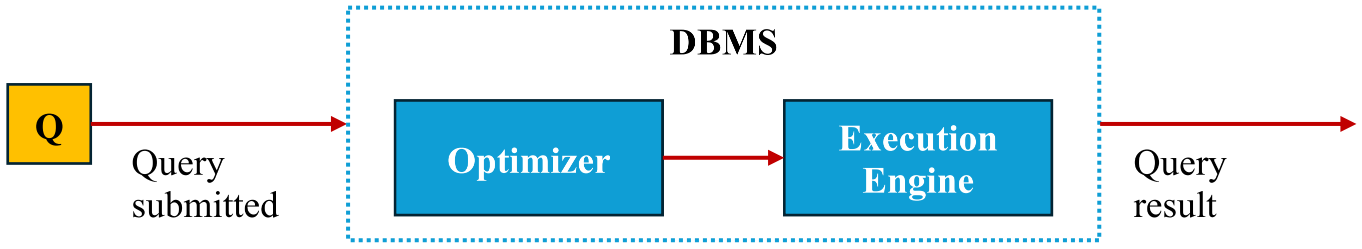
The Unreasonable Effectiveness of LLMs for Query Optimization

Peter Akioyamen, Zixuan Yi, Ryan Marcus
Database Group at The University of Pennsylvania

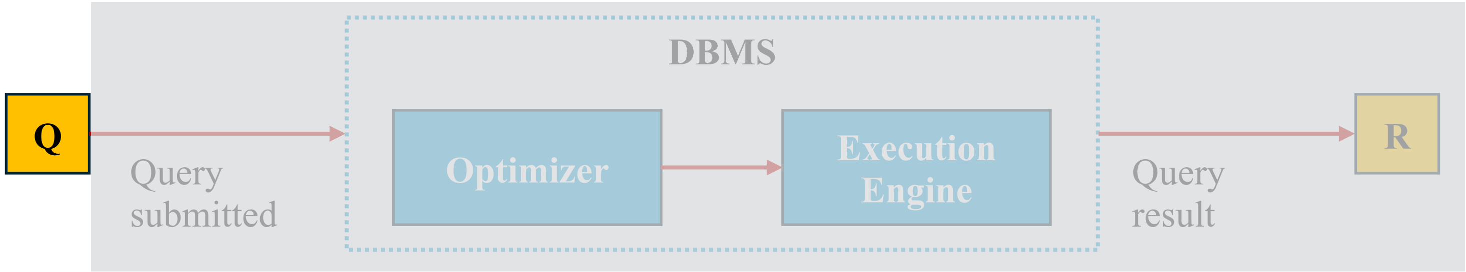
ML for Systems Workshop at NeurIPS 2024

Background: What is a query optimizer?

Background: What is a query optimizer?

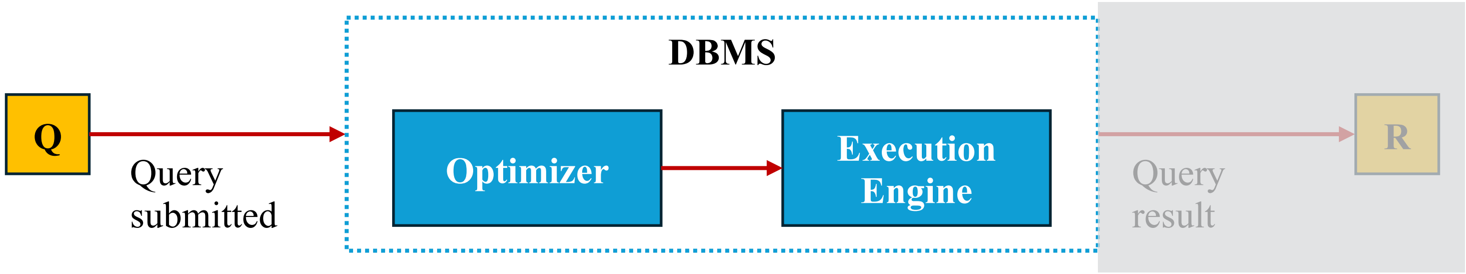


Background: What is a query optimizer?

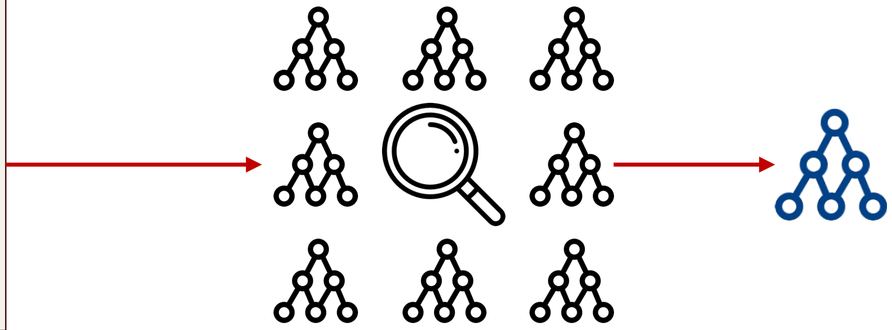



```
SELECT t.title AS movie,
       cn.country_code AS country
FROM company_name AS cn,
     movie_companies AS mc,
     title AS t
WHERE t.production_year > 2005
      AND t.id = mc.movie_id
      AND cn.id = mc.company_id;
```

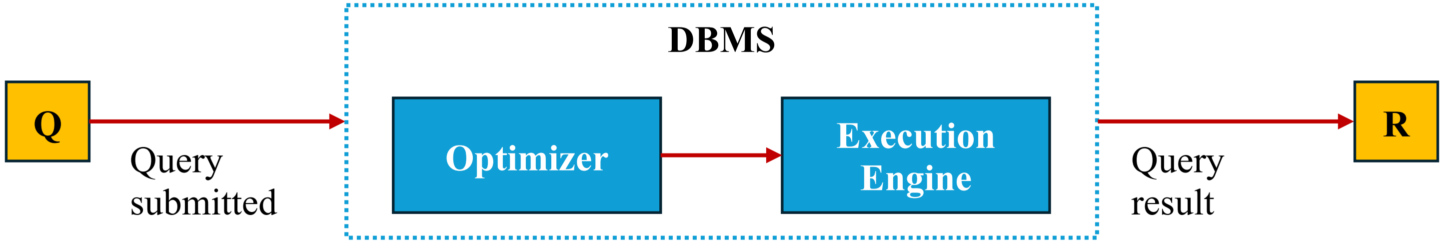
Background: What is a query optimizer?



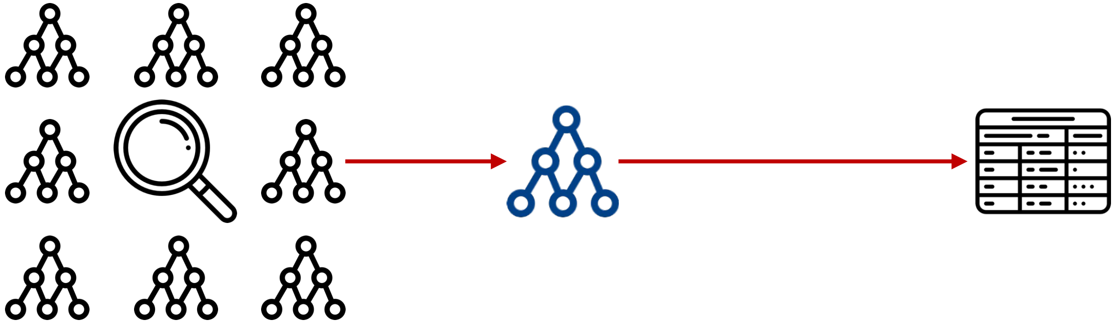

```
SELECT t.title AS movie,
       cn.country_code AS country
FROM company_name AS cn,
     movie_companies AS mc,
     title AS t
WHERE t.production_year > 2005
      AND t.id = mc.movie_id
      AND cn.id = mc.company_id;
```



Background: What is a query optimizer?




```sql
SELECT t.title AS movie,
       cn.country_code AS country
FROM company_name AS cn,
     movie_companies AS mc,
     title AS t
WHERE t.production_year > 2005
      AND t.id = mc.movie_id
      AND cn.id = mc.company_id;
```



Background: Why is query optimization difficult?

Background: Why is query optimization difficult?

There are many ways we can execute a query – "Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

FLIGHT

f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT

ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE

p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE

ar_id	name
AC	Air C
AA	Amer
…	…

Background: Why is query optimization difficult?

There are many ways we can execute a query – "Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

FLIGHT

f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT

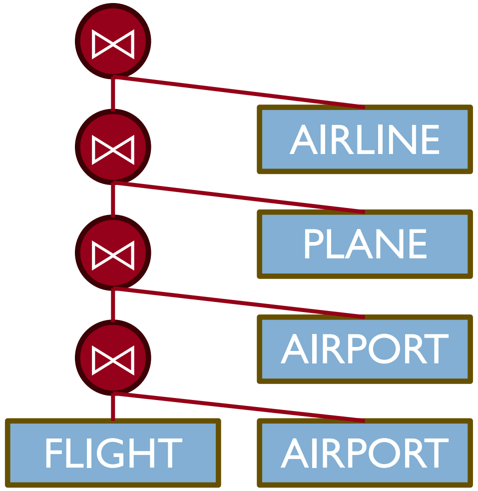
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE

p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE

ar_id	name
AC	Air C
AA	Amer
…	…



Background: Why is query optimization difficult?

