جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الامن السيبراني

Lecture: (10)

Subject: Database
Level: Second
Lecturer: Asst. Lecturer Qusai AL-Durrah

Indexing and Hashing

6 Indexing and Hashing

Many queries reference only a small proportion of the records in a file. For example , the queries " Find all accounts at the Tech branch" reference only a fraction of the account records. It is inefficient for the system to have to read every record and to check the branch names. The system should be able to locate these records directly. To allow that, we design additional structures with the files.

An index for a file in the system works like a catalog for a book in a library, if we are looking for a book , the catalog of the name of the books tells us where to find the book.

To assist us searching the catalog, the names in the catalog listed in an alphabetic order.

There are two basic kinds of indices :

Ordered indices: such indices are based on a sorted ordering of the values.

Hash index : such indices are based on some values , these values calculated by a function called hash function.

We often want to have more than one index for the file. Return to the example of the library, there can be a catalog for the names of the books and another catalog for the others of the books and third one for the subjects of the books.

Indexing and Hashing

6 Indexing and Hashing

Many **queries** reference only a small proportion of the records in a file. For example , the queries " Find all accounts at the Tech branch" reference only a fraction of the account records. It is inefficient for the system to have to read every record and to check the branch names. The system should be able to locate these records directly. To allow that, we design additional structures with the files.

An index for a file in the system works like a catalog for a book in a library, if we are looking for a book , the catalog of the name of the books tells us where to find the book.

To assist us searching the catalog, the names in the catalog listed in an alphabetic order.

There are two basic kinds of indices :

Ordered indices: such indices are based on a sorted ordering of the values.

Hash index : such indices are based on some values , these values calculated by a function called hash function.

We often want to have more than one index for the file. Return to the example of the library, there can be a catalog for the names of the books and another catalog for the others of the books and third one for the subjects of the books.

Indexing and Hashing

6 Indexing and Hashing

Many queries reference only a small proportion of the records in a file. For example , the queries " Find all accounts at the Tech branch" reference only a fraction of the account records. It is inefficient for the system to have to read every record and to check the branch names. The system should be able to locate these records directly. To allow that, we design additional structures with the files.

An index for a file in the system works like a catalog for a book in a library, if we are looking for a book , the catalog of the name of the books tells us where to find the book.

To assist us searching the catalog, the names in the catalog listed in an alphabetic order.

There are two basic kinds of indices :

Ordered indices: such indices are based on a sorted ordering of the values.

Hash index : such indices are based on some values , these values calculated by a function called hash function.

We often want to have more than one index for the file. Return to the example of the library, there can be a catalog for the names of the books and another catalog for the others of the books and third one for the subjects of the books.

Indexing and Hashing

6 Indexing and Hashing

Many queries reference only a small proportion of the records in a file. For example , the queries " Find all accounts at the Tech branch" reference only a fraction of the account records. It is inefficient for the system to have to read every record and to check the branch names. The system should be able to locate these records directly. To allow that, we design additional structures with the files.

An index for a file in the system works like a catalog for a book in a library, if we are looking for a book , the catalog of the name of the books tells us where to find the book.

To assist us searching the catalog, the names in the catalog listed in an alphabetic order.

There are two basic kinds of indices :

Ordered indices: such indices are based on a sorted ordering of the values.

Hash index : such indices are based on some values , these values calculated by a function called hash function.

We often want to have more than one index for the file. Return to the example of the library, there can be a catalog for the names of the books and another catalog for the others of the books and third one for the subjects of the books.

Indexing and Hashing

6.1 Ordered Indices :

These are used to gain fast random access to records in a file. Each index structure is associated with a particular ***search key***. The index stores the values of the search keys in sorted order.

The record in the indexed file may themselves be sorted in some way. The file may have several indices of different search key.

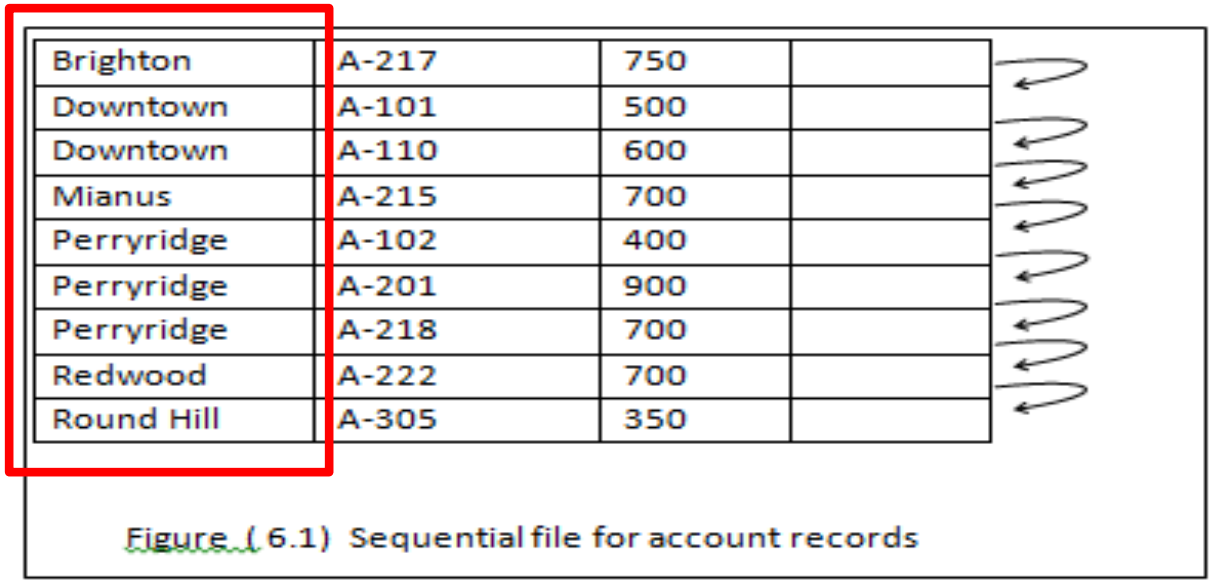
Primary index : if the file containing the record is sequentially ordered , the index whose search key specifies the sequential order of the file , this index is a primary index for that file.

Secondary index : is the index of the file whose search key specifies an order different from the order of the file.

Indexing and Hashing

6.1.1 : Ordered Primary Index

Figure 6.1 shows an ordered file for ***account*** records.



Brighton	A-217	750	
Downtown	A-101	500	
Downtown	A-110	600	
Mianus	A-215	700	
Perryridge	A-102	400	
Perryridge	A-201	900	
Perryridge	A-218	700	
Redwood	A-222	700	
Round Hill	A-305	350	

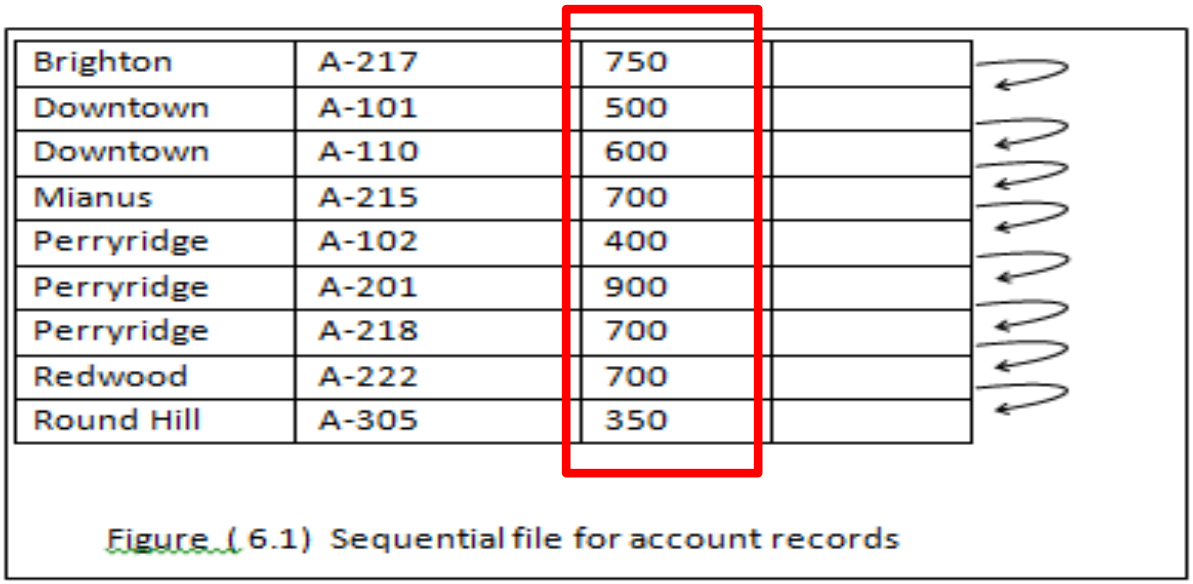
Figure (6.1) Sequential file for account records

The file of figure 6.1 is sorted on a search key order, with branch name is used as the search key.

Indexing and Hashing

6.1.1 : Ordered Primary Index

Figure 6.1 shows an ordered file for ***account*** records.



Brighton	A-217	750	
Downtown	A-101	500	
Downtown	A-110	600	
Mianus	A-215	700	
Perryridge	A-102	400	
Perryridge	A-201	900	
Perryridge	A-218	700	
Redwood	A-222	700	
Round Hill	A-305	350	

Figure (6.1) Sequential file for account records

The file of figure 6.1 is sorted on a search key order, with branch name is used as the search key.

Indexing and Hashing

6.1.1.1 : Indices Types : There are two types of ordered indices : ***Dense*** and ***Sparse*** indices.

Dense index : an index entry appears for every search key value in the file. The index record contains the search key value and a pointer to the first data record with that search key value, as shown in figure (6.2) for the account file.

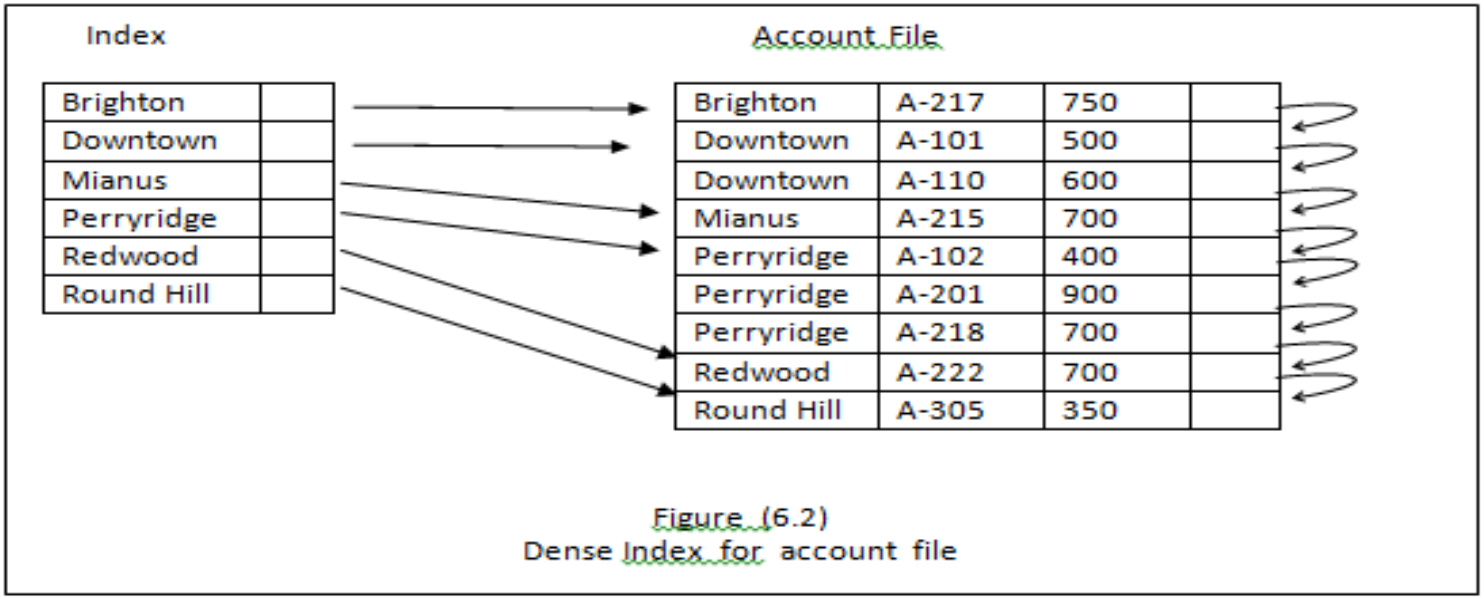


Figure (6.2)
Dense Index for account file

Indexing and Hashing

6.1.1.1 : Indices Types : There are two types of ordered indices : ***Dense*** and ***Sparse*** indices.

