Reflection Results

Ibrahim Ozkan
Lamin O. Touray
Mahdi Ouchrahou

Task 1.3 - Our Query

- **Experience**:
  - Took database courses during bachelor's degree.
  - Use of ORMs over time led to a decline in SQL construction skills.
- **Development Duration & Steps:**
  - Query construction in Task 1.1 took one day.
  - Involved analyzing the database schema, reading PostgreSQL documentation, fixing SQL errors, and execution.
- **Quality of Implementation:**
  - Results from Task 1.1 were less accurate compared to the LLM-generated version.
  - Query execution was slower, indicating inefficiency.
- **Automation Tool:**
  - None used.

Task 1.3 - LLMs Query

- **Experience**:
  - Familiar with using ChatGPT and adept at crafting prompts to achieve desired results.
  - Gained familiarity with PostgreSQL before using ChatGPT effectively.
- **Duration & Steps**:
  - Achieved the final result in a 20-minute session with ChatGPT, including query executions.
  - Tested every query given to us by ChatGPT and did some error fixing
  - Also, did some result analysing to interpret the correctness of SQL queries
- **Quality**:
  - ChatGPT's implementation outperformed ours in terms of both accuracy and performance.
- **Misunderstandings**:
  - Initially, ChatGPT used the **post** table in its query, which caused execution errors.
  - Proper prompts directed it to use the correct tables.
  - The initial query was inefficient; prompting for optimization resulted in a more efficient version.
- **Automation Tools:**
  - None used.

