Supplementary Material for Understanding Student Errors in Graph Query Formulation

Amedeo Pachera
Lyon1 University, CNRS Liris
Lyon, France
amedeo.pachera@univ-lyon1.fr

Stefania Dumbrava
ENSIIE
Paris, France
stefania.dumbrava@ensiie.fr

Angela Bonifati
Lyon1 University, CNRS Liris & IUF
Lyon, France
angela.bonifati@univ-lyon1.fr

Andrea Mauri
Lyon1 University, CNRS Liris
Lyon, France
andrea.mauri@univ-lyon1.fr

PVLDB Reference Format:
Amedeo Pachera, Stefania Dumbrava, Angela Bonifati, and Andrea Mauri. Supplementary Material for Understanding Student Errors in Graph Query Formulation. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/pake97/Understanding-Student-Errors-in-Graph-Query-Formulation.git.

1 QUERIES GROUND TRUTH

In this Section, we report the ground truth answer to the questions of our study. We manually write the correct query to answer each question. Note that there are multiple solutions to obtain the correct results. We did not directly compare our ground truth with the students' answers to label them, instead, we used the following queries and guidelines.

1.1 Schema Discovery Queries

- Q1: Write a Cypher query that returns the set of all distinct node labels. State also how many answers it returns.
```
MATCH (n)
RETURN DISTINCT labels(n)
//ANSWERS: 9
```
- Q2: Write a Cypher query that returns the set of all distinct relationship types. State also how many answers it returns.
```
MATCH ()-[r]->()
RETURN DISTINCT type(r)
//ANSWERS: 10
```
- Q3: Write a Cypher query that returns each distinct node label together with its corresponding set of properties. State also how many answers it returns.
```
MATCH (n)
RETURN DISTINCT labels(n) AS label, properties(n) AS properties
//ANSWERS: 18
```
- Q4: Write a Cypher query that returns each distinct relationship type together with its corresponding set of properties. State also how many answers it returns.
```
MATCH ()-[r]->()
RETURN DISTINCT type(r) AS type, properties(n) AS properties
//ANSWERS: 10
```
- Q5: Write at least three additional Cypher queries that are needed for you to understand the underlying schema structure of the dataset. You can also write more than three queries in case. For each of them, state also how many answers it returns.
```
MATCH (m)-[l]->(n)-[k]->(:Genome)
RETURN DISTINCT type(k), type(l), labels(n), labels(m)
//ANSWERS: 1

MATCH (m)-[l]->(n)-[k]->(:Family)
RETURN DISTINCT type(k), type(l), labels(n), labels(m)
//ANSWERS: 6

MATCH (m)-->()-->(n)-->(:Family)
RETURN DISTINCT labels(m), labels(n)
//ANWERS: 2
```

1.2 Analytical Queries

- Q6: Write a Cypher query to return each pangenome, together with its associated genes. State also how many answers it returns.
```
MATCH (g:Gene)-[:IS_IN_FAMILY]->(f:Family)-[:IS_IN_PANGENOME]->(p:Pangenome)
RETURN distinct p.name, g.name
//ANSWERS: 5477
```
- Q7: Write a Cypher query to return each pangenome, together with its associated spots. State also how many answers it returns.
```
MATCH (sp:Spot)<-[:IS_IN_SPOT]-()<-[:IS_IN_RGP]-()-[:IS_IN_FAMILY]->()
-[:IS_IN_PANGENOME]->(p:Pangenome)
RETURN distinct p.name, sp.name
//ANSWERS: 656
```
- Q8: Write a Cypher query to return each pangenome and its associated modules that contain at least a pair of gene families that are more than 80% identical. State also how many answers it returns.
```
MATCH (p:Pangenome)<--()-->(m:Module)<--(f1:Family)<-[r:IS_SIMILAR]-(f2:Family)
WHERE f1 <> f2 AND r.coverage >= 0.8
RETURN p.name, COUNT(DISTINCT(m.name))
//ANSWERS: 2 - Enterobacter.cloacae (236 modules)
//and Acinetobacter.baumannii (278 modules)
```
- Q9: Write a Cypher query that returns pairs of inter-pangenome families that both contain annotations and that are more than 80% identical. State also how many answers it returns.
```
MATCH (p1:Pangenome)-[:IS_IN_PANGENOME]-(f1:Family)-[r1:IS_SIMILAR]-(f2:Family)
-[:IS_IN_PANGENOME]-(p2:Pangenome)
WHERE r1.identity >= 0.8
AND p1.taxid <> p2.taxid
AND f1.annotation IS NOT NULL
AND f2.annotation IS NOT NULL
```


Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

```
RETURN  f1.name, f2.name
//ANSWERS: 14
```

- Q10: Write a Cypher query that returns, for each pangenome, the neighborhood of its gene families containing annotations. State also how many answers it returns.

```
MATCH (f1:Family)-[:NEIGHBOR]-(f2:Family)-[:IS_IN_PANGENOME]->(p:Pangenome),
WHERE f1.annotation IS NOT NULL
AND f2.annotation IS NOT NULL
RETURN p, f1, f2
//ANSWERS: 153
```

1.3 Aggregation Queries

- Q11: For every pangenome, write a Cypher query that return its name, together with its number of families that contain annotations. State also how many answers it returns.

```
MATCH (p:Pangenome)-[:IS_IN_PANGENOME]-(f:Family)
WHERE f.annotation IS NOT NULL
RETURN p.name, count(f)
//ANSWERS: ("Enterobacter.cloacae": 110, "Acinetobacter.baumannii": 92)
```

- Q12: Write a Cypher query that returns all modules together with the average similarity of the gene families they are comprised of, in descending order. State also how many answers it returns.

```
MATCH (m:Module) <-[]-(a:Family)<-[k:IS_SIMILAR]-(f:Family)
RETURN m.name, COLLECT(f.name), AVG(r.identity) AS idnt
ORDER BY idnt DESC
//ANSWERS: 406
```

- Q13: Write a Cypher query that returns the names of the modules that contain the pairs of gene families that have the minimum and, respectively, the maximum similarity (as indicated by the value of the "identity" property of the "IS_SIMILAR" edge connecting them). State also how many answers it returns.

```
MATCH (m1:Module)<-[:IS_IN_MODULE]-(f:Family)-[sim:IS_SIMILAR]->(f2:Family)
-[:IS_IN_MODULE]->(m2:Module)
RETURN m1.name, m2.name, sim.identity
ORDER BY sim.identity DESC
LIMIT 1
UNION ALL
MATCH (m1:Module)<-[:IS_IN_MODULE]-(f:Family)-[sim:IS_SIMILAR]->(f2:Family)
-[:IS_IN_MODULE]->(m2:Module)
RETURN m1.name, m2.name, sim.identity
ORDER BY sim.identity LIMIT 1
//ANSWERS: Modules 2 and 177 with similarities 40 and, respectively,  0.8
```

- Q14: Write a Cypher query that returns the top 10 gene families with the highest number of associated modules. State also how many answers it returns.

```
MATCH (f:Family)-[:IS_IN_MODULE]-(m:Module)
WITH f, count(m) AS num
RETURN f, num
ORDER BY num DESC
LIMIT 10
//ANSWERS: 10
```

- Q15: Return the names of the top 10 pairs of similar families (as defined in Q13), whose associated identity and coverage metrics surpass 0.8, together with the names of their partitions (as indicated by the value of the "subpartition" property associated to the nodes of type "Partition"). State also how many answers it returns.

```
MATCH (p:Partition)<-[:HAS_PARTITION]-(f1:Family)-[r:IS_SIMILAR]->(f2:Family)
-[:HAS_PARTITION]->(p2:Partition)
WHERE r.identity >=0.8 AND r.coverage >=0.8
RETURN f1.name, f2.name, p.partition, p2.partition, r.identity
ORDER BY r.identity DESC
LIMIT 10
//ANSWERS: 10
```

2 TABLE EXTENDED VERSIONS

Table 1 is the extended version of Table 4 of the paper, reporting the *complications* and the *semantic return errors* performed by the students in the **schema discovery queries**. This extended table shows that none of the errors were related to advanced query concepts (as expected from the type of questions) such as the `LIMIT` and `ORDER BY` clauses. Moreover, none of the errors consists of a *semantic where error* or a *semantic pattern error*.

Table 2 is the extended version of Table 5 of the paper, reporting the *complications* and the *semantic errors* performed by the students in the **analytical queries**. In this phase, the *complications* caused by the overuse of `WITH` and `UNWIND` are absent. None of the students used a *Wrong Edge Label* when defining the pattern of the queries. *Additional Condition* errors were also absent in this phase. Finally, as for the schema discovery queries, none of the students struggled with the `LIMIT` and `ORDER BY` clauses.

