Multi-Query Optimization for the Novel ANALYZE Operator

Marios Iakovidis · Panos Vassiliadis

University of Ioannina, Greece

DOLAP 2026 · March 24 2026, Tampere, Finland

In This Paper

I The ANALYZE Operator

Novel intentional operator

360° view of a data subspace in a single invocation

II Multi-Query Optimization

Optimization strategies

Optimization strategies to reduce the number of queries issued

III Evaluation

Four datasets · 70 workloads

Algorithm performance evaluation + areas of competence

Related Work & Context

ANALYZE in the context of related work: intentional analytical model [1] and highlight extraction

DIFF [2]	ASSESS [3]	MetaInsight [4]	EXPLAIN [5]	ANALYZE ★
Why do two aggregates differ across subspaces?	*How does a cube cell compares to a benchmark?*	*What structured knowledge is hidden across a dataset?*	*Why does this measure behave this way?*	*Give me a 360° view of this subspace*
Explains differences in multidimensional aggregates by pinpointing contributing subspaces.	Cube cells rated vs. expected values.	Automatic discovery of structured knowledge patterns for exploratory data analysis.	Statistical model of cause-effect.	Original + siblings + drill-downs.
Sarawagi 1999	*Francia et al. 2021/2023*	*Ma et al. 2021*	*Francia et al. 2024*	*This paper*

[1]: Vassiliadis, Marcel & Rizzi, Inf. Syst. 2019. **[2]:** Sarawagi, VLDB 1999. **[3]:** Francia et al., EDBT 2021 / TKDE 2023. **[4]:** Ma et al., SIGMOD 2021. **[5]:** Francia et al., Inf. Syst. 2024. **[6]:** Vassiliadis, Marcel & Rizzi, Inf. Syst. 2019.