Dense index : an index entry appears for every search key value in the file. The index record contains the search key value and a pointer to the first data record with that search key value, as shown in figure (6.2) for the account file.

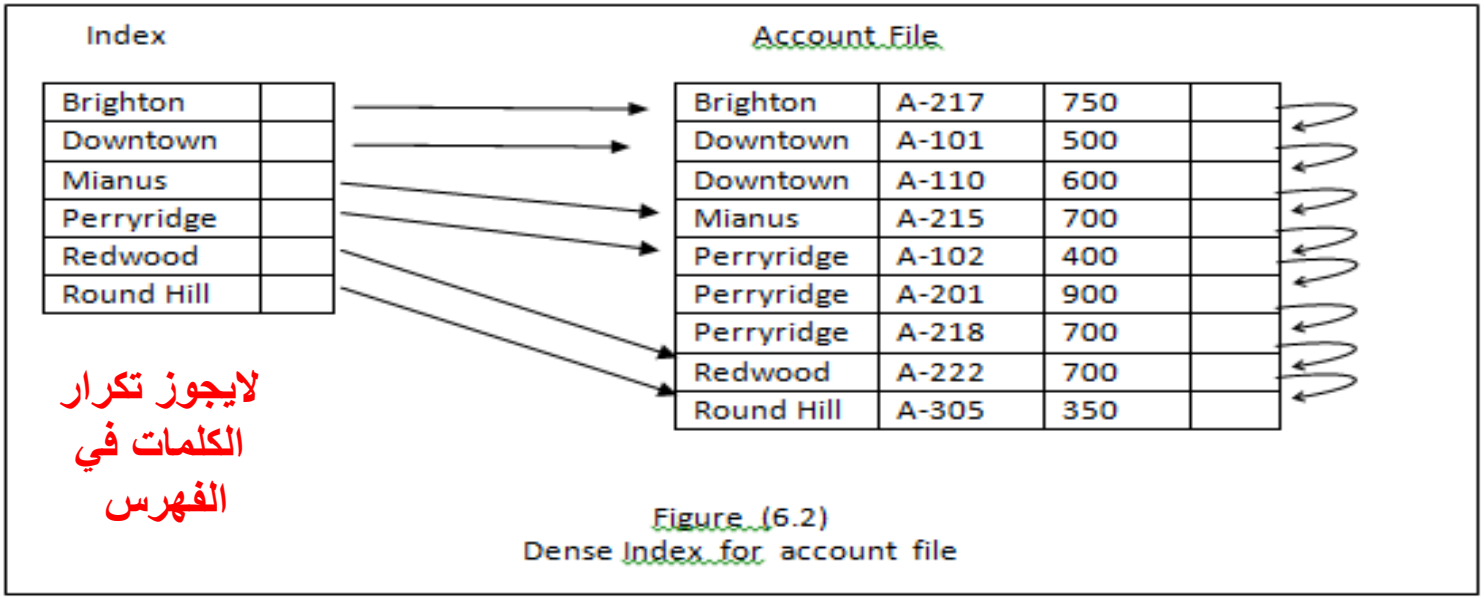


Figure (6.2)
Dense Index for account file

Indexing and Hashing

Table

Doctor name	Age	Specialism
David	32	Tooth
David	30	Tooth
John	34	Bone
John	29	Stomach
Mary	37	Stomach
Zain	25	Bone

Indexing and Hashing

Ordered **Dense Primary index**

Index

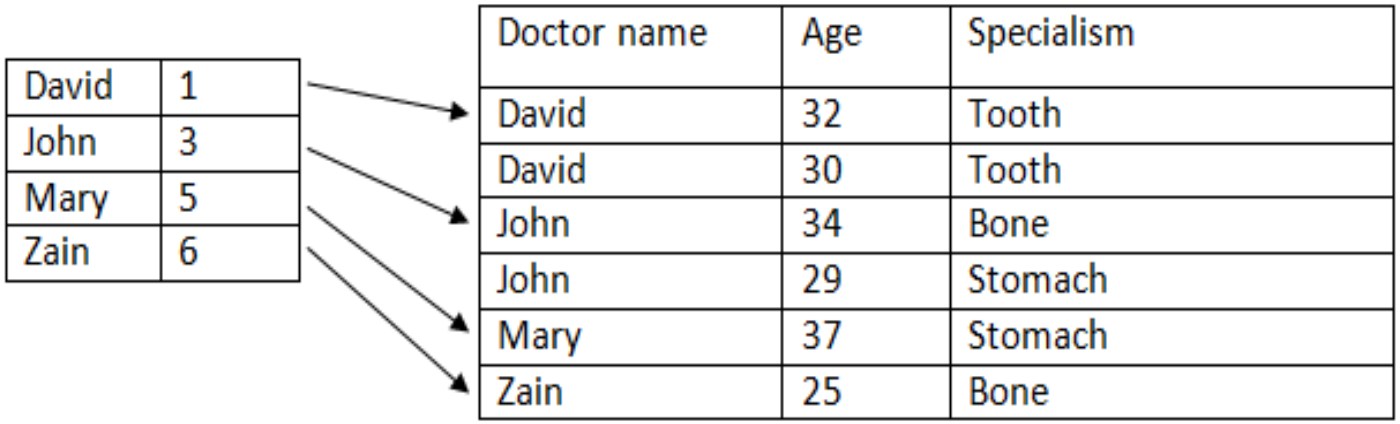
David	1

Table

Doctor name	Age	Specialism
David	32	Tooth
David	30	Tooth
John	34	Bone
John	29	Stomach
Mary	37	Stomach
Zain	25	Bone

Indexing and Hashing

Ordered **Dense Primary index**

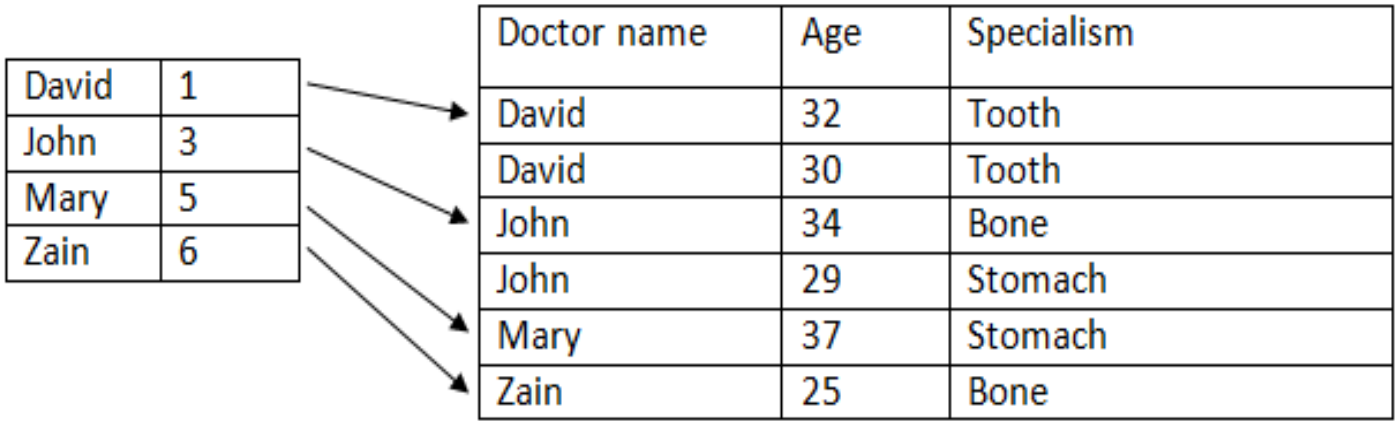


David	1
John	3
Mary	5
Zain	6

Doctor name	Age	Specialism
David	32	Tooth
David	30	Tooth
John	34	Bone
John	29	Stomach
Mary	37	Stomach
Zain	25	Bone

Indexing and Hashing

Ordered **Dense Primary index**



David	1
John	3
Mary	5
Zain	6

Doctor name	Age	Specialism
David	32	Tooth
David	30	Tooth
John	34	Bone
John	29	Stomach
Mary	37	Stomach
Zain	25	Bone

John

Indexing and Hashing

6.1.1.1 : Indices Types : There are two types of ordered indices : ***Dense*** and ***Sparse*** indices.

Dense index : an index entry appears for every search key value in the file. The index record contains the search key value and a pointer to the first data record with that search key value, as shown in figure (6.2) for the account file.

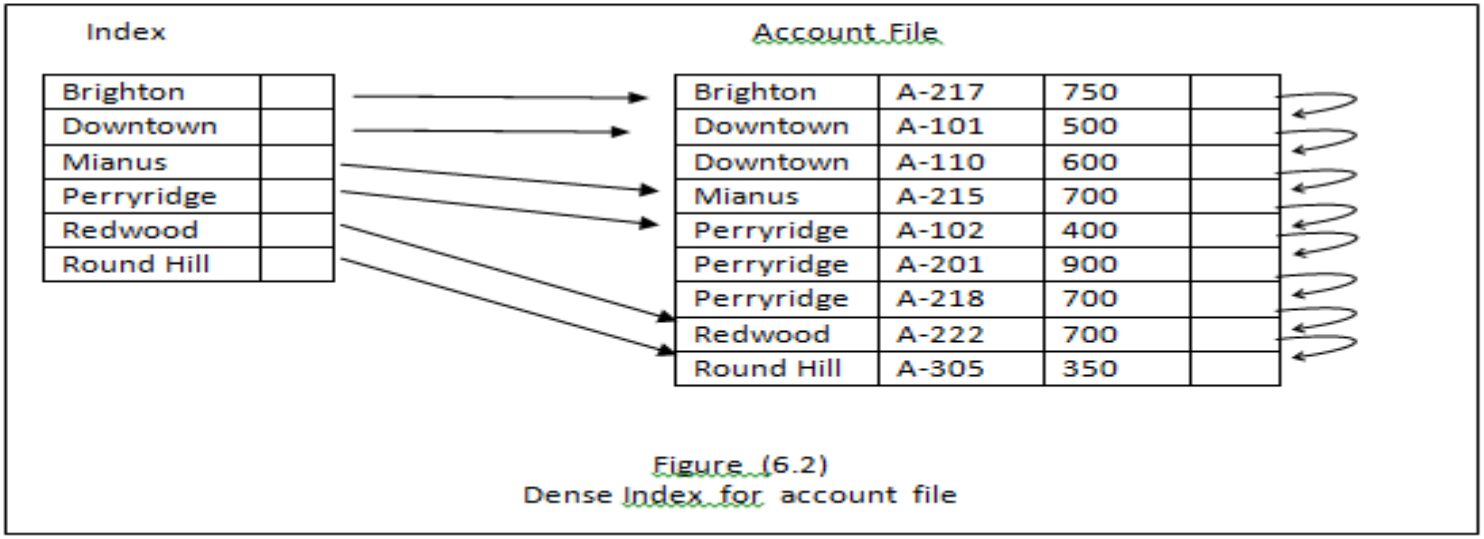


Figure (6.2)
Dense Index for account file

Perryridge

Indexing and Hashing

6.1.1.1 : Indices Types : There are two types of ordered indices : ***Dense*** and ***Sparse*** indices.

Dense index : an index entry appears for every search key value in the file. The index record contains the search key value and a pointer to the first data record with that search key value, as shown in figure (6.2) for the account file.

