# Measuring Query Complexity in SQLShare Workload

**Aditya Vashistha**
adityav@cs.washington.edu

**Shrainik Jain**
shrainik@cs.washington.edu

## INTRODUCTION

The *Database-as-a-Service* paradigm has gained a lot of popularity in the past few years[1]–[5]. There are multiple providers for a generic *Database-as-a-Service* like SQLAzure[1], Amazon RDS[3], Fusion Tables[4] & Big Query[5]. These systems are used by experts and non-experts for wide variety of tasks. Amongst them, the most popular use of these systems are for analytical and science workloads. We analyze one such workload on our homegrown *Database-as-a-Service* platform, SQLShare[6], for long tail sciences, with the aim to find ways of measuring complexity of queries written by SQLShare users. In past few years, SQLShare has seen a diverse workload of real science queries. Current benchmarks for evaluating the complexity of queries focus solely on system performance. In order to build systems that focus on user-productivity, there is a need to analyze the workloads on current systems from the point of view of how easy it is for users to use them.

In this paper, we provide an analysis of queries from the SQLShare workload, to come up with an empirical formula to measure the query complexity as a function of the *cognitive load on users* and *performance load on the system*. Having a measure for query complexity, helps us measure users' skill of writing queries and how it changes over time. It will also enable us writing smarter query recommendation systems for suggesting queries to users they understand better on the basis of their SQL skill level. Another use of this work is to have a quantifiable proof for the hypothesis that ad-hoc queries by non-experts can be *"complex"* too.

Our work has the following contributions:

1. We determine the metrics that impact query complexity.
2. We write a tool for analyzing the importance of these metrics.
3. We adapt Halstead measures for determining complexity of SQL queries.
4. We determine an empirical formula for measuring query complexity in high-variety workloads.
5. We compare the query complexity perceived by users (ground truth) with the query complexity determined using the metrics and their relative importance (step 1 and step 2), the query complexity computed using Halstead measures (step 3), and the query complexity computed using the empirical formula (step 4).

This paper is structured as follows. In the next section we provide a concrete definition of what we mean by query complexity and how *user-intent* is missing from the current definition. Following that, we present various properties of a query that affect query complexity, and mechanisms to measure them. We also look at what experts think are the most important measures of complexity. Next we look at methodologies to calculate the relative importance of these metrics in determining the query complexity. Following this are 4 ways of measuring query complexity for a subset of SQLShare queries and an evaluation of these measures and their correctness when compared to ground truth. We end with a discussion on the future directions and conclusions.

## QUERY COMPLEXITY

Query complexity is often measured in terms of the resources required by a database server for executing the query. All database systems measure the complexity of a query in terms of space and time required for executing a query in the query optimization phase. However, we are interested in measuring the complexity of a query from the perspective of database users who are authoring these queries. For database users, we define query complexity as the cognitive load on a user while writing the query. In this paper, we will refer to *query complexity* as the complexity of a query from the perspective of a user rather than the system.

## FINDING A SET OF REPRESENTATIVE QUERIES FROM SQLSHARE

SQLShare is a *Database-as-a-Service* for long tail science. It allow users to upload raw tabular data and write SQL queries on it, without the any need for setup and schemas. The users don't need to worry about schema, the system automatically generates one. Howe et. al.[6], [7] have shown how SQLShare has resulted in improvement of productivity of researchers across multiple labs at the University of Washington. SQLShare workload consists of queries written by experts and non-experts (a total of 1178 users). SQLShare has a unique high-variety workload[8], with 19000 queries (18500 distinct queries) over 6434 tables and 66632 columns across all tables.

We have been trying to understand the properties this workload with the eventual aim of building better systems for handling high-variety data. In the context of this paper, we will talk only about the analysis of query complexity for SQLShare workload. We rely heavily on the tools we wrote for the overall analysis of SQLShare. One such tool is the QWLA (Query WorkLoad Analysis) tool[9]. Details of how the tool works are not in the scope this paper, but at a high level, a part of its implementation does the following:

- Take a set of queries on a system (like SQLShare) as an input.
- Generate XML query plans.
- Parse them to extract interesting metrics from the queries like number of operators and number of expressions.

