References

A Query Examples

Our benchmark suite incorporates a broad range of query types. We show examples of each query type as follows.

LLM filter. This query type leverages LLM for filtering data within a `WHERE` clause. The LLM processes and analyzes information to meet some specified criteria, such as identifying whether a movie is suitable for kids. This query type illustrates typical use cases in sentiment analysis and content filtering, which are important for application tasks, such as customer feedback analysis and content moderation.

```
SELECT t.movietitle
FROM MOVIES
WHERE LLM(
    'Given the following fields,
    ↪ determine whether the movie is
    ↪ suitable for kids. Answer ONLY
    ↪ with "Yes" or "No".',
    movieinfo,
    reviewcontent,
    reviewtype,
    movietitle
) = 'Yes'
```

LLM projection. This query type makes calls to an LLM within a `SELECT` statement to process information from specified database column(s). It reflects common tasks in data analytics in which the LLM is used for summarization and interpretation based on certain data attributes.

```
SELECT LLM(
    'Given the following information,
    ↪ summarize good qualities in
    ↪ this movie that led to a
    ↪ favorable rating.',
    reviewcontent, movieinfo
)
FROM MOVIES
```

Multi-LLM invocation. This query type involves multiple LLM calls in different parts of the query and addresses scenarios in which several layers of data processing or analysis are required. It represents advanced analytical tasks, such as combining different data insights.

```
SELECT LLM(
    'Given the information about a
    ↪ movie, summarize the good
    ↪ qualities that led to a
    ↪ favorable rating.',
    reviewtype,
    reviewcontent,
    movieinfo,
    genres
)
FROM MOVIES
WHERE LLM(
    'Given the following review, answer
    ↪ whether the sentiment is
    ↪ "POSITIVE" or "NEGATIVE".
    ↪ Respond ONLY with "POSITIVE" or
    ↪ "NEGATIVE", in all caps.',
    reviewcontent
) = 'NEGATIVE'
```

LLM aggregation. This query type incorporates an AVG operator that incorporates LLM outputs into further query processing. For example, one could use LLMs to assign sentiment scores to individual reviews and then aggregate these scores to calculate an average sentiment for overall customer feedback. This query type is essential for tasks that need to extract insights from complex textual data.

```
SELECT AVG(
    LLM(
        'Rate sentiment in numerical
        ↪ values from 1 (bad) to 5
        ↪ (good).',
        reviewcontent, movieinfo
    )
) AS AverageScore
FROM MOVIES
```

Retrieval-augmented generation (RAG). This query type leverages external knowledge bases for enhanced LLM processing, enriching LLM queries with a broader context. It simulates use cases where queries need to pull in relevant information from external sources, such as document databases or knowledge graphs, to provide comprehensive answers.

```
SELECT LLM(
    'Given a question and four
    ↪ supporting contexts, answer the
    ↪ provided question.',
    ↪ VectorDB.search(question, k=4),
    ↪ question)
FROM FEVER
```

B Dataset Information

We detail the fields and functional dependencies (FDs) used for each dataset as follows.

MOVIES

```
columns:
genres, movieinfo, movietitle,
productioncompany, reviewcontent,
reviewtype, rottentomatoeslink,
topcritic

FDs:
movieinfo, movietitle,
rottentomatoeslink
```

PRODUCTS

```
columns:
description, id, parent_asin,
product_title, rating, review_title,
text, verified_purchase


FDs:
parent_asin, product_title
```

