Automatic Index Creation

Saksham Rathi, Kavya Gupta, Shravan S, Mayank Kumar

(22B1003) (22B1053) (22B1054) (22B0933)

CS349: DataBase and Information Systems
Under Prof. Sudarshan and Prof. Suraj

Indian Institute of Technology Bombay
Spring 2024-25

Goals and what we implemented from a user's eyes

- Indexes are crucial for efficient query execution in relational databases.
- However, developers sometimes forget to create indexes for frequently queried columns.
- This can lead to repeated full relation scans, significantly degrading performance.
- **Goal:** Modify the application layer of PostgreSQL to detect such patterns and automatically create indexes when beneficial.
- **Another Goal** was to understand and implement the paper *"An Auto-Indexing Technique for Databases Based on Clustering"* [ZSG04].
- We implemented an interface that can take and submit queries from users as usual as well as automatically create (and remove) indices appropriately, thereby improving performance without any user intervention.

Calculating the frequency of relations and their attributes

For Stage 1 of this project, we made a simple parser, which gives us the list of relations, and the attributes involved in some query. This helps us know the number of times, a particular attribute of a relation is accessed, this in turn can be used to make the decision on whether we should construct an index on that attribute.
For a sample run, the `\show` command can be used to display the frequency results.

Results

Here is the sample output:

```
~/Desktop/CS349/project/code  main !2
./run
pgshell# SELECT id FROM student ;
Table name: student
Attributes: id
SELECT id FROM student ;
Query executed successfully. No results to display.
pgshell# SELECT id FROM instructor ;
Table name: instructor
Attributes: id
SELECT id FROM instructor ;
Query executed successfully. No results to display.
pgshell# \show
Table name: instructor
id 1

Table name: student
id 1

pgshell#
```

Figure: Displaying the output of show command

An Auto-Indexing Technique for Databases Based on Clustering

- Automate the physical design so that the task of the database administrator (DBA) is minimized.
- The first category is external tools which use linear programming optimization techniques and other cost minimization techniques to solve the Index Selection Problem.
- The second category is the tools that utilize the query optimizer to give cost estimates for various index configurations and suggest a configuration with the least cost estimation.
- In this technique the optimizer is invoked only once for each query in the workload to choose the final set of indexes from a set of externally determined index configurations.
-

Identifying Candidate Indexes

- A query attribute matrix is created.
- The presence of an indexable attribute is created by 1 and absence by a 0.
- The condition applied is:
  `Freq > threshold1 OR Freq * T > threshold2`
- `Freq` is the frequency of each indexable attribute in the workload and `T` is proportional to the size of the table in rows to which the column belongs.
- Weights of 3, 2, 1 are given to the columns occurring in a `WHERE` clause, `GROUP BY` or `ORDER BY` clauses and aggregate functions, respectively.
- During the clustering phase queries that are similar based on common and frequently occurring attributes are clustered together.

Candidate index suggestion

- During this phase, those candidate indexable attributes which are common to all the queries clustered together during the clustering phase are suggested as indexes.
- The optimizer uses its statistics and cost estimates to choose indexes for each query.
- Those indexes not being picked up by the optimizer are dropped because the presence of these unused indexes will cause an overhead of space and maintenance in the database.

References

M. Zaman, J. Surabattula, and L. Gruenwald.
An auto-indexing technique for databases based on clustering.
In *Proceedings. 15th International Workshop on Database and Expert Systems Applications, 2004.*, pages 776–780, 2004.