Finding a set of representative queries from a high variety workload present a challenge because we want to make sure we get a subset of queries that are most diverse. In order to do this, we analyzed SQLShare query dataset and looked at the ordered list of queries based on each metric. We generated samples from each list capturing queries with high and low values of metrics. As a single query can have multiple metrics with high (or low) values, we got duplicated queries as well. Removing these duplicates from a list of 180 queries resulted in a query set of 117 unique queries. We refer to this set as the representative set of queries. An example query looks like the following:

```
SELECT *
FROM   (SELECT sql_query,
               sql_query_hash,
               x_col_name,
```

```
                y_col_name,
                vizlet_type,
                Row_number()
                  OVER (
                    partition    BY    sql_query,
sql_query_hash, x_col_name,
                  y_col_name
                    ORDER BY score DESC) AS rnk
        FROM   [billhowe].[vizlet scores]) x
WHERE  x.rnk > 1
ORDER  BY sql_query,
          sql_query_hash,
          x_col_name,
          y_col_name,
          x.rnk DESC
```

In the following sections, we will look at how we used this query set to figure out what metrics contribute most to the complexity of queries, what metrics are most important from the point of view of database experts, and a comparison of these.

## METRICS FOR QUERY COMPLEXITY

In our prior work, we have provided an operational definition of variety based on the observation that the complexity of user intent and user effort matters as much as the complexity of the data itself—that variety is a function of both workload and data. We analyzed three relational workloads (SQLShare, SDSS, TPC-H) representing different points on the variety spectrum to evaluate candidate metrics for quantifying query complexity and schema complexity, and therefore effort. We proposed a Variety Coefficient V with respect to an environment Env consisting of a dataset D, a catalog S and a set of tasks T as

$$V_{Env}(D; S; T) = \Omega catalog(S) + \Omega code\,(S,T) + \Omega manual\,(D,S,T) + \Omega execution(D,T)$$

Here Ωcatalog(S) is the effort to comprehend the catalog, Ωcode (S,T) is the effort to write the code, Ωmanual (D,S,T)is the effort to perform any manual work, and Ωexecution(D,T) is the effort to run the code. Together, the sum of these dimensions determine V.

There are several metrics that could be derived from a query for estimating Ωcode (S,T). Some of them include number of operators, number of distinct operators, types of operators and expressions, number of tables, number of columns, and number of nodes per weakly connected component in a table connect graph[10]. Many of these metrics are also relevant for measuring query complexity. For our class project, we have selected the following metrics to impact the query complexity:

- Number of tables in a query or table touch
- Number of columns in a query or column touch
- The length of a query
- The number of operators in a query like Scan, Join, Filter
- The number of expression operators in a query like LE, LIKE, GT, OR, AND, Count*
- The runtime of a query

We posit that a query with higher number of tables, higher number of referenced columns, longer length and runtime, and higher number of operators and expressions will considered to be more complex in nature. We have selected these metrics on the basis of our intuition and past experiences in authoring the queries.

## Relative Importance of Metrics

We conducted a survey of database experts to identify the relative importance of metrics for query complexity. We send an email to members of database research group (~25 members) at the University of Washington and requested them to complete an email survey. In the survey, we asked the experts to rate each of the six metrics on a scale of 10 to indicate how well the metric determine the complexity of a query as per their experiences. We requested them to assign a high score to indicate high correlation between the query complexity and the metric, and a low score to indicate a weak correlation. A high correlation implied that as per their experience, the measure significantly impact the overall query complexity.

Eight experts consisting of two faculty, one post-doctoral researcher and five database PhD students responded to our survey. The scores for each of the experts are indicated in Table 1. We also computed the mean score and normalized score for each metric. We also analyzed the outcomes of the survey on three dimensions:

1. How much database experts agree to each other on the relative importance of metrics?
2. How similar or different is the importance of various metrics?
3. Is there any correlation among metrics on the basis of user rating?

### *Agreement between Experts*

In order to measure agreement in the scores of database experts, we performed Kendall's coefficient of concordance (Kendall's W) test. If the test statistic W is 1, then all the survey respondents have been unanimous, and each respondent has assigned the same order to the metrics. If W is 0, then there is no overall trend of agreement among the experts, and their responses may be regarded as essentially

