Chapter 7 Indexing

Simple Index – Single Keys

Multiple Keys

Secondary Keys

Inverted Lists

Classes

Index – A tool for finding records in a file.

It consists of a ***key field*** on which the index is searched and a ***reference field*** that tells where to find the data file record associated with a particular *key*

Key	Reference

The *key* is usually in canonical form (i.e. uppercase, lowercase, #of digits)

The *reference field* indicates the location of the data record in either ***RRN*** or ***Offset*** format.

ENTRY – SEQUENCED FILES

Records occur in the order they are entered into the file

The index is sorted in key order.

EXAMPLE: Record Albums or CDs

Data records contain:

Identification Number

Title

Composer or Composers

Artist or Artists

Label (publisher)

The *Primary Key* will be the Label concatenated with the label companies ID number

The canonical form of the Key is Uppercase Label and any characters in the ID number will also be uppercase.

Index		Address of Record	Recording File
Key	Reference Field		Actual Data Record
ANG3795	152	17	LON | 2312 | Romeo and Juliet | Prokofiev | …
COL31809	338	62	RCA | 2626 | Quartet in C Sharp Minor | Beethoven | …
COL38358	196	117	WAR | 23699 | Touchstone | Corea | …
DG139201	382	152	ANG | 3795 | Symphony No. 9 | Beethoven | …
DG18807	241	196	COL | 38358 | Nebraska | Springsteen | …
FF245	427	241	DG | 18807 | Symphony No. 9 | Beethoven | …
LON2312	17	285	MER | 75016 | Coq d'Or Suite | Rimsky-Korsakov | …
MER75016	285	338	COL | 31809 | Symphony No. 9 | Dvorak | …
RCA2626	62	382	DG | 139201 | Violin Concerto | Beethoven | …
WAR23699	117	427	FF | 245 | Good News | Sweet Honey in the Rock | …

Operations Required to Maintain the Indexed Files

- Create the original empty index and data files
- Load the index file into memory before using the index file
- Rewrite the index file from memory after using the index file
- Add data records to the data file
- Delete records from the data file
- Update records in the data file, and
- Update the index to reflect changes in the data file
- Create an index for a file

Creating the Files:

Two files are needed: One for the Index, and one for the data.

Use the *create* method from the BufferFile class.

Loading the Index into Memory:

Use the IOBuffer class. This will be a fixed field buffer.

Rewriting the Index File from Memory:

Use the *Rewind* and *Write* operations of class BufferFile.

The entire Index must fit in a single record.

What happens if the program should fail before the index is written?

You need to store an indication that the index is out of date.

If the program detects an out of date index, the program must rebuild it.

Record Addition

- Requires addition of a record to the index as well as to the data file.
- Data file records can be added as discussed in the previous chapter.
- Index record must be put in the correct spot so we can use the binary search.

    We must move the entries out of the way. This isn't so bad because the records are in memory, so no file access is required.

Record Deletion

- requires removal of the record in the index, as well as removal of the data file.
- Again, we can use the techniques for deletion from the previous chapter.
- For the index, we don't have to move the records. Just mark the entry as unused, and don't copy it when we put the index back into the file upon completion.

Record Updating: there are two categories:

- The update changes the value of the key field
    - The index may need to be reordered
    - It is essentially a delete followed by insert
- The update does not affect the key field
    - If the records are fixed size, nothing is required.
    - If the records are of variable length, then the new record most likely won't be placed in the same location. So, a new location needs to be added to the Index
    - Delete followed by Insert can be used to appropriately place the changed record for variable length records