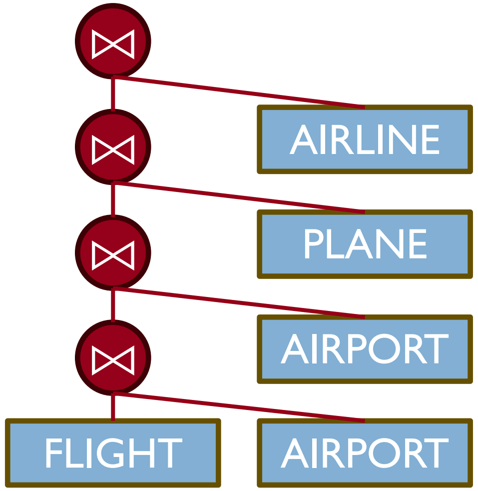
There are many ways we can execute a query – "Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

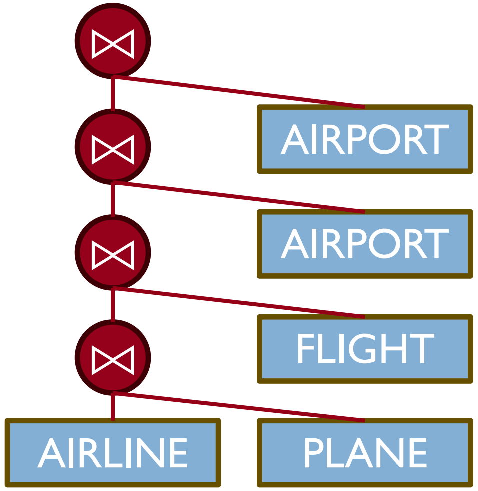
FLIGHT			
f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT		
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE		
p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE	
ar_id	name
AC	Air C
AA	Amer
…	…





Background: Why is query optimization difficult?

There are many ways we can execute a query – "Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

FLIGHT

f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT

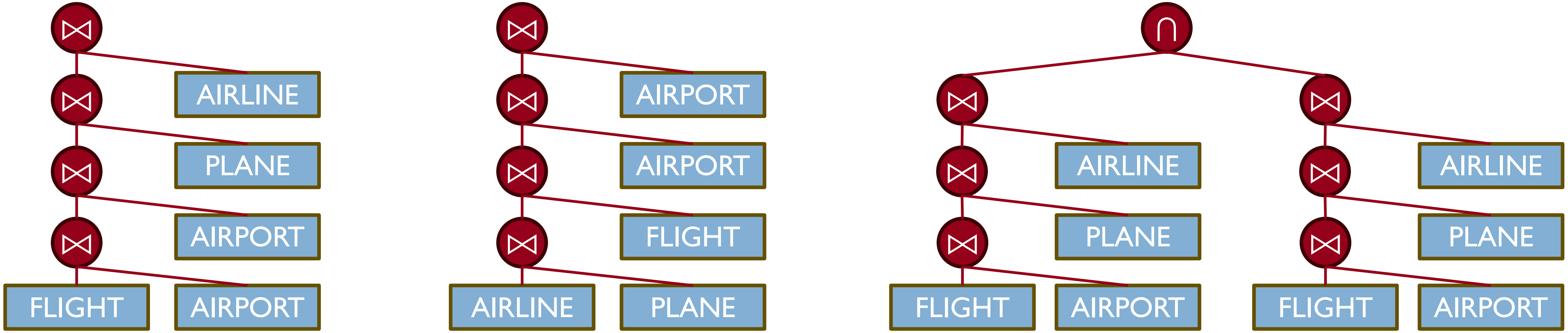
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE

p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE

ar_id	name
AC	Air C
AA	Amer
…	…



Background: Why is query optimization difficult?

Additionally, the query optimizer must still choose the physical operators for each join, that is, *how* to perform each join – hash join, nested for loop, sort then merge

FLIGHT

f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT

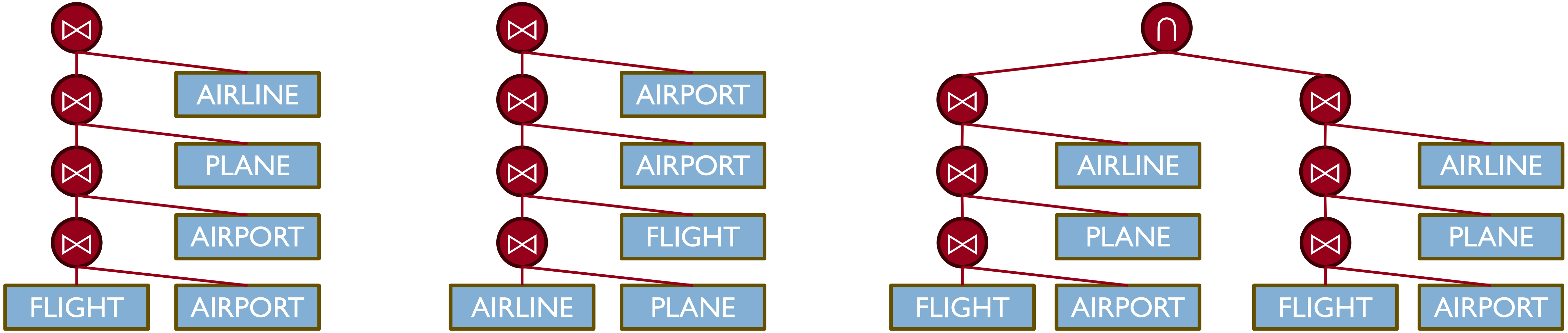
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE

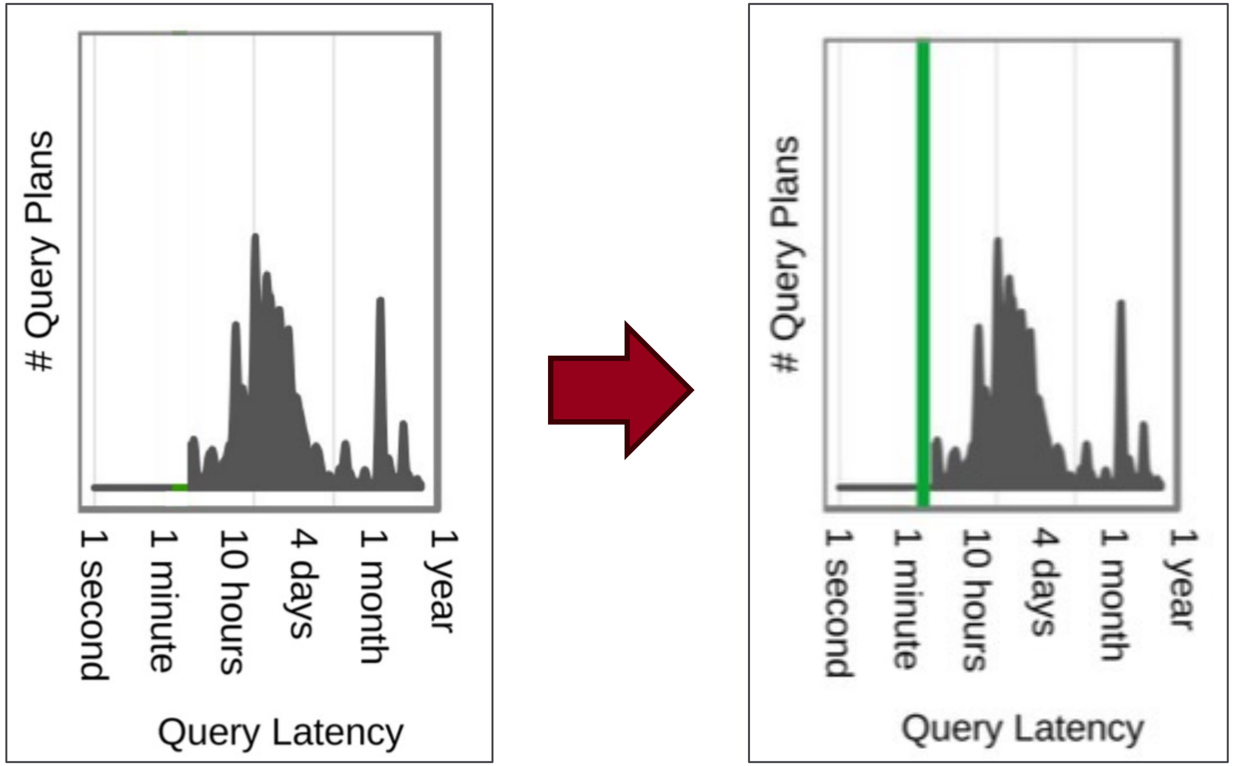
p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE

ar_id	name
AC	Air C
AA	Amer
…	…



Background: Why is query optimization difficult?