I

The ANALYZE Operator

II Multi-Query Optimization

III Evaluation

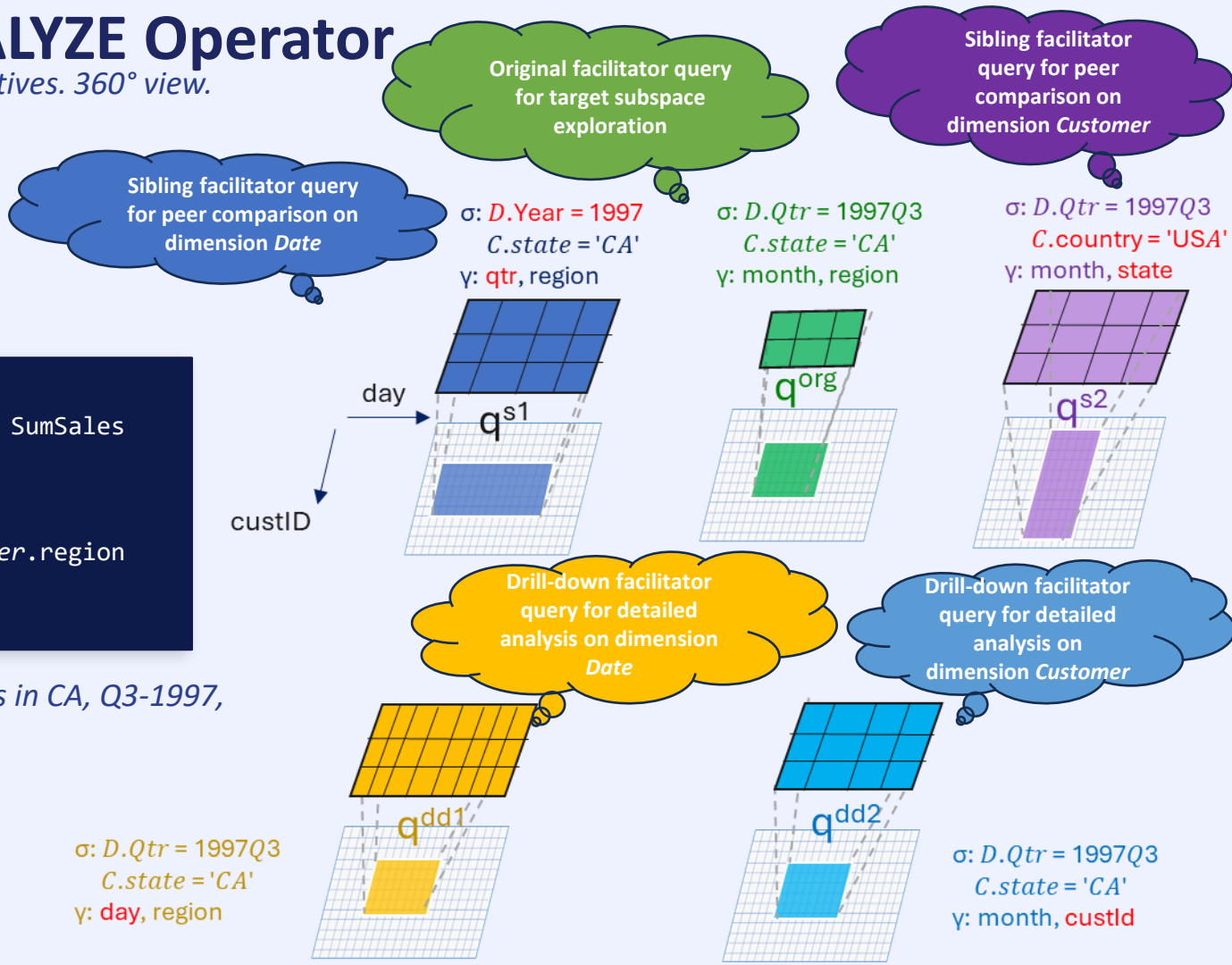
IV Conclusions

Meet the ANALYZE Operator

One invocation. Five perspectives. 360° view.

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧
Customer.state=CA
GROUP BY Date.month, Customer.region
AS AnalyzeQueryExample
```