Table 3 is the extended version of Table 6 of the paper, reporting the *complications* and the *semantic errors* performed by the students in the **aggregation queries**. In this last phase, none of the *complications* was due to the use of `COLLECT` and `UNWIND`. As for the analytical queries, none of the students used a *Wrong Edge Label* when defining the pattern of the queries. For what concerns the *semantic where errors*, none of the students put a condition on a *wrong property* or used a *wrong* or *additional condition*. Finally, among the *semantic return errors*, `Wrong DISTINCT` and `Missing DISTINCT` are absent. Differently from the previous phases, some of the errors (10) are due to `LIMIT` on the wrong variable, while none of them concerns the `ORDER BY` clause.

3 VISUALIZING THE GRAPH QUERY FORMULATION ERROR FLOW

The Sankey diagram in Figure 1 visualizes the flow of errors made by students across a series of queries (Q1 to Q15). The diagram tracks different types of errors and correct responses as they propagate through each query. Each query is represented as a vertical section, with the types of responses and errors flowing horizontally.

Starting with Q1, there is a mix of *Syntax Errors*, *Compilation*, *Semantic Returns*, and *Correct* answers. As we move from Q2 to Q15, each subsequent query shows how students' responses from the previous query translate into new errors or correct responses. The diagram highlights both the persistence and changes in the types of errors made by students. Correct responses generally persist as correct or transition into different types of errors, showing areas where students may have misunderstandings after initially correct responses. Syntax Errors and Compilation Errors often carry over to subsequent queries, while some error types like Semantic Return and Semantic Where appear recurrently, indicating common trouble areas for students. Certain errors diminish over time, such as Syntax Errors, suggesting learning and improvement in syntax understanding. In the *schema discovery queries* (Q1-Q4) the majority of the the answers are correct or contain complications. *Schema discovery queries* especially Q6, Q9, and Q10 have notable shifts and distributions in the type of errors, indicating these queries were particularly challenging for the students. The persistence of certain errors and their transitions highlight specific areas where students struggle, which could be key focus areas for future teaching. Overall, this Sankey diagram effectively visualizes the error dynamics in student responses over multiple queries, providing valuable insights into common pitfalls and learning progression.

Category	Type	Count	Count / Category Total	Count / Total
Complications	Added `COLLECT`	49	0.422	0.395
	Added `WITH`	40	0.345	0.323
	Added `UNWIND`	27	0.233	0.218
	Additional information	0	0	0
	Total	116	1	0.935
Semantic Return Errors	Missing `DISTINCT`	6	0.75	0.048
	Missing Information	1	0.125	0.008
	Wrong Information	1	0.125	0.008
	Missing `LIMIT`	0	0	0
	Missing `ORDER BY`	0	0	0
	Wrong `LIMIT`	0	0	0
	Wrong `ORDER BY`	0	0	0
	Wrong `DISTINCT`	0	0	0
	Total	8	1	0.065
Total		124	1	1

Table 1: Number of error categories and type occurred writing schema discovery queries. We report the absolute number, the percentage of occurrence both within the category and with respect to the whole set of errors.

Category	Type	Count	Count/ Category Total	Count / Total
Complications	Added `COLLECT`	26	0.963	0,096
	Added `WITH`	0	0	0
	Added `UNWIND`	0	0	0
	Additional Information	1	0.037	0,004
	Total	27	1	0.1
Semantic Pattern Errors	Wrong Pattern w.r.t. Schema	18	0.29	0,067
	Wrong Pattern	21	0.339	0,078
	Missing Node	4	0.065	0,015
	Additional Node	5	0.081	0,019
	Wrong Edge Direction	14	0.226	0,052
	Wrong Edge Label	0	0	0
	Total	62	1	0.230
Semantic Where Errors	Wrong Property	31	0.34	0,115
	Missing Condition	49	0.538	0,181
	Wrong Condition	3	0.033	0,011
	Wrong Operator	8	0.088	0,030
	Additional Condition	0	0	0
	Total	91	1	0.337
Semantic Return Errors	Missing `DISTINCT`	34	0.378	0,126
	Missing Information	13	0.144	0,048
	Wrong Information	41	0.456	0,152
	Wrong `DISTINCT`	2	0.022	0,007
	Missing `LIMIT`	0	0	0
	Missing `ORDER BY`	0	0	0
	Wrong `LIMIT`	0	0	0
	Wrong `ORDER BY`	0	0	0
	Total	90	1	0.333
Total		270	1	1

Table 2: Number of error categories and type occurred writing analytical queries. We report the absolute number, the percentage of occurrence both within the category and with respect to the whole set of errors.