- The number of possible query plans follows Catalan numbers
- At $n = 19$ there are more than 2^{32} query plans
    - Traditional QOs use complex heuristics to eliminate very bad plans
    - But often select suboptimal plans, leaving performance on the table

Note: Figures from *Machine Learning for Query Optimization* by Ryan Marcus (https://rm.cab/brown22)

Background: SQL hints can be used to improve performance

Background: SQL hints can be used to improve performance

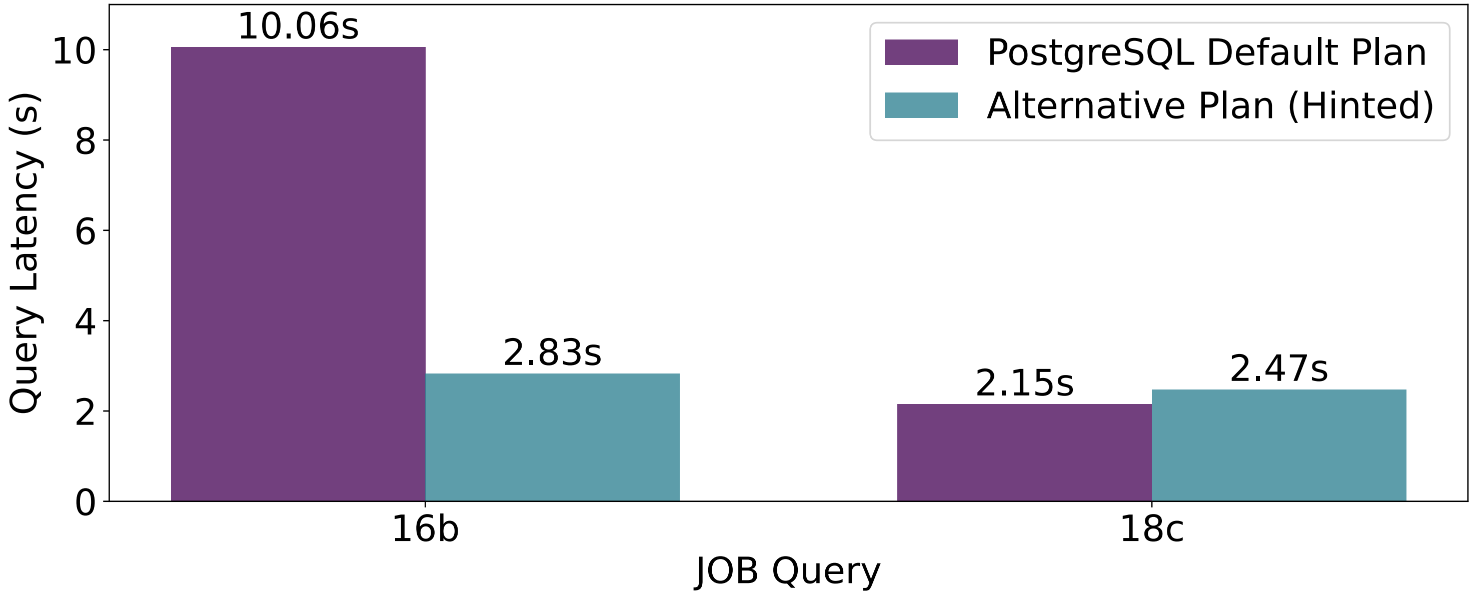
- Hints are optional clauses that can be inserted into a query to guide the optimizer into generating plans with specific characteristics

Background: SQL hints can be used to improve performance

- Hints are optional clauses that can be inserted into a query to guide the optimizer into generating plans with specific characteristics

- SQL hints provide a coarse-grained way to influence a query's execution plan, often chosen based on *a priori* knowledge of the data

Background: SQL hints can be used to improve performance



- Selecting hints can be extremely complicated for users, and providing the optimizer with incorrect hints can severely degrade query latency
- Different hints improve performance of some queries and degrade performance of others – this difference is often asymmetric

Background: SQL hints can be used to improve performance

"Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

FLIGHT

f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT

ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE

p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE

ar_id	name
AC	Air C
AA	Amer
…	…

Background: SQL hints can be used to improve performance

"Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

FLIGHT			
f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT		
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE		
p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE	
ar_id	name
AC	Air C
AA	Amer
…	…

```
SELECT *
FROM flight AS fl,
        plane AS pl, airline AS al
        airport AS ap_1, airport AS ap_2,
WHERE al.name = "Air Canada"
        AND ap_1.city = "Vancouver"
        AND ap_1.cntry = "CAN"
        AND ap_2.city = "Tokyo"
        AND ap_2.cntry = "JAP"
        AND pl.airline = al.ar_id
        AND fl.orig = ap_1.ap_id
        AND fl.dest = ap_2.ap_id
        AND fl.plane = pl.p_id;
```

Background: SQL hints can be used to improve performance

"Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

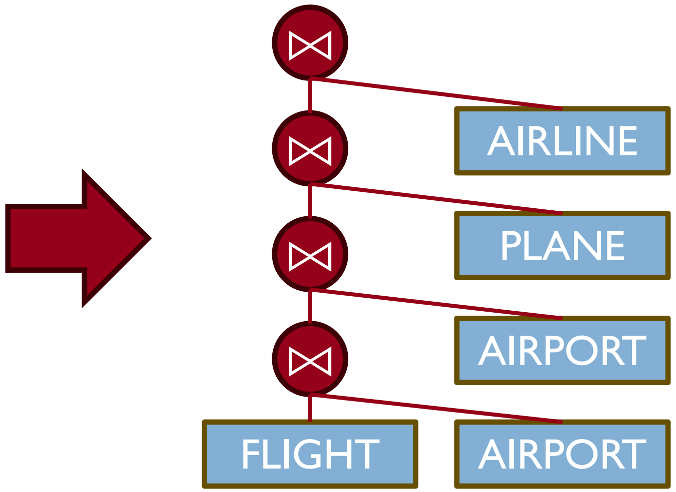
FLIGHT			
f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT		
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE		
p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE	
ar_id	name
AC	Air C
AA	Amer
…	…

```
SELECT *
FROM flight AS fl,
        plane AS pl, airline AS al
        airport AS ap_1, airport AS ap_2,
WHERE al.name = "Air Canada"
        AND ap_1.city = "Vancouver"
        AND ap_1.cntry = "CAN"
        AND ap_2.city = "Tokyo"
        AND ap_2.cntry = "JAP"
        AND pl.airline = al.ar_id
        AND fl.orig = ap_1.ap_id
        AND fl.dest = ap_2.ap_id
        AND fl.plane = pl.p_id;
```



Background: SQL hints can be used to improve performance

"Find all flights by Air Canada that originate in Vancouver, CA with a destination of Tokyo, JP"

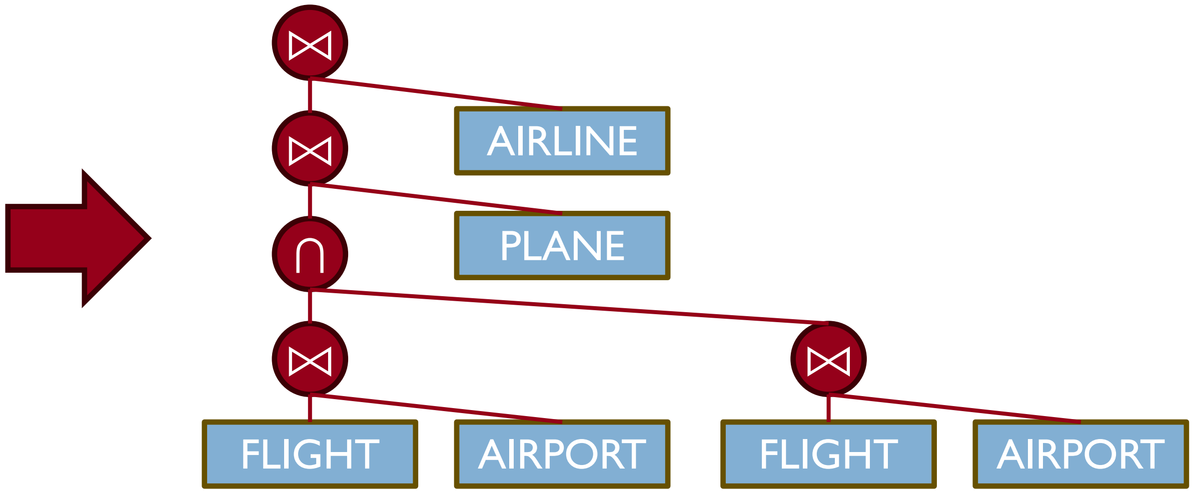
FLIGHT			
f_id	orig	dest	plane
1	LGA	YYZ	32
2	YVR	HND	1
…	…	…	…

AIRPORT		
ap_id	city	cntry
YVR	Van	CAN
HND	Tok	JAP
…	…	…

PLANE		
p_id	airline	model
1	AC	B747
2	UA	A350
…	…	…

AIRLINE	
ar_id	name
AC	Air C
AA	Amer
…	…

```
/*+ Parallel(ap_1, 3, hard) Parallel(ap_2, 3, hard) */
SELECT *
FROM flight AS fl,
        plane AS pl, airline AS al
        airport AS ap_1, airport AS ap_2,
WHERE al.name = "Air Canada"
        AND ap_1.city = "Vancouver"
        AND ap_1.cntry = "CAN"
        AND ap_2.city = "Tokyo"
        AND ap_2.cntry = "JAP"
        AND pl.airline = al.ar_id
        AND fl.orig = ap_1.ap_id
        AND fl.dest = ap_2.ap_id
        AND fl.plane = pl.p_id;
```



Motivation: Simplify query optimization through steering

Motivation: Simplify query optimization through steering

Current State-of-the-Art

- Modern methods use supervised learning[1], RL[2], or hybrid approaches[3], but perform sophisticated feature engineering on internal database statistics

[1] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner. FASTgres: Making Learned Query Optimizer Hinting Effective. Proceedings of the VLDB Endowment, Aug. 2023.
[2] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[3] C. Anneser, N. Tatbul, D. Cohen, Z. Xu, P. Pandian, N. Laptev, and R. Marcus. Autosteer: Learned query optimization for any sql database. Proceedings of the VLDB Endowment, Aug. 2023.
[4] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. arXiv preprint arXiv:1809.00677, 2018.
[5] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