Concrete example: store sales in CA, Q3-1997, grouped by month × region



II

Multi-Query Optimization

I The ANALYZE Operator

III Evaluation

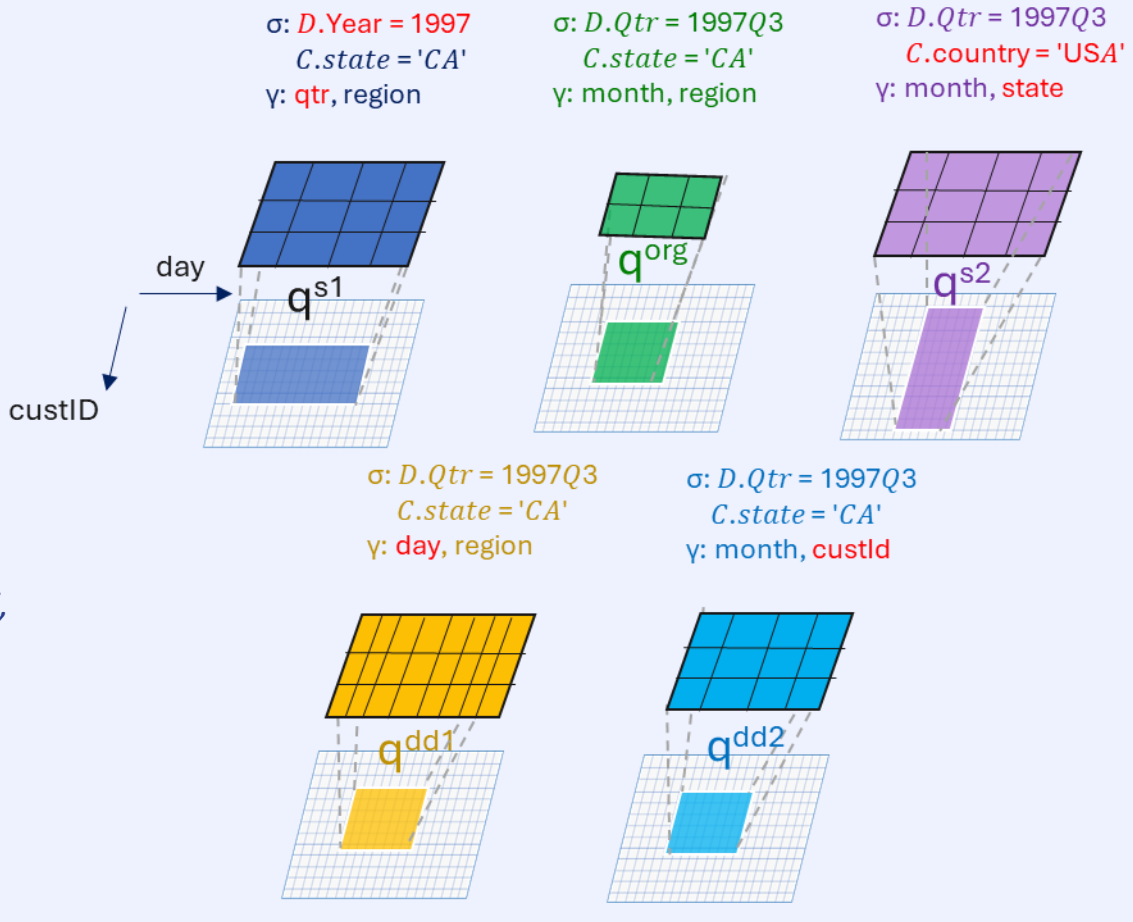
IV Conclusions

Min-MQO

One actual cube query per facilitator

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧
Customer.state=CA
GROUP BY Date.month, Customer.region
AS AnalyzeQueryExample
```