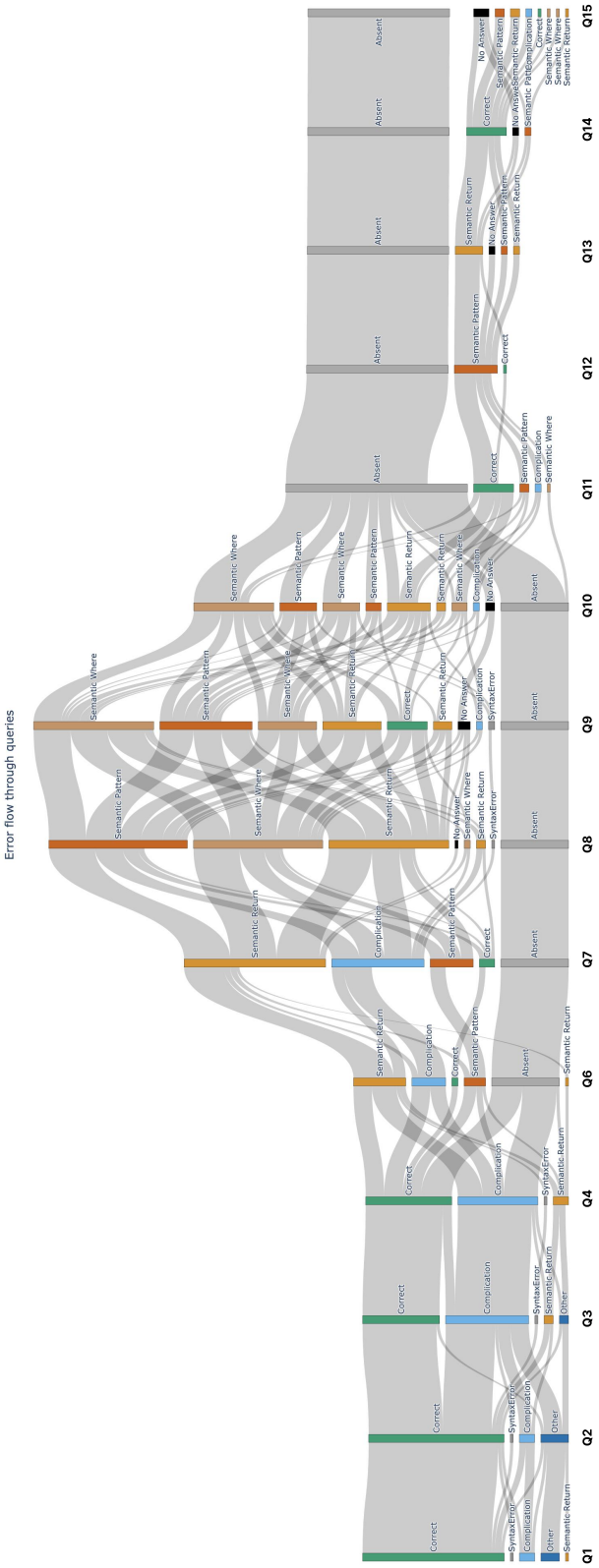

Figure 1: Sankey diagram of the errors among the queries.

Category	Type	Count	Count/ Category Total	Count / Total
Complications	Added `COLLECT`	0	0	0
	Added `WITH`	2	0.5	0.038
	Added `UNWIND`	0	0	0
	Additional Information	2	0.5	0.038
	Total	4	1	0.075
Semantic Pattern Errors	Wrong Pattern w.r.t. Schema	3	0.143	0.057
	Wrong Pattern	1	0.048	0.019
	Missing Node	1	0.048	0.019
	Additional Node	3	0.0143	0.057
	Wrong Edge Direction	13	0.619	0.245
	Wrong Edge Label	0	0	0
	Total	21	1	0.396
Semantic Where Errors	Wrong Property	0	0	0
	Missing Condition	2	0.667	0.038
	Wrong Condition	0	0	0
	Wrong Operator	1	0.333	0.019
	Additional Condition	0	0	0
	Total	3	1	0.057
Semantic Return Errors	Missing `DISTINCT`	0	0	0
	Missing Information	14	0.56	0.264
	Wrong Information	1	0.040	0.019
	Wrong `DISTINCT`	0	0	0
	Missing `LIMIT`	0	0	0
	Missing `ORDER BY`	0	0	0
	Wrong `LIMIT`	10	0.4	0.189
	Wrong `ORDER BY`	0	0	0
	Total	25	1	0.472
Total		53	1	1

Table 3: Number of error categories and type occurred in writing aggregate queries. We report the absolute number, the percentage of occurrence both within the category and with respect to the whole set of errors