Motivation: Simplify query optimization through steering

Current State-of-the-Art

- Modern methods use supervised learning[1], RL[2], or hybrid approaches[3], but perform sophisticated feature engineering on internal database statistics
- Common wisdom within the database community is that complex features such as cardinality estimates[4] or operator models[5] are required for query optimization

[1] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner. FASTgres: Making Learned Query Optimizer Hinting Effective. Proceedings of the VLDB Endowment, Aug. 2023.
[2] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[3] C. Anneser, N. Tatbul, D. Cohen, Z. Xu, P. Pandian, N. Laptev, and R. Marcus. Autosteer: Learned query optimization for any sql database. Proceedings of the VLDB Endowment, Aug. 2023.
[4] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. arXiv preprint arXiv:1809.00677, 2018.
[5] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

Motivation: Simplify query optimization through steering

Current State-of-the-Art

- Modern methods use supervised learning[1], RL[2], or hybrid approaches[3], but perform sophisticated feature engineering on internal database statistics
- Common wisdom within the database community is that complex features such as cardinality estimates[4] or operator models[5] are required for query optimization
- Often there is a need to materialize at least some of the potential query plans before optimizing the decisions of the query optimizer

[1] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner. FASTgres: Making Learned Query Optimizer Hinting Effective. Proceedings of the VLDB Endowment, Aug. 2023.
[2] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[3] C. Anneser, N. Tatbul, D. Cohen, Z. Xu, P. Pandian, N. Laptev, and R. Marcus. Autosteer: Learned query optimization for any sql database. Proceedings of the VLDB Endowment, Aug. 2023.
[4] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. arXiv preprint arXiv:1809.00677, 2018.
[5] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

Motivation: Simplify query optimization through steering

Current State-of-the-Art

- Modern methods use supervised learning[1], RL[2], or hybrid approaches[3], but perform sophisticated feature engineering on internal database statistics
- Common wisdom within the database community is that complex features such as cardinality estimates[4] or operator models[5] are required for query optimization
- Often there is a need to materialize at least some of the potential query plans before optimizing the decisions of the query optimizer
- These requirements have hindered practical adoption, *so how else can we reclaim the performance that traditional QOs leave on the table?*

[1] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner. FASTgres: Making Learned Query Optimizer Hinting Effective. Proceedings of the VLDB Endowment, Aug. 2023.
[2] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[3] C. Anneser, N. Tatbul, D. Cohen, Z. Xu, P. Pandian, N. Laptev, and R. Marcus. Autosteer: Learned query optimization for any sql database. Proceedings of the VLDB Endowment, Aug. 2023.
[4] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. arXiv preprint arXiv:1809.00677, 2018.
[5] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

Motivation: Simplify query optimization through steering

Current State-of-the-Art

- Modern methods use supervised learning[1], RL[2], or hybrid approaches[3], but perform sophisticated feature engineering on internal database statistics
- Common wisdom within the database community is that complex features such as cardinality estimates[4] or operator models[5] are required for query optimization
- Often there is a need to materialize at least some of the potential query plans before optimizing the decisions of the query optimizer
- These requirements have hindered practical adoption, *so how else can we reclaim the performance that traditional QOs leave on the table?*

Simplify feature engineering and learn to steer the query optimizer using hints!

[1] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner. FASTgres: Making Learned Query Optimizer Hinting Effective. Proceedings of the VLDB Endowment, Aug. 2023.
[2] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[3] C. Anneser, N. Tatbul, D. Cohen, Z. Xu, P. Pandian, N. Laptev, and R. Marcus. Autosteer: Learned query optimization for any sql database. Proceedings of the VLDB Endowment, Aug. 2023.
[4] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. arXiv preprint arXiv:1809.00677, 2018.
[5] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data

* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data

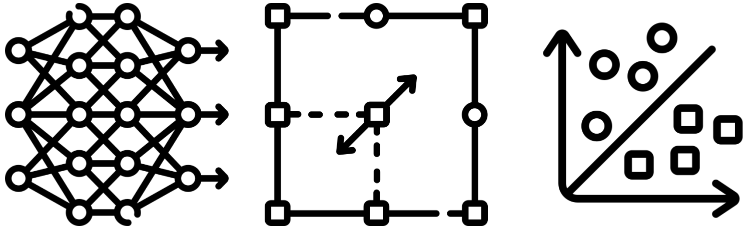
1. ~3000 SQL queries from the join order[1] and cardinality estimation[2] benchmarks
2. Gathered 48 well-known PostgreSQL hints used in prior work[3-4]
3. Executed queries 5 times per hint – mean latency was used for analysis
4. The hint with the best performance gains relative to the default PostgreSQL plan was selected *a priori* as the Alternative plan

* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data



Modelling

1. ~3000 SQL queries from the join order[1] and cardinality estimation[2] benchmarks
2. Gathered 48 well-known PostgreSQL hints used in prior work[3-4]
3. Executed queries 5 times per hint – mean latency was used for analysis
4. The hint with the best performance gains relative to the default PostgreSQL plan was selected *a priori* as the Alternative plan

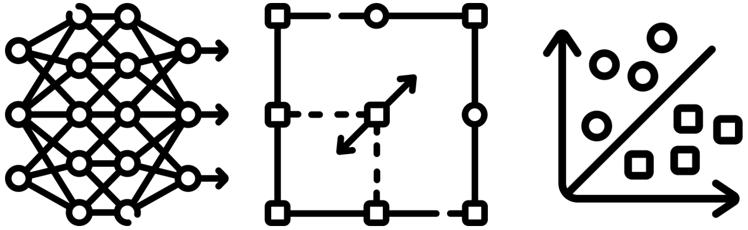
* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data

1. ~3000 SQL queries from the join order[1] and cardinality estimation[2] benchmarks
2. Gathered 48 well-known PostgreSQL hints used in prior work[3-4]
3. Executed queries 5 times per hint – mean latency was used for analysis
4. The hint with the best performance gains relative to the default PostgreSQL plan was selected *a priori* as the Alternative plan



Modelling

1. Embed raw SQL queries using an LLM
2. Truncate embeddings using PCA (120 dimensions captures ~90% of variance)
3. Trained binary classifiers to steer between the Default Plan and the Alternative Plan produced by the selected hint*

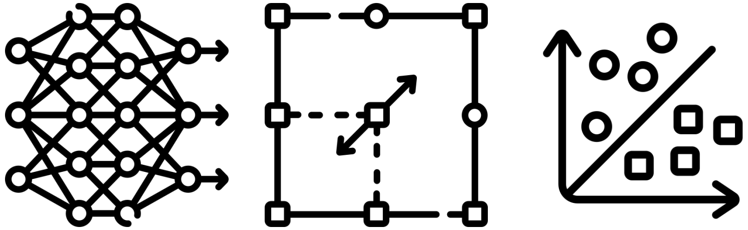
* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data

1. ~3000 SQL queries from the join order[1] and cardinality estimation[2] benchmarks
2. Gathered 48 well-known PostgreSQL hints used in prior work[3-4]
3. Executed queries 5 times per hint – mean latency was used for analysis
4. The hint with the best performance gains relative to the default PostgreSQL plan was selected *a priori* as the Alternative plan



Modelling

1. Embed raw SQL queries using an LLM
2. Truncate embeddings using PCA (120 dimensions captures ~90% of variance)
3. Trained binary classifiers to steer between the Default Plan and the Alternative Plan produced by the selected hint*

LLMSteer

* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data

1. ~3000 SQL queries from the join order[1] and cardinality estimation[2] benchmarks
2. Gathered 48 well-known PostgreSQL hints used in prior work[3-4]
3. Executed queries 5 times per hint – mean latency was used for analysis
4. The hint with the best performance gains relative to the default PostgreSQL plan was selected *a priori* as the Alternative plan

Modelling

1. Embed raw SQL queries using an LLM
2. Truncate embeddings using PCA (120 dimensions captures ~90% of variance)
3. Trained binary classifiers to steer between the Default Plan and the Alternative Plan produced by the selected hint*

LLMSteer

Evaluation

* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

Training Data

1. ~3000 SQL queries from the join order[1] and cardinality estimation[2] benchmarks
2. Gathered 48 well-known PostgreSQL hints used in prior work[3-4]
3. Executed queries 5 times per hint – mean latency was used for analysis
4. The hint with the best performance gains relative to the default PostgreSQL plan was selected *a priori* as the Alternative plan

Modelling

1. Embed raw SQL queries using an LLM
2. Truncate embeddings using PCA (120 dimensions captures ~90% of variance)
3. Trained binary classifiers to steer between the Default and Alternative Plans produced by the selected hint*

LLMSteer

Evaluation

1. Stratified cross-validation
2. Query Optimization Metrics:
   I. Cumulative execution time of queries (*total latency*)
   II. 90th percentile latency of queries (*P90 latency*)
3. Classification Metrics:
   I. Recall
   II. AUROC
   III. Accuracy

* Results shown for SVM with RBF kernel only, which was the best performing model.

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? Proceedings of the VLDB Endowment, Nov. 2015
[2] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh. Flow-loss: Learning cardinality estimates that matter. Proceedings of the VLDB Endowment, July 2021.
[3] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data, New York, NY, USA, 2021.
[4] R. Heinrich, M. Luthra, H. Kornmayer, and C. Binnig. Zero-shot cost models for distributed stream processing. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems. June 2022.

