Tool Giving Intuitive and Analytical Index Suggestions

By:

Jayant Shelke

Mustafa Badshah

Muthaiah Ramanathan

Project Specifications

- Tool that analyses a workload of queries that are executed against a Database Management System and suggests indexes to improve performance.

- Console based application developed in Visual Studio, Dot Net Framework 4.5
- Back-end development database – Microsoft SQL Server
- Language Used – C#
- Sample Database – AdventureWorks2016
- Collaboration Base - Github

Motivation

- Improving the performance of database transactions remains a challenge in practice.
- Challenge is associated with how data records are organized in a storage device and how quickly those records can be accessed.
- Common task undertaken by DB Administrators is selection of indexes to speed up performances of queries submitted to DBMS.
- Difficult to manually tune performance for extremely large database schema as possible configurations grow exponentially.

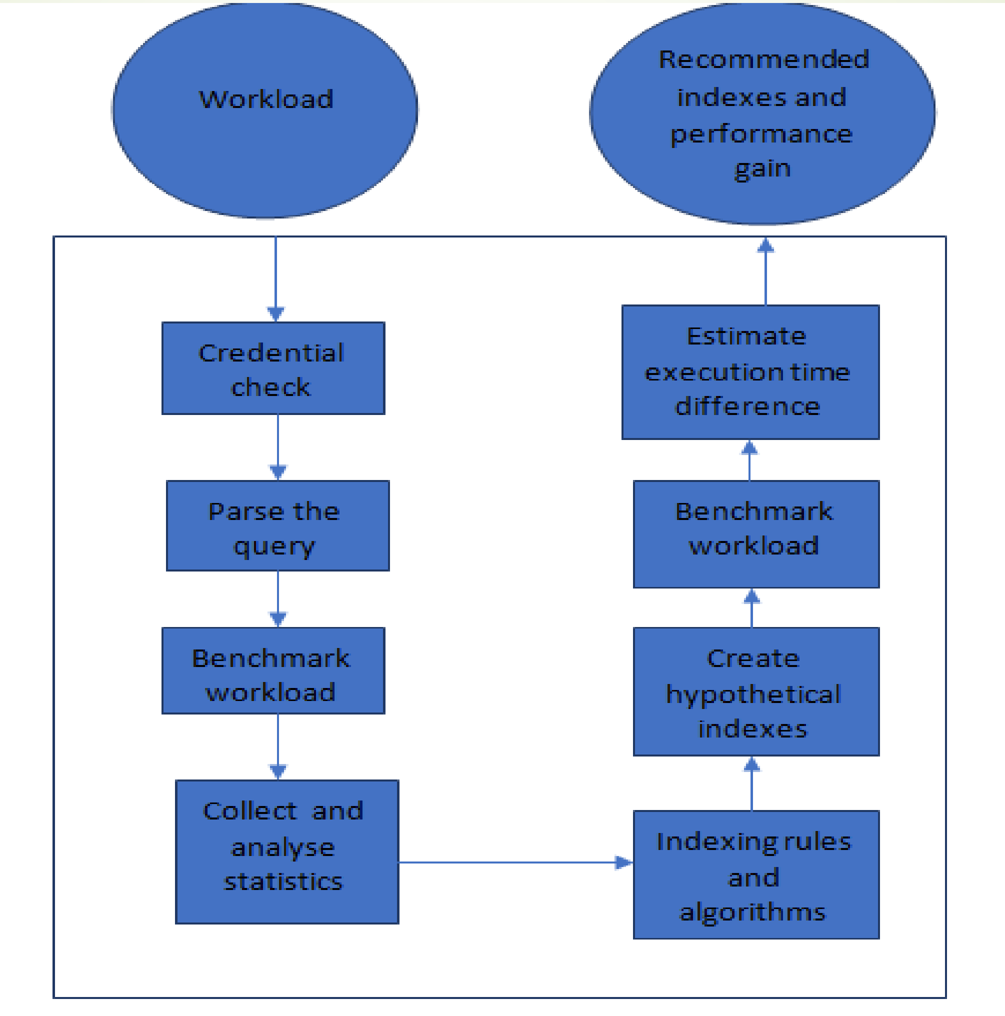
Challenges

- Index Selection is not straightforward and involves a number of factors.

- Need to take into account the workload, data model, schema design, logical organization of data records in storage devices.

- Extends on the approach of the referenced paper to cover the drawback it had.

Modules and Execution Flow


Workload
Recommended indexes and performance gain
Credential check
Parse the query
Benchmark workload
Collect and analyse statistics
Indexing rules and algorithms
Create hypothetical indexes
Benchmark workload
Estimate execution time difference

- **Workload:** The input to this project will be the workload – a set of queries that must be triggered against the database chosen.

- **Check admin privilege:** We check if the user has the permission to trigger those queries in the workload and we also validate the credentials of the user with the database

- **Parse the queries:** We parse the queries to develop the statistical data for each of the attributes in the queries.

- **Execute the workload and benchmark:** We execute the queries in the workload against the database with the system generated existing indexes and capture the statistics for each query.

- **Analyze the Workload:** We traverse through the workload and collect the various statistics like the selectivity, reduction factors, predicates, type of data, repetition of the columns, keys, etc., for the attributes and relations involved.

- **Create Hypothetical Indexes:** Based on the statistics, we rank columns of each relation to find out the columns for which index creation would seem useful. We create these indexes for the relation.

- **Bench mark the workload, again:** To quantify the success of the intuitive indexes created, we benchmark the workload again and record I/O statistics.

Index Suggestion Algorithm

- For each table:
  - For each column:
    - For each Clause:
      - If the column occurs for that specific clause we assign it a specific score based on the clause score and number of occurrences.
      - If column occurs in Where Clause:
        - Assign operator score to column depending on the operator it is associated with and number of occurrences.
        - Assign column a hash score or binary score based on association with operators.
  - Find top scoring column and binary and hash score of that column:
    - If binary score is higher → Create binary index
    - If hash index is higher → Check if difference in score between binary and hash score is more than tolerance threshold.
    - If yes → Create hash index

- Composite index:
  - If Binary Index:
    - Check all other columns of table whose binary score is higher and based on a tolerance threshold we add the column to our composite binary index suggestion.
  - If Hash Index:
    - Check all other columns of table whose hash score is higher and based on a tolerance threshold we add the column to our composite hash index suggestion.

- Clause Score
  - Select Clause – 1
  - Having Clause – 2
  - GroupBy, OrderBy – 3
  - Join – 4
  - Where – 5

- Operator Score
  - Equal – 5
  - Greater than Less than – 4
  - Like, Between – 3
  - Not Equal – 2
  - Unknown - 1

Future Works

- Graphical interface which will allow database administrators to check hypothetical performance of database
- Monitor the database workload and create or remove indexes autonomously, effectively providing self-tuning capabilities to the database system.
- Make the tool available online.
- Support for different Database Management Systems.