**Table 1: Scores by Database Experts**

| | Domink | Magda | Bill | Shrainik | Brandon | Prasang | Sudeepa | Laurel | Avg. | Norm. Score |
|---|---|---|---|---|---|---|---|---|---|---|
| Table Touch | 8 | 5 | 6 | 7 | 3 | 5 | 7 | 8 | 6.12 | 0.18 |
| Column Touch | 7 | 4 | 6 | 7 | 5 | 1 | 6 | 2 | 4.75 | 0.14 |
| Length | 6 | 5 | 7 | 8 | 9 | 8 | 6 | 2 | 6.38 | 0.19 |
| Number of Operators | 7 | 7 | 9 | 8 | 3 | 1 | 7 | 9 | 6.38 | 0.19 |
| Number of Expressions | 9 | 3 | 8 | 9 | 5 | 10 | 5 | 6 | 6.88 | 0.20 |
| Query Runtime | 2 | 5 | 2 | 3 | 7 | 1 | 6 | 4 | 3.8 | 0.11 |

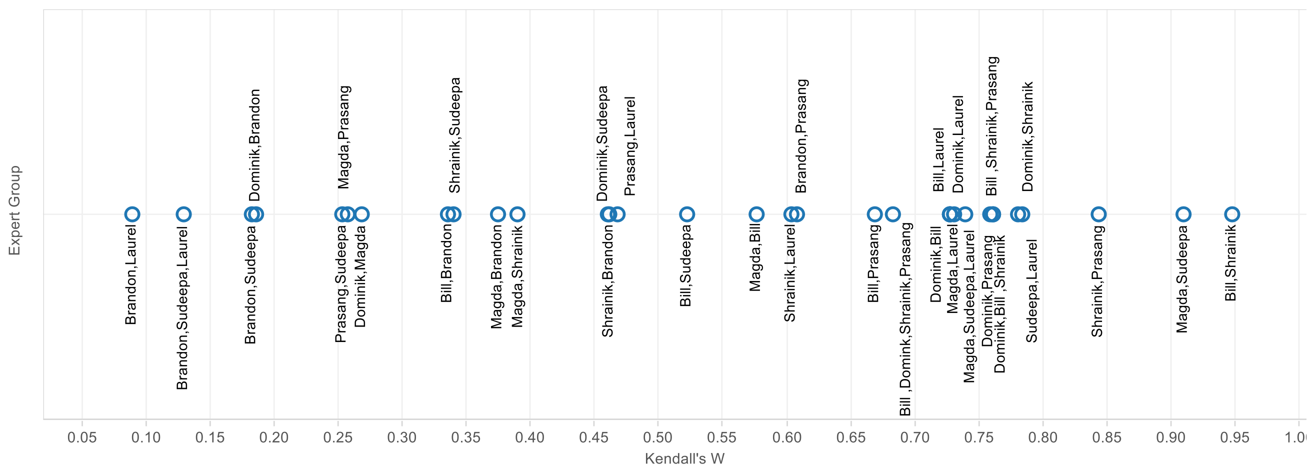



**Figure 1: Measuring Agreement among Experts in Ranking Metrics**

random. Intermediate values of W indicate a greater or lesser degree of unanimity among the various responses. For our survey, we find a lesser degree of unanimity among the experts (W=0.19, p = 0.1, n.s).

This prompted us to determine the agreement between various subsets of our expert group. We considered all pairs of experts (28 pairs) and four sub-group of three experts and one sub-group of four experts. We evaluated Kendall's W for 33 sub-group. The value of W for each sub-group is presented in Figure 1. The figure indicates the consensus among several people in the database group. The agreement between Shrainik and Bill, Magda and Sudeepa was highest wherein the agreement between Brandon and Laurel was lowest.

### *Important Metrics*

We also wanted to measure how similar or different metrics are from each other. We conducted Friedman test to evaluate the chance that random sampling would result in sums of ranks as far apart as observed in this experiment. If the p value is small, we can reject the idea that all of the differences between metrics are due to random sampling and conclude instead that at least one of the metrics differ from the rest. On conducting the test ($\chi 2(5) = 7.596$, $p = 0.18$, n.s.), we found no compelling evidence that the metrics differ from each other. The failure to achieve statistical significance could also be because of insufficient data.

### *Correlation among Metrics*

We were also interested in measuring any evidence for correlation among several metrics on the basis of expert rating. We conducted Kendall rank correlation coefficient and Spearman's rank correlation coefficient tests to measure the correlation between metrics. The Kendall's tau ($\tau$) coefficient is -0.72 ($p < 0.05$) and Spearman's rho coefficient is -0.867 ($p < 0.01$) indicating a statistically significant negative correlation between runtime of a query and number of expressions.