LLMSteer: A simpler approach to query optimization using large language models

```
SELECT t.title AS movie, cn.
    country_code AS country
FROM company_name AS cn,
    movie_companies AS mc,
    title AS t
WHERE t.production_year > 2005
    AND t.id = mc.movie_id
    AND cn.id = mc.company_id;
```

LLMSteer: A simpler approach to query optimization using large language models

```sql
SELECT t.title AS movie, cn.
    country_code AS country
FROM company_name AS cn,
     movie_companies AS mc,
     title AS t
WHERE t.production_year > 2005
     AND t.id = mc.movie_id
     AND cn.id = mc.company_id;
```

1 LLM embedding

$$[-0.04813609, -0.43741802, \ldots, -0.28336727, 0.98100264]$$

d-dimensional query embedding

LLMSteer: A simpler approach to query optimization using large language models

```sql
SELECT t.title AS movie, cn.
    country_code AS country
FROM company_name AS cn,
     movie_companies AS mc,
     title AS t
WHERE t.production_year > 2005
     AND t.id = mc.movie_id
     AND cn.id = mc.company_id;
```

1 LLM embedding

$$[-0.04813609, -0.43741802, \ldots, -0.28336727, 0.98100264]$$

d-dimensional query embedding

2 Truncate embeddings

$$[0.74970049, -1.36168475, \ldots, 0.78163968, 0.66720773]$$

120 principal components

LLMSteer: A simpler approach to query optimization using large language models

```sql
SELECT t.title AS movie, cn.
    country_code AS country
FROM company_name AS cn,
    movie_companies AS mc,
    title AS t
WHERE t.production_year > 2005
    AND t.id = mc.movie_id
    AND cn.id = mc.company_id;
```