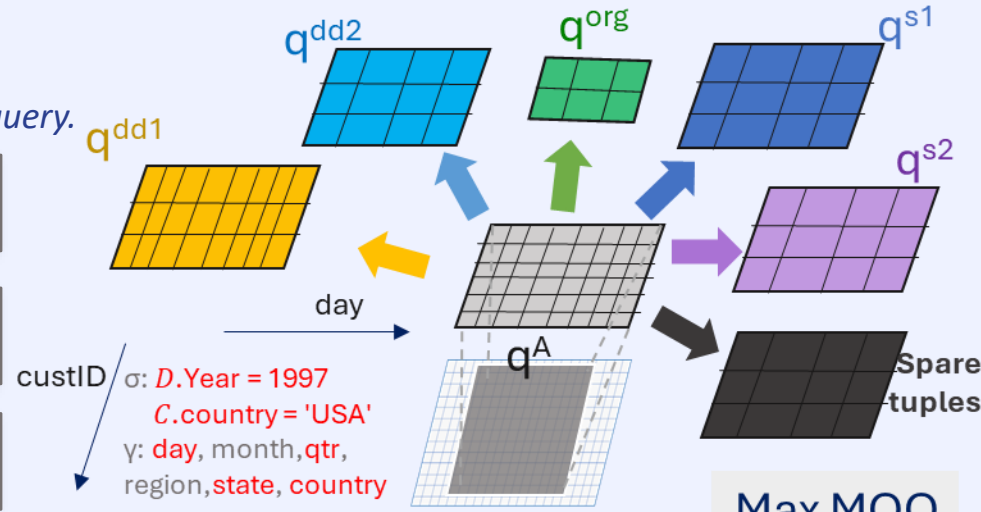
Concrete example: store sales in CA, Q3-1997, grouped by month × region

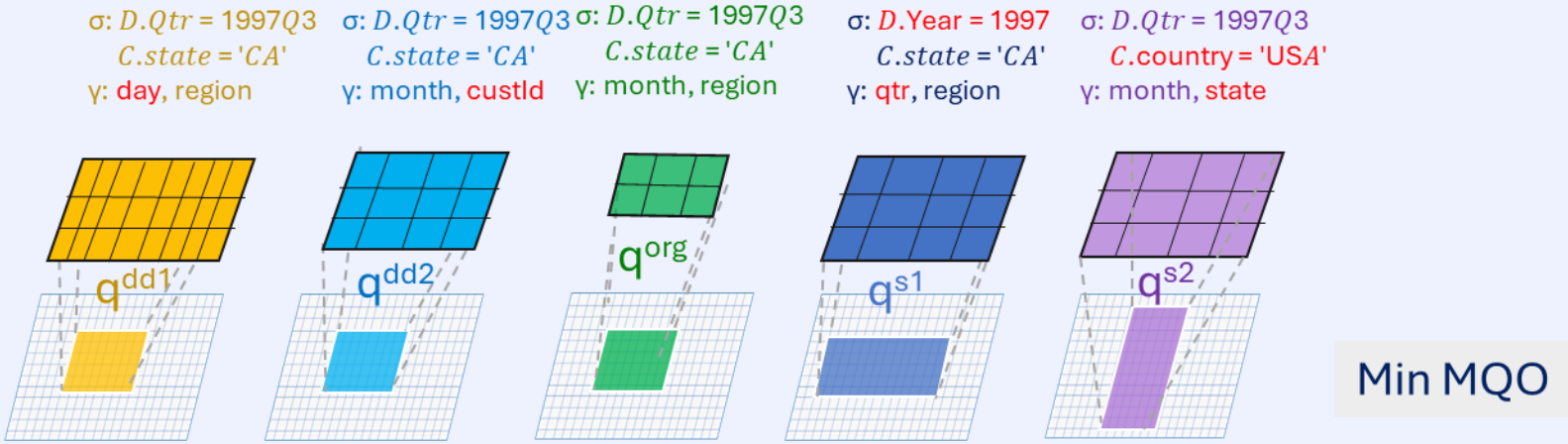


Max-MQO

Multi-Query Usability Theorem –
*All five queries can be answered from **one** single cube query.*

1. Build result maps for all 5 facilitator cube queries
2. Construct & execute q^A
3. Distribute tuples to the respective result maps