## COMPUTING QUERY COMPLEXITY

### Using Ground Truth Coding

We computed the complexity score for 117 distinct queries in the representative set. We rated each query on a scale of 100 and assigned a higher number to indicate higher complexity. We considered cognitive effort in writing the queries as the measure of complexity. Before assigning the complexity score, we read each query and also accessed various features of the query like number of tables, number of columns, number of expressions etc. The rating process consisted of three passes. In the first pass, the authors rated all the queries independently. In the second pass, both the authors reviewed the score of the other one to highlight inconsistent ratings given by the other independent of their score. In the third pass, the authors reviewed the queries highlighted by the other author to either revise their rating or keep the old rating.

We computed the final complexity score as the average of the two ratings and normalized it on a scale of ten. The mean score of the distribution is 4.79, the minimum score is 0.45, the maximum score is 9.8 and the median score is 4. We ran two non-parametric test, Kendall rank correlation coefficient and Spearman's rank correlation coefficient, on the ranks obtained using the complexity score of first author and the ranks obtained using the complexity score of the second author. The results indicate significant correlation (Kendall's tau_b =.875, $p < 0.01$; Spearman's rho = .965, $p < 0.01$) in the complexity rank computed on the basis of scores assigned by the authors and thus, high agreement among them. We use these complexity scores as our ground truth for further experiments. Some example query ratings are:

**Query:**

```
SELECT *
FROM   [bomanis@washington.edu].[testdatalist.csv]
WHERE  column2 < 150
   AND column1 > 1200
```

**Rating:**

**1.1**

**Query:**

```
SELECT *
FROM   (SELECT sql_query,
               sql_query_hash,
               x_col_name,
               y_col_name,
               vizlet_type,
               Row_number()
                 OVER (
                   partition     BY     sql_query,
sql_query_hash, x_col_name,
                 y_col_name
                   ORDER BY score DESC) AS rnk
```

```
        FROM    [billhowe].[vizlet scores]) x
WHERE   x.rnk > 1
ORDER   BY sql_query,
           sql_query_hash,
           x_col_name,
           y_col_name,
           x.rnk DESC
```

**Rating:**

**4.25**

Our calculation for ground truth complexity score has limitations because of a fewer number of coders and potential bias. The ground truth score could be coded more rigorously by 1) including more experts for coding the complexity score, 2) conducting a user study where users are requested to explain the query and then code the complexity score, and 3) series of experiments where users are asked to write several queries and then code the complexity score for each of the query. In this paper, because of time constraints and lack of IRB approval, we will use the ground truth calculated by us.

## Using Experts' Ranking

As described in table 1, we computed a normalized weight for each metric by analyzing the scores given by experts. For all queries in the representative set of queries, we also computed the values for each metric. We computed the complexity score as the summation of multiplications of weight for each metric and its value.

$$Complexity\ Score = \sum W_i * V_i$$

Here $W_i$ is the normalized weight for a metric and $V_i$ is the value for that metric. For example, if a query references 1 table and 3 columns, and has 1 operator, 3 expressions. 0-second runtime and a length of 97 characters, then the complexity score will be computed as:

0.18*1+0.14*3+0.18*3+0.20*3+0.11*0+0.19*97=19.44.

We computed a ranked-order of queries by using the complexity scores computed on the basis of experts' vote.

## Using Halstead Measures

Halstead measures are software metrics used for estimating the number of errors in a program and measuring complexity in a program. Cyclomatic complexity measures are also used for estimating the number of defects in a program and determining the complexity of a code. Cyclomatic complexity is computed using the number of edges, vertices and loops in a control flow graph. Halstead complexity is computed using the number of operators and operands in a program. Though these measures are not used to measure the complexity of SQL queries, we adapted the Halstead complexity measure to determine the query complexity. The query complexity is calculated as:

$$Query\ Complexity = \frac{n1}{2} * \frac{N2}{n2} * \log_2(n1 + n2)$$

Here n1 is the number of distinct operators, n2 is the number of distinct operands, N1 is the total number of operators and N2 is the total number of operands. We considered number of columns referenced in a query as the operands, and number of operators and expressions as Halstead operators for computing the Halstead complexity score for each query. We computed a ranked-order of queries in the representative set by using the complexity scores computed on the basis of Halstead measures.