1 LLM embedding

$$[-0.04813609, -0.43741802, \ldots, -0.28336727, 0.98100264]$$

d-dimensional query embedding

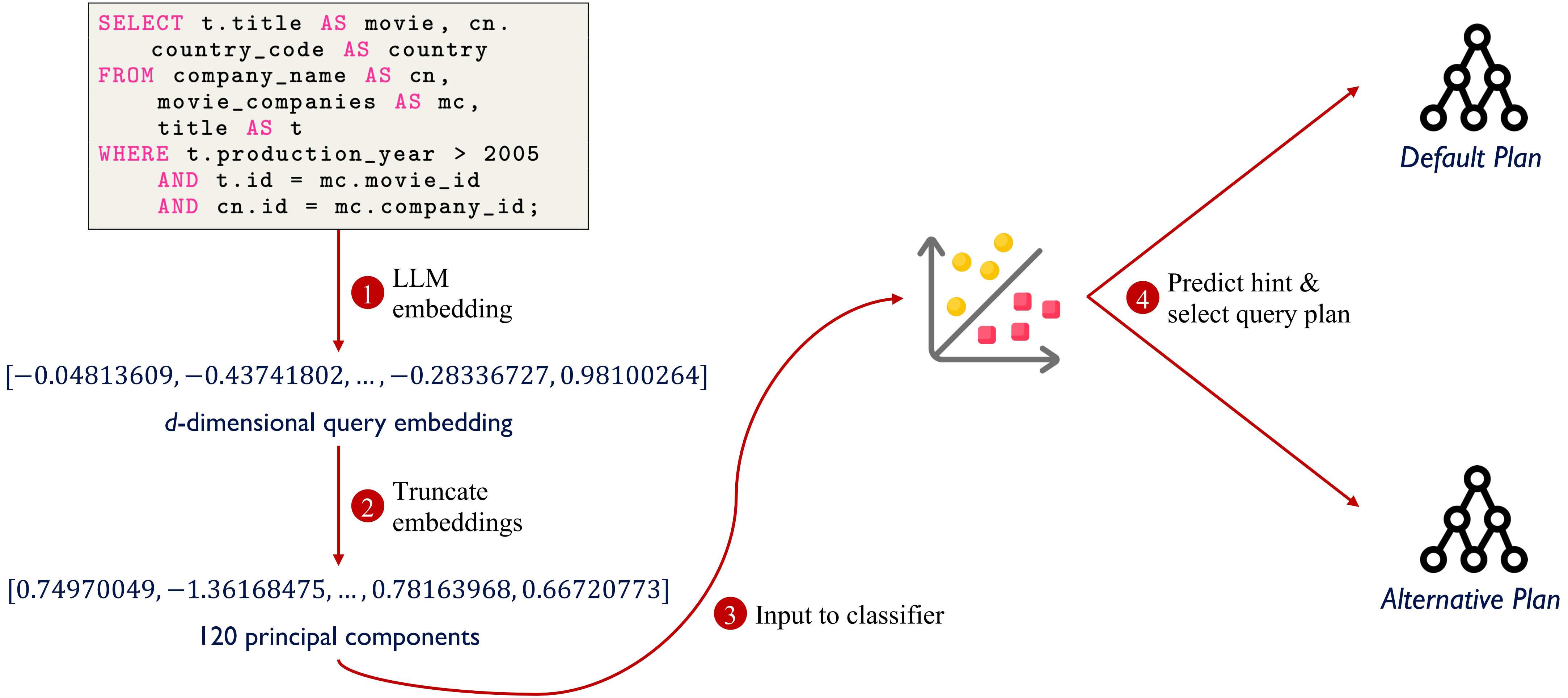
2 Truncate embeddings

$$[0.74970049, -1.36168475, \ldots, 0.78163968, 0.66720773]$$

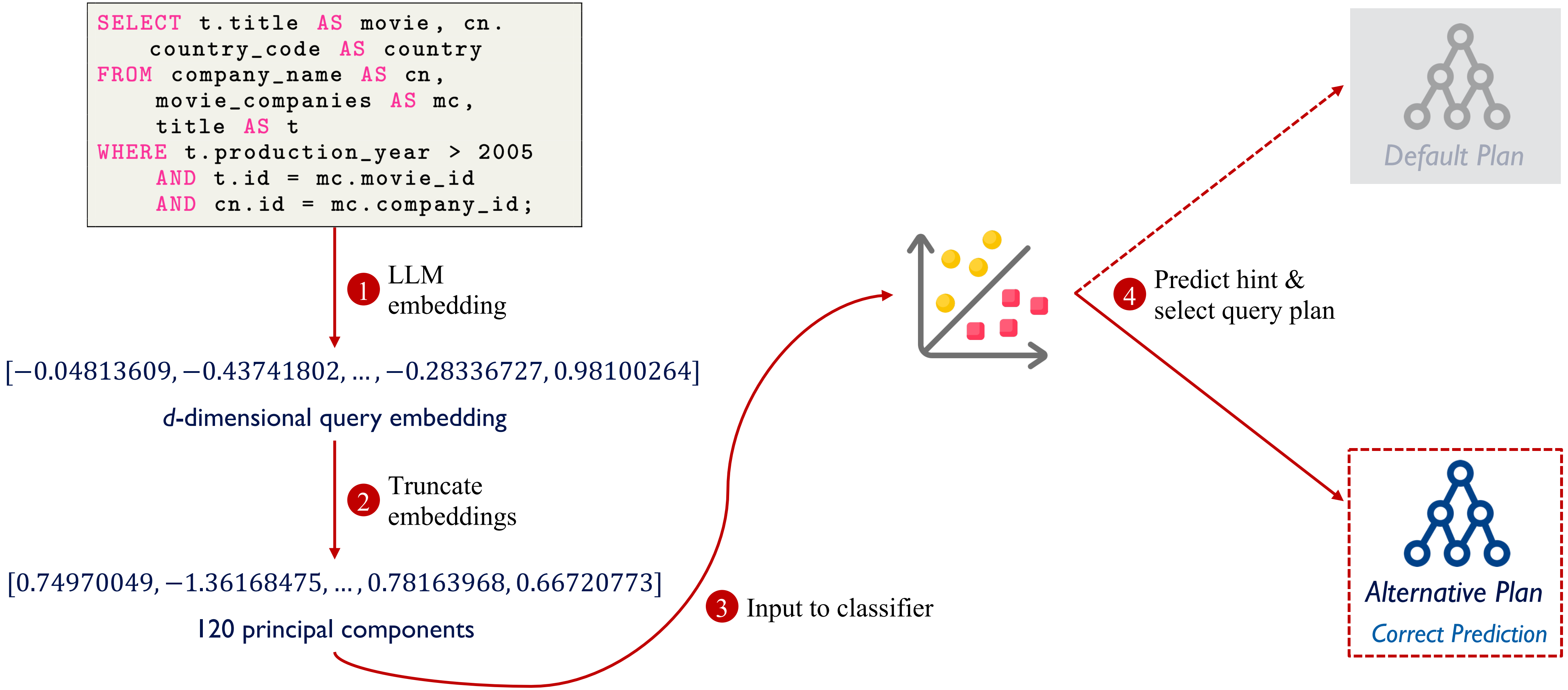
120 principal components

3 Input to classifier

LLMSteer: A simpler approach to query optimization using large language models



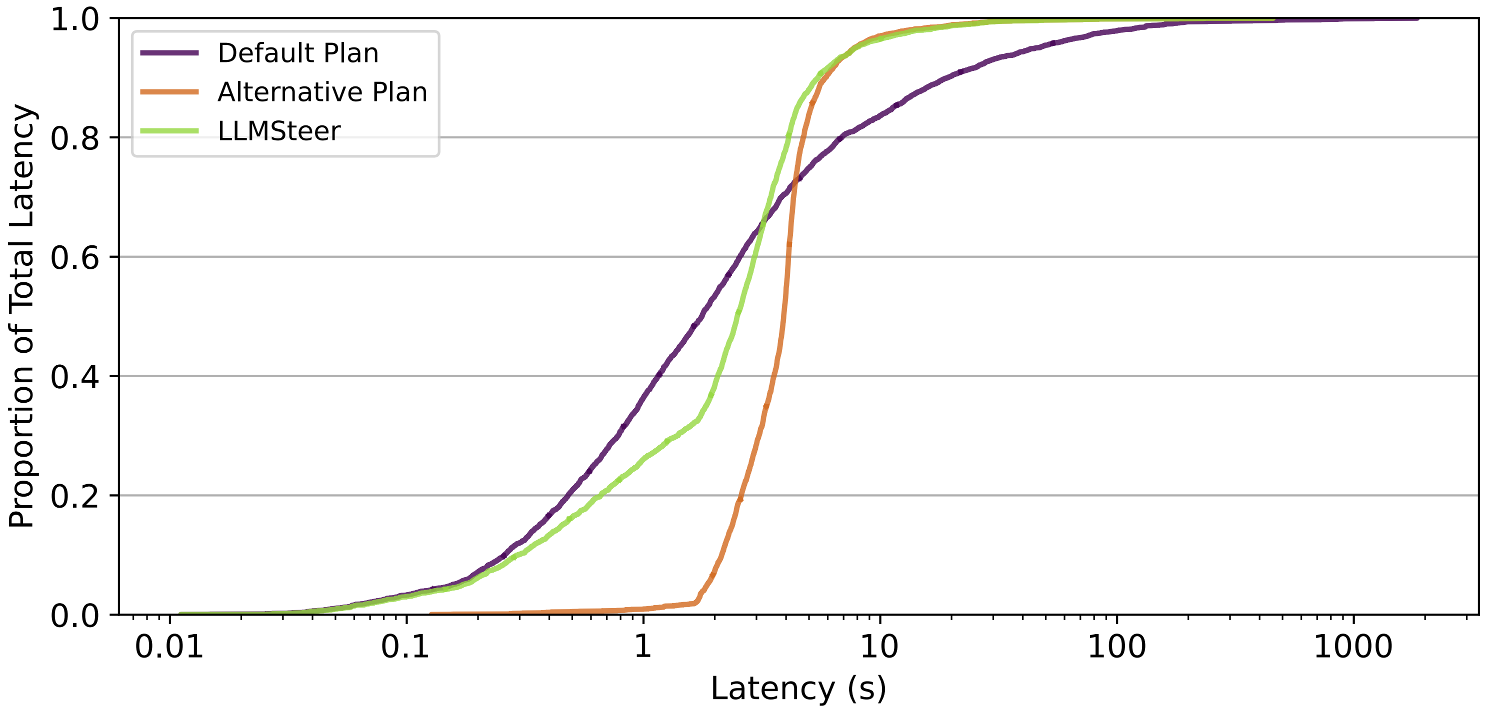
LLMSteer: A simpler approach to query optimization using large language models



Penn Engineering

Experiment 1: Comparative performance of LLMSteer

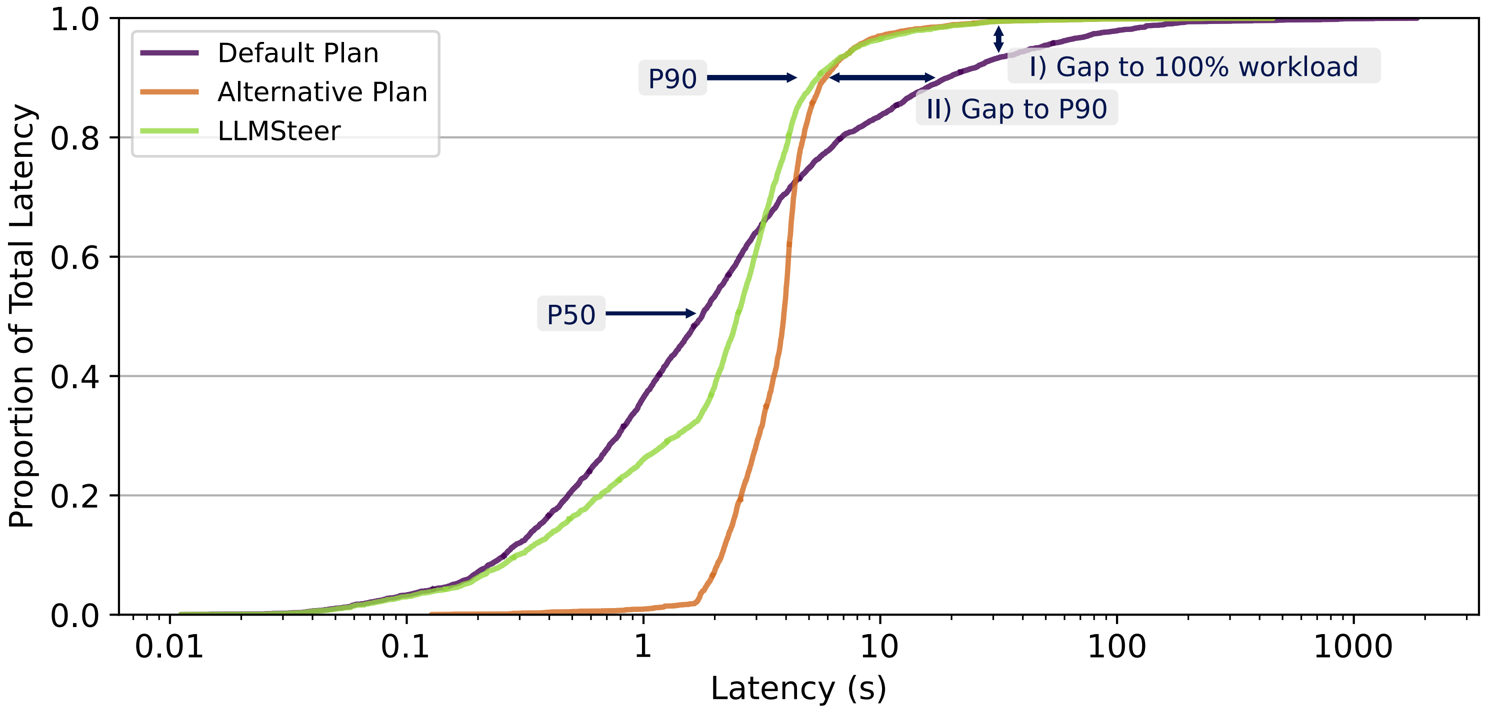
Experiment 1: Comparative performance of LLMSteer



Empirical CDF of latency across cross-validation testing workloads

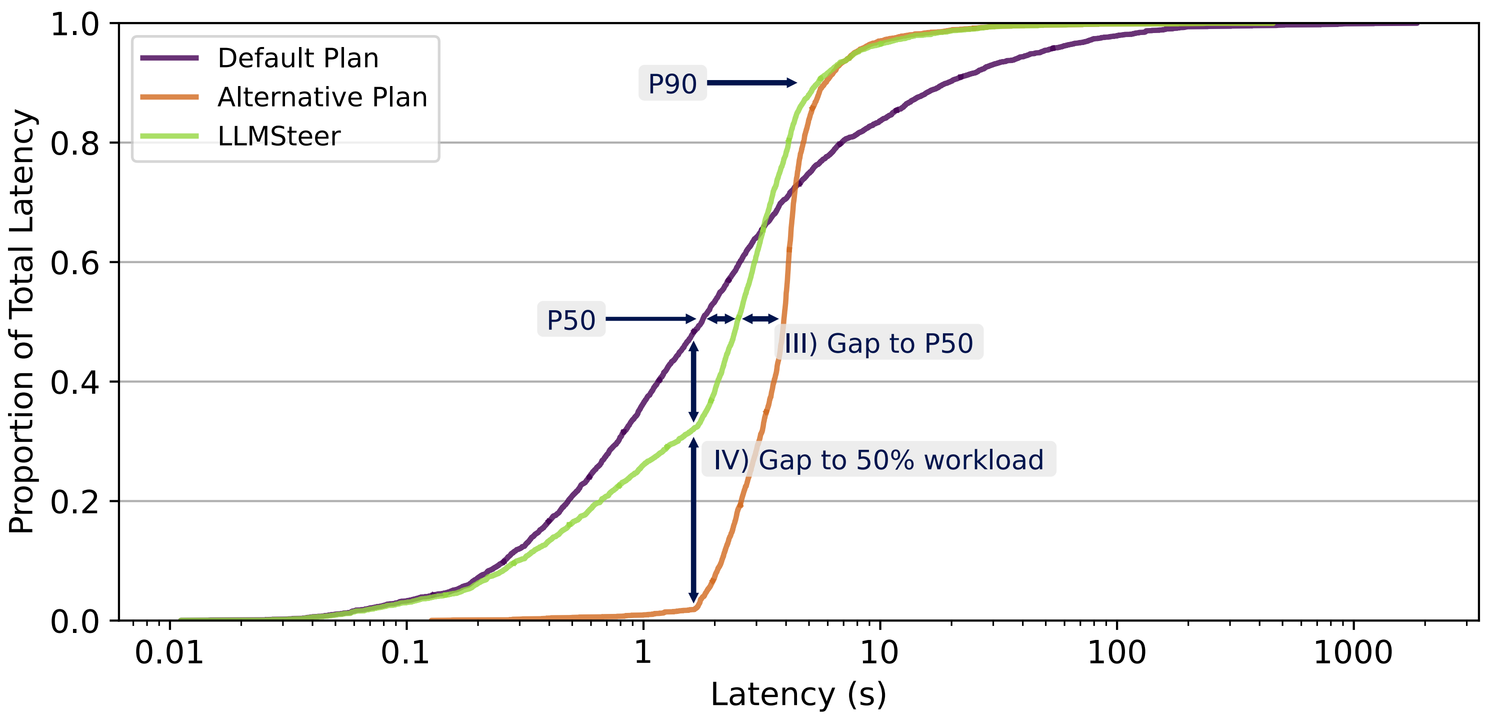
- Purple indicates selecting the default plan for all queries
- Orange indicates selecting the alternative plan for all queries