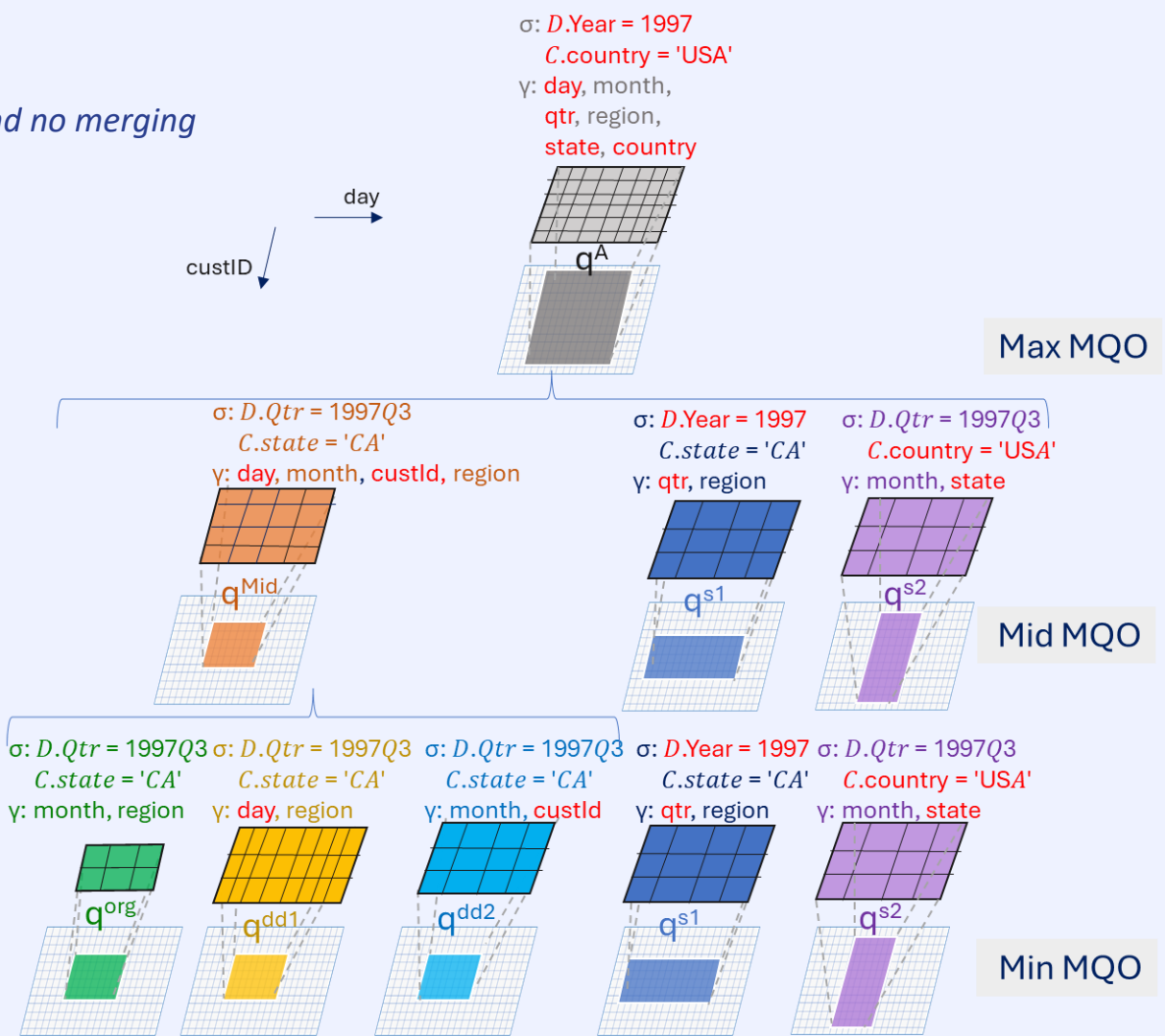




Mid-MQO

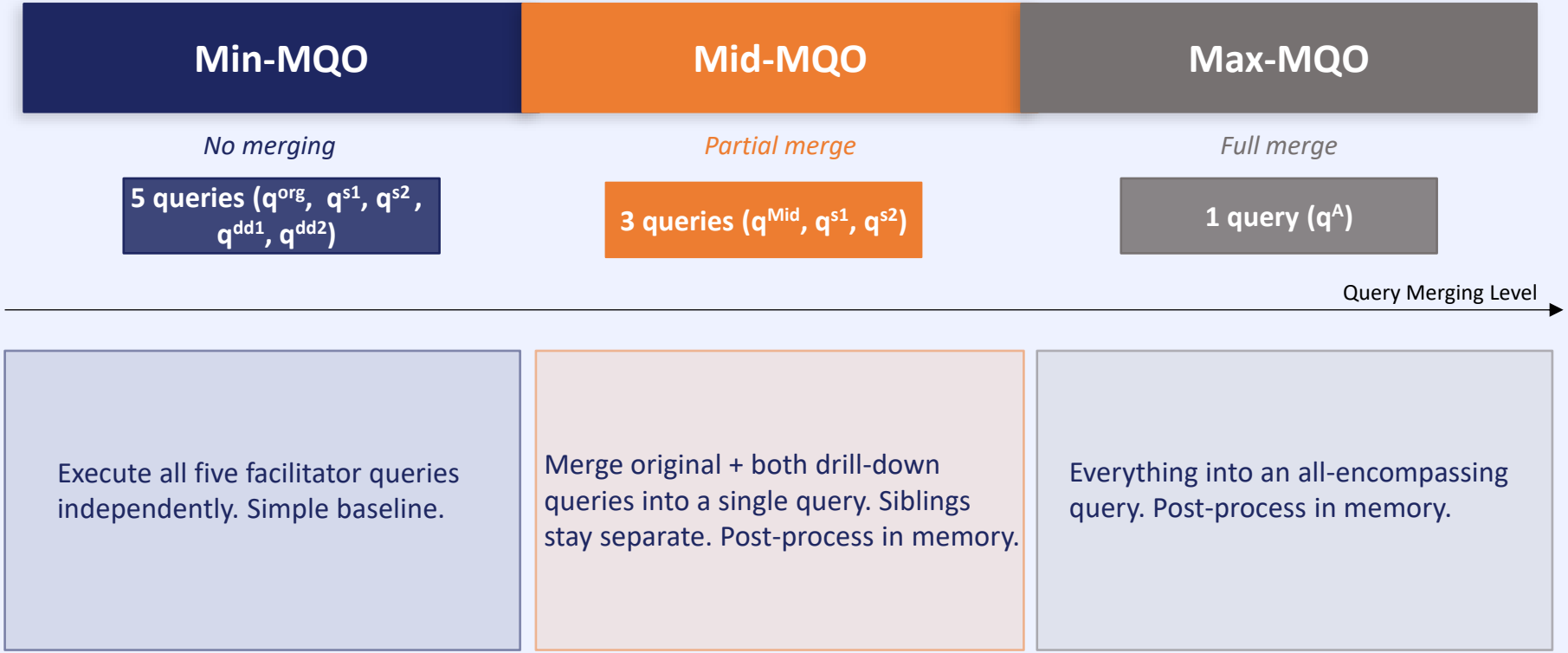
Strikes balance between full merging and no merging

1. Build result maps for original and drill-down cube queries
2. Construct & execute q^{Mid}
3. Distribute tuples to the respective result maps
4. Execute q^{s1}
5. Execute q^{s2}