## Regression for Query Complexity Formula

We ran linear regression on the labeled complexity scores (ground truth) and the values of metrics for 117 queries. A complexity score (QC) is dependent on the 6 metrics: number of tables ($N_{table}$), number of columns ($N_{column}$), query string length ($Q_{length}$), number of operators ($N_{operator}$), number of expressions ($N_{expression}$) and query runtime ($Q_{runtime}$).

$$QC = a * N_{table} + b * N_{column} + c * Q_{length} + d * N_{operator} + e * N_{expression} + f * Q_{runtime}$$

To estimate the values of these coefficients, we used regression with 10-fold cross validation. The regression yielded the following coefficients:

*a = -0.00248, b = 0.000168, c = 0.001571, d = 0.012903, e = 0.000355, f = 8.96E-07.*

Based on this model, we conclude that the most important metric in measuring the query complexity is the number of operators, followed by the query length. The query runtime is almost negligible in determining the complexity of a query.

It is also interesting to note that the number of tables were assigned a negative weight while determining the overall complexity score. This indicate that the number of tables may not be an important metric to determine the query complexity or its importance might be compensated by other metric like number of columns or number of operators. We found that the simple linear regression yielded a very high RMSE even on training data, suggesting that either linear regression is possibly not a good tool or the features are not linearly related or the data is faulty/insufficient. For future work, we will look at non-linear models to mitigate these limitations.

Using the formula obtained using the regression, we computed complexity score for each query in the representative set and ranked them on the basis of complexity score.

# EVALUATING QUERY COMPLEXITY

In this section, we compare the ranks of the query set obtained using

- Ground truth query complexity score
- Complexity score computed using experts' rating of metrics

  Complexity score computed using Halstead measure
- Complexity score computing using formula obtained from regression

We conducted Kendall rank correlation coefficient and Spearman's rank correlation coefficient tests to measure the correlation between several complexity ranks. The results are depicted in Table 2. We found a statistically significant correlation between the rank-order obtained using ground truth score and the rank obtained using Halstead measure. The results also indicate that the ranks obtained using our version of Halstead measures (that relies on number of columns, expressions and operators in a query) has the most agreement with the rank obtained using ground truth score. We also saw a significant correlation among complexity rank computed using empirical formula derived from regression and ground truth.

Another way of comparing different measures is to analyze the trends in complexity vs individual metrics. In Figure 2, 3 and 4, we show how change in values of different metrics relate to the overall complexity score of the query. This comparison is done for both complexity score obtained using experts' score and ground truth complexity score. The key things to note from these graphs are:

- When a metric is considered individually, increase in that metric alone should increase the overall complexity.

**Table 2: Correlation among Rank-order Obtained using Various Complexity Scores**

| | | Ground Truth | Expert Metrics | Halstead | Regression |
|---|---|---|---|---|---|
| Kendall's tau_b | Ground Truth | 1 | .799** | .832** | .826** |
| | Expert Metrics | .799** | 1 | .785** | .872** |
| | Halstead | .832** | .785** | 1 | .751** |
| | Regression | .826** | .872** | .751** | 1 |
| Spearman's rho | Ground Truth | 1 | .942** | .956** | .958** |
| | Expert Metrics | .942** | 1 | .932** | .958** |
| | Halstead | .956** | .932** | 1 | .914** |
| | Regression | .958** | .958** | .914** | 1 |