Experiment 1: Comparative performance of LLMSteer



- LLMSteer outperforms both the default and alternative plans at the higher end of the distribution, achieving a lower P90 (II) and saturating just as fast the Alternative plan (I)

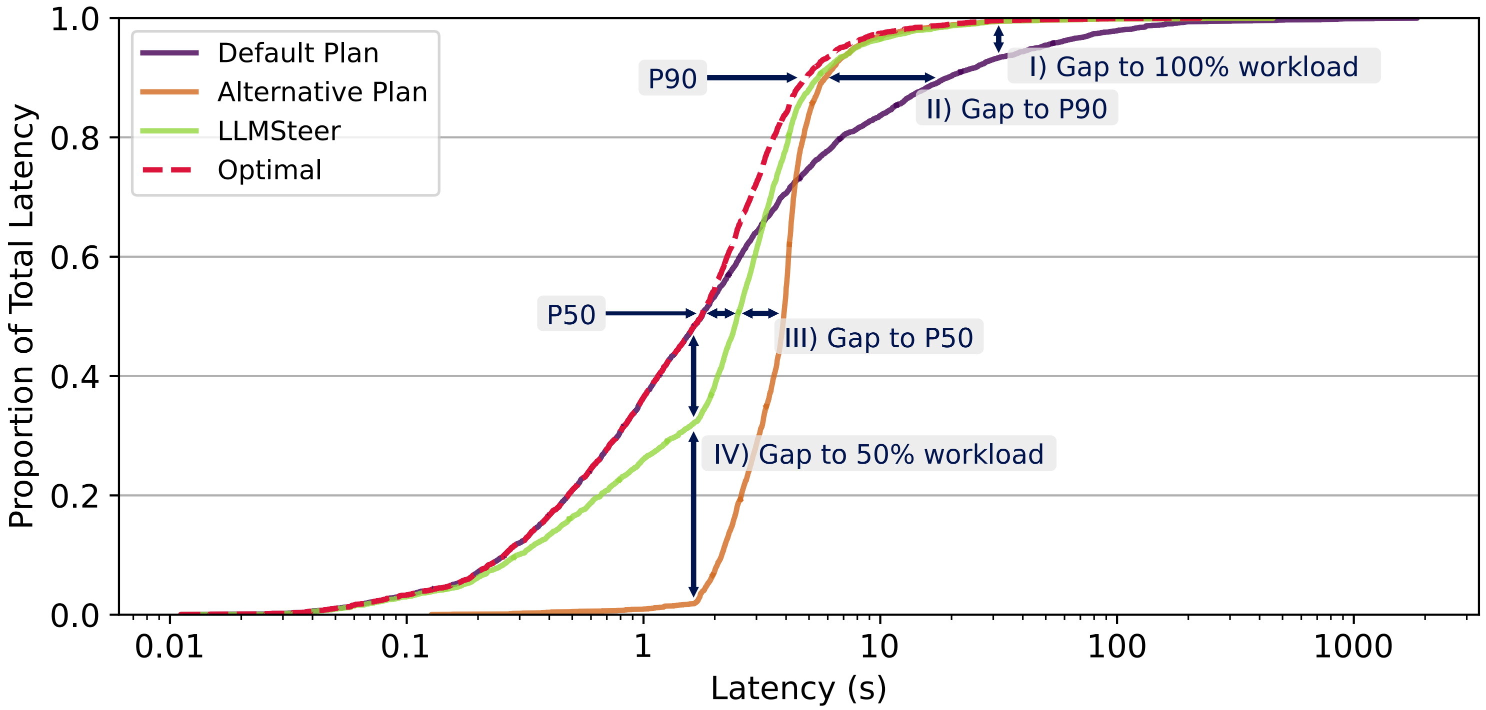
Experiment 1: Comparative performance of LLMSteer



- LLMSteer improves on the alternative plan, lowering the performance gap to the default plan, capturing more of the total latency earlier and improving the median latency

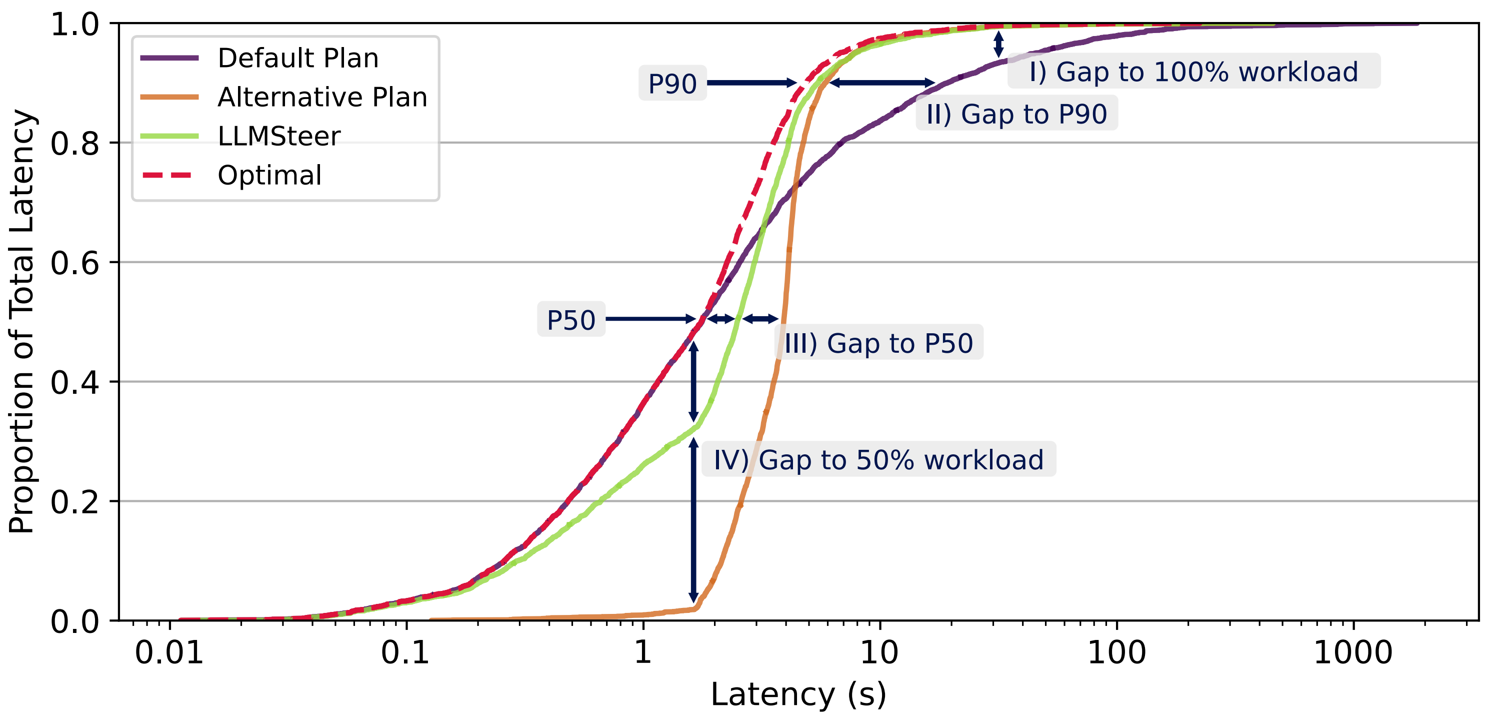
Penn Engineering

Experiment 1: Comparative performance of LLMSteer



- LLMSteer falls short of the optimal steering strategy, but effectively combines the benefits of the default PostgreSQL plan and the alternative

Experiment 1: Comparative performance of LLMSteer



- The system can be seen as trading a small increase in median latency for a large reduction in P90 and total latency, a trade-off that is worthwhile in most practical applications

Cautious Optimism and Next Steps

Cautious Optimism and Next Steps

Challenges and Limitations

- Internet-scale language models
  - Were LLMs trained on the JOB and CEB benchmark data?
  - More broadly, how do we create benchmarks in this new LLM-era?