Min-MQO, Mid-MQO, Max-MQO — Three Ways to Merge

Fewer database round-trips ≠ always faster - trade-offs ahead



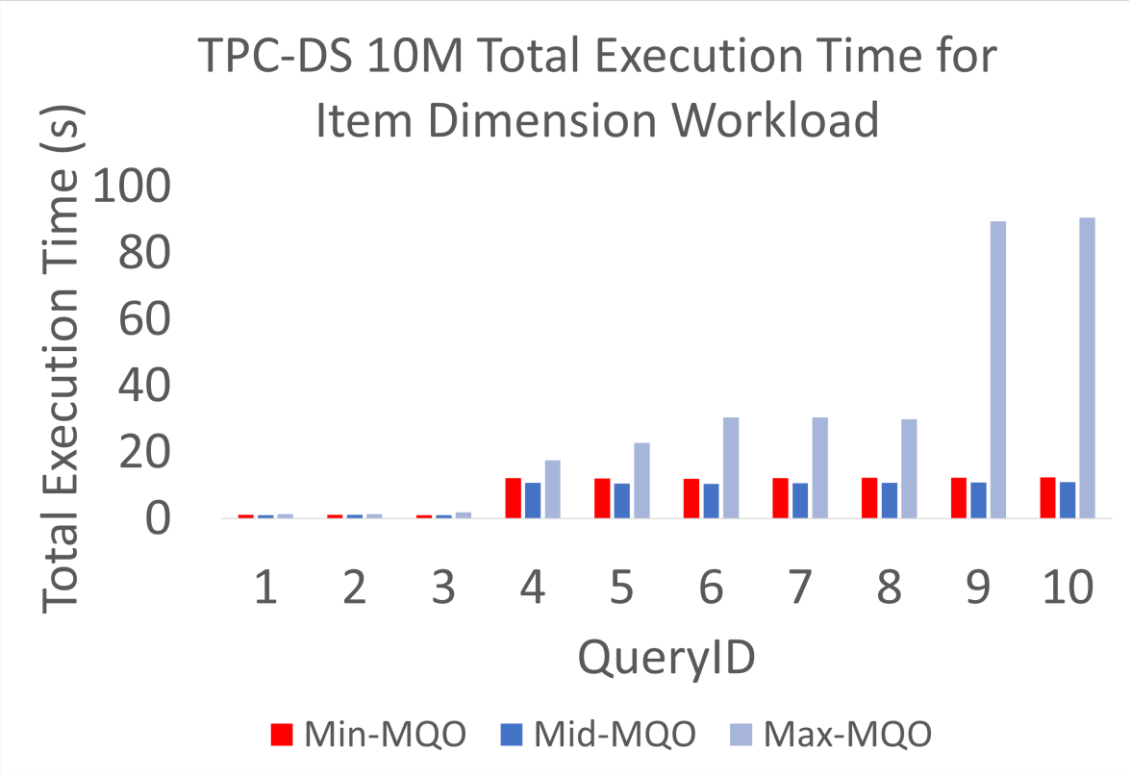
III

Evaluation

Delian Cubes engine · MySQL 8.0 · Windows 11, 32GB, 14-core

Min-MQO is never the best algorithm

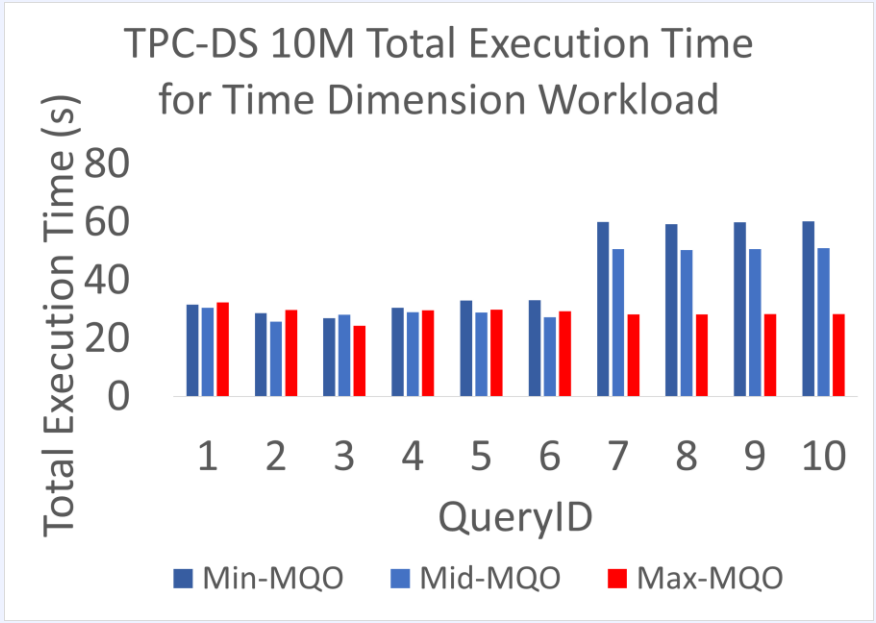
Metric: Total Execution Time per ANALYZE query in seconds.
Selectivity: increases as QueryID increases (1→10).