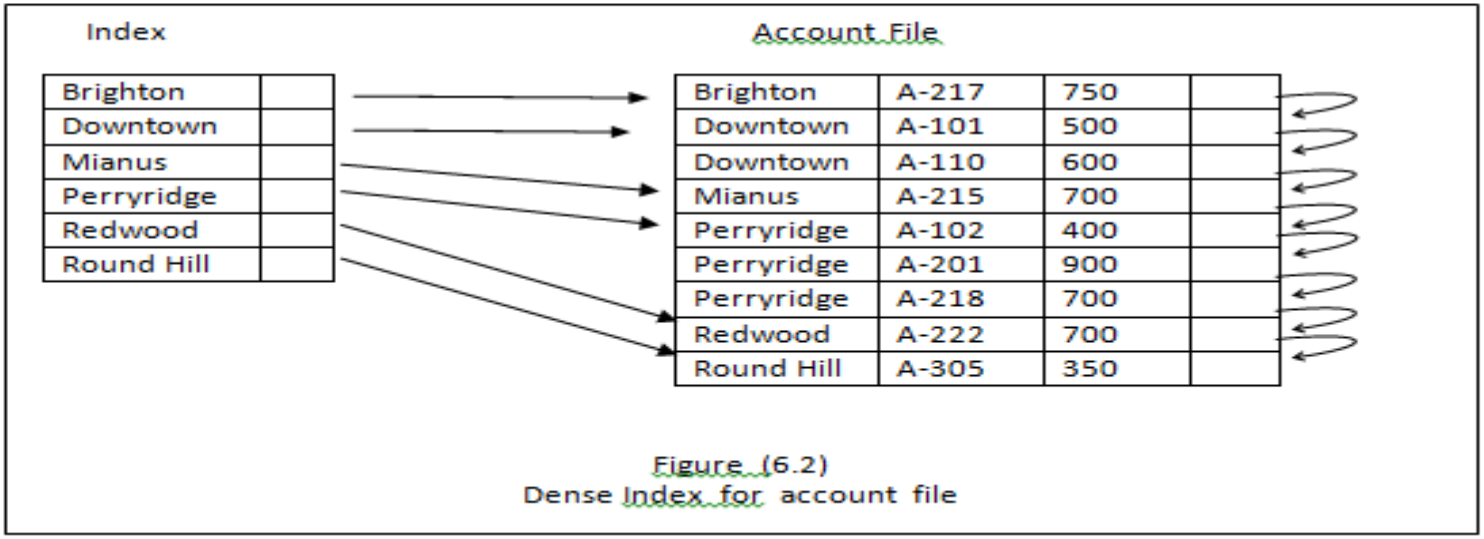


Figure (6.2)
Dense Index for account file

New York

Indexing and Hashing

Spars index : An index record is created for only some of the values. To locate a record we find the index entry with the largest search key value that is less than or equal to the search key value for which we are locking . We start at the record pointed to by that index entry, and follow the pointer in the file until we find the desired record. As shown in figure (6.3) for the account file.

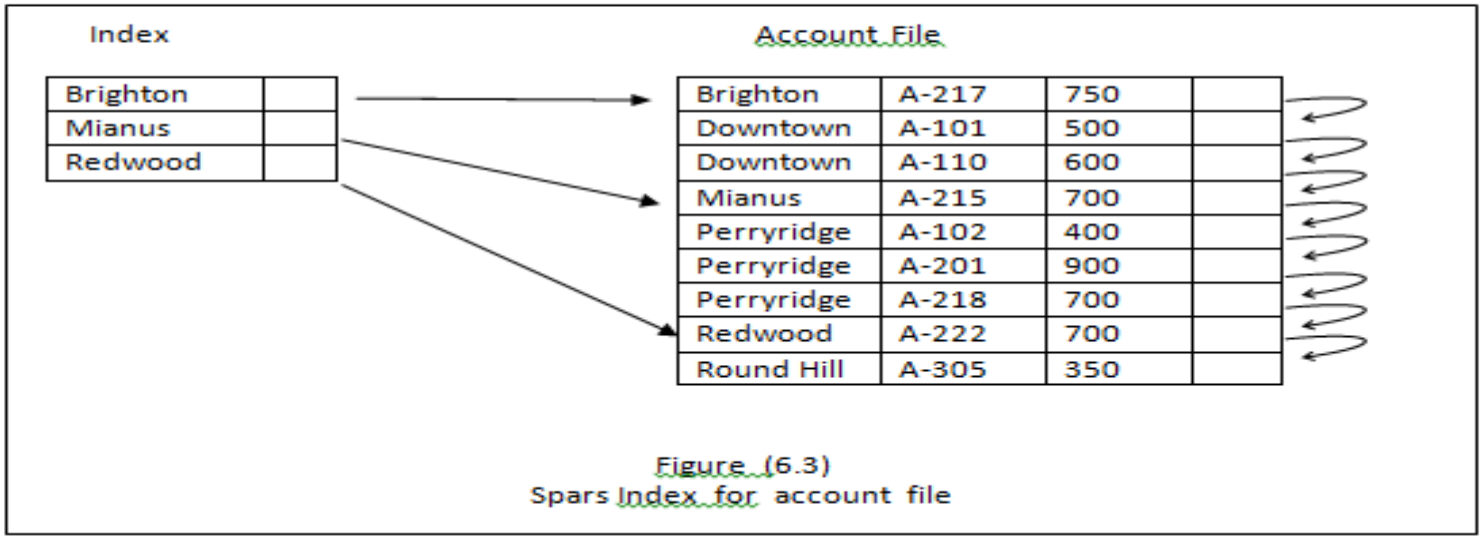


Figure (6.3)
Spars Index for account file

Indexing and Hashing

Spars index : An index record is created for only some of the values. To locate a record we find the index entry with the largest search key value that is less than or equal to the search key value for which we are locking . We start at the record pointed to by that index entry, and follow the pointer in the file until we find the desired record. As shown in figure (6.3) for the account file.

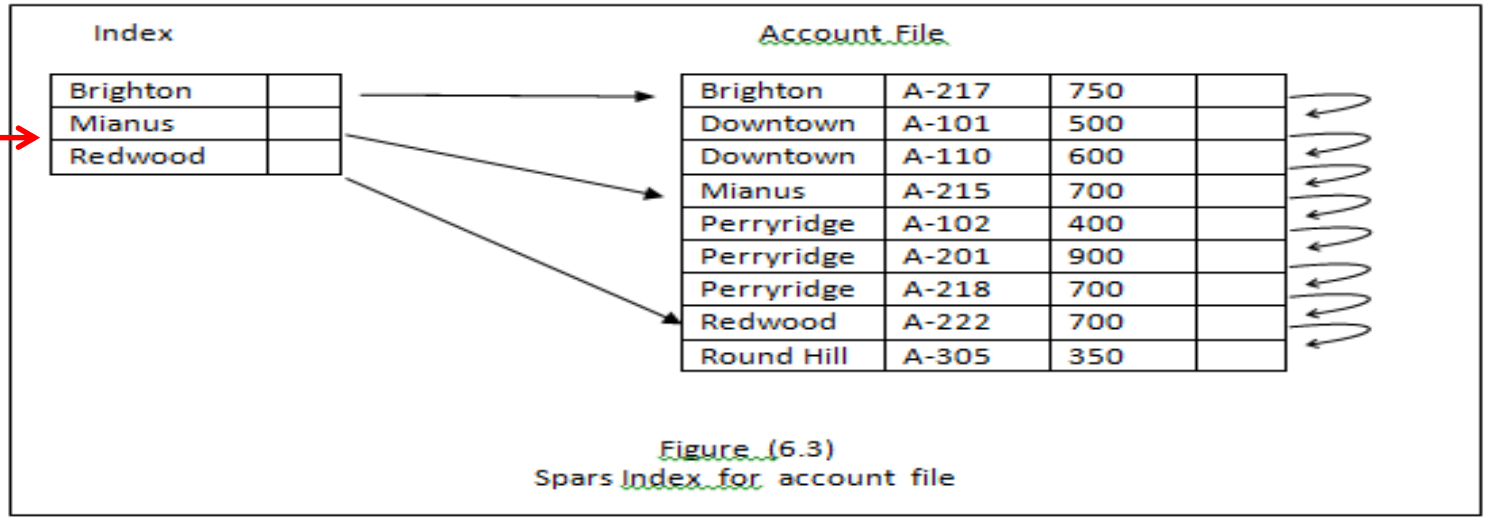


Figure (6.3)
Spars Index for account file

Perryridge

Indexing and Hashing

Spars index : An index record is created for only some of the values. To locate a record we find the index entry with the largest search key value that is less than or equal to the search key value for which we are locking . We start at the record pointed to by that index entry, and follow the pointer in the file until we find the desired record. As shown in figure (6.3) for the account file.

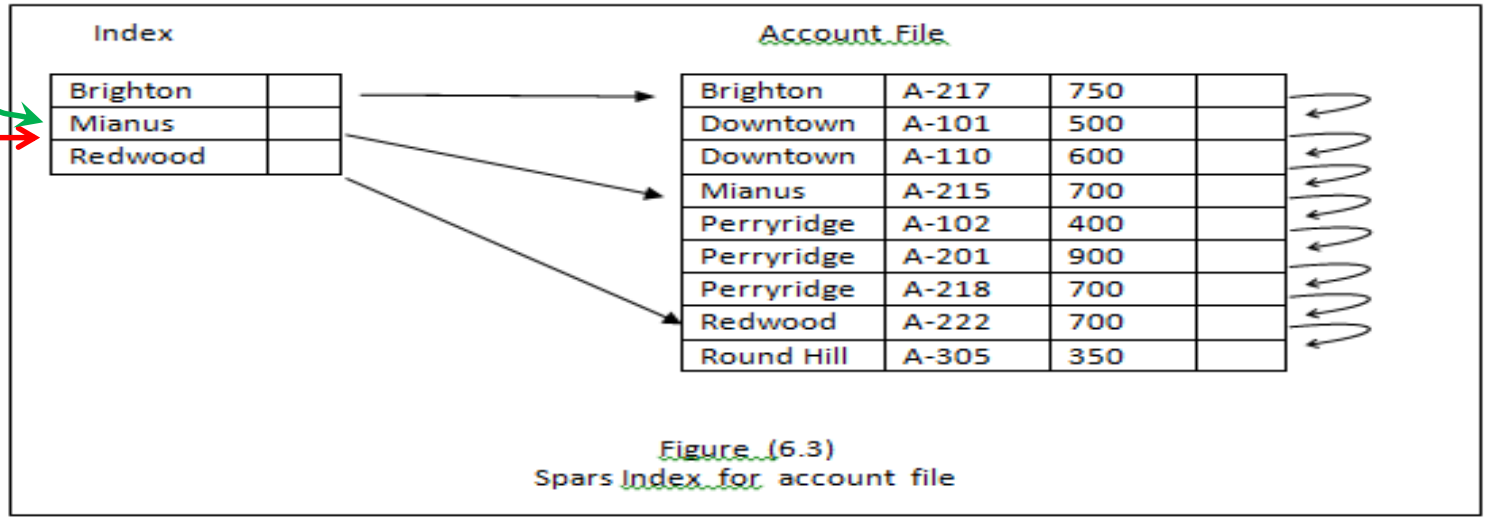


Figure (6.3)
Spars Index for account file

Perryridge

Indexing and Hashing

Spars index : An index record is created for only some of the values. To locate a record we find the index entry with the largest search key value that is less than or equal to the search key value for which we are locking . We start at the record pointed to by that index entry, and follow the pointer in the file until we find the desired record. As shown in figure (6.3) for the account file.

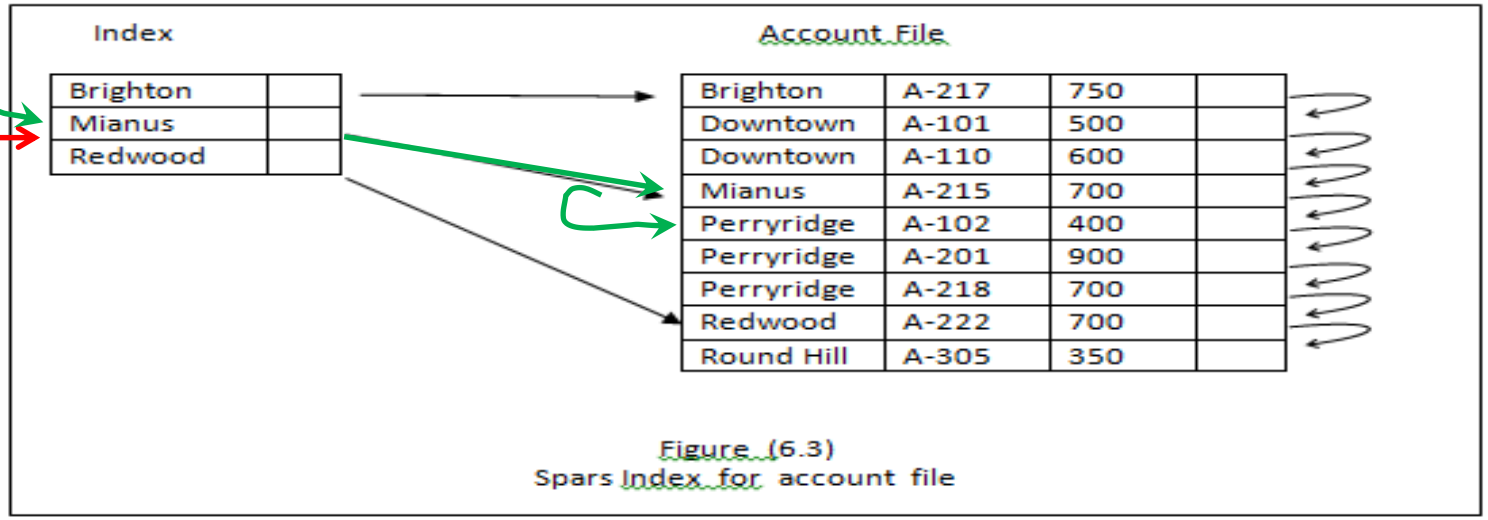


Figure (6.3)
Spars Index for account file

Perryridge

Indexing and Hashing

Spars index : An index record is created for only some of the values. To locate a record we find the index entry with the largest search key value that is less than or equal to the search key value for which we are locking . We start at the record pointed to by that index entry, and follow the pointer in the file until we find the desired record. As shown in figure (6.3) for the account file.