Cautious Optimism and Next Steps

Challenges and Limitations

- Internet-scale language models
  - Were LLMs trained on the JOB and CEB benchmark data?
  - More broadly, how do we create benchmarks in this new LLM-era?
- Integration of LLMs into query pathways
  - No longer need to materialize query plans to perform optimization
  - No longer require internal database statistics
  - Must now integrate LLMs into query workflows and perform inference

Cautious Optimism and Next Steps

Challenges and Limitations

- Internet-scale language models
  - Were LLMs trained on the JOB and CEB benchmark data?
  - More broadly, how do we create benchmarks in this new LLM-era?
- Integration of LLMs into query pathways
  - No longer need to materialize query plans to perform optimization
  - No longer require internal database statistics
  - Must now integrate LLMs into query workflows and perform inference

Reasons for Optimism

- We did not expect this to work, but clearly LLMs *can* represent something meaningful about program semantics that is helpful for query optimization

Cautious Optimism and Next Steps

Challenges and Limitations

- Internet-scale language models
  - Were LLMs trained on the JOB and CEB benchmark data?
  - More broadly, how do we create benchmarks in this new LLM-era?
- Integration of LLMs into query pathways
  - No longer need to materialize query plans to perform optimization
  - No longer require internal database statistics
  - Must now integrate LLMs into query workflows and perform inference

Reasons for Optimism

- We did not expect this to work, but clearly LLMs *can* represent something meaningful about program semantics that is helpful for query optimization
- Quantization may play an essential role in improving latency and developing LLM-powered QOs

Summary

Summary

- We introduce LLMSteer, a simpler approach to query optimization using LLMs rather than internal database statistics.

Summary

- We introduce LLMSteer, a simpler approach to query optimization using LLMs rather than internal database statistics.
- LLMSteer is effective on two query benchmarks – relative to the PostgreSQL default plan the system reduces total and P90 latency by 72% and reduces median latency by 35% relative to the alternative.

Summary

- We introduce LLMSteer, a simpler approach to query optimization using LLMs rather than internal database statistics.
- LLMSteer is effective on two query benchmarks – relative to the PostgreSQL default plan the system reduces total and P90 latency by 72% and reduces median latency by 35% relative to the alternative.

There are still far more open questions than answers!

Thank you! Questions?

Our group: https://db.cis.upenn.edu

Our code: https://github.com/peter-ai/LLMSteer

Reach me at: peterai@seas.upenn.edu

ArXiv **Code** **Me**

Appendix

Experiment II: Robustness to non-semantic syntactic changes in SQL query formatting

Experiment II: Robustness to non-semantic syntactic changes in SQL query formatting

- Syntax A is the original query formatting – single-line declarative statements with no newlines or indentation

```
"SELECT t.title AS movie, cn.country_code AS country FROM company_name AS cn, movie_companies AS mc, title AS t WHERE t.production_year > 2005 AND t.id = mc.movie_id AND cn.id = mc.company_id;"
```

Experiment II: Robustness to non-semantic syntactic changes in SQL query formatting

- Syntax A is the original query formatting – single-line declarative statements with no newlines or indentation

```
"SELECT t.title AS movie, cn.country_code AS country FROM company_name AS cn, movie_companies AS mc, title AS t WHERE t.production_year > 2005 AND t.id = mc.movie_id AND cn.id = mc.company_id;"
```

- Syntax B introduces newline characters and uses whitespace for indentation

```
"SELECT t.title AS movie,
        cn.country_code AS country
FROM company_name AS cn,
        movie_companies AS mc,
        title AS t
WHERE t.production_year > 2005
        AND t.id = mc.movie_id
        AND cn.id = mc.company_id;"
```

Experiment II: Robustness to non-semantic syntactic changes in SQL query formatting

- Syntax A is the original query formatting – single-line declarative statements with no newlines or indentation

```
"SELECT t.title AS movie, cn.country_code AS country FROM company_name AS cn, movie_companies AS mc, title AS t WHERE t.production_year > 2005 AND t.id = mc.movie_id AND cn.id = mc.company_id;"
```

- Syntax B introduces newline characters and uses whitespace for indentation

```
"SELECT t.title AS movie,
        cn.country_code AS country
FROM company_name AS cn,
        movie_companies AS mc,
        title AS t
WHERE t.production_year > 2005
        AND t.id = mc.movie_id
        AND cn.id = mc.company_id;"
```

- Syntax C introduces newline characters and uses tabs for indentation

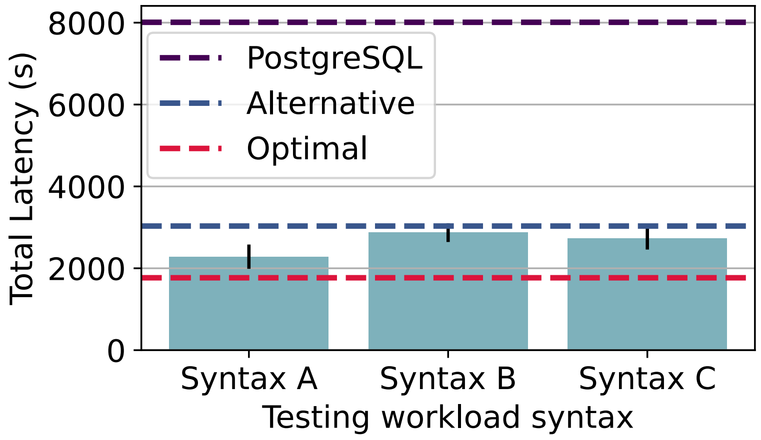
```
"SELECT t.title AS movie,
\tcn.country_code AS country
FROM company_name AS cn,
\tmovie_companies AS mc,
\ttitle AS t
WHERE t.production_year > 2005
\tAND t.id = mc.movie_id
\tAND cn.id = mc.company_id;"
```

Penn Engineering

Experiment 11: Robustness to non-semantic syntactic changes in SQL query formatting

The original SQL queries were single line statements – we evaluate LLMSteer on query formats that align more closely with how queries are written in production systems

- Syntax A is the original query formatting – single-line declarative statements with no newlines or indentation
- Syntax B introduces newline characters and uses whitespace for indentation
- Syntax C introduces newline characters and uses tabs for indentation

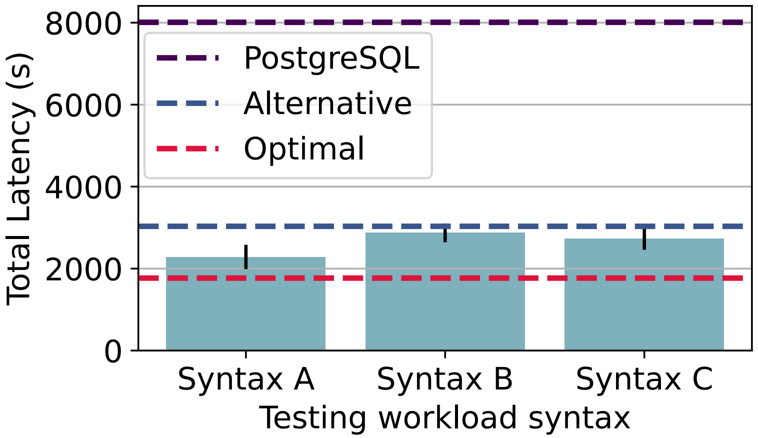


a) Trained on Syntax A

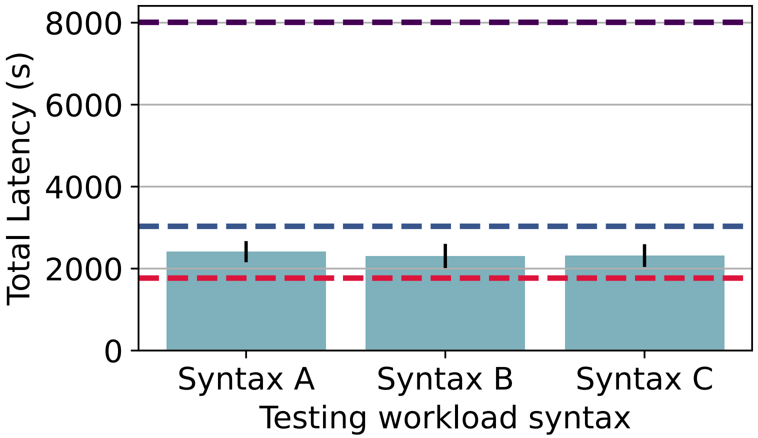
Experiment II: Robustness to non-semantic syntactic changes in SQL query formatting

The original SQL queries were single line statements – we evaluate LLMSteer on query formats that align more closely with how queries are written in production systems

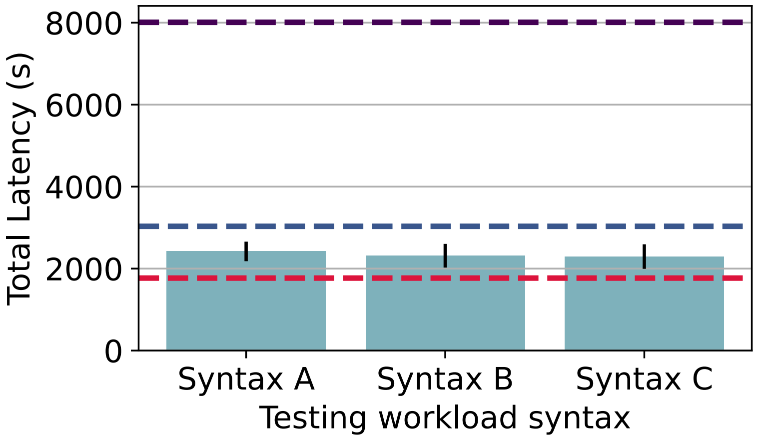
- Syntax A is the original query formatting – single-line declarative statements with no newlines or indentation
- Syntax B introduces newline characters and uses whitespace for indentation
- Syntax C introduces newline characters and uses tabs for indentation



a) Trained on Syntax A



b) Trained on Syntax B



c) Trained on Syntax C