Recommandation Engine for SQL queries

Rahmane Ousmane

Supervisor : Xie Hongwei

College of Software
Taiyuan University of Technology

January 6, 2018

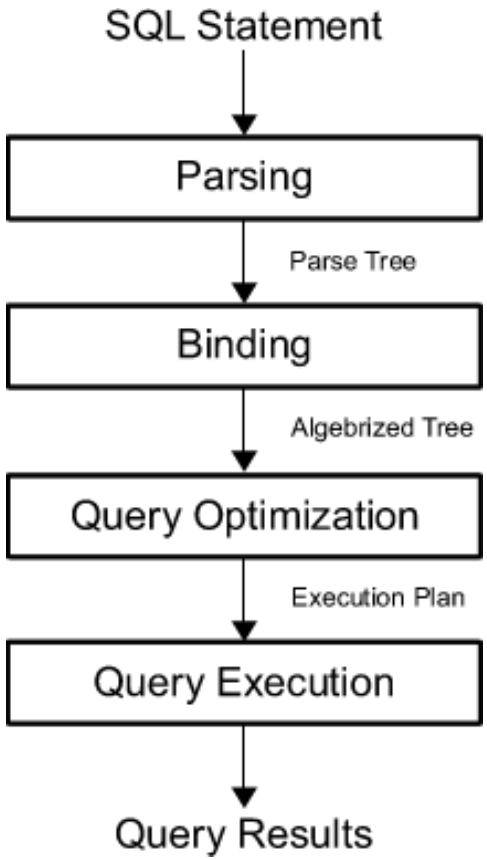
Summary

SQL influence

According to 70% of the respondents of the **2016 OReilly Data Science Salary Survey**, they use SQL in their professional context.

That survey also points out the fact that the SQL language stands out way above the R (57%) and Python (54%) programming languages.

SQL queries processing architecture

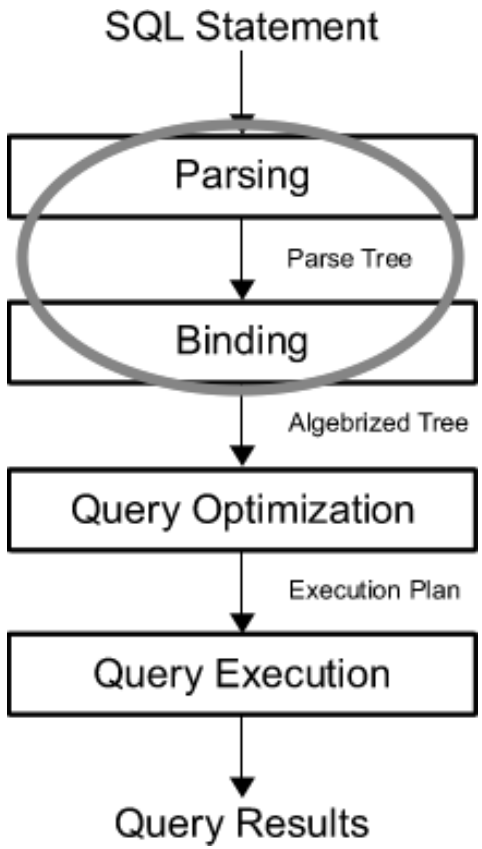


Parsing and binding

The query is parsed and bound.

Assuming the query is valid, the output of this phase is a logical tree, with each node in the tree representing a logical operation that the query must perform, such as reading a particular table, or performing an inner join.

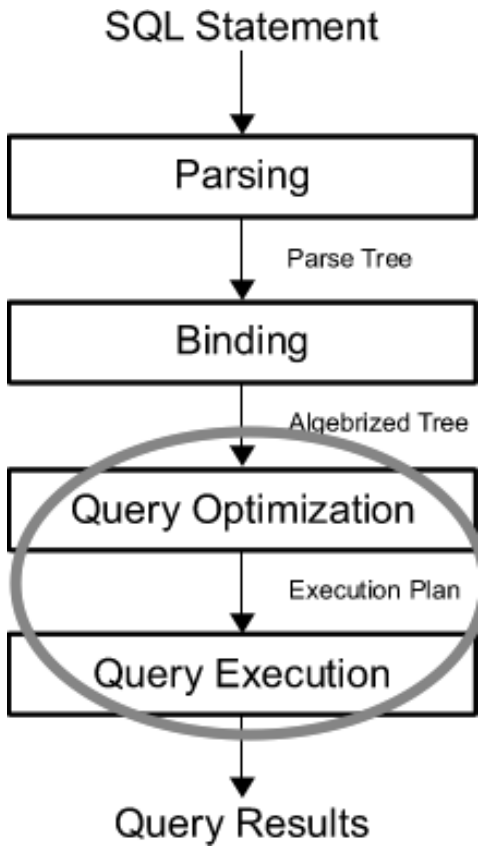
This logical tree is then used to run the query optimization process.



Generate possible execution plans

Using the logical tree, the Query Optimizer devises a number of possible ways to execute the query i.e. a number of possible execution plans.