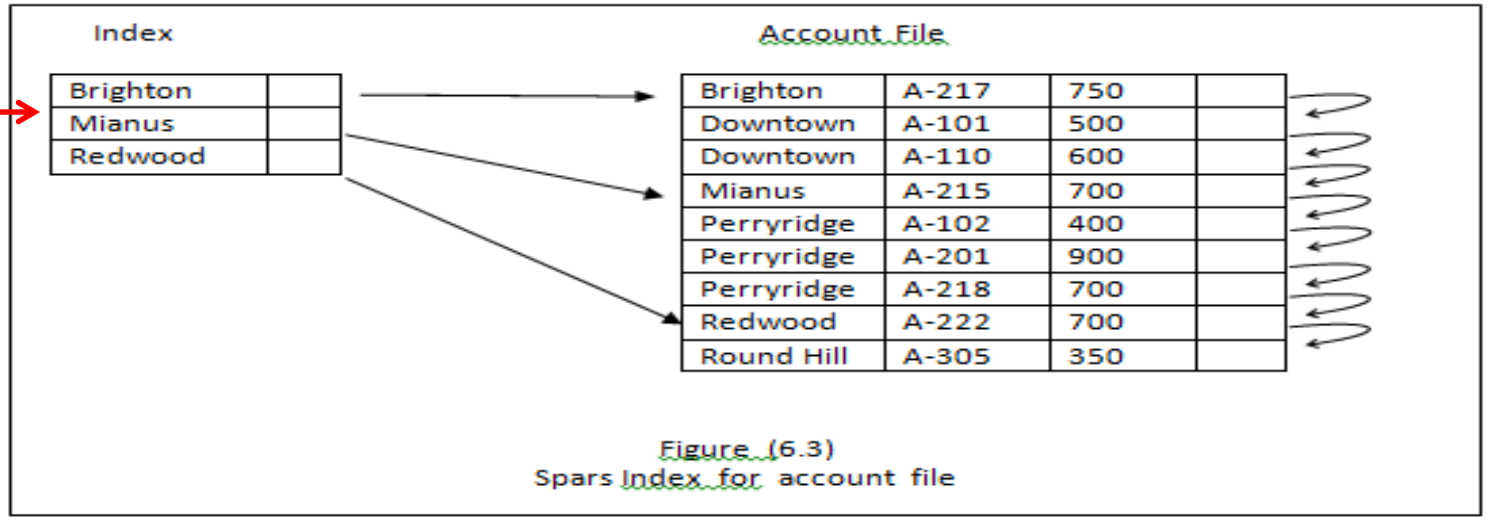
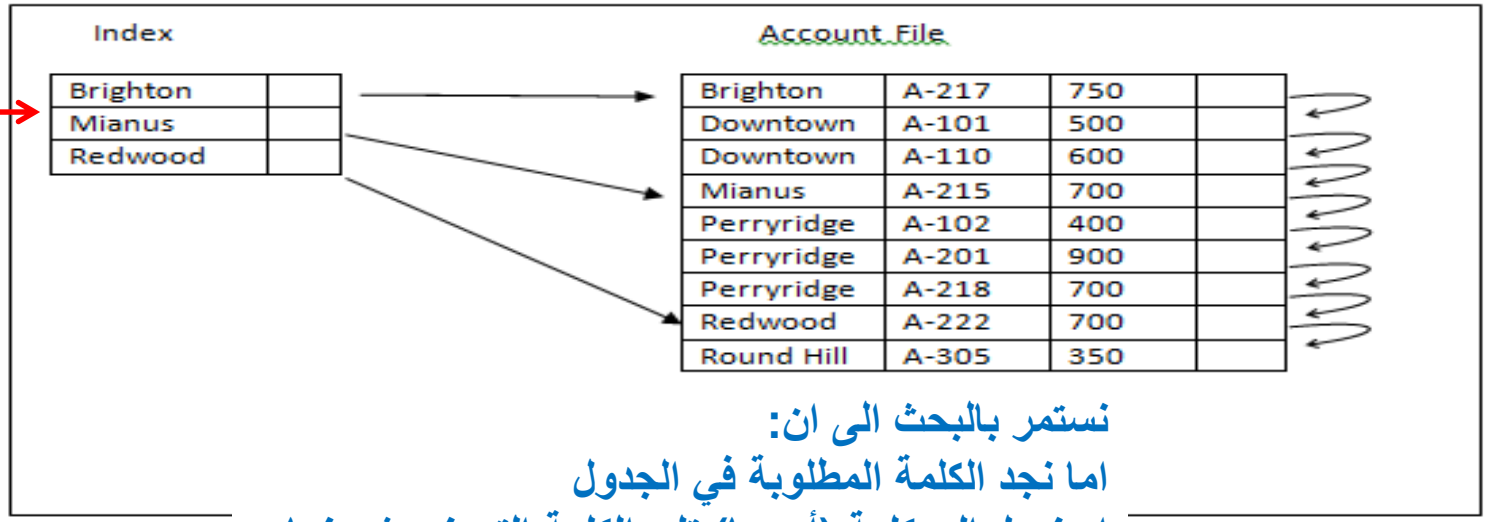


Figure (6.3)
Spars Index for account file

Hollywood

Indexing and Hashing

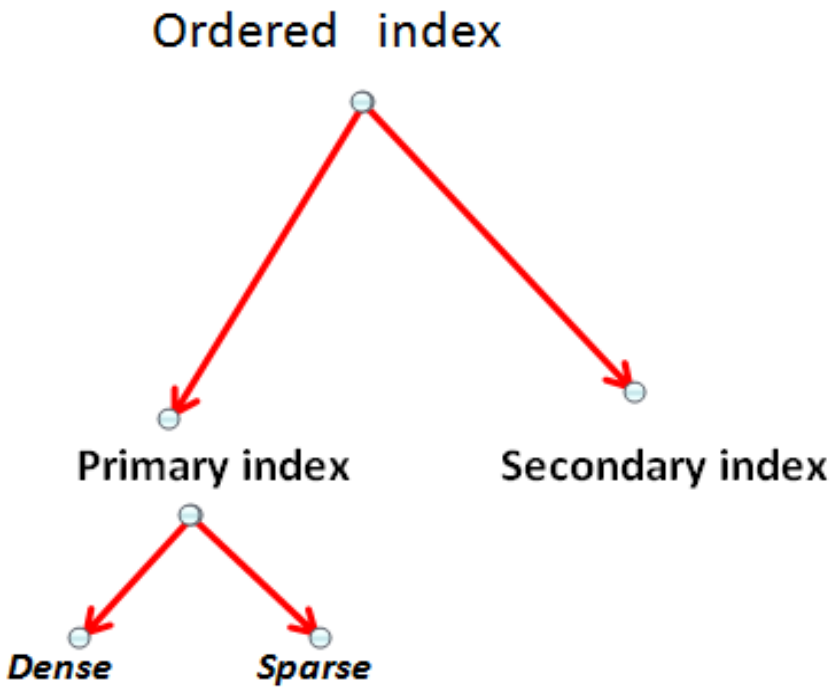
Spars index : An index record is created for only some of the values. To locate a record we find the index entry with the largest search key value that is less than or equal to the search key value for which we are locking . We start at the record pointed to by that index entry, and follow the pointer in the file until we find the desired record. As shown in figure (6.3) for the account file.



نستمر بالبحث الى ان:
اما نجد الكلمة المطلوبة في الجدول
او نصل الى كلمة (أبجديا) تلي الكلمة التي نبحث عنها