BIRD

```
columns:
Body, PostDate, PostId, Text

FDs:
Body, PostId
```

PDMX

```
columns:
artistname, bestarrangement, bestpath,
bestuniquearrangement, composername,
complexity, genre, grooveconsistency,
groups, hasannotations, hascustomaudio,
hascustomvideo, haslyrics, hasmetadata,
haspaywall, id, isbestarrangement,
isbestpath, isbestuniquearrangement,
isdraft, isofficial, isoriginal,
isuserpro, isuserpublisher, isuserstaff,
license, licenseurl, metadata,
nannotations, ncomments, nfavorites,
nlyrics, notesperbar, nnotes, nratings,
ntracks, ntokens, nviews, path,
pitchclassentropy, postdate, postid,
publisher, rating, scaleconsistency,
songlength, songlengthbars,
songlengthbeats, songlengthseconds,
songname, subsetall, subsetdeduplicated,
subsetrated, subsetrateddeduplicated,
subtitle, tags, text, title, tracks,
version


FDs:
[metadata, path],
[hasannotations, hasmetadata, isdraft,
isofficial, isuserpublisher, subsetall
]
```

BEER

```
columns:
beer/beerId, beer/name, beer/style,
review/appearance, review/overall,
review/palate, review/profileName,
review/taste, review/time

FDs:
[beer/beerId, beer/name]
```

FEVER

```
-- FEVER --
columns:
claim, evidence1, evidence2,
evidence3, evidence4

FDs: []
```

SQuAD

```
columns:
question, context1, context2,
context3, context4, context5

FDs: []
```

C Prompts

We detail the system and user prompts for each query type and dataset as follows.

System Prompt

```
You are a data analyst. Use the provided JSON data
to answer the user query based on the specified
fields. Respond with only the answer,
no extra formatting.

Answer the below query:
{QUERY}

Given the following data:
{fields}
```

User Prompt - LLM Filter

```
MOVIES: Given the following fields, answer in one
word, 'Yes' or 'No', whether the movie would be
suitable for kids.  Answer with ONLY 'Yes' or 'No'.

PRODUCTS: Given the following fields determine if
the review speaks positively ('POSITIVE'),
negatively ('NEGATIVE'), or netural ('NEUTRAL')
about the product. Answer only 'POSITIVE',
'NEGATIVE', or 'NEUTRAL', nothing else.

BIRD: Given the following fields related to posts
in an online codebase community, answer whether the
post is related to statistics. Answer with only
'YES' or 'NO'.

PDMX: Based on following fields, answer 'YES' or
'NO' if any of the song information references a
specific individual. Answer only 'YES' or 'NO',
nothing else.

BEER: Based on the beer descriptions, does this
beer have European origin? Answer 'YES' if it does
or 'NO' if it doesn't.
```

User Prompt - LLM Projection

```
MOVIES: Given information including movie
descriptions and critic reviews, summarize the good
qualities in this movie that led to a favorable
rating. (also used in multi-invocation)

PRODUCTS: Given the following fields related to
amazon products, summarize the product, then answer
whether the product description is consistent with
the quality expressed in the review. (also used
in multi-invocation)

BIRD: Given the following fields related to posts
in an online codebase community, summarize how the
comment Text related to the post body.

PDMX: Given the following fields, provide an
overview on the music type, and analyze the given
scores. Give exactly 50 words of summary.

BEER: Given the following fields, provide an
high-level overview on the beer and review in a
20 words paragraph.
```

User Prompt - LLM Aggregation

```
MOVIES: Given the following fields of a movie
description and a user review, assign a sentiment
score for the review out of 5. Answer with ONLY a
single integer between 1 (bad) and 5 (good).

PRODUCTS: Given the following fields of a product
description and a user review, assign a sentiment
score for the review out of 5. Answer with ONLY a
single integer between 1 (bad) and 5 (good).
```

User Prompt - Multi-LLM Invocation

```
MOVIES/PRODUCTS: Given the following review, answer
whether the sentiment associated is 'POSITIVE' or
'NEGATIVE'. Answer in all caps with ONLY 'POSITIVE'
or 'NEGATIVE':
```

User Prompt - RAG

```
FEVER: You are given 4 pieces of evidence as
{evidence1}, {evidence2}, {evidence3}, and
{evidence4}. You are also given a claim as {claim}.
Answer SUPPORTS if the pieces of evidence support
the given {claim}, REFUTES if the evidence refutes
the given {claim}, or NOT ENOUGH INFO if there is
not enough information to answer. Your answer
should just be SUPPORTS, REFUTES, or NOT ENOUGH
INFO and nothing else.

SQuAD: Given a question and supporting contexts,
answer the provided question.
```

D ABLATIONS

We present two sets of ablation experiments: one comparing the prefix hit rate (PHR) between GGR and an optimal oracle, and another examining the impact of using a smaller LLM model.

