Cover Letter of

"Towards a Converged Relational-Graph Optimization Framework"

Anonymous Authors

Dear Reviewers and Meta Reviewer,

Firstly, we would like to thank the reviewers and the meta reviewer for their insight and valuable comments which enable us to greatly improve the quality of our manuscript. We have carefully taken your comments into consideration in preparing this revision. Below we summarize the major changes in each section in this revision.

In Section 1, we have mainly made the following modifications:

1. We have made the query example on Page 1 (in the original manuscript) a figure float. (R3-M1)
2. We have removed RGMapping from the contributions of this paper. (R2-i.6, R3-O1.1)

In Section 2, we have mainly made the following modifications:

1. We move Table 1 earlier to make the notations in this section easier to understand. (R3-M1)
2. We have simplified the contents about the definition of RGMapping. (R2-i.6, R3-O1.1, R3-M1, R3-C1, M-C7)
3. We have added an ER diagram for the Person, Knows, Likes, and Message tables to Figure 2 (Figure 1 in the original paper), and have included corresponding descriptions in the main text to explain the relationships between vertex relations and edge relations in E-R terms. (R3-O1.1)
4. We have enlarged the font sizes in Figure 2 (Figure 1 in the original manuscript) from 28pt to 32pt. (R1-D1)

In Section 3, we have mainly made the following modifications:

1. We have enlarged the font sizes in Figure 3 (Figure 2 in the original manuscript) from 24pt to 28pt. (R1-D1)
2. We add more contents in Figure 3 to show what is given up when using the tree decomposition approach in RelGo. (R2-ii.1, M-C4)
3. We add several sentences to discuss the differences between tree decomposition applied in this paper and the techniques used by EmptyHeaded and CLFTJ. (R2-i.1, M-C2)
4. We add a new subsection (i.e., Section 3.1.4) to compare the optimization of RelGo and Calcite and add two new figures, i.e. Figure 4(b) and Figure 4(c), to show the experimental results. (R2-S1, R2-ii.1, M-C6)
5. We add a sentence to emphasize that the intersection evaluation using worst-case optimal join is implementaed on relational tables. (R2-i.5)

In Section 4, we have made the following modifications:

1. We have enlarged the font sizes in Figure 6 (Figure 5 in the original manuscript) from 28pt to 32pt. (R1-D1)
2. We add a new paragraph to explain why HASH_JOIN is used for the entire plan in the absence of a graph index. (R3-O2.5)
3. We explain why we limit the extensive depth, methods, and literature on relational query optimization. (R3-O4.1, R3-C3)
4. We add several sentences to explain that FilterIntoMatchRule is a global optimization while ExpandGetVFusionRule is a local optimization rule. Besides, we briefly describe two more optimization rules we applied in RelGo. (R2-i.3)
5. We introduce the higher-order graph statistics in more detail. (R1-O1, M-C5)

In Section 5, we have made the following modifications:

1. We add two new baselines, i.e., Umbra and Kùzu, add description about them, and conduct compreshensive experiments on them. The experimental results are shown in the revision and compared with RelGo. (R2-i.4, R2-iii.1, R2-C1, R2-C3, R3-O2.3, R3-C2, M-C1, M-C3). We find that Kùzu is slower than RelGo on both the LDBC and JOB benchmarks, even slower than DuckDB. We have raised an issue on their GitHub to inquire about this problem [1], but as of now, they have not responded. Therefore, we reported our experimental results in the manuscript.
2. We conduct new compreshensive experiments on a larger instance, i.e., LDBC100 to make the conclusions more convincing. Besides, we add the statistics of LDBC100 to Table 2. (R3-O2.6, R3-C2)
3. We have add some explanations about the procedures for data loading and graph index construction in thie revision. (R3-O2.2)
4. We add the description about the versions of DuckDB used in the experiments. (R2-iii.2)
5. We have fixed the typo in this revision and modified the size of the RAM to 256GB.
6. We enlarge the font sizes in Figures 7–11 (Figures 6–10 in the original manuscript) from 20/22pt to 26/28pt. (R1-D1)
7. We have redrawn Figure 11 (Figure 10 in the original manuscript), normalized all runtimes to that of DuckDb, and plot speedups achieved by RelGo and the baselines. (R1-M2)
8. We add two paragraphs to discuss how to extend RelGo to deal with queries wihout explicit PGQ component. Besides, we explain about the potential tradeoffs between global and local optimizations (R2-i.3, M-C6, R3-O3.1, R3-C3)
9. We add a new section, i.e., Section 5.4 (Case Study), and a new figure (i.e., Figure 13) to demonstrate why plans generated by RelGo are better than those generated by the baselines. (R1-O2, M-C4)
10. We provide the query statement for a representative SQL/PGQ query (i.e. JOB[17]) in Figure 12 in Section 5.4. (R3-O2.1, M-C4)

In Section 6, we have made the following modifications:

1. We add the citations of the related work mentioned by Reviewer #2 in this section and describe these works briefly. (R2-C2)
2. We explain the relationships between DuckPGQ and DuckDB in more detail in this revision. (R2-i.2, R2-C2, M-C2)

[1]https://github.com/kuzudb/kuzu/issues/3865

Sincerely yours,

Anonymous Authors