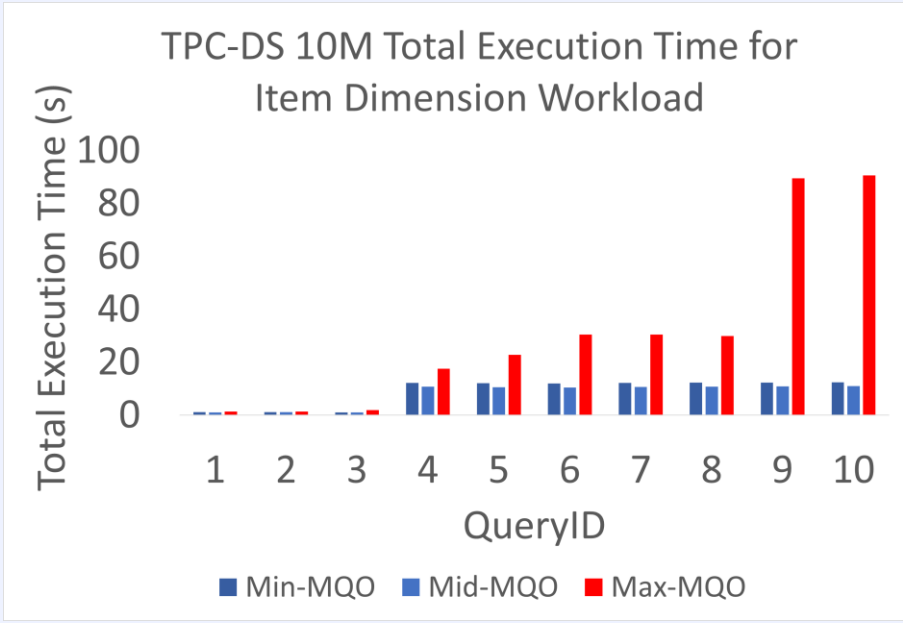


Min-MQO executes all 5 facilitator queries and is never the best algorithm. However, Min-MQO follows the performance of Mid-MQO closely.

Max-MQO can win big or lose big

Metric: Total Execution Time per ANALYZE query in seconds.
Selectivity: increases as QueryID increases (1→10).

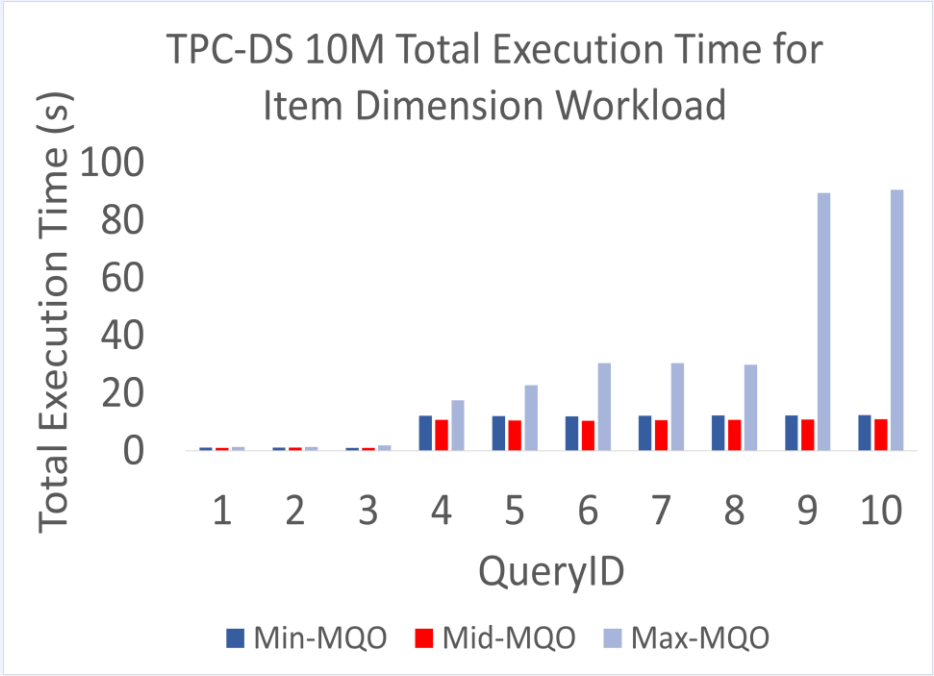


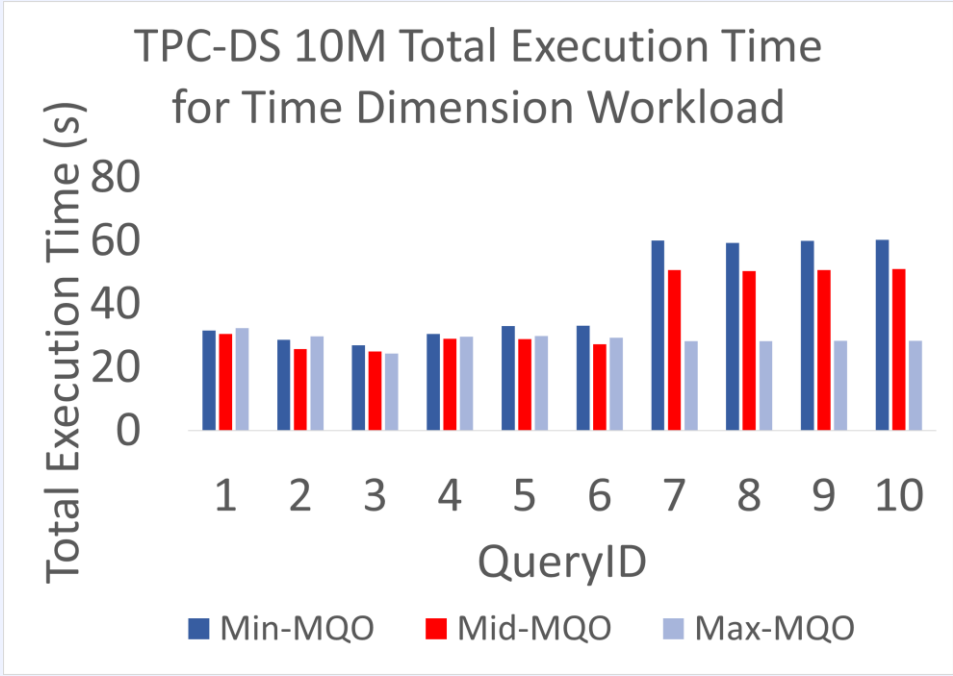


Max-MQO shows different behavior when we change filtering and grouping dimensions withing the same dataset.