D.1 PHR of GGR v.s. OPHR

OPHR is a very expensive brute-force oracle algorithm that iterates through all possible combinations of value groups and calculates the prefix hit count. In our empirical evaluation, it is impractical to run on larger datasets.

Thus, we test the first (10, 25, 50, 100, 200) rows for each dataset and terminate OPHR runs exceeding 2 hours, reporting the result of the successful run with the most rows. For PDMX, we reduce 57 columns to 10 to enable runs on even as few as 10 rows. The PHR (prefix hit rate) and solver runtime in seconds across datasets are reported in Table 1, with the dataset labeled as {*dataset*}-{*#rows*}.

Dataset	**PHR (%)**			**Solver Runtime (s)**	
	OPHR	**GGR**	**Diff**	**OPHR**	**GGR**
Movies-50	80.6	80.6	0%	2556	0.05
Products-25	19.7	18.5	-1.2%	357	0.06
BIRD-50	77.5	76.2	-1.3%	0.43	0.05
PDMX-25	29.4	28.6	-0.8%	822	0.05
Fever-50	7.3	6.9	-0.4%	110	0.23
Beer-10	25.7	25.6	-0.1%	1269	0.08
SQuAD-10	34.0	34.0	0%	1.6	0.05

Table 1. Comparison of Prefix Hit Rate (PHR) and solver runtime across datasets. GGR achieves near-optimal PHR while being orders of magnitude faster than OPHR.

We can see that on these small samples of the datasets, our algorithm (GGR) achieves within 2% of the optimal, but can be up to *hours faster* on solver runtime.

D.2 Results of Smaller Model

To analyze the impact of using a smaller model, we run the Filter Query described in Fig.3a with the Llama-3.2-1B model, using the same setup as with Llama-3 8B (i.e., single L4 instance), and compare the prefix hit rate and end-to-end query execution time of GGR with the default vLLM baseline (i.e. Cache Original). The results are reported in Table 2.

Metric	**BIRD**	**Movies**	**PDMX**
Runtime (orig/GGR)	1.5×	1.3×	1.3×
Orig PHR (%)	10.41	29.32	11.97
GGR PHR (%)	83.99	82.10	56.00
Metric	**Products**	**BEER**	
Runtime (orig/GGR)	1.4×	1.2×	
Orig PHR (%)	24.06	47.98	
GGR PHR (%)	82.10	73.93	

Table 2. Cache runtime ratio and prefix hit rate (PHR) (%) comparison between original and GGR ordering for Llama-3.2-1B.

We observe similar prefix hit rates with Llama-3.2-1B compared to our previous 8B model runs. This consistency arises from the effectiveness of GGR field reordering, which converts non-reusable field contents (0 hits) into reusable prefixes within the cache. We also observe that under the same GPU instance setup (e.g., L4 with 24 GB memory), larger models like Llama-8B (7.6 GB) exhibit larger relative performance gains from GGR compared to smaller models like Llama-1B (1.8 GB), despite seeing similar prefix hit rates. This is because prefix caching benefits from reducing computational overhead on shared prefixes and enabling larger batch sizes for LLM generation by reducing memory usage through sharing. For smaller models, the availability of ample GPU memory diminishes the relative impact of prefix caching, as larger batch sizes can be achieved without relying on caching. But for larger models, or when there is less available GPU space, prefix caching benefits become more pronounced.