implementor-name-1

[ORGANIZATION IS] INDEXED

[ACCESS MODE IS SEQUENTIAL]

RECORD KEY IS data-name-1

- SELECT to create indexed file

SELECT Statement

- ORGANIZATION INDEXED
 - Indicates index file to be created along with data file
 - Index file must be established to be able to randomly access file later
- ACCESS MODE SEQUENTIAL
 - Records written in sequence by key field
 - Optional since SEQUENTIAL is default mode

SELECT Statement

- RECORD KEY clause
 - Names key field within disk record used to form index
 - Must be in same physical location in each record (usually first field)
 - Value must be unique for each record
 - Best to use numeric field as key

WRITE ... INVALID KEY

- INVALID KEY clause required when writing indexed records to handle I/O errors
 - Key field not in sequence
 - Key field same as one already in file
- If error detected with WRITE
 - Record not written
 - Statement(s) following INVALID KEY executed

WRITE ... INVALID KEY

Format

```
WRITE record-name-1 [FROM identifier-1]  
    [INVALID KEY imperative-statement-1]  
    [NOT INVALID KEY imperative-statement-2]  
[END-WRITE]
```

- Statement(s) following NOT INVALID KEY executed if WRITE is successful

Updating Index File Randomly

- Master records can be updated directly without creating a new file
- With index file, changes can be made in any sequence

Updating Index File Randomly

- Read transaction record (or get update data interactively)
- Move key field value to RECORD KEY of master file
- Read master record into storage (READ ... INVALID KEY)
- Make needed changes to fields
- REWRITE record to master file

Updating Index File Randomly

- In SELECT statement specify ACCESS MODE IS RANDOM
- Open indexed file for I-O
 - I (Input) to read in records
 - O (Output) to rewrite or update records

READ ... INVALID KEY

- To locate record with key field equal to value stored in record key

Move Trans-No To Master-No

Read Indexed-File

Invalid Key Perform 600-Err-Rtn

Not Invalid Key Perform 500-OK-Rtn

End-Read

REWRITE ... INVALID KEY

Format

```
REWRITE record-name-1 [FROM identifier-1]  
  [INVALID KEY imperative-statement-1]  
  [NOT INVALID KEY imperative-statement-2]  
[END-REWRITE]
```

- To update existing indexed record
- INVALID KEY occurs if programmer has changed key field of record

Add New Record

- To add new record to indexed file
 - Move data to master record fields
 - Use `WRITE ... INVALID KEY` to create new record

Delete Existing Record

Format

DELETE index-file-name-1 RECORD

[INVALID KEY imperative-statement-1]

[NOT INVALID KEY imperative-statement-2]

[END-DELETE]

- To delete record with key field equal to value stored in record key

Debugging Tips

- Must run program to create indexed file
 - Cannot be created using text editor
- To test an update program, always run index file creation program first
- May not be able to DISPLAY or print indexed records on your system directly
 - Move data to standard sequential record first

Printing Indexed File Sequentially

- Process file in same way as a sequential file
- Specify `ACCESS IS SEQUENTIAL`
- `SORT` file before printing if report is to be in sequence by field other than key field

Printing Indexed File Randomly

- May need to access only records about which inquiries have been made
- Specify `ACCESS IS RANDOM`

Printing Indexed File Randomly

- Read inquiry record
- Move inquiry key field value to RECORD KEY of master file
- Read master record into storage (READ ... INVALID KEY)
- Move desired fields to output record
- WRITE output record to print file

Random Interactive Inquiries

- Inquiries often made interactively with output displayed on screen
- Assume indexed accounts receivable file with customer's Acct-No as key field
- Interactive inquiries made about balance due for a customer
- Code using random access for one inquiry follows

Random Interactive Inquiries

Display 'Enter account number'

Accept Acct-No

Read Accts-Receiveable

Invalid Key

Display 'Account not on file'

Not Invalid Key

Display 'Balance due = ', Bal-Due

End-Read

ALTERNATE RECORD KEY

- Clause to enable file to be accessed randomly using more than one key field
 - May want to access accounts receivable records by account number or name
- Add to `SELECT` statement after `RECORD KEY` clause to establish multiple key fields for indexing

ALTERNATE RECORD KEY

Format

[ALTERNATE RECORD KEY IS

data-name-2 [WITH DUPLICATES]] ...

- Multiple ALTERNATE keys allowed
- Need not be unique
- Access records by RECORD KEY or any ALTERNATE RECORD KEYS

SELECT Example

Select Accts-Receiveable
Assign To 'C:\AcctRec.ndx'
Organization Is Indexed
Access Is Sequential
Record Key Is Acct-No
Alternate Record Key
Is Cst-Last-Name
With Duplicates.