** Correlation is significant at the 0.01 level (2-tailed)

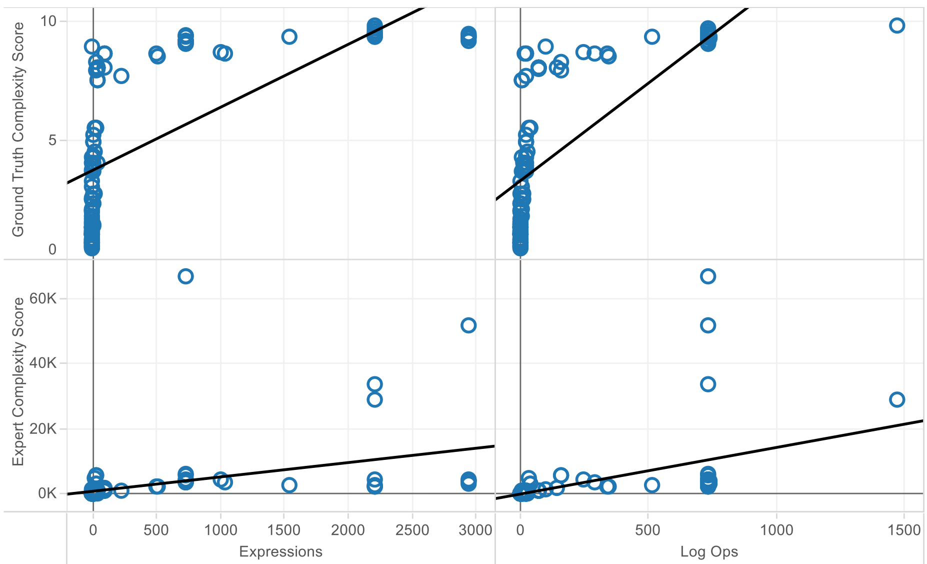
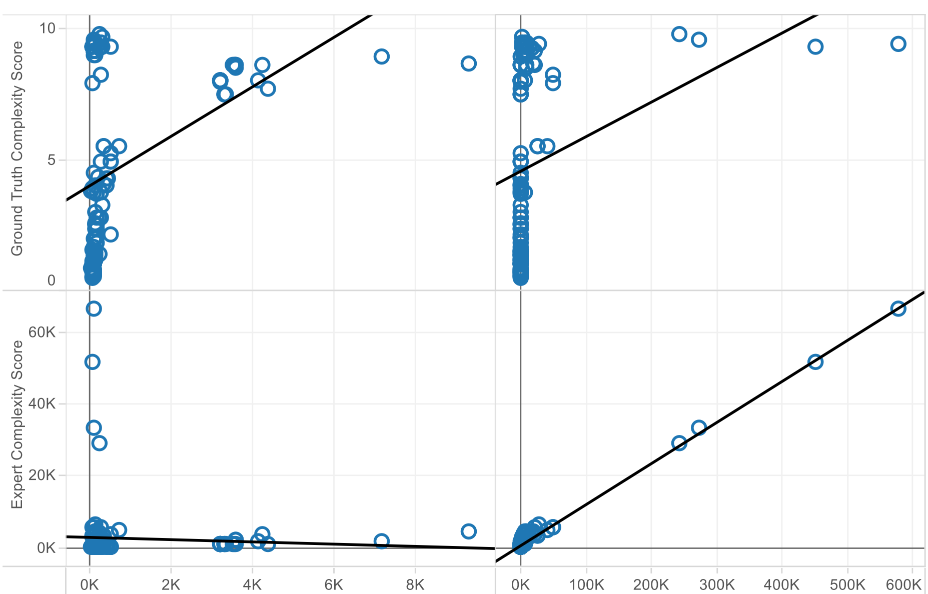



**Figure 2: Comparison of Ground Truth vs Expert Scores for Expression and Log Ops**



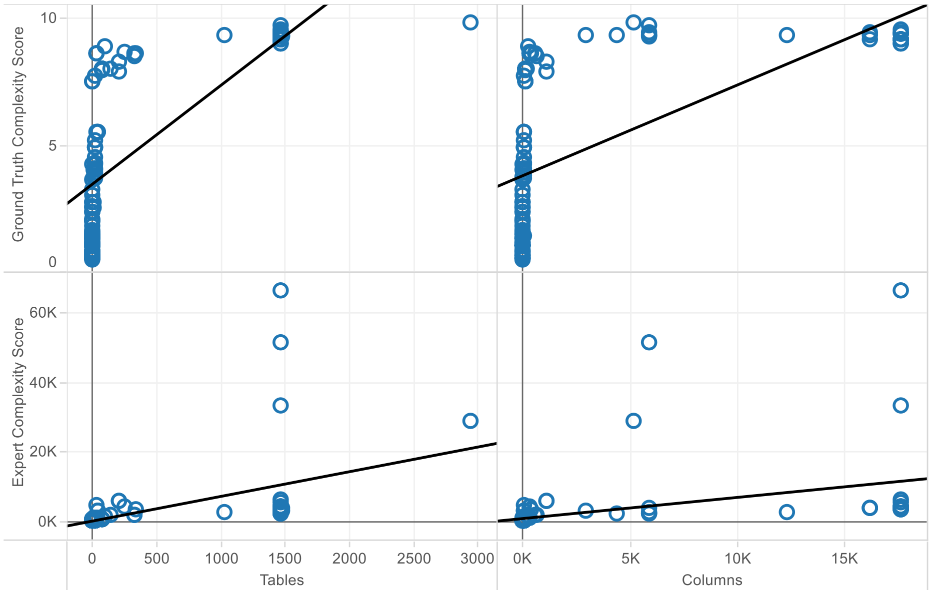



**Figure 4: Comparison of Ground Truth vs Expert Scores for Tables and Columns**

- The ground truth is available for queries that are either very complex, or very simple, presenting a bi-modal graph. We expect that with more diverse queries in the query set for ground truth, we will see better results and thus generate a better model.
- Experts tend to give query length lower importance probably because it doesn't relate directly to how complex a query plan is going to be, but users find it difficult to write longer queries which is probably the reason for a negative slope of trend line in Figure 3.

