Lecture 10

Chapter 8: Indicies

John Connor

November 7, 2018

Indexes

Indexes (8.3)

An index allows database queries to be carried out efficiently.

Indexes (8.3)

An index allows database queries to be carried out efficiently.

You may think of an index for a set of attributes as begin similar to a binary search tree of (values, locations) pairs, where “values” is an assignment of values to the attributes and “locations” is the locations of the tuples with these values in the specified attributes.

Indexes (8.3)

An index allows database queries to be carried out efficiently.

You may think of an index for a set of attributes as begin similar to a binary search tree of (values, locations) pairs, where "values" is an assignment of values to the attributes and "locations" is the locations of the tuples with these values in the specified attributes.

This is very similar to the index in a book, where "values" is simply a word, and "locations" is the page numbers where the word occures in the text.

Motivation

When relation are very large, it becomes expensive to search all of the tuples that match a given condition.

For example, consider the first query we examined:

```
SELECT *
FROM Movies
WHERE studioName = 'Disney' AND year = 1990;
```

Motivation

When relation are very large, it becomes expensive to search all of the tuples that match a given condition.

For example, consider the first query we examined:

```
SELECT *
FROM Movies
WHERE studioName = ‘Disney’␣AND␣year␣=␣1990;
␣␣␣␣
```

There might be hundereds of thousands of tuples in the database, but only a fraction of the match the conditions.

Synatx (8.3.2)

Suppose we want an index on `year` and on `studioName`:

```
CREATE INDEX YearIndex ON Movies(year);
CREATE INDEX StudioIndex ON Movies(studioName);
```

Synatx (8.3.2)

Suppose we want an index on `year` and on `studioName`:

```
CREATE INDEX YearIndex ON Movies(year);
CREATE INDEX StudioIndex ON Movies(studioName);
```

To create a multiattribute index on `year` and `studioName`:

```
CREATE INDEX KeyIndex ON Movies(year, studioName);
```

Synatx (8.3.2)

Suppose we want an index on `year` and on `studioName`:

```
CREATE INDEX YearIndex ON Movies(year);
CREATE INDEX StudioIndex ON Movies(studioName);
```

To create a multiattribute index on `year` and `studioName`:

```
CREATE INDEX KeyIndex ON Movies(year, studioName);
```

To remove an index, we do it by name. For example:

```
DROP INDEX YearIndex;
```

A Cost Model

An index on an attribute may greatly increase the speed of queries involving the attribute. However inserting, updating, and deleting records will require more become time.

A Cost Model

An index on an attribute may greatly increase the speed of queries involving the attribute. However inserting, updating, and deleting records will require more become time.

Assume tuples are uniformly distributed on disk, and that one page holds many tuples. Loading a page into memory is time consuming, but it costs little more time to examine all of the tuples on a page, or just one.

A Cost Model

An index on an attribute may greatly increase the speed of queries involving the attribute. However inserting, updating, and deleting records will require more become time.

Assume tuples are uniformly distributed on disk, and that one page holds many tuples. Loading a page into memory is time consuming, but it costs little more time to examine all of the tuples on a page, or just one.

Try to work out which attributes will gain the most from being indexed. Also try to work out if we can use an index to work out a more efficient way to store tuples than "uniformly on disk".

A Cost Model

An index on an attribute may greatly increase the speed of queries involving the attribute. However inserting, updating, and deleting records will require more become time.

Assume tuples are uniformly distributed on disk, and that one page holds many tuples. Loading a page into memory is time consuming, but it costs little more time to examine all of the tuples on a page, or just one.

Try to work out which attributes will gain the most from being indexed. Also try to work out if we can use an index to work out a more efficient way to store tuples than "uniformly on disk".

1. Generally the best attributes to index are the keys (or sets of attributes that are "almost" keys).
2. We have the option of "clustering" the storage of tuples based on the indexed attributes.

Index Maintenance

One way to somewhat mitigate the cost of indexes is to create an mutate data in batches, and to rebuild the indices after the batch is processed.

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.
2. You have a high volume of queries, and its eating up your CPU and memory. What questions do you ask?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.
2. You have a high volume of queries, and its eating up your CPU and memory. What questions do you ask?
   2.1 Are there too many database connections?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.
2. You have a high volume of queries, and its eating up your CPU and memory. What questions do you ask?
   2.1 Are there too many database connections? Can queries be queued?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.
2. You have a high volume of queries, and its eating up your CPU and memory. What questions do you ask?
   2.1 Are there too many database connections? Can queries be queued?
   2.2 Are the queries changing the data, or just reading it?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.
2. You have a high volume of queries, and its eating up your CPU and memory. What questions do you ask?
   2.1 Are there too many database connections? Can queries be queued?
   2.2 Are the queries changing the data, or just reading it?
      2.2.1 If just reading, are some queries being executed more than once?

Performance Questions

1. You have a low volume of queries, and plenty of CPU and memory, but your queries are slow. What questions do you ask?
   You see that most of the query time is spent doing IO. How can you fix this?
   1.1 If the queries are mostly reads, one possibility is to add indexes.
   1.2 If the queries are mostly writes, one possibility is to *remove* indexes.
2. You have a high volume of queries, and its eating up your CPU and memory. What questions do you ask?
   2.1 Are there too many database connections? Can queries be queued?
   2.2 Are the queries changing the data, or just reading it?
      2.2.1 If just reading, are some queries being executed more than once? If so, maybe the results can be cached.
      2.2.2 If changing, can the changes be batched? Can index maintenance be deferred?