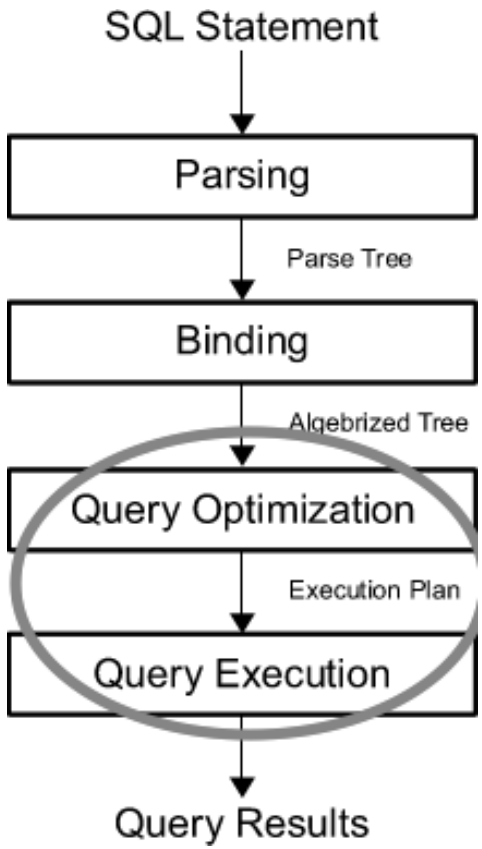
An execution plan is, in essence, a set of physical operations (an index seek, a nested loop join, and so on), that can be performed to produce the required result, as described by the logical tree



Cost-assessment of each plan

While the Query Optimizer does not generate every possible execution plan, it assesses the resource and time cost of each plan it does generate.

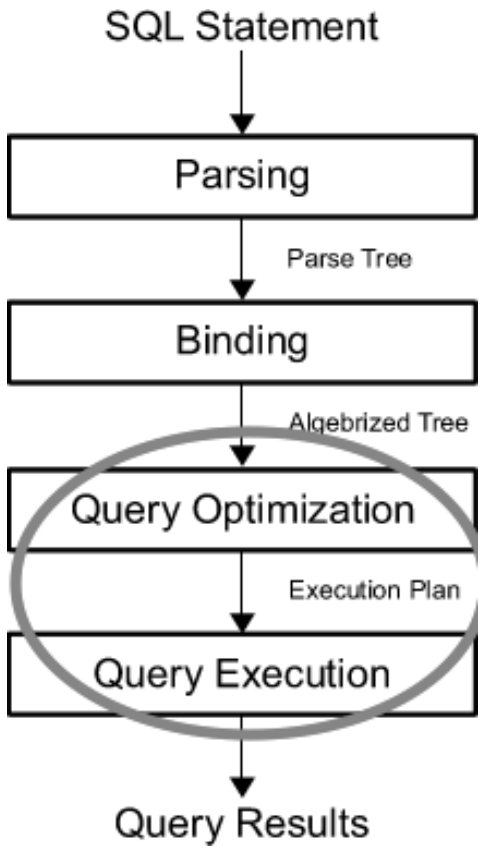
The plan that the Query Optimizer deems to have the lowest cost of those it's assessed is selected, and passed along to the Execution Engine



Query execution, plan caching

The query is executed by the Execution Engine, according to the selected plan.

The plan may be stored in memory, in the plan cache.



The query optimizer is not enough

A problem of Query Optimizer is that, even after more than 30 years of research, they are highly complex pieces of software which still face some technical challenges.

As a result, there may be cases when, even after providing the Query Optimizer with all the information it needs and there doesn't seem to be any apparent problem, we are still not getting an efficient plan.

Combinatorial Explosions

The first major impediment to a query optimizer finding an optimal plan is the fact that, for many queries, it is just not possible to explore the entire search space.

An effect known as combinatorial explosion makes this exhaustive enumeration impossible, as the number of possible plans grows very rapidly depending on the number of tables joined in the query.

To make the search a manageable process, heuristics are used to limit the search space. However, if a query optimizer is not able to explore the entire search space, there is no way to prove that you can get an absolutely optimal plan, or even that the best plan is among the candidate being considered.

Better SQL queries

The Garbage In, Garbage Out (GIGO) principle surfaces within the query processing and execution. It means that, If the optimizer gets a badly formulated query, it will only be able to do as much as it can.

Bad queries Vs Good queries : Index

An Index is a datastructure used to quickly locate or look up data without having to search every row in a table every time that table is accessed. Some SQL operators like *OR*, *NOT*, *AND* ... don't make use of indexes ; and that makes queries slower !

Bad query ; not using indexes

```
SELECT pay, name
FROM employes
WHERE pay = 123456
OR pay = 678910;
```

Better query ; using indexes

```
SELECT pay, name
FROM employes
WHERE pay IN (123456, 678910);
```

Bad queries Vs Good queries : Correlated subqueries

A correlated subquery is a subquery that uses values from the outer query.

Because the subquery may be evaluated once for each row processed by the outer query, it can be inefficient.

