indices

October 19, 2023

1 RDB lab: Indices

You have just heard about indices today. This lab intends to give you a quick demonstration of their benefits.

This demo will be done using SQLite. SQLite is an open-source, serverless, and lightweight database management system. It is embedded within the application that makes use of it, so it doesn't require a separate server process to operate - that's why it's often referred to as a self-contained, file-based database.

Python has built-in support for SQLite 3 with its `sqlite3` library (documentation).

[3]:
```
import sqlite3
import time
import os
```

First, we remove the file that contains the database before each run. This way, we can run the script multiple times and start with a fresh, empty db.

[4]:
```
DB_NAME = 'test_database.db'

if os.path.exists(DB_NAME):
    os.remove(DB_NAME)
    print("Database deleted successfully")
```

```
Database deleted successfully
```

1.0.1 Create & populate db

Next, we create a db table and populate it with a few million records.

This takes a little while…

[5]:
```
# Create a new SQLite3 database
connection = sqlite3.connect(DB_NAME)

# Create a new cursor
cursor = connection.cursor()

# Create a table
cursor.execute("CREATE TABLE students (name TEXT, age INTEGER)")
```

```
# Populate the table with lots of random data
for i in range(20000000):
    age = i % 60 # Let's assume the ages range from 0 to 60
    cursor.execute(f"INSERT INTO students VALUES ('Student{i}', {age})")

# Commit the changes and close the connection
connection.commit()
connection.close()
```

1.0.2 Query (without index)

Then, we query the database. First, without index.

[6]:
```
connection = sqlite3.connect(DB_NAME)
cursor = connection.cursor()

start = time.time()
cursor.execute("SELECT * FROM students WHERE age = 20 AND name LIKE 'Student%'")
results = cursor.fetchall()
end = time.time()

print(f"Time taken without index: {end - start}s")
print(f"Number of results: {len(results)}")

connection.close()
```

```
Time taken without index: 0.7092139720916748s
Number of results: 333333
```

Note the time that it took to perform the `SELECT` query.

1.0.3 Index creation

Now, let's create an index on the the `age` column of the table `students`

[7]:
```
connection = sqlite3.connect(DB_NAME)
cursor = connection.cursor()

cursor.execute("CREATE INDEX age_index ON students (age)")

connection.commit()
connection.close()
```

1.0.4 Query (with index)

Then, we query the database. This time, there is an index.

[8]:
```
connection = sqlite3.connect(DB_NAME)
cursor = connection.cursor()

start = time.time()
cursor.execute("SELECT * FROM students WHERE age = 20 AND name LIKE 'Student%'")
results = cursor.fetchall()
end = time.time()

print(f"Time taken with index: {end - start}s")
print(f"Number of results: {len(results)}")

connection.close()
```

```
Time taken with index: 0.28774476051330566s
Number of results: 333333
```

1.0.5 Results

You should observe that the query execution time with an index is significantly shorter than without indices.

I must admit that this demo is not very impressive: I get a 2-3x speed-up on my machine. In practice, adding an index to a complex query on a large database, with multiple joins usually speeds things up from a couple of minutes to a few seconds…

Caution: Indices speed up search queries but slow down inserts and updates. Therefore, use them wisely, especially in situations where reading data is more frequent than writing data.

[ ]:
```

```