CS 564 – Fall 2019 Assignment 4

zhihan

Assignment 4: Query Tuning

- Write-up is posted on course website https://kyle-klassy.github.io/cs564-fall19/assignments/p4/p4.pdf
- Slides will be posted after class
- Due Date: Nov. 22 (Friday) @11:59PM
- Individual assignment
- 15% of final grade

Assignment 4: Query Tuning

- Goal: understand query optimization from hands-on experience
- Given a query, you will rewrite it to make it as fast as possible
- Database:
    - Scaled TPC-H database (~ 1.2GB)
- Queries:
    - TPC-H Queries (#2, #3, and #4)

Steps

1. Download the database, three queries and the template report (report.txt)
2. Run each provided query 5 times and, for each query, write down the average user time as ***t_base***

Run the Provided Query

```
>> sqlite3 TPC-H.db < query2.sql
1922.82|Supplier#000000719|CHINA|8460|Manufacturer#4|nQoXFQ,z
toTyboWFmO,a|28-664-720-1497|jole about the requests. quickly
ironic

Run Time: real 0.045 user 0.033880 sys 0.010552
```

command

query results

statistics

Take down the user time and run 5 times to take the average.

Steps

1. Download the database, three queries and the template report (report.txt)
2. For each provided query, run 5 times and take down the average user time as ***t_base***
3. Any SQL statements that modify the database (e.g. “create index”) must be in ***preprocessing.sql***. And provide a ***clean.sql*** to revert all the changes you made.

Modify the Database (e.g. create indexes)

- SQL statements like this (e.g., creating indexes)

  ```
  CREATE INDEX index_name ON table(column);
  ```

  Must be put to ***preprocessing.sql***
- Prepare a file ***clean.sql*** that can revert all the changes you made on the database;
- Rules on creating indexes:

  - All the rewritten queries should share exactly the same set of indexes!

  (i.e. all three optimized queries share the same preprocessing.sql)

  - No more than three indexes can be created per table !

Steps

1. Download the database, three queries and the template report (report.txt)
2. For each provided query, run 5 times and take down the average user time as ***t_base***
3. Any SQL statements that modify the database (e.g. “create index”) must be in ***preprocessing.sql***. And provide a ***clean.sql*** to revert all the changes you made.
4. For each provided query,
    a. rewrite and measure the performance (average of 5 runs)

Rewrite and Measure the Performance

1. Rewrite it to make it as fast as possible and save rewritten query in a separate file (e.g. ***query2_opt.sql***).
   - *You can apply strategies like using a different join orders, push down grouping operations, adding indexes, etc.*
   - any SQL statements modifying the database, e.g. adding indexes, must be in ***preprocessing.sql***. (Make sure you follow the rule of having a single set of indexes for all queries.)
2. Run ***preprocessing.sql*** -> run your rewritten query 5 times and write down the average user time as ***t_opt*** -> run ***clean.sql***

Steps

1. Download the database, three queries and the template report (report.txt)
2. For each provided query, run 5 times and take down the average user time as ***t_base***
3. Any SQL statements that modifies the database (e.g. create indexes) must be in ***preprocessing.sql***. Provide a ***clean.sql*** to revert all the changes you made.
4. For each provided query,
   a. rewrite and measure the performance ***t_opt***
   b. Calculate percentage of improvement ((***t_base*** - ***t_opt***) / ***t_base***) and record in ***report.txt***

Sample of filled report.txt

## basic information

- name: [zhihan]

- netid: [zguo]

## query 2

- A Description of how you optimize the query

[There is an index created on column A of table X; I changed the join order to ...]

- The optimized query shows [90] % improvement over baseline.

Steps

1. Download the database, three queries and the template report (report.txt)
2. For each provided query, run 5 times and take down the average user time as ***t_base***
3. Any SQL statements that modifies the database (e.g. create indexes) must be in ***preprocessing.sql***. Provide a ***clean.sql*** to revert all the changes you made.
4. For each provided query,
    a. rewrite and measure the performance ***t_opt***
    b. Calculate percentage of improvement ((***t_base*** - ***t_opt***) / ***t_base***) and record in ***report.txt***
5. Final check: make sure your queries output the same results as the provided ones! Otherwise may result in ZEROs.

Turn in the Assignment

- Files:
    - query2_opt.sql
    - query3_opt.sql
    - query4_opt.sql
    - Report.txt
    - preprocessing.sql
    - clean.sql
- Put in folder named <netid>_P4; Compressed the folder into <netid>_P4.tar.gz and submit to canvas.

Rubric

- Correctness
  - Optimized queries return the correct results
  - Provided sql files do not have errors
- Run Time
  - Each query can achieve more than 0% improvement over baseline (20% * 3)
    - For average run time, we will grade it based on our own measurements over your queries. And we will only use the numbers in your report as reference. So please make the improvements a bit more significant to reduce the influences of variances.
- Other
  - At least three different types of optimizing strategies are used in the assignment. (15%)
  - no more than three indexes are used per table. (15%)
  - Follow all other instructions (10%)
    - Correct filename extension; correct preprocessing.sql and clean.sql; etc.

Q&A

Have Fun!