LESSON 4.3

98-364 Database Administration Fundamentals

Understand Indexes

Lesson Overview

In this lesson, you will learn:

- What is an index?
- Why do we use indexes?
- Common data structures for indexes
- Non-clustered indexes
- Clustered indexes

What is an index?

- An index is a list of objects.
  - — *Example:* an index for a book is a list of the topics within the book.
  - — We use an index so that we can go directly to the topic that we want to look at in the book or database.
- An index is a data structure.
  - — A listing of key words and associated data that point to the location of more comprehensive information, such as files and records on a disk or record keys in a database.

Why do we use indexes?

- **Life without an index**
  - — Data is difficult to locate because it is stored in the order entered.
  - — There is no easy or efficient method for retrieving records.
  - — Searching is blind and haphazard. An example of a blind search: Suppose you want to try a recipe that uses a food that you have never heard of. All you know is that it is a type of food. If you had an index of foods, it would speed up your efforts to find this particular item, and if you also had data on specialty food stores that carry that item, you would be able to obtain it even faster.
- **Life with an index**
  - — Significantly improve the speed of data retrieval.
  - — Writing of records can be slower. In addition, indexes take up more storage, just as a book index requires more pages in the book.
  - — The Internet would not be possible without indexes. If we had to search the Internet in the order that items were added, we would never find anything on this giant database on the World Wide Web.

Common Data Structures for Indexes

Four common data structures for indexes: bitmap, dense, sparse, and reverse.

- **Bitmap index**
  - — A bitmap index stores the data in bit arrays.
  - — The most common form is a B-tree.
  - — B-trees can be a highly efficient data structure.

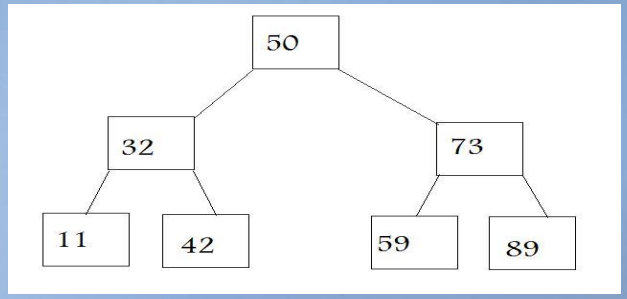
- **Dense index**
  - — A dense index works with pairs of keys and pointers for each record.
  - — Every key has a pointer tied directly to a record.

Common Data Structures for Indexes (continued)

- **Sparse index**
  - — A sparse index works with pairs of keys and pointers for each block (a sequence of bytes or bits).
  - — Every key has a pointer tied to a block of data.
  - — Less costly in resources and less effective (more generalized) than a dense index.

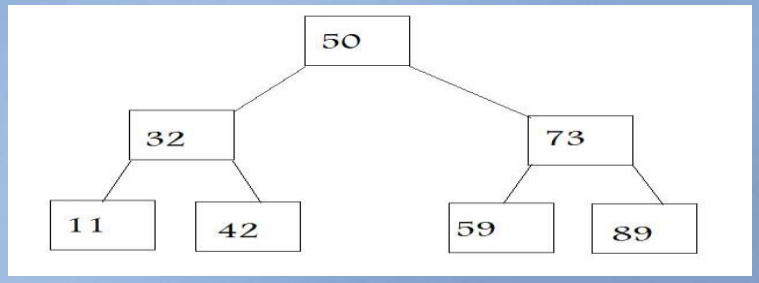
- **Reverse index**
  - — A reverse index reverses the key value; for instance, 12345 becomes 54321.
  - — This method is useful when keys are set in a sequence where new key values change by a uniform amount.

B-tree



- A tree structure for storing database indexes.
- The *B* in *B-tree* standards for *balance*.
- Each node in the B-tree contains a sorted list of key values and links that correspond to ranges of key values between the listed values.
- To find a specific data record given its key value, the program reads the first node, or root, from the disk and compares the desired key with the keys in the node to select a subrange of key values to search.
- It repeats the process with the node indicated by the corresponding link. At the lowest level, the links indicate the data records.
- The database system can thus rapidly skip down through the levels of the tree structure to find the simple index entries that contain the location of the desired records or rows.

B-tree (continued)



- This B-tree cuts search time significantly.
- The data is organized in a balance method.
- Each side of the tree has half the keys.
- If you are looking for 59, you would travel down the right branch until you hit 73 and then travel down the left branch to reach 59.
- The lowest layer is the tree is referred to as the *leaf layer* of the tree.
  At this layer, you can find the pointer to the location of the desired records or rows.

Non-clustered Indexes

- Similar to the index of a book.
- Has a keyword and pointer to the stored location of the information.
  - Example: If I wanted information about the topic of "primary key" for these lessons, I would look at the following:

Keyword	Location of the information
Primary key	Review lesson 4.2 slide 4-6, 10

- Non-clustered indexes can use the B-tree structure.
  - The leaf layer of the tree is made up of index pages or pointers.
- Non-clustered indexes are less efficient in searches than clustered indexes.

Non-clustered Indexes (continued)

- Syntax:

```
CREATE INDEX index_name
ON table_name (column_name)
```

- Example:

```
CREATE INDEX NamesIndex
ON employees (LastName, FirstName)
```

The above statement creates an index on the `employees` table by `LastName` and any duplicate last names are then sorted by `FirstName`.

Clustered Indexes

- If your address book is ordered by nicknames, this is similar to a clustered index.
- Clustered indexes are indexes that are in a special order to make retrieval faster.
- These indexes can be stored in memory.
- Clustered indexes can use the B-tree data structure.
  - — In the clustered B-tree, the data records of the underlying table are sorted in order based on their clustered keys.
  - — The leaf layer (bottom layer) of clustered indexes is made up of data pages or records.
  - — At this layer, data access will be direct.
  - — This is the major functional differences between clustered and non-clustered. Non-clustered indexes have a pointer value at the leaf layer rather than the value itself.

Clustered Indexes (continued)

- Syntax:

```
CREATE [ UNIQUE ] CLUSTERED INDEX index_name
 ON table_name (column_name)
```

  [ ] = Optional: A unique index is one in which no two rows have the same index key value.

- Example:

```
CREATE CLUSTERED INDEX NamesIndex
 ON employees (LastName, FirstName)
```

- The above statement has `CLUSTERED` before `INDEX`, hence creating a clustered index.
  - If the word `CLUSTERED` is left out, then a non-clustered index is created.

Lesson Review

1. What is an index?
2. Why do we use indexes?
3. List the four common index structures.
4. What is a B-tree?
5. What is the difference between a clustered and non-clustered index?
6. What factors influence which index method is used?
7. List several indexes that you use and predict the structure employed.