# RELATED WORK

Complexity of queries in terms on cognitive load on the users is relatively untouched. This is because most of the database systems research is focused on performance of systems rather than productivity of users. Over the past few years, even non-computer scientists have realized the power of databases, as Howe et. al. [6][11] showed the power of building systems which required minimal setup. With this new found interest in this field, we feel that its time user productivity and ease of use of databases become 'first class design constraint'. To measure productivity and ramp up efforts of users we need a strong quantitative measures of complexity of composing queries. Siau et. al. [12][13] looked at cognitive mapping techniques for user-database interaction, and effects of query complexity on novice database users, but concrete query complexity measurements were missing from their work.

Cyclomatic Complexity [14] and Halstead complexity [15] have been referred in literature as usual ways to measure the complexity of code. Our work is motivated from the fact that it is important to measure the complexity of the code both from the users' point of view and systems' point of view.

Query optimizers analyze the complexity of queries internally. However, this analysis focuses on runtime and space complexity of queries and completely ignores the efforts users have to put in writing queries. Our eventual goal is to make better systems that give same amount of importance to user productivity as they give to system performance. Another reason for coming with newer complexity metrics is to have measures specific to relational queries, something that Cyclomatic complexity isn't designed for.

[16] and [17] show how usage logs and information about user sessions can be used to infer properties of a workload as it evolves over time. We use the similar approach of looking at logs, but our aim is to find out ways to measure the effort required by users to write a query. The idea of measuring query complexity based on user effort is new and few researchers have addressed it.

# CONCLUSION AND FUTURE WORK

Query complexity is inherently difficult to quantify. We showed in this work how even experts in the field have different notions of what counts as complex. We presented different ways to measure query complexity and computed the accuracy of each way by comparing it with the ground truth (hand labeled queries). Our current analysis points out factors that are most important for query complexity (i.e. number of operators & expressions) and, factors least important (i.e., query runtime). We also presented a way in which Halstead complexity can be calculated for relational queries. We found that this measure works marginally better than just plain regression. This further hints that operators and expressions in a query are indeed the dominating factors in computing query complexity.

Our efforts to compute query complexity is definitely a major step in the right direction, however there is still room for improvements. In future, we will consider more number of queries and much diverse queries while computing query set for measuring ground truth. We will also improve the methodology to compute ground truth complexity score by conducting user experiments and taking inputs from more experts. We will also compute models more complex than simple linear regressions for determining important metrics for computing query complexity.

The analysis of query complexity is fundamental for making better systems. We plan to use this analysis in a number of ways in future. The measure of query complexity has several implication in benchmarking workloads or users. Coming up with a benchmark for science workloads that focus on user cognitive load require us to find a representative set of queries from workloads containing

