Department of UIC

Database Management Systems

Pawandeep Sharma

Assistant Prof., UIC

INDEXING & HASHING IN DBMS

Basic Concepts

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to set of attributes used to look up records in a file.
- An **index file** consists of records (called **index entries**) of the form

Basic Concepts

- Index files are typically much smaller than the original file
- Two basic kinds of indexes:
  - **Ordered indexes:** search keys are stored in sorted order. They are further of two types:
    - **Dense Index**
    - **Sparse Index**
  - **Hash indexes:** search keys are distributed uniformly across “buckets” using a “hash function”.

Types of Ordered Indexing

- Indexing is defined based on its indexing attributes. Indexing can be one of the following types:
- **Primary Index:** If index is built on ordering 'key-field' of file it is called Primary Index. Generally it is the primary key of the relation.
- **Secondary Index:** If index is built on non-ordering field of file it is called Secondary Index.
- **Clustering Index:** If index is built on ordering non-key field of file it is called Clustering Index. Ordering field is the field on which the records of file are ordered. It can be different from primary or candidate key of a file.

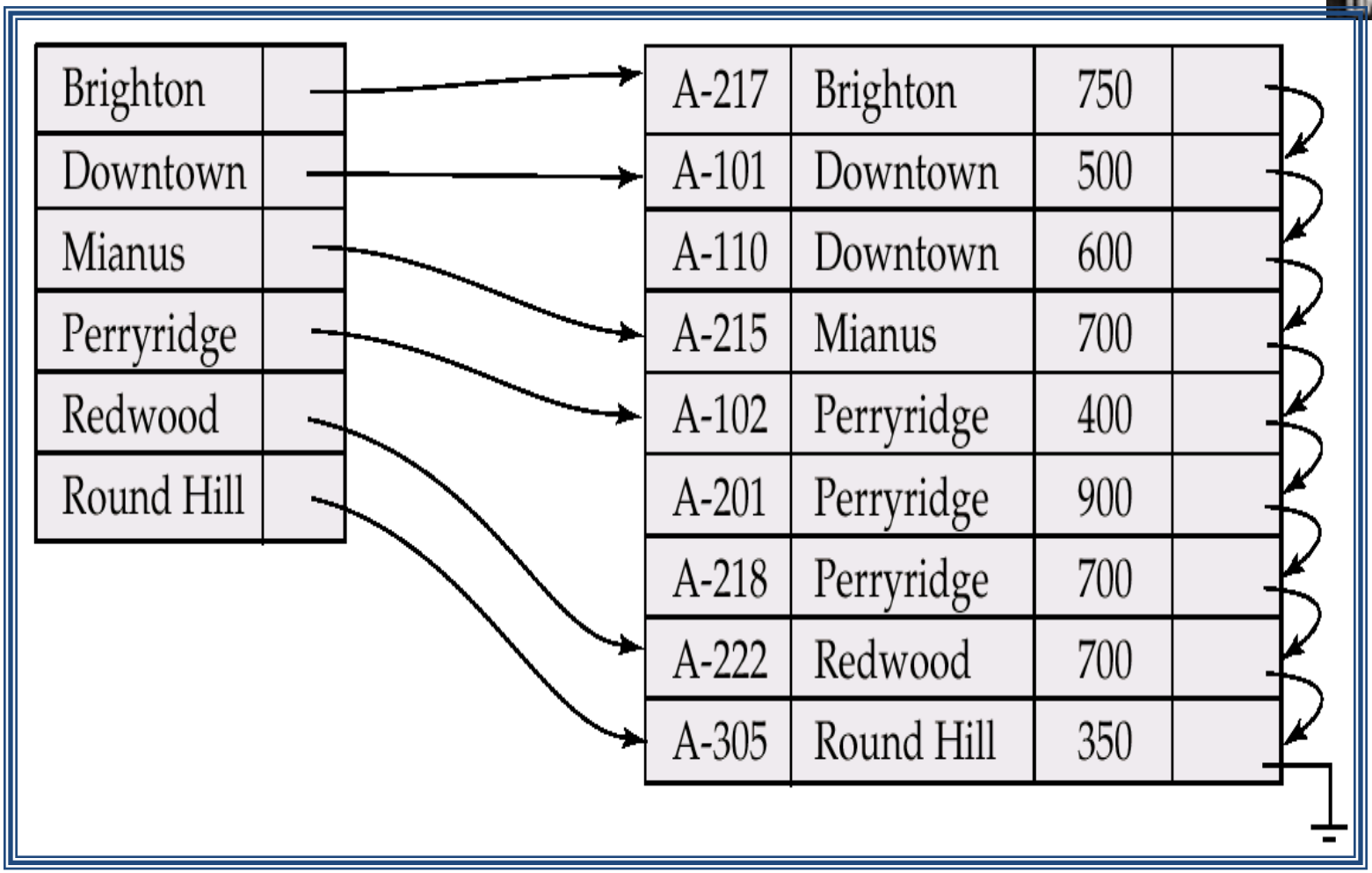
Primary Indexing is of two types:

- Dense Index
- Sparse Index

Dense Index Files

- Index record appears for every search-key value in the file.
- This makes searching faster but requires more space to store index records itself.
- Index record contains search key value and a pointer to the actual record on the disk.

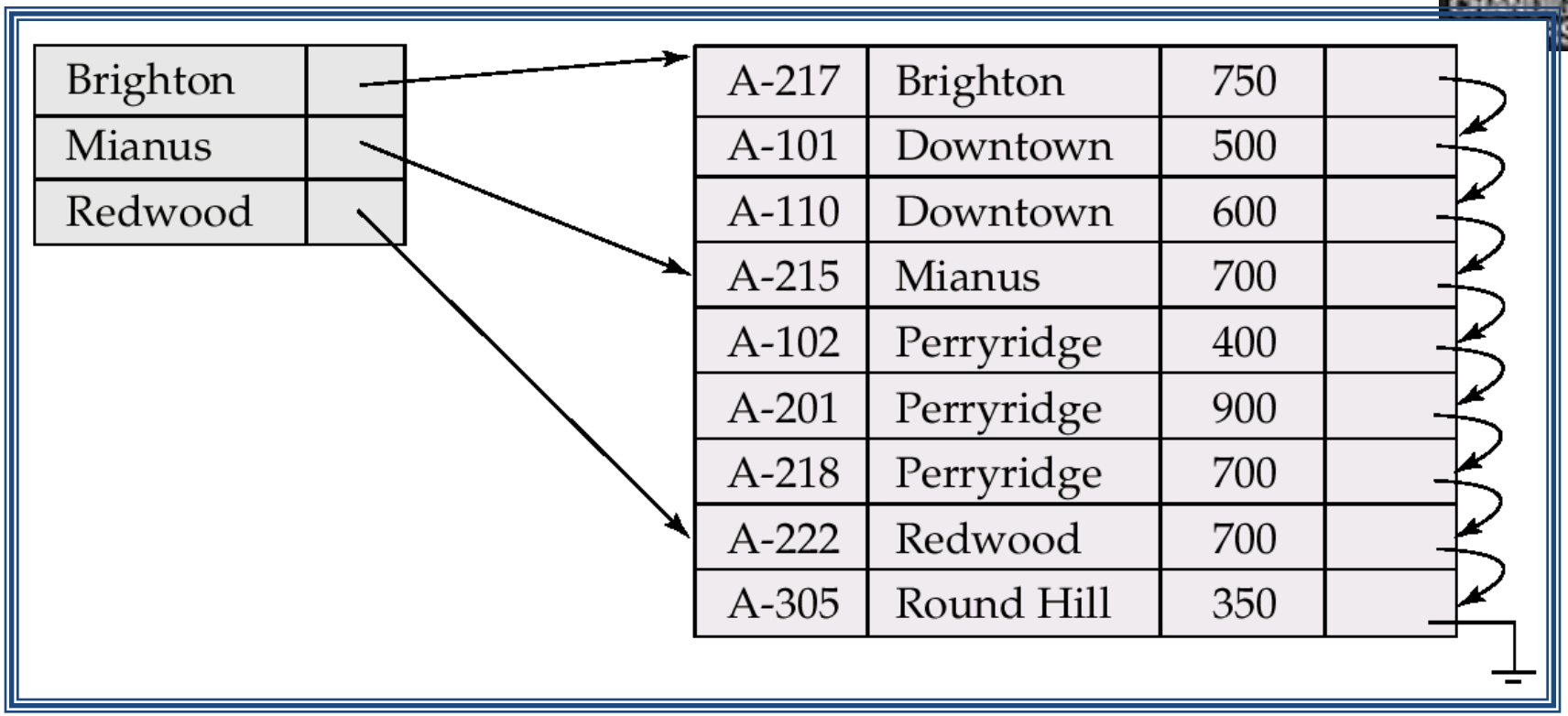
Example of Dense Index Files



Sparse Index Files

- It contains index records for only some search-key values.
  - Applicable when records are sequentially ordered on search-key
- To locate a record with search-key value *K* we:
  - Find index record with largest search-key value < *K*
  - Search file sequentially starting at the record to which the index record points
- Less space and less maintenance overhead for insertions and deletions.
- Generally slower than dense index for locating records.