Hollywood

Indexing and Hashing

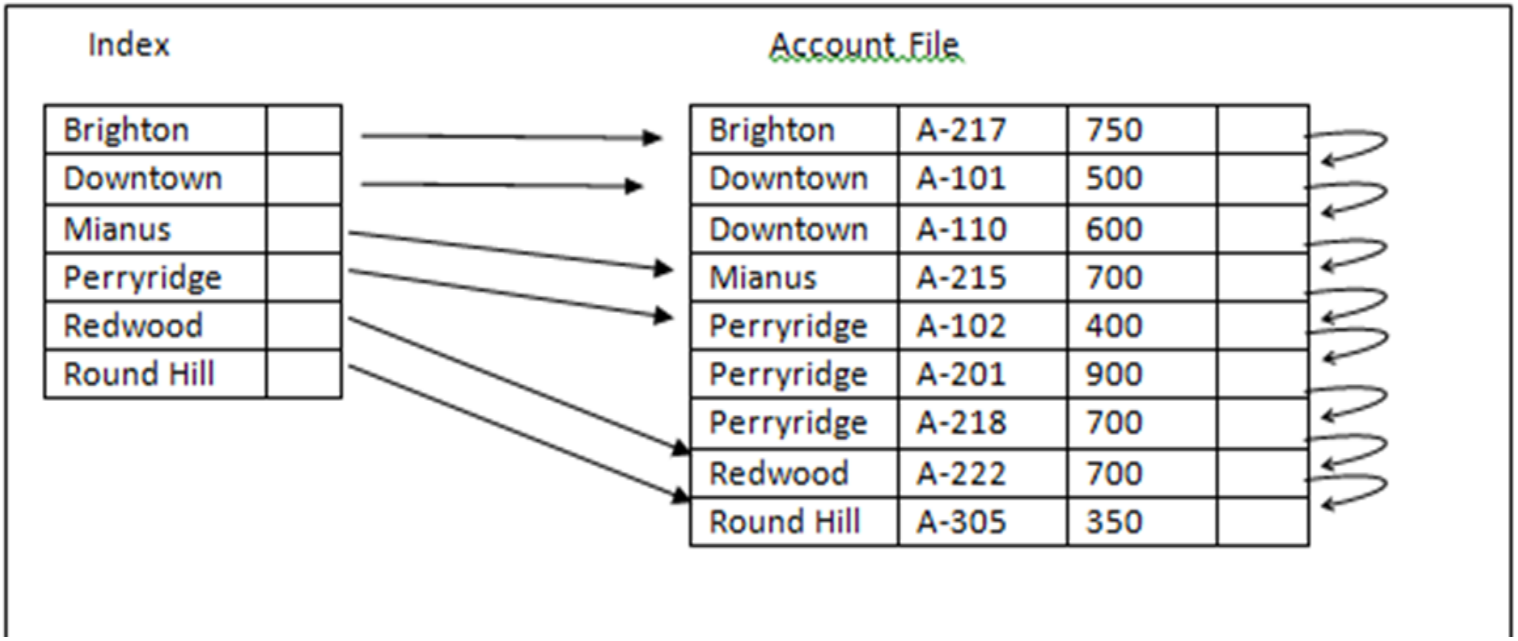


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices

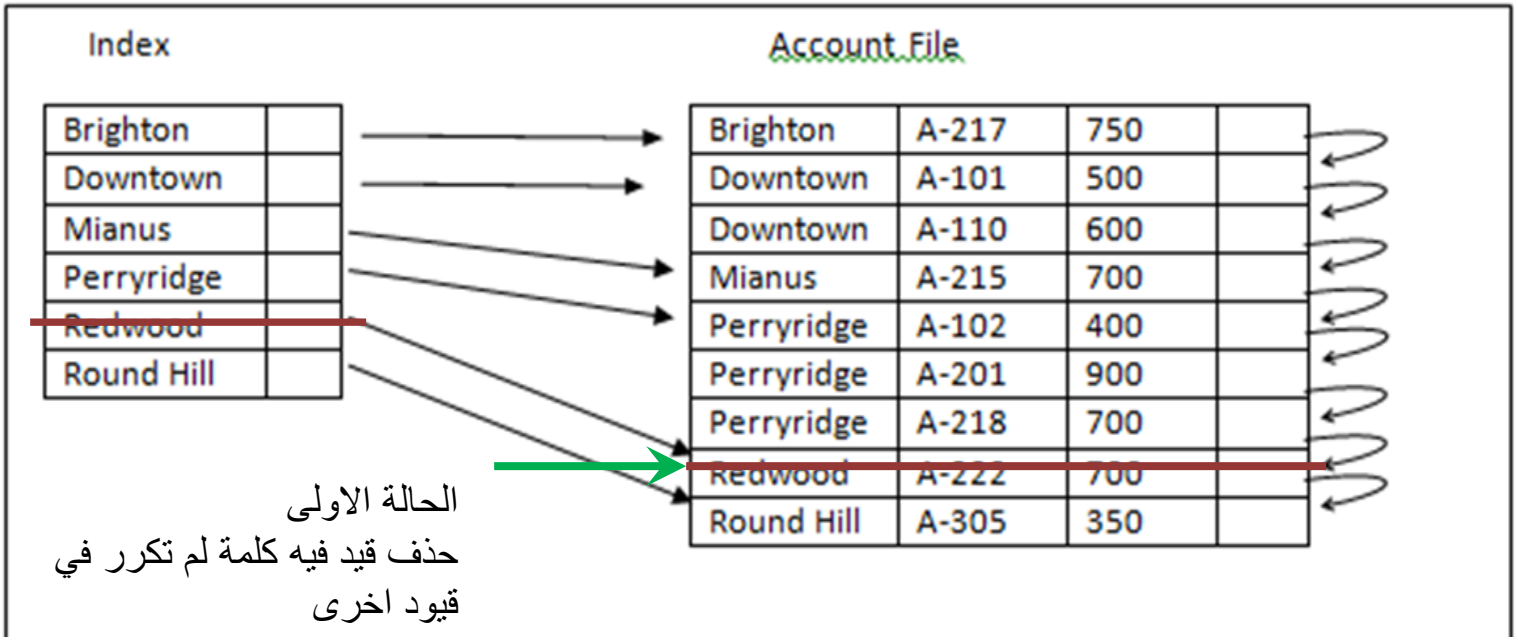


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices

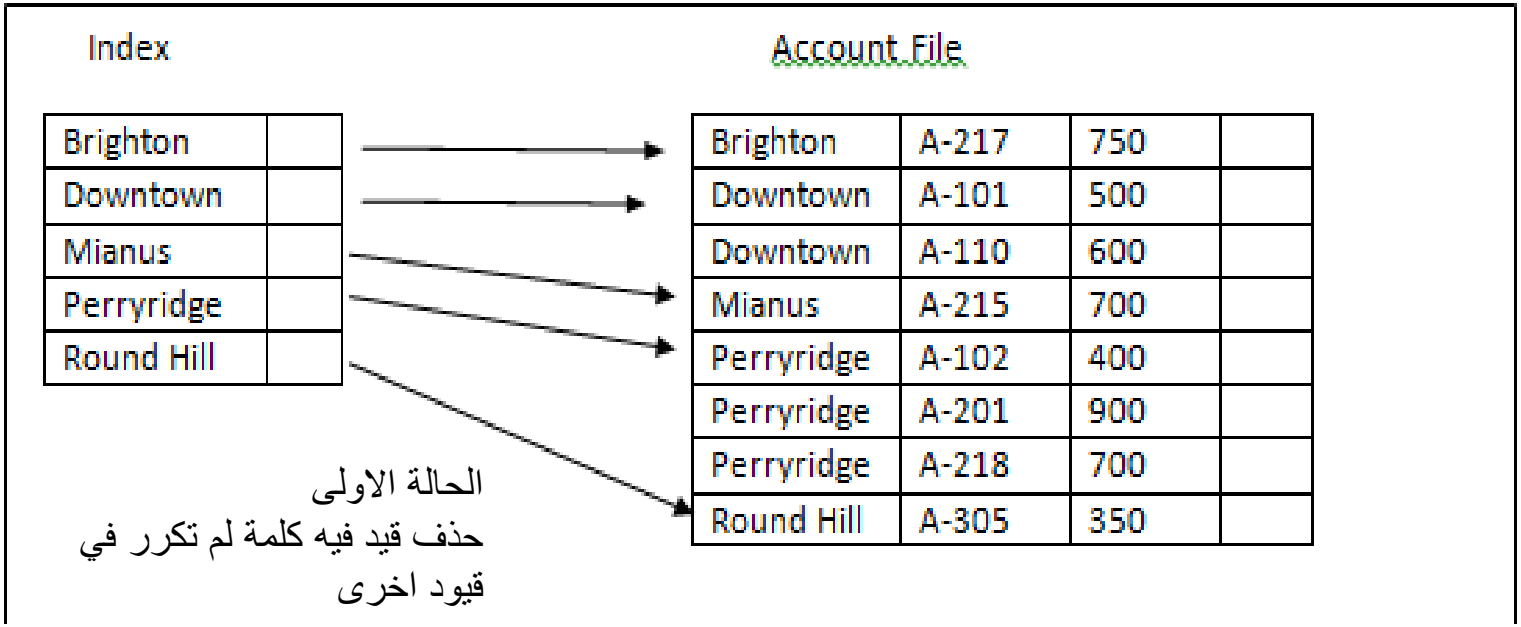


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices

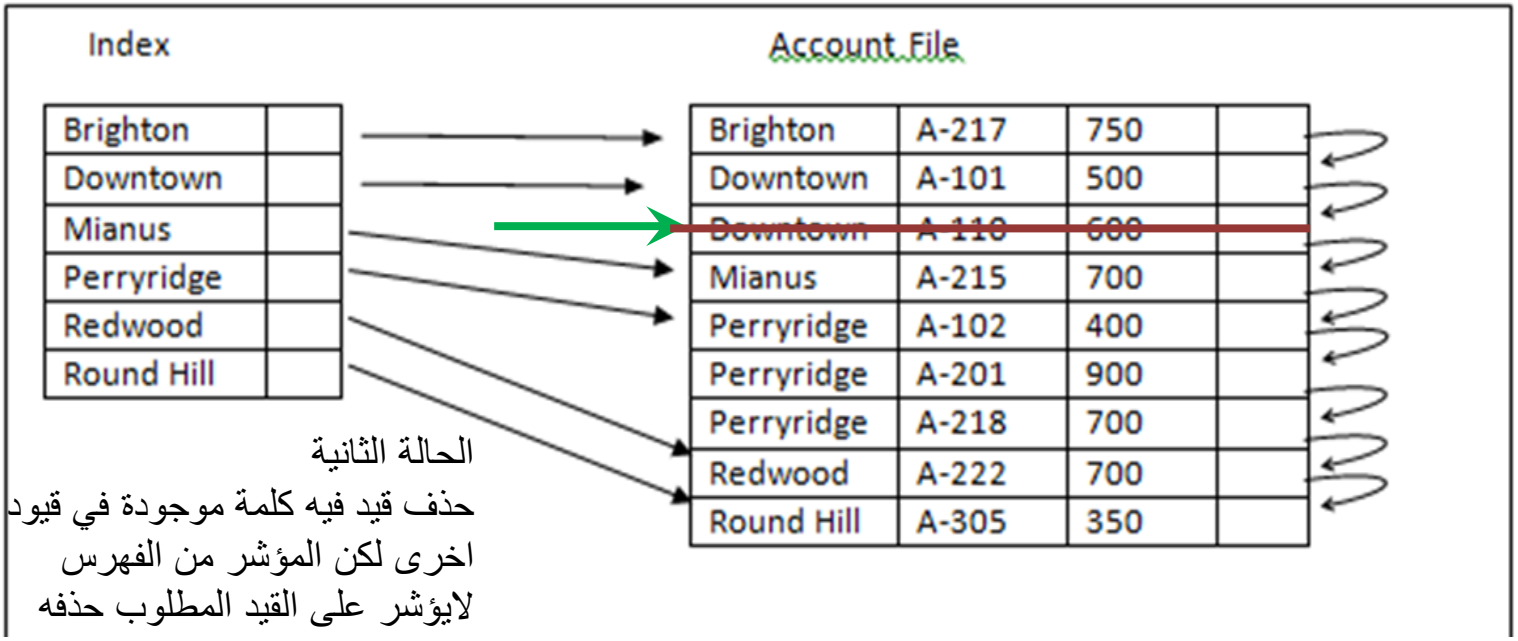


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices

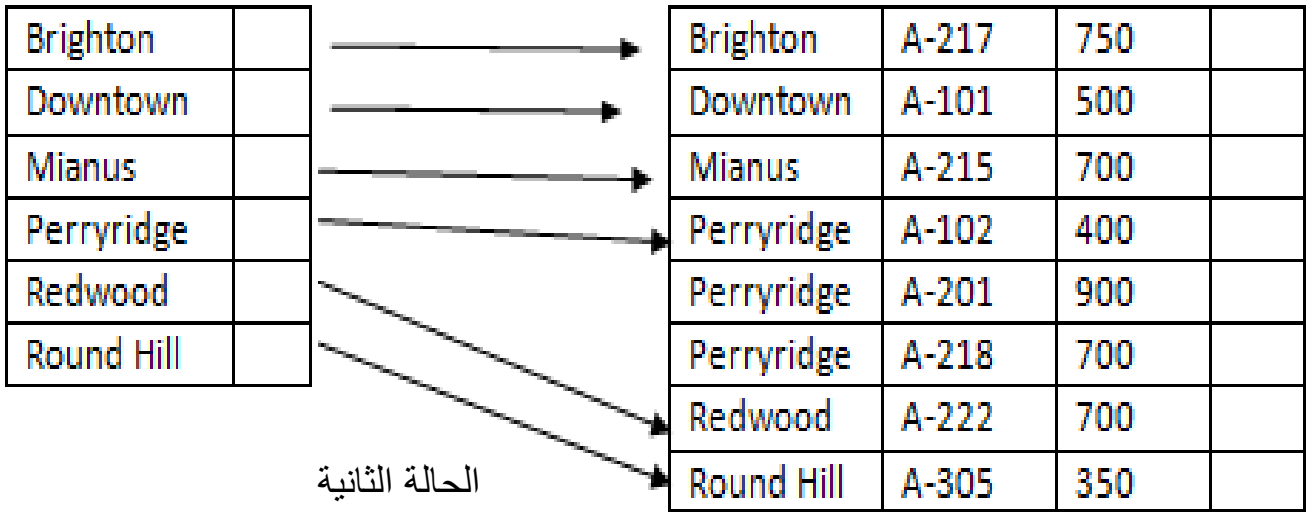


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices



الحالة الثانية

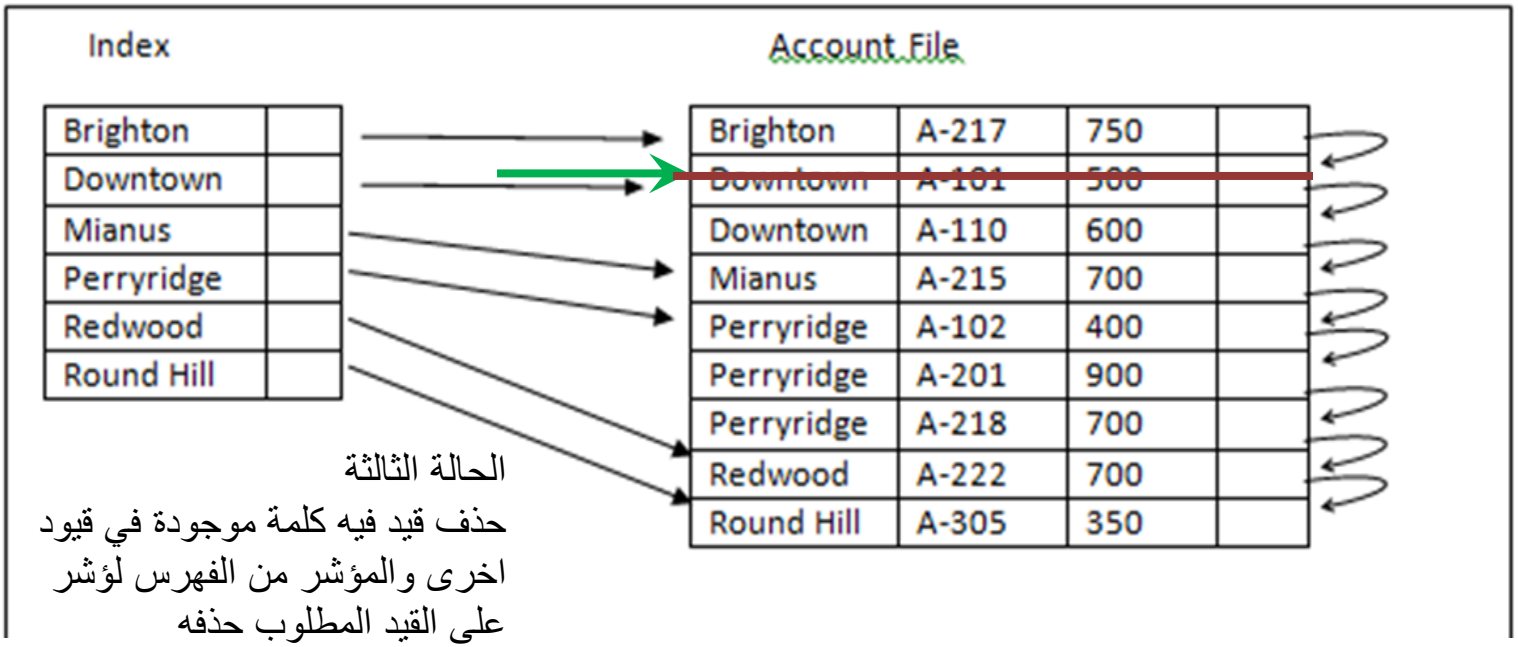
حذف قيد فيه كلمة موجودة في قيود اخرى لكن المؤشر من الفهرس لايؤشر على القيد المطلوب حذفه

Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices

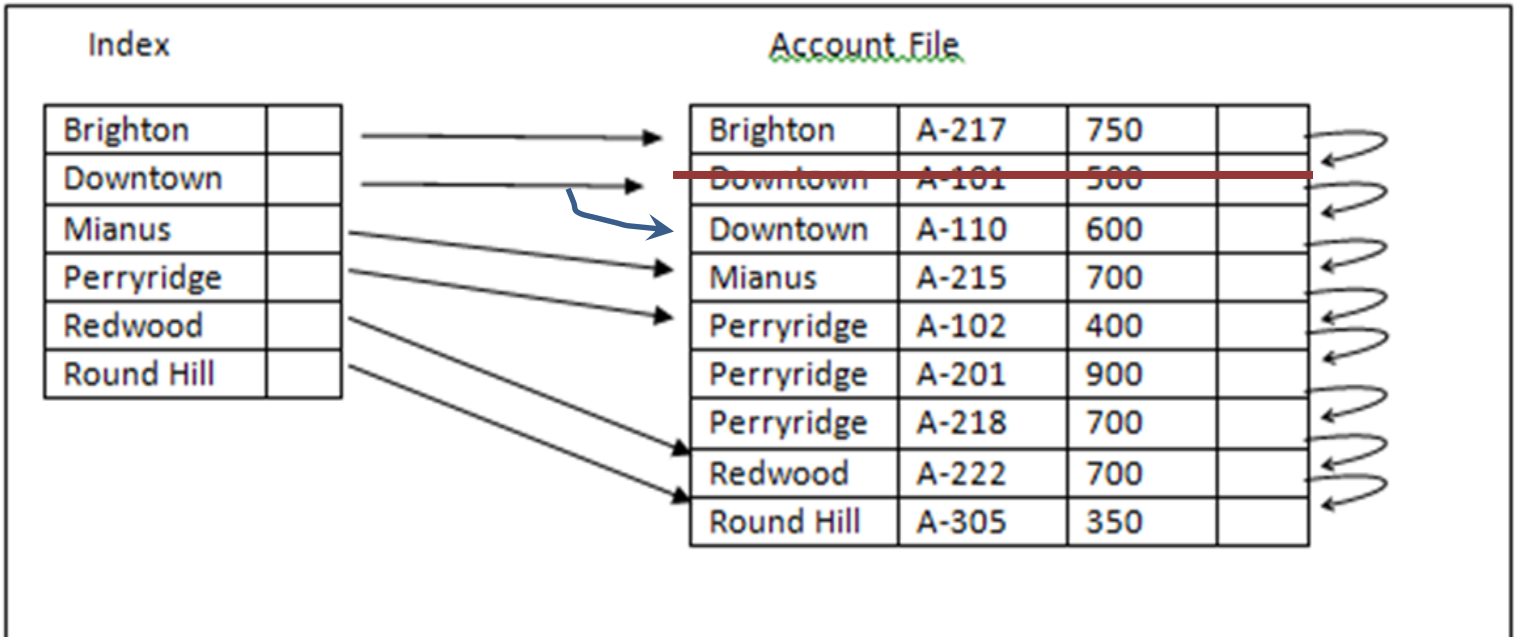


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Deletion In dense indices

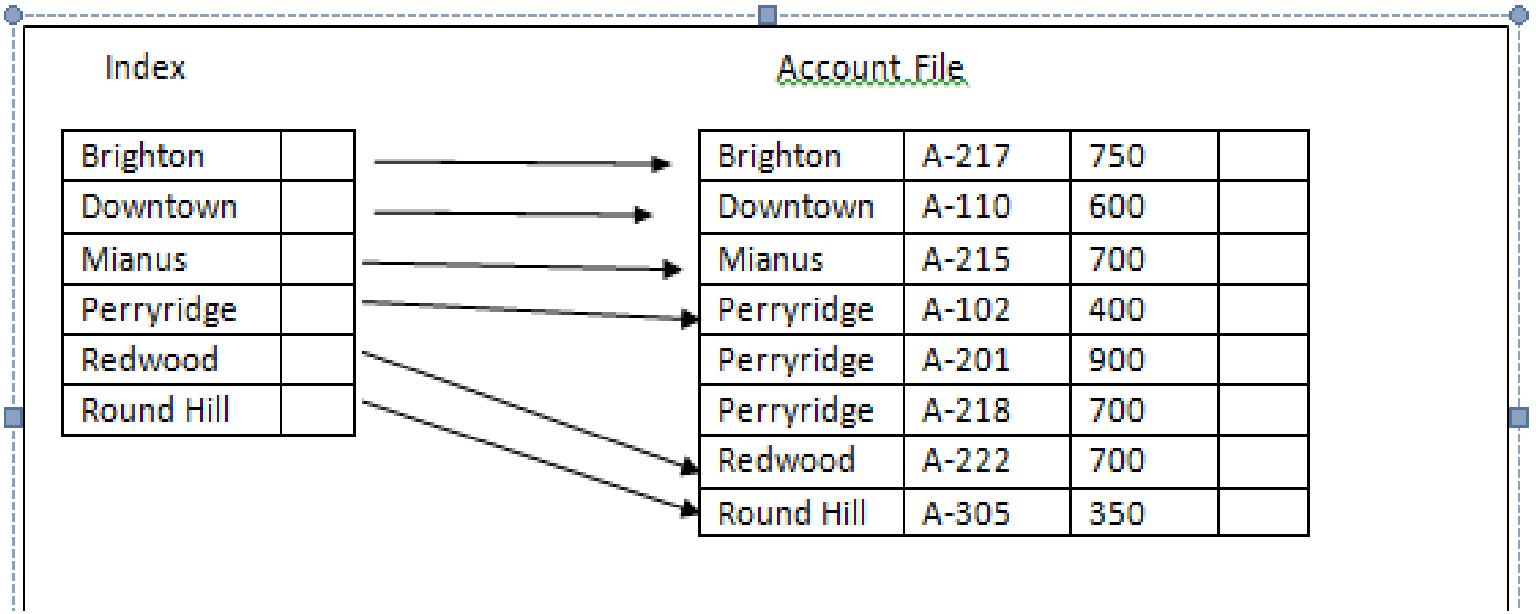


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

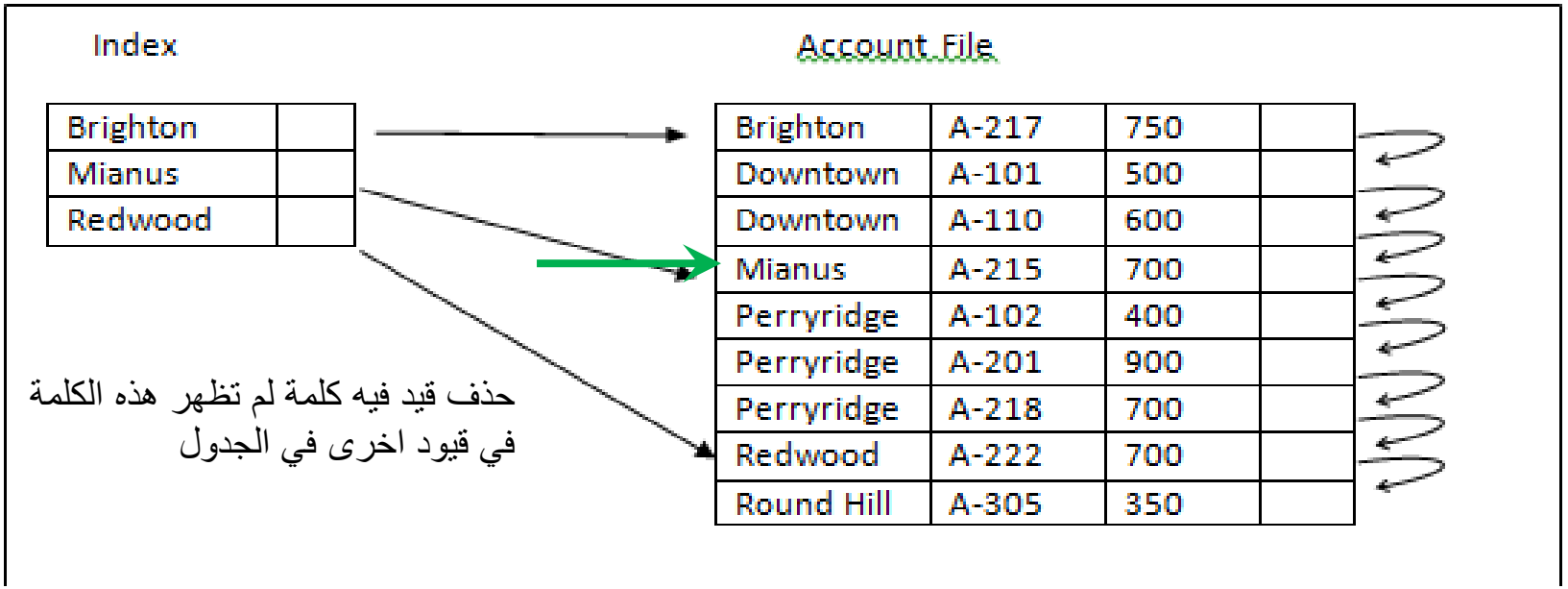
Deletion In dense indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

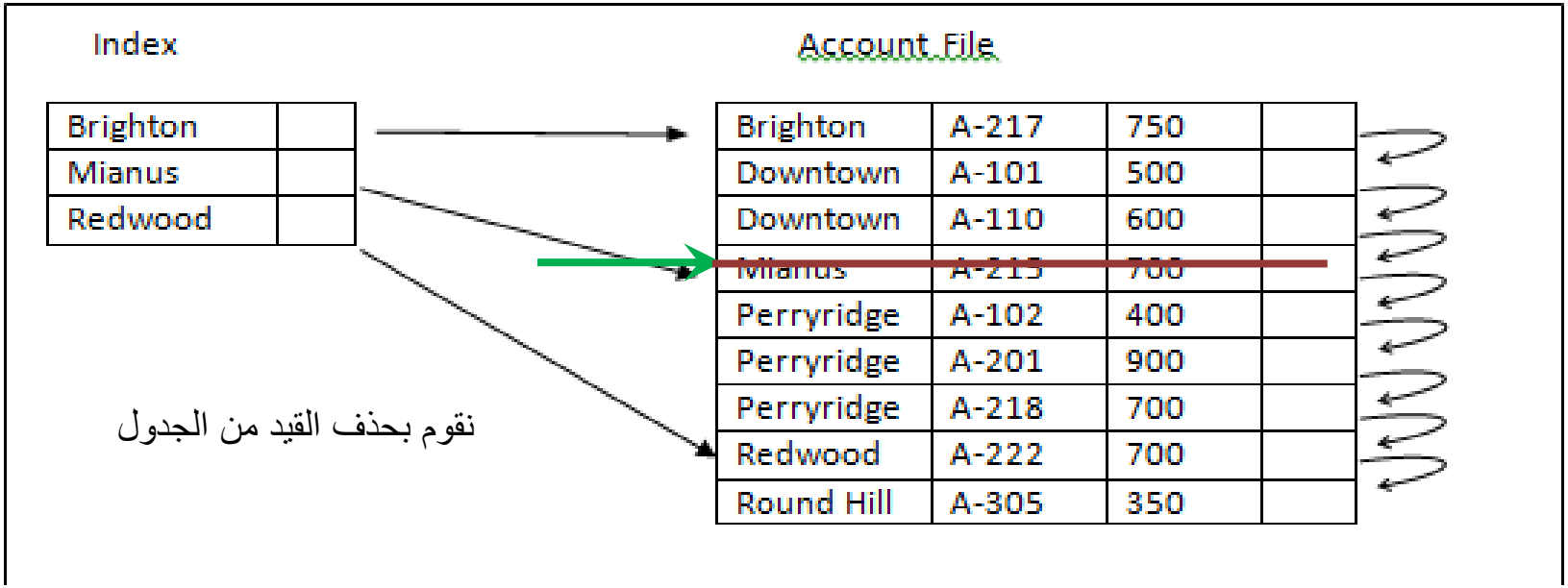
- Deletion In **sparse** indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