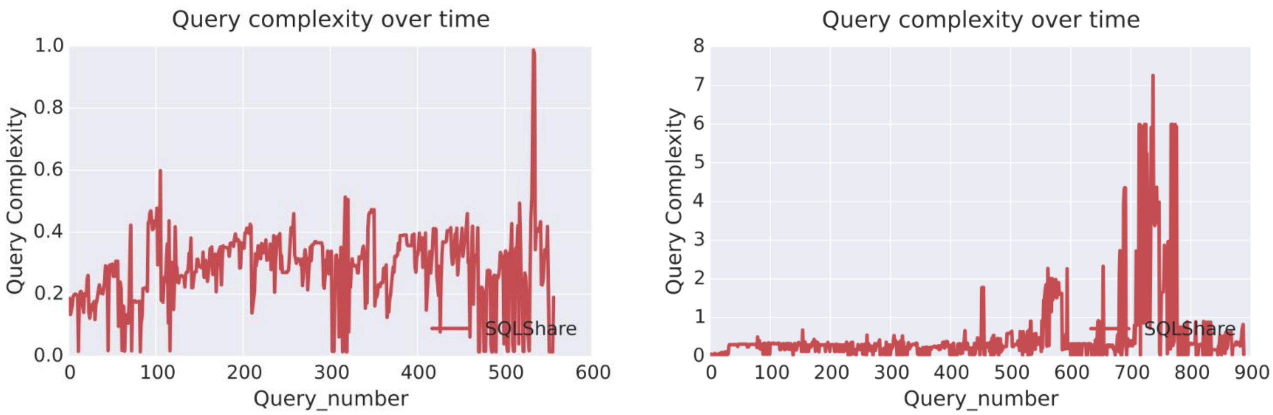



**Figure 5a and Figure 5b Complexity over time for 2 sample users from SQLShare**

queries of varying complexity. This is would only be possible after we have a way to measure the complexity of queries. There are several other implications of measuring query complexity for high-variety workloads. The measures of query complexity can be used for measuring how SQL skill of database users evolve over time by tracking the complexity of queries written by them. This can be particularly useful in classroom settings for tracking the progress of students. The measure of query complexity can also be used for making smarter query recommendation. Once we measure the SQL skill set of a user on the basis of complexity of queries the user is authoring, we can recommend auto-completion and suggestions that relates well to the level of skills of the user.

Figures 5 show some early results for a sample use case of query complexity analysis, i.e. complexity variation over time. We generated these figures by calculating complexity using the empirical formula found by regression. In figure 5a we see a user who has never written a query with complexity more than 1 (on a scale of 10). A query recommendation system should ideally take this into account and not recommend queries which are too complex. Some users (say students in a Database class) learn SQL over time, however it is difficult to prove this quantitatively. With a measure of query complexity we can now see this variation chronologically. Figure 5b shows this for a real user from SQLShare who is has learned SQL overtime.

In future, we will explore these directions further.

## REFERENCES


[1] "SQLAzure." [Online]. Available: http://azure.microsoft.com/en-us/services/sql-database/.

[2] "Azure tables." [Online]. Available: http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-tables/#what-is-the-table-service.

[3] "Amazon RDS." [Online]. Available: http://aws.amazon.com/rds/.

[4] "Fusion Tables." [Online]. Available: https://support.google.com/fusiontables/answer/2571232.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst. TOCS*, vol. 26, no. 2, p. 4, 2008.

[6] B. Howe, G. Cole, E. Souroush, P. Koutris, A. Key, N. Khoussainova, and L. Battle, "Database-as-a-Service for Long-Tail Science," in *Scientific and Statistical Database Management*, J. B. Cushing, J. French, and S. Bowers, Eds. Springer Berlin Heidelberg, 2011, pp. 480–489.

[7] B. Howe, G. Cole, N. Khoussainova, and L. Battle, "Automatic example queries for ad hoc databases," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011, pp. 1319–1322.

[8] "Gartner's 3D Data Management," *3D Data Management*. .

[9] *Query Workload Analysis*. .

[10] S. Jain, D. Moritz, B. Howe, and D. Halperin, "High Variety Data Management." [Online]. Available: https://www.dropbox.com/s/73t3l07w81qg13g/paper.pdf?dl=0.

[11] A. Key and B. Howe, "VizDeck: Self-Organizing Data Dashboards for eScience."

[12] K. Siau and X. Tan, "Cognitive mapping techniques for user-database interaction," *IEEE Trans. Prof. Commun.*, vol. 49, no. 2, pp. 96–108, Jun. 2006.

[13] K. L. Siau, H. C. Chan, and K. K. Wei, "Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces," *IEEE Trans. Syst. Man Cybern. Part Syst. Hum.*, vol. 34, no. 2, pp. 276–281, Mar. 2004.

[14] T. J. McCabe, "A Complexity Measure," *IEEE Trans. Softw. Eng.*, vol. SE-2, no. 4, pp. 308–320, Dec. 1976.

[15] "Halstead complexity measures," *Wikipedia, the free encyclopedia*. 24-Oct-2014.

[16] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "SnipSuggest: Context-aware Autocompletion for SQL," *Proc VLDB Endow*, vol. 4, no. 1, pp. 22–33, Oct. 2010.

[17] K. Morton, M. Balazinska, D. Grossman, R. Kosara, and J. Mackinlay, "Public Data and Visualizations: How Are Many Eyes and Tableau Public Used for Collaborative Analytics?," *SIGMOD Rec*, vol. 43, no. 2, pp. 17–22, Dec. 2014.