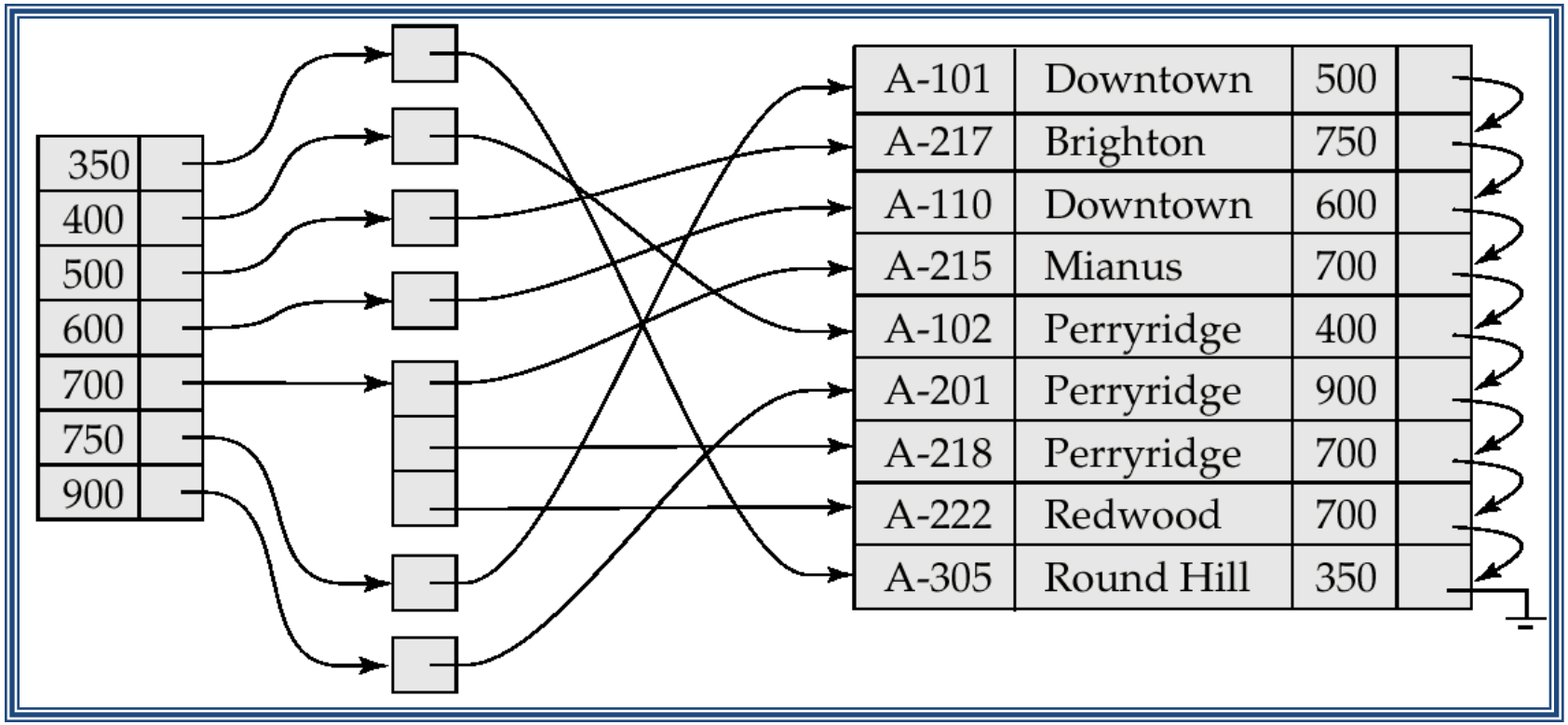
Example of Sparse Index Files



Secondary Indexes

- Frequently, one wants to find all the records whose values in a certain field (which is not the search-key of the primary index satisfy some condition.
  - Example 1: In the *account* database stored sequentially by account number, we may want to find all accounts in a particular branch
  - Example 2: as above, but where we want to find all accounts with a specified balance or range of balances
- We can have a secondary index with an index record for each search-key value; index record points to a bucket that contains pointers to all the actual records with that particular search-key value.

Secondary Index on ***balance*** field of ***account***



HASHING

Hashing

- The idea behind hashing is to provide a function h, called a hash function or randomizing function, which provides the address of the disk block in which the record is stored.
- This organization is usually called a hash file.
- Hashing provides very fast access to records on certain search conditions .
- The search condition must be specified on a single field of the file called the hash field.
- In most cases ,the hash field is also a key field of the file ,in which case it is called the hash key.

Types of Hashing

- Static Hashing (External Hashing)
- Dynamic Hashing (Internal Hashing)

Static Hashing

- Hashing provides a means for accessing data without the use of an index structure.
- Data is addressed on disk by computing a function (hash function) on a search key instead.

Organization

- A unit of storage is known as bucket that can hold one or more records.
- The hash function, h, is therefore a function from the set of all search-keys, K, to the set of all bucket addresses, B.
- Insertion, deletion, and lookup operations can be performed on this bucket.

Querying and Updates

- To insert a record into the structure compute the hash value $h(K_i)$, and place the record in the bucket address returned.
- For lookup operations, compute the hash value as above and search each record in the bucket for the specific record.
- To delete simply lookup and remove.

Properties of the Hash Function

- The distribution should be uniform.
  - An ideal hash function should assign the same number of records in each bucket.
- The distribution should be random.
  - Regardless of the actual search-keys, the each bucket has the same number of records on average
  - Hash values should not depend on any ordering or the search-keys

Condition of Bucket Overflow

- How does bucket overflow occur?
  - Not enough buckets to handle data
  - A few buckets have considerably more records than others. This is referred to as skew.
    - Due to Multiple records have the same hash value

Solutions

- Provide more buckets than are needed.
- Overflow chaining
  - If a bucket is full, link another bucket to it. Repeat as necessary.
  - The system must then check overflow buckets for querying and updates. This is known as **closed hashing**.