Bad query using subquery

```
SELECT pay, name, dep FROM agents
WHERE pay IN (SELECT pay FROM
managers WHERE dep = agents.dep)
```

Better query using INNER JOIN

```
SELECT agents.pay, agents.name,
agents.dep FROM agents
INNER JOIN managers ON
agents.dep = managers.dep
WHERE agents.pay = managers.pay;
```

Approach

In order to have good performing database querying operations, we need to write optimized SQL queries.

SQL is a big language with a big learning curve. It will take someone years of experience to be able to write very performant sql queries. Effectively writing optimized queries requires knowing top SQL statements, top wait types, SQL plans, blocked queries, resource contention, and the effect of missing indexes.

To do all this, we need an effective SQL queries recommandation tool that will take SQL queries as input and give better SQL queries as output. This is a must-have tool for writing top performing SQL queries.

Existing studies and tools

Improving SQL queries execution performance is a two-fold issue :

- Improving the database query optimizer
- Write better SQL queries

Existing works are focused on one or another face of the problem. But our approach is to focus on the SQL queries quality.

Studies

Some interesting studies about improving sql queries execution are:

- Inside the SQL Server Query Optimizer, by Benjamin Nevarez
- Database Tuning Advisor for Sql Server 2005, Toronto University, SIGMOD 05
- Reducing the Braking Distance of an SQL Query Engine, Michael Cary, IBM Research Cente

Downside:
Most of the studies are focused on improving the database query optimizer but not on helping writing high quality sql queries.

Tools

Some interesting tools about improving sql queries execution are:

EverSQL : `https://www.sqlserver.com`

SQL : `https://sql-tuning.com`

Percona : `https://www.percona.com/`

DataPine : `https://www.datapine.com/sql-query-analyzer`

Downside:
These tools are very limited. They parse the queries to find some keywords or patterns and make some not very accurate recommendations based on that.

Why is our Approach better ?

Our approach is better than focusing on the query optimizer because :

- Despite many years of research, the query optimiser is still facing some very complex challenges
- Each RDBMS vendor, implement its own proprietary query optimizer, thus it will be very hard to standardize any improvements
- Better SQL queries quality, will indeed lead to better execution performance
- Our tool will be 'database-independant', since the same queries will run on several databases
- Our tool will make developers and database administrators care more about their sql queries quality
- Our tool will be powered by machine learning. Thus, it will be able to learn and improve by itself

Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

We have chosen to use machine learning because:

- It gives a system the ability to keep learning and improving by itself.
- It's a fast-growing field
- The more the system is used the better it gets

Machine Learning applications

Some of the machine learning applications are:

- Image Recognition
- Speech Recognition
- Medical Diagnosis
- Statistical Arbitrage
- Learning Associations
- Classification
- Prediction
- Extraction
- Regression

Machine Learning algorithms

Some of the machine learning algorithms classes are:

- Supervised machine learning algorithms
- Unsupervised machine learning algorithms
- Semi-supervised machine learning algorithms
- Reinforcement machine learning algorithms

Applications

Our recommandation engine, will combine two main machine learning applications :

Extraction :
The system will be able to extract useful informations from queries it receives as input and represent them in its own way.

Prediction :
The system will be able use extracted informations from queries, to eventually predict a better version.

Steps

For each application (Extraction and Prediction), we will be going through the following steps.

1. Gathering data
2. Preparing the data
3. Choosing a model
4. Training
5. Evaluation
6. Hyperparameter tuning
7. Prediction or Extraction

Steps

We need to gather a huge set of data here.

That data will mainly be a set of queries and better versions of these queries.

Preparing the data

Here, we will focus on randomizing the data and filter it, to make sure we only keep the relevant records.

Choosing a model

There are many models that researchers and data scientists have created over the years. Here, we are going to choose a machine learning model that suit our case. There is a big am

Training

In this step, we will use our data to incrementally improve our model's ability to match a sql query with a better version of it.

Evaluation

Once training is complete, it's time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

Hyperparameter tuning

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

Prediction

Machine learning is using data to answer questions. So Prediction, or inference, is the step where we get to answer some questions. This is the point of all this work, where the value of machine learning is realized.

Bibliography

O'Reilly 2016 Data Science Survey

Inside the SQL Server Query Optimizer, by Benjamin Nevarez

Reducing the Braking Distance of an SQL Query Engine, Michael Cary, IBM Research Center

Optimization of nested SQL queries revisited, Harry K.T Wong, Univ. of California, Berkeley, SIGMOD 87

An Overview of Machine Learning, Jaime G. CarbonellRyszard S. MichalskiTom M. Mitchell, Symbolic Computation book series

Machine Learning for Information Extraction in Informal Domains, Dayne Freitag, Kluwer Academic Publishers

Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems : Sholom M. Weiss, Casimir A. Kulikowski, Morgan Kaufmann Publishers Inc

Oracle SQL High-Performance Tuning, Second Edition, Guy Harrison, The ACM Guide to computing Literature

[Patent] SQL performance analyzer, Peter Belknap, Benoit Dageville, Karl Dias, Khaled Yagoub, Oracle International Corporation

[Patent] Database execution cost and system performance estimator , Rainer Eberhard, Harold Hall, Seetha Lakshmi, International Buisness Machines Corporation

Thank you!