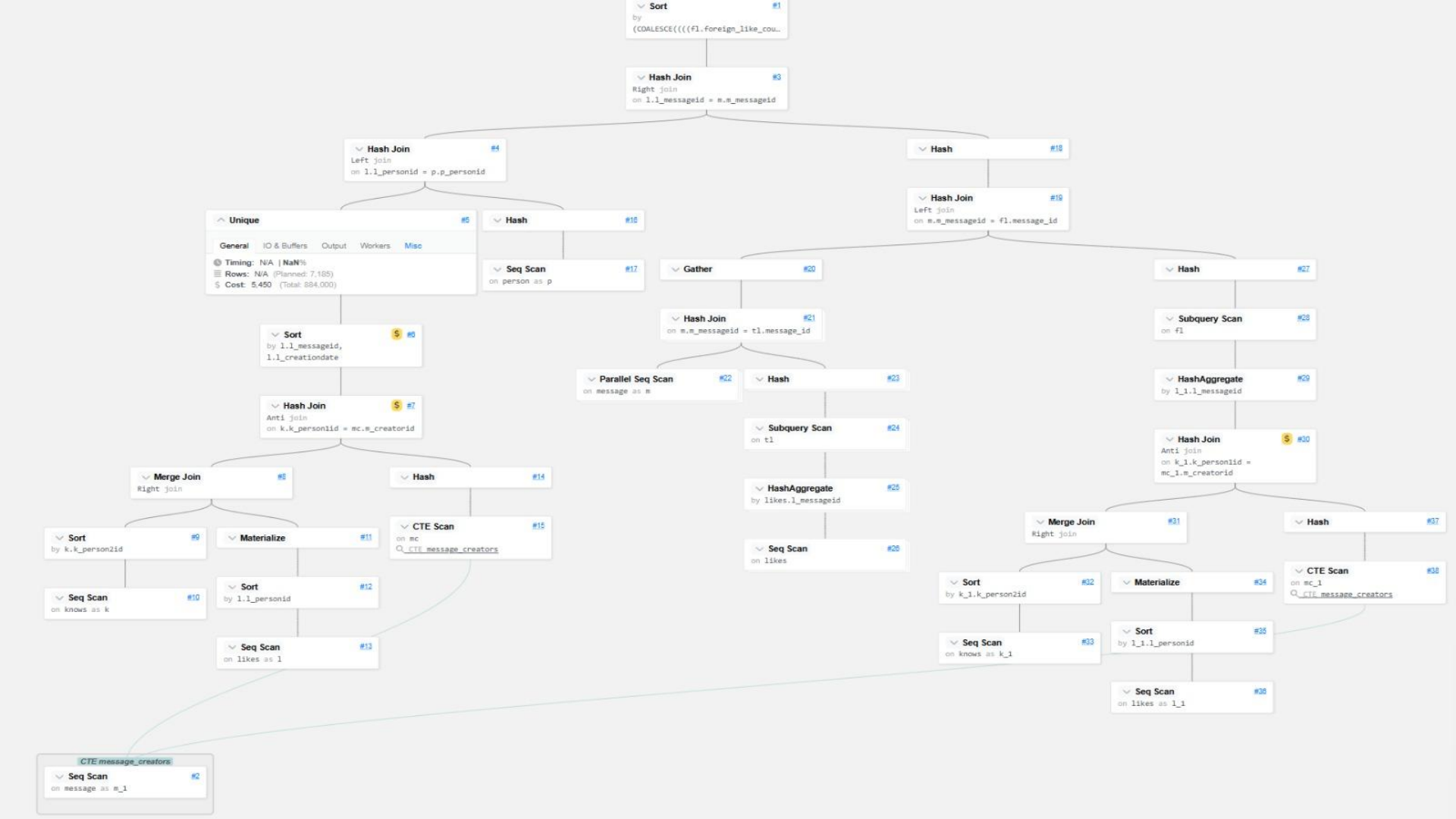
Task 2.3

- **Query Development Process**:
  - In sub-task 1, the query development process was straightforward due to prior heavy lifting, requiring only minor adjustments to the query.
  - For sub-task 2, the process was also simple because:
    - The LLM retained knowledge of the previous query.
    - With a prompt instruction, it efficiently adjusted the previous query to produce the desired result.

- **Role of LLM Automation**:
  - The LLM can integrate into an interactive query development process, acting as a source of ground truth to compare and refine human-generated queries for better optimization.
- **Efficiency Through LLM**:
  - With accurate and concise prompts, the LLM can leverage its vast knowledge to create accurate and efficient queries, speeding up the development process through automation.

Task 3.2

- **Result:**
  - The LLM made a good effort in creating the execution plan.
  - However, it gave a **hypothetical execution plan**, which looks like the structure of a real PostgreSQL execution plan but is not a real one.
  - The output was more like a template, so it is not fully usable for real-world tasks.
- **Explanatory Value:**
  - The LLM's explanation was helpful for understanding execution plans.
  - It showed a template of how an execution plan should look and explained its parts.
  - It also explained the query the plan was based on, which added clarity.
  - Query execution plan help in further optimization, query debugging and index optimization.
    - Fundamental difference.


Sort #1
by (COALESCE((((fl.foreign_like_cou...
Hash Join #3
Right join
on l.l_messageid = m.m_messageid
Hash Join #4
Left join
on l.l_personid = p.p_personid
Hash #18
Unique #6
General IO & Buffers Output Workers Misc
Timing: N/A | NaN%
Rows: N/A (Planned: 7,185)
$ Cost: 5,450 (Total: 884,000)
Hash #16
Seq Scan #17
on person as p
Hash Join #19
Left join
on m.m_messageid = fl.message_id
Gather #20
Hash #27
Sort #5
by l.l_messageid, l.l_creationdate
Hash Join #21
on m.m_messageid = tl.message_id
Subquery Scan #28
on fl
Hash Join #7
Anti join
on k.k_personid = mc.m_creatorid
Parallel Seq Scan #22
on message as m
Hash #23
HashAggregate #29
by l_1.l_messageid
Subquery Scan #24
on tl
Hash Join #30
Anti join
on k_1.k_personid = mc_1.m_creatorid
Merge Join #8
Right join
Hash #14
HashAggregate #25
by likes.l_messageid
Merge Join #31
Right join
Hash #37
Sort #9
by k.k_person2id
Materialize #11
CTE Scan #15
on mc
CTE message_creators
Seq Scan #26
on likes
Sort #32
by k_1.k_person2id
Materialize #34
CTE Scan #38
on mc_1
CTE message_creators
Seq Scan #10
on knows as k
Sort #12
by l.l_personid
Sort #35
by l_1.l_personid
Seq Scan #33
on knows as k_1
Seq Scan #13
on likes as l
Seq Scan #36
on likes as l_1
CTE message_creators
Seq Scan #2
on message as m_1