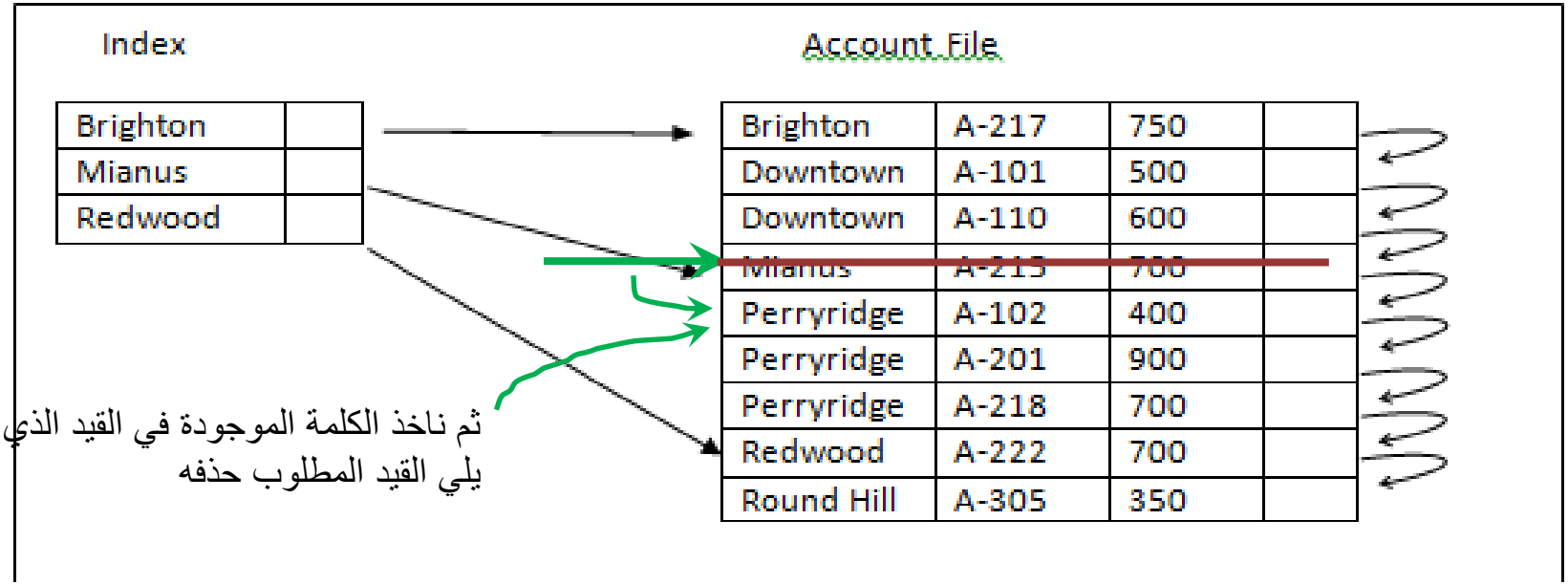
- Deletion In **sparse** indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

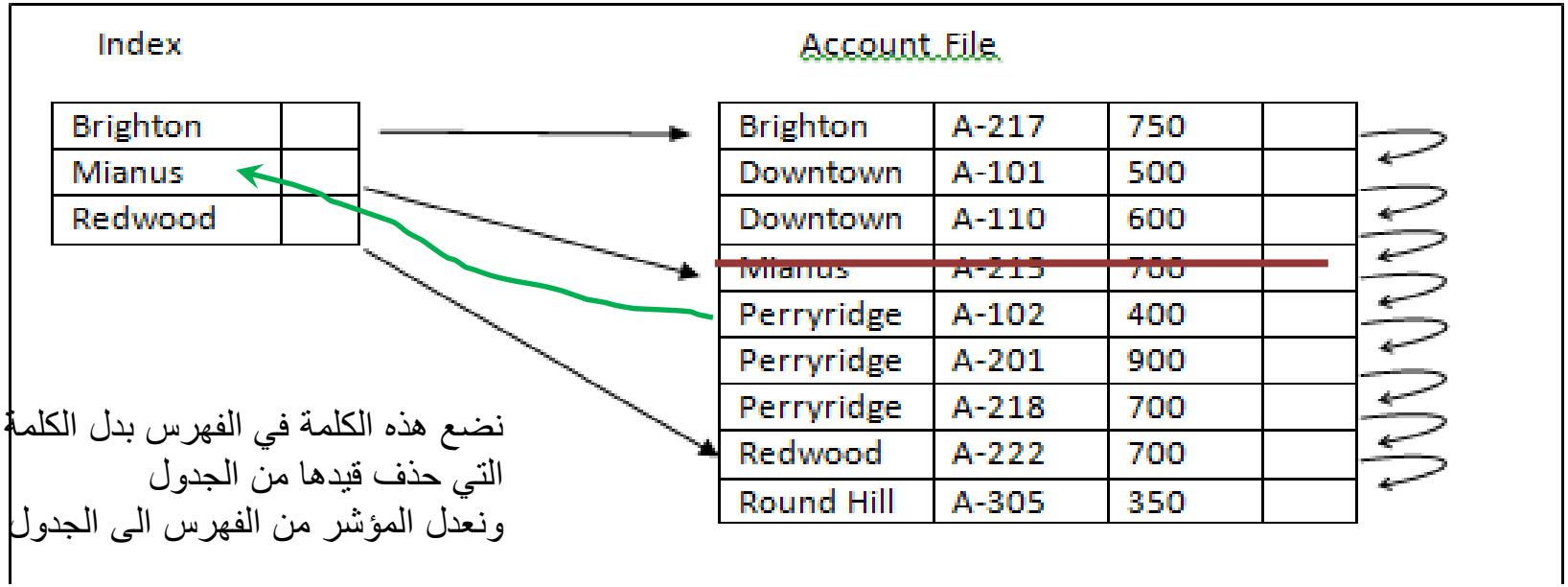
- Deletion In **sparse** indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

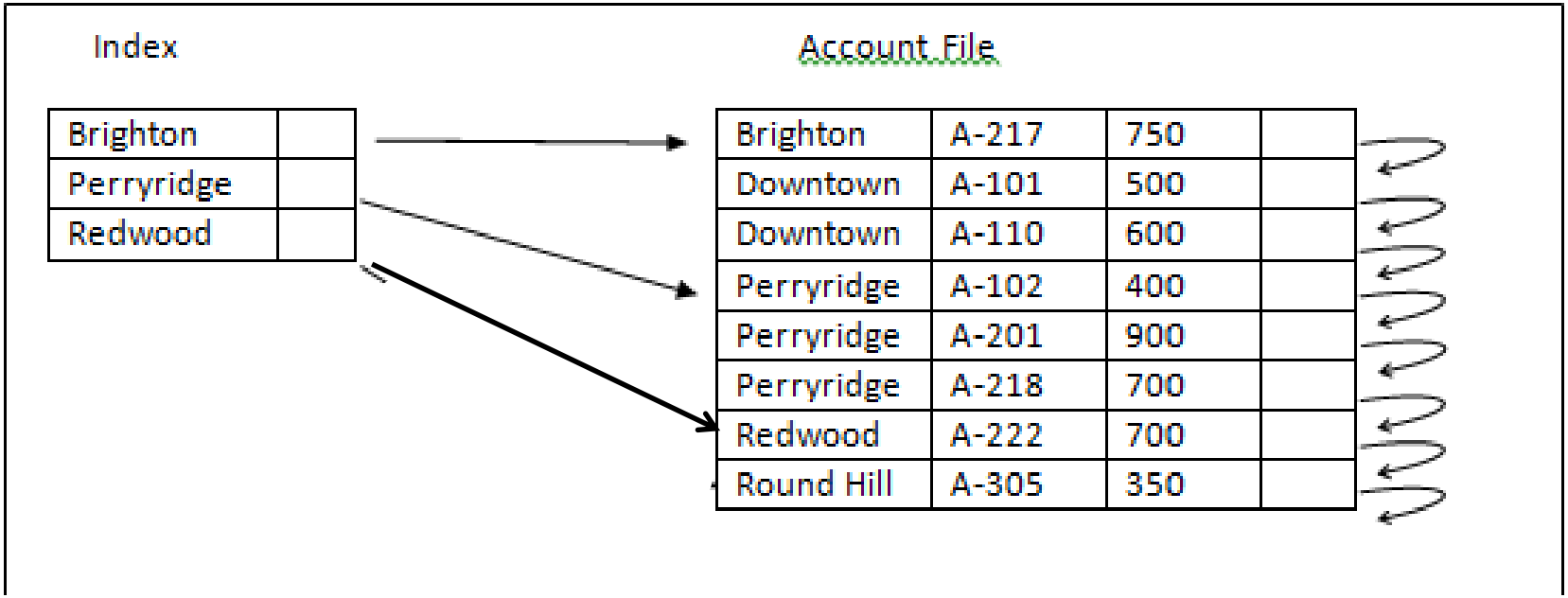
- Deletion In **sparse** indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

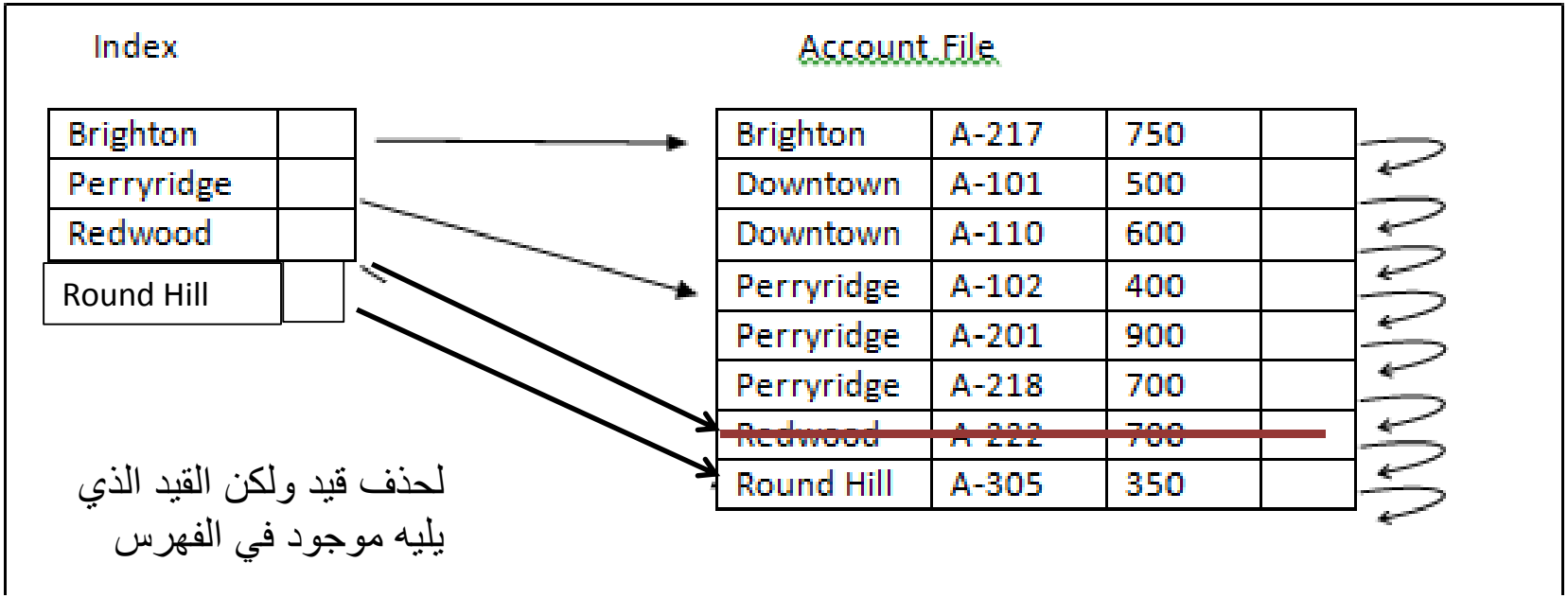
- Deletion In **sparse** indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