Random Access by ALTERNATE

- To access file by Cst-Last-Name

Display 'Enter last name'

Accept Cst-Last-Name

Read Accts-Receiveable

Key is Cst-Last-Name

Invalid Key Display 'No record found'

Not Invalid Key

Display 'Balance due = ', Bal-Due

End-Read

Random Access by ALTERNATE

- KEY clause used with READ to specify access by ALTERNATE RECORD KEY
- Since WITH DUPLICATES specified
 - May be more than one record with same last name
 - Record retrieved is first one placed on disk
- If KEY clause omitted, uses RECORD KEY field (Acct-No) to find record

START Statement

- To begin processing indexed file sequentially starting from any record location
 - Print file beginning with customer record with Acct-No = 025
 - Print all customers with Cst-Last-Name beginning with letter 'S'

START Statement

Format

START file-name-1

$\left(\begin{array}{l} \text{KEY} \\ \text{IS } = \\ \text{IS } > \\ \text{IS } \underline{\text{NOT}} < \\ \text{IS } \geq \end{array} \right) \text{ data-name-1}$

[INVALID KEY imperative-statement-1]

[NOT INVALID KEY

imperative-statement-2]

[END-START]

START Statement

- To begin processing with record whose account number equals 025

Move 025 To Acct-No

Start Accts-Receiveable

Invalid Key

Display 'Acct-No 025 not found'

Not Invalid Key

Perform 300-Proc-Rec

End-Start

START Statement

- START locates record with Acct-No = 025
- INVALID KEY clause executed only if no such record found
- START locates record but does not READ it
- 300-Proc-Rec must include READ ... AT END to bring record into storage for processing

START Statement

- KEY clause can be omitted only if checking for value equal to RECORD KEY value
- To locate record with Acct-No > 100:
Move 100 To Acct-No
Start Accts-Receiveable
Key > Acct-No
Invalid Key ...
...

ACCESS IS DYNAMIC

- Mode used to access indexed file both randomly and sequentially in single program
- For example, update selected records, then print control listing of entire indexed file
 - Random access used for updating
 - Sequential access used for printing report

ACCESS IS DYNAMIC

- Mode required for reading records in sequence by ALTERNATE RECORD KEY
- Also required when records accessed by both RECORD KEY and ALTERNATE RECORD KEY

READ ... NEXT RECORD

- To perform sequential read of indexed file when ACCESS MODE IS DYNAMIC
- To sequentially read from file by its ALTERNATE RECORD KEY
- To begin reading sequentially from some point other than beginning of file

READ ... NEXT RECORD

- Assume first record with Acct-No > 100 has been located using START
- Use READ ... NEXT RECORD to read in records sequentially from this position
- Only word NEXT required

READ ... NEXT RECORD

Perform Until More-Records = 'NO'

Read Accts-Receiveable Next Record

At End

Move 'NO' To More-Records

Not At End

Perform 300-Proc-Rec

End-Read

End-Perform

FILE STATUS Clause

- To determine exact type of input or output error that occurred when accessing a file
- Included in SELECT statement for a file as last clause

- Forma

^t
SELECT ...

[FILE STATUS IS data-name]

FILE STATUS Clause

- Data-name must appear in WORKING-STORAGE as two-position alphanumeric field

- Exam

Select Indexed-Pay-File ...

File Status Is WS-Status.

...

Working-Storage Section.

01 WS-Status

Pic X(2).

FILE STATUS Clause

- When input or output operation performed on Indexed-Pay-File
 - Value placed in WS-Status
 - Can be tested by programmer in PROCEDURE DIVISION
- Several FILE STATUS field values and their meaning follow

FILE STATUS Values

File Status

field value

Meaning

00

No error occurred

21

Sequence error - keys not
in correct order

22

Attempt to write record
creating duplicate primary
record key

Checking FILE STATUS

Write Indexed-Pay-Rec

Invalid Key

 If WS-Status = '21'

 Display 'Key not in sequence'

 End-If

 ...

Not Invalid Key

 Perform 600-OK-Rtn

End-Write

Exception Handling

- Most comprehensive method for handling input/output errors is to establish separate section(s) for this
- Place exception handling routines in **DECLARATIVES** segment
 - Always appears first in **PROCEDURE DIVISION**
 - Must begin with section-name

DECLARATIVES Format

DECLARATIVES.

section-name SECTION.

USE AFTER STANDARD

{ EXCEPTION } PROCEDURE
{ ERROR }

ON file-name-1 ...

END DECLARATIVES.

DECLARATIVES Example

Procedure Division.

Declaratives.

A000-Exception-Handling Section.

 Use After Error Procedure
 On Indexed-Pay-File

A100-Check-It.

 If WS-Status = '21'

 Display 'Key not in sequence'

 End-If ...

End Declaratives.

DECLARATIVES Example

B000-Regular-Processing Section.

B100-Main-Paragraph.

...

Read Indexed-Pay-File

...

Write Indexed-Pay-File

...

DECLARATIVES Example

- Once section header used, rest of PROCEDURE DIVISION must be divided into sections
- Statements in paragraph A100-Check-It in DECLARATIVES test value of FILE STATUS field
- INVALID KEY not needed for READ or WRITE since errors handled in DECLARATIVES

Chapter Summary

- Indexed Files - SELECT clauses
 - ORGANIZATION IS INDEXED
 - ACCESS IS RANDOM
 - For nonsequential updates, inquiries
 - ACCESS IS SEQUENTIAL
 - For creating, reporting, updating sequentially

Chapter Summary

- Indexed Files - SELECT clauses
 - RECORD KEY - Key field for establishing index, accessing records
 - FILE STATUS IS data-name for indicating success of input or output operation

Chapter Summary

- Indexed Files - PROCEDURE DIVISION
 - Creating indexed file
 - ACCESS IS SEQUENTIAL
 - Use READ ... AT END
 - Reading from indexed file
 - In sequence - READ ... AT END
 - Randomly - READ ... INVALID KEY