Appendix: Research Plan

After I am admitted as a graduate student, apart from serving my supervisor's project, I want to develop my own research in the following areas:

Direction 1: Optimize least-frequently-used (LFU) strategy with weights to fit for erratic data access pattern

FlexPushdownDB[1] [VLDB'21] (FPDB) focuses on how to combine caching with pushdown to optimize cloud-native databases with storage-disaggregation architecture. The article proposed a hybrid query executor based on separable operators. The query is divided into the local cache plan and the pushdown plan, and the final output is combined from caching and pushdown by a merge operator. This design achieves better runtime performance under whatever cache size.

Additionally, the paper also proposed a novel Weighted-LFU cache replacement policy. Compared to traditional LFU policy, which simply increases the frequency counter of a segment by 1/*segment.size*, FPDB's Weighted-LFU strategy takes into account cost of pushdown. The basic rationale is that since higher pushdown cost indicates higher benefit of caching, higher weight is assigned for each data segment access.

Ignited by FPDB, there is another promising application of weighting strategy that could be further studied for LFU policy improvement, which is to break the overall period into two parts: historical period and most-recent period. **Here is the analysis of why this research matters:** Traditional LFU policy simply calculates the accumulated frequency of data access in a single period. Although this algorithm is robust under most data access scenarios, when the pattern of data access changes, due to LFU needs longer time to fit for the new pattern, it may result in a sharp decrease in cache hit ratio. To solve the bottleneck of failing to decrease the influence of historical period, and failing to admit the role that most-recent period plays, I want to utilize the idea of weighting to depict the relation between most-recent period and historical period, and introduce the concept of accumulated access frequency, which is calculated via the formula as shown in Equation 1:

$$AccuFreq_t = \alpha \times \frac{AccessFreq_t}{CurrentPeriod} + (1-\alpha) \times AccuFreq_{t-1}, \text{where } 0 \le \alpha \le 1 \quad (1)$$

In the formula, $CurrentPeriod$ is the duration of most-recent period, and $AccessFreq_t$ refers to the number of times that the data segment is accessed in the most-recent period. $AccuFreq_{t-1}$ represents the cumulative historical access frequency of the segment. Parameter α is the coefficient used to determine the respective weights of access frequency in the most-recent period and historical period, and the historical access frequency decays at the rate of coefficient $(1-\alpha)$ in the most-recent period. Therefore, a larger α attaches greater importance to the data access pattern in the most-recent period, and indicates less impact of historical access pattern. That may facilitate the caching system to monitor and adjust to new

[1] Yifei Yang, Matt Youill, Matthew Woicik, Yizhou Liu, Xiangyao Yu, Marco Serafini, Ashraf Aboulnaga, and Michael Stonebraker. 2021. FlexPushdownDB: hybrid pushdown and caching in a cloud DBMS. Proc. VLDB Endow. 14, 11 (July 2021), 2101–2113. https://doi.org/10.14778/3476249.3476265

access pattern more swiftly than traditional LFU policy and eventually maintain high cache-hit ratio.

Direction 2: Apply query selection for workload compression task under materialized view recommendation

ISUM[2] [SIGMOD'22] presents an outstanding framework on workload compression under index recommendation. The key idea of ISUM is to select *k* queries from input workload to minimize total estimated cost. A query is selected based on the following principles: 1. High Utility, which significantly contributes to the cost reduction of the entire workload; 2. High Influence (measured by the similarity between queries using *weighted Jaccard*). Finally, the experiment of ISUM proves the effectiveness of using Benefit, which is the sum of Utility and Similarity to select queries. This work is a milestone to me since it originates some innovative ideas such as using summary-features based greedy approach to prevent all-pairs comparison, using statistics-based method to assign reasonable weight for each column. Based on the core idea of this milestone work, I propose to make additional contributions to workload compression under materialized view (MV) recommendation.

Here is my primary idea on upgrading the workload compression framework inherited from ISUM to fit better for MV: I want to adjust the statistics-based method used to assign weights for each column. Particularly, apart from selectivity and density, **I argue that Rewriting Frequency (RF) should be considered as a crucial factor in the case of MV.** Rewriting is a unique optimizer module in MV, which is used to adjust users' input query to fit for the given MV. For example, under *expression adjustment* scenario, suppose query q_1 asks for expression *ln(revenue)*, whereas MV stores *abs(revenue)*. Therefore, converting values from absolute scale to logarithm scale increases the RF of field *revenue*. Additionally, other scenarios such as JOIN, GROUP BY, Filter are also involved in the evaluation of RF. My core assumption is that columns that are frequently rewritten indicates higher importance among user's query than other columns, consequently, should be assigned higher weight.

I believe one of the key tasks to be studied in this research is how to systematically define and measure RF. Especially one of the challenges is how to normalize this index when facing a set of queries. Materialized view has been widely adopted as a common query acceleration tactic, which has even been practiced many times by me. Therefore, I believe this direction will fill the blank on MV recommendation with minimum cost and could be applied to industry in the future.

---

[2] Tarique Siddiqui, Saehan Jo, Wentao Wu, Chi Wang, Vivek Narasayya, and Surajit Chaudhuri. 2022. ISUM: Efficiently Compressing Large and Complex Workloads for Scalable Index Tuning. In Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22). Association for Computing Machinery, New York, NY, USA, 660–673. https://doi.org/10.1145/3514221.3526152