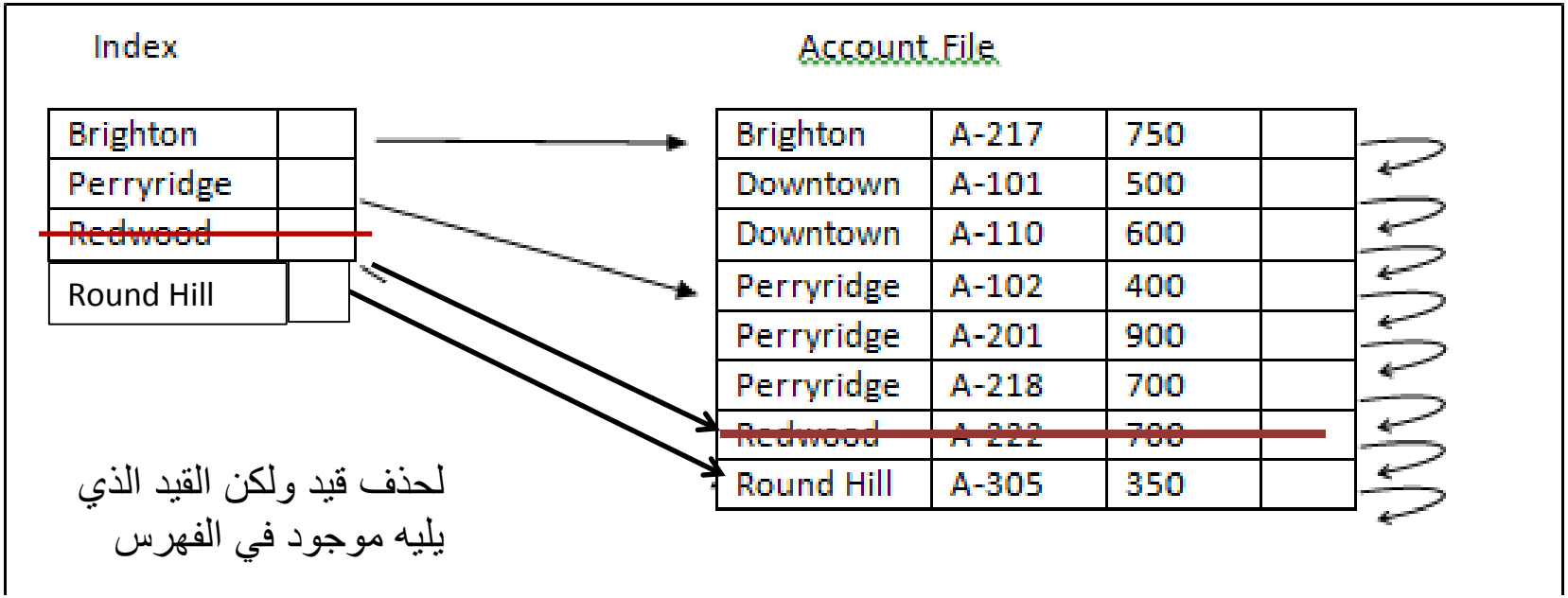
- Deletion In **sparse** indices



Indexing and Hashing

- **6.1.1.2 Index Update**
-
- Every index must be updated whenever a record is inserted into the file or deleted from the file

- Deletion In **sparse** indices

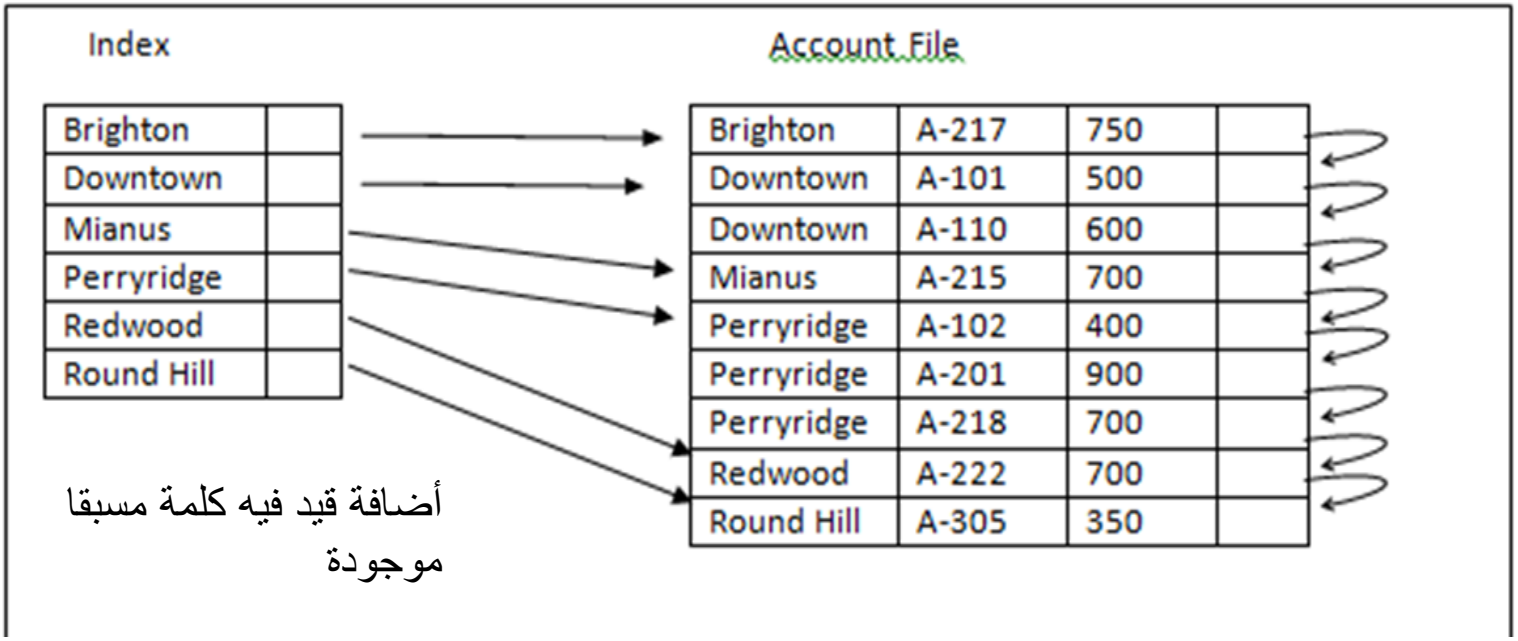


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Insertion In dense indices

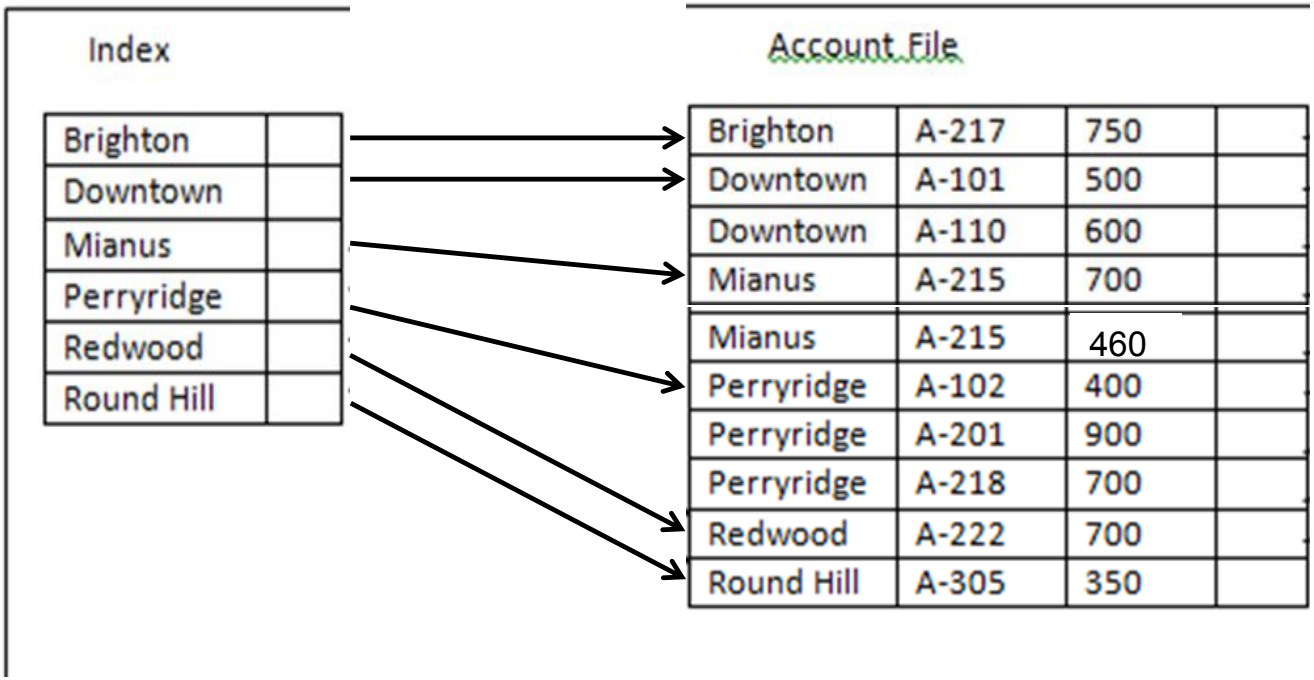


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Insertion In dense indices

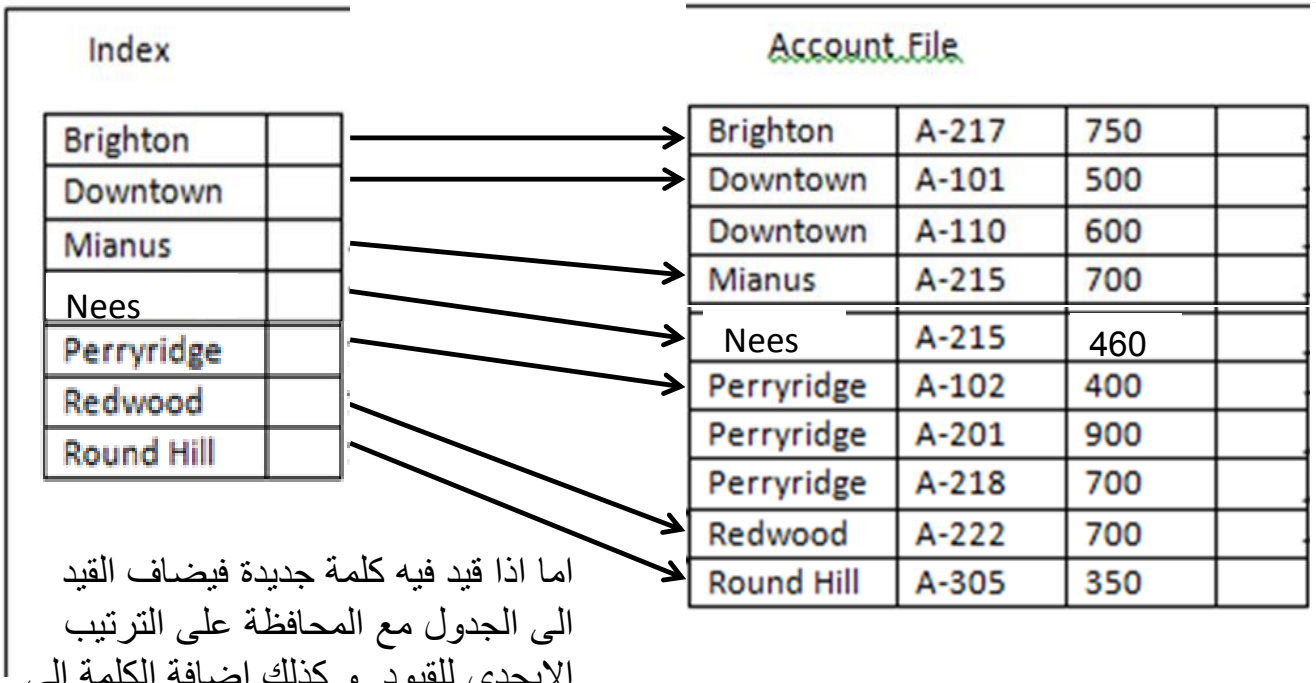


Indexing and Hashing

6.1.1.2 Index Update

Every index must be updated whenever a record is inserted into the file or deleted from the file

Insertion In dense indices



اما اذا قيد فيه كلمة جديدة فيضاف القيد الى الجدول مع المحافظة على الترتيب الابجدي للقيود و كذلك اضافة الكلمة الى الفهرس مع المؤشر