Chapter 7 Secondary Indexes

Simple Index – Single Keys

Multiple Keys

Secondary Keys

Inverted Lists

Classes

What happens if the index doesn't fit completely into memory?

We can't do a binary search because it unfeasibly requires seeking.

Index rearrangement from insertions and deletions is infeasible too.

The Solution

Use hashed organization (later)

Use a tree-structured index B-tree (later)

Multiple Keys

What happens if we want to use something other than the primary key to access a record?

We build *secondary* indices to access the records.

How do we handle the fact that a given key won't refer to a single record?

Composer Index

Secondary Key	Primary Key
BEETHOVEN	ANG3795
BEETHOVEN	DG139201
BEETHOVEN	DG18807
BEETHOVEN	RCA2626
COREA	WAR23699
DVORAK	COL31809
PROKOFIEV	LON2312
RIMSKY-KORSIKOV	MER75016
SPRINGSTEEN	COL38358
SWEET HONEY IN THE R	FF245

Notice that the Secondary Key leads to the Primary Key not the address in the file of the record.

This is called Entry Sequenced.

After you consult the secondary index, you must then consult the primary index to locate a record in the file.

Use of this technique makes deletion, updating, and addition easier.

The Secondary Key is in canonical form and of a fixed size.

Search algorithms for the Index must know the canonical form with all its restrictions.

Title Index

Secondary Key	Primary Key
COQ D'OR SUITE	MER75016
GOOD NEWS	FF245
NEBRASKA	COL38358
QUARTET IN C SHARP M	RCA2626
ROMEO AND JULIET	LON2312
SYMPHONY NO. 9	ANG3795
SYMPHONY NO. 9	COL31809
SYMPHONY NO. 9	DG18807
TOUCHSTONE	WAR23699
VIOLIN CONCERTO	DG139201

Index			Recording File
Key	Reference Field	Address of Record	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 | Drorak | …
RCA2626	62	382	DG | 139201 | Violin Concerto | Beethoven | …
WAR23699	117	427	FF | 245 | Good News | Sweet Honey in the Rock | …

Operations That Require Maintenance of the Secondary Index

- Add data records to the data file
- Delete records from the data file
- Update records in the data file

Record Addition

- requires addition of a entry to the secondary index as well as to the primary index.
- the index must be rearranged following the ordering of the secondary index.
- this is works well if the index can be read into memory prior to updating.
- the duplicate keys must remain in order by the value of the Primary Key.

Record Deletion

- Usually requires removing all references to that record in the file system.
- We can leave the secondary index alone since the reference to the primary index informs us that the record is deleted.
- We can periodically rebuild the secondary index files.
- In a highly volatile file, the secondary indices should be kept in a B-tree. (later)

Record Updating - this manifests in three categories:

- The update changes the secondary key
    - The index may need to be reordered.
    - The process is essentially record deletion followed by record insertion.
- The update changes the primary key
    - Requires updating the primary key in all secondary indices.
    - Does not require reordering the secondary indices unless the corresponding secondary key occurs more than once in the index.
    - Reordering the secondary keys in this instance is only a local reordering since the number of records with this secondary key remains the same.
- The update is for a field not associated with any of the keys
    - No changes to the secondary index are required

Using Combinations of Secondary Keys

Find all recordings of Beethoven's Symphony No. 9

Find all entries for Beethoven

ANG3795
DG139201
DG18807
RCA2626

Find all entries for Symphony No. 9

ANG3795
COL31809
DG18807

Now find the intersection of the lists (Chapter 8)

ANG3795
DG18807

Now we can use this list to get the records for the list.

Inverted Lists:

- Can we avoid rearranging the index every time we add a record?
- Can we avoid repeating the secondary key when there is more than on record with the given secondary key?

Make each Secondary Key reference a linked list of primary key references.

Secondary Index File

BEETHOVEN	3
COREA	2
DVORAK	7
PROKOFIEV	10
RIMSKY-KORSIKOV	6
SPRINGSTEEN	4
SWEET HONEY IN THE R	9

Label ID List file

0	LON2312	-1
1	RCA2626	-1
2	WAR23699	-1
3	ANG3795	8
4	COL38358	-1
5	DG18807	1
6	MER75016	-1
7	COL31809	-1
8	DG139201	5
9	FF245	-1
10	ANG36193	0

- The only time we need to rearrange the Secondary Index is when a new composer is added to the list.
- Deleting and Adding records otherwise only requires adding to the linked list.
- Removing all records for a given Secondary Index entry means removing the entries from the Label ID file. You don't need to remove the Secondary Index entry.
- Those fewer records in the Secondary Index are easier to rearrange.
- This makes keeping the Secondary Index easier.
- The Label ID file never needs to be sorted since its order is defined by the linked list.
- Since the Label ID records are fixed length adding and deleting entries is easy.

- There can be problems if the index doesn't fit in memory because of the seek time problem

Selective Indexes

You can create temporary secondary indexes for special uses.

- Courses for a particular term
- Students who entered during a specified term

Binding: When do we bind a particular key to a physical address of its associated record?

Answer: the question of when a Key will be bound must be dealt with *intentionally* and *early* in the design process.

- Primary Keys are usually bound at the time the file is created
  - termed Binding Tightly
  - Eliminates the need to search on the primary key
  - Requires reconstruction of all indexes when the data is reorganized
- Secondary Keys are bound when the key is used i.e. when the record is retrieved
- You can bind Primary Keys at retrieval time.
- Then, rearranging the file only requires changing the primary key index file.
- Now record retrieval only needs one more reference.
- Of course if the reference means a *seek,* this can take time but this time is traded for the time not needed to change all the indexes.