Alternatives

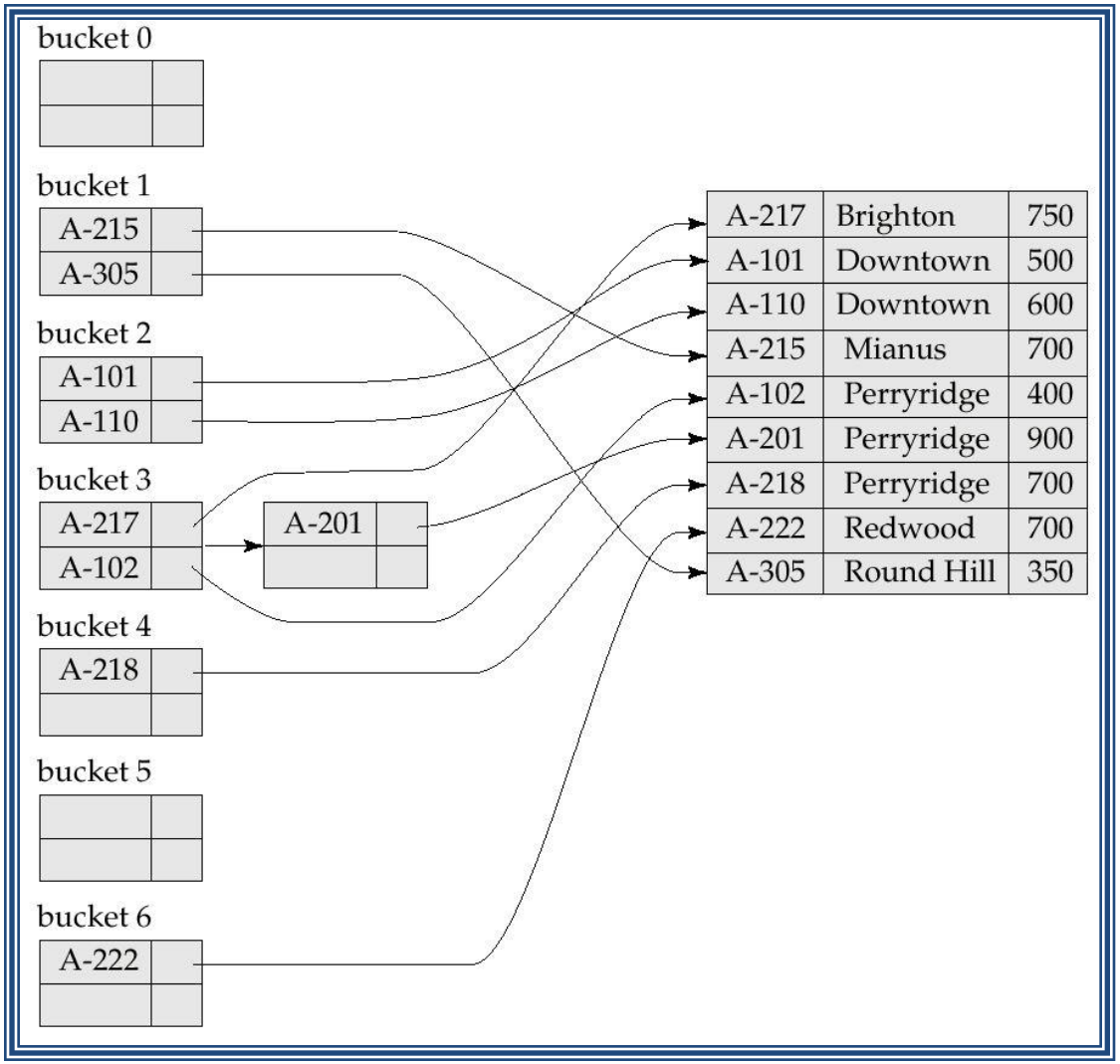
- Open hashing
  - The number of buckets is fixed
  - Overflow is handled by using the next bucket in cyclic order that has space.
    - This is known as **linear probing**.

Note: Closed hashing is preferred in database systems.

Hash Index

- A **hash index** organizes the search keys, with their pointers, into a hash file.
- Hash indices never primary even though they provide direct access.

Example of Hash Index



Dynamic Hashing

- More effective than static hashing when the database grows or shrinks
- **Actually an Extendable hashing** in which buckets are added and deleted on demand.

The Hash Function in dynamic hashing

- Typically produces a large number of values, uniformly and randomly.
- Only part of the value is used depending on the size of the database.

Comparison to Other Hashing Methods

- Advantage: performance does not decrease as the database size increases
- Disadvantage:Complex implementation

Ordered Indexing vs. Hashing

- Hashing is less efficient if queries to the database include ranges as opposed to specific values.
- Otherwise hashing provides faster insertion, deletion, and lookup than ordered indexing.

Reference Books

- Fundamentals of Database Systems by R.Elmasri and S.B.Navathe, 3rd Edition, Pearson Education, New Delhi.
- An Introduction to Database Systems by C.J. Date, 7th Edition, Pearson Education, New Delhi.
- A Guide to the SQL Standard, Data, C. and Darwen, H.3rd Edition, Reading, Addison-Wesley Publications, New Delhi.
- Introduction to Database Management system by Bipin Desai, Galgotia Pub, New Delhi.
- Database System Concepts by A. Silberschatz, H.F.Korth and S.Sudarshan, 3rd Edition, McGraw-Hill, International Edition.
- SQL / PL/SQL, by Ivan Bayross, BPB Publications.

QUERIES ??

THANK YOU