Mid-MQO is never the worst strategy

Metric: Total Execution Time per ANALYZE query in seconds.
Selectivity: increases as QueryID increases (1→10).





Mid-MQO is a reliable solution since it is never the worst performing algorithm.

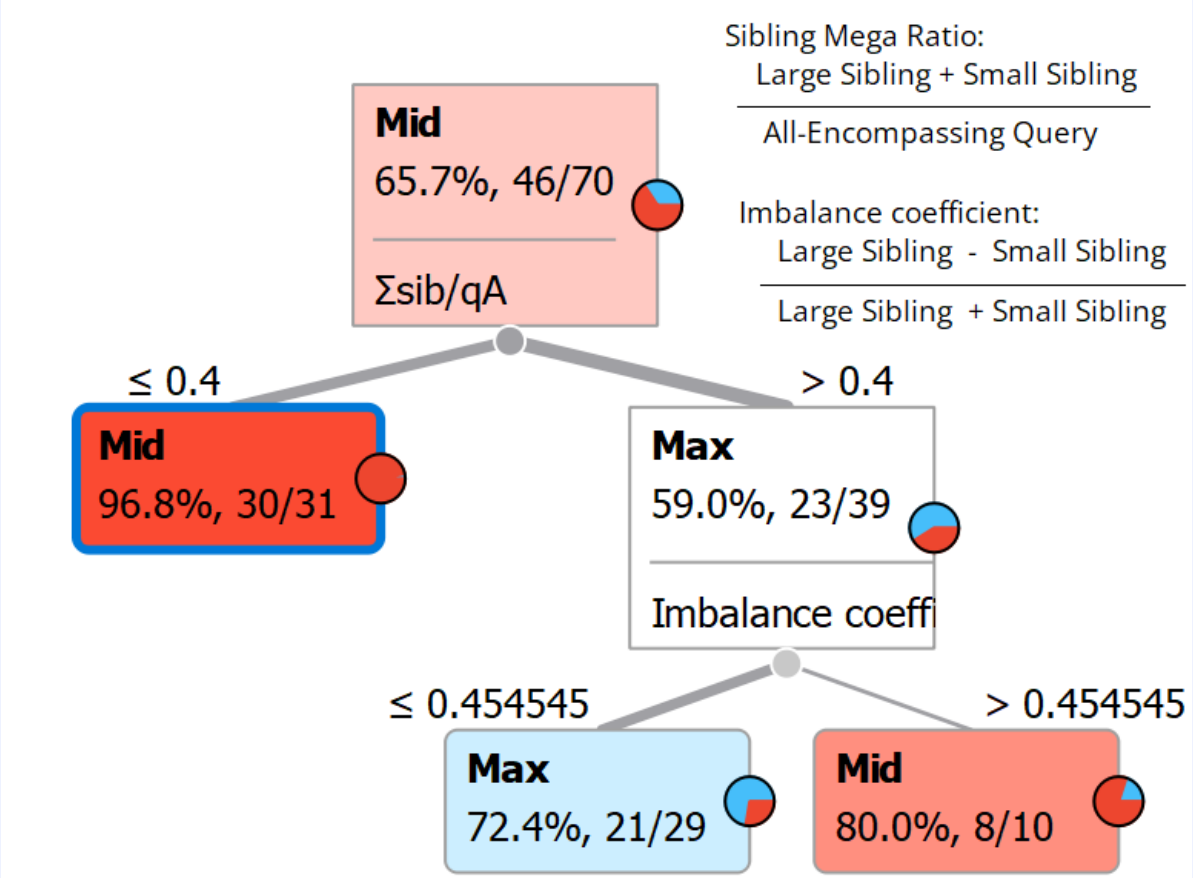
Decision Metrics

Two metrics that help us determine whether it is better to merge siblings (Max-MQO) or keep them separated (Mid-MQO)

Sibling Mega Ratio (Σsib/qA)	Imbalance Coefficient (Imbalance coeff.)
Definition: $(\|q^{s1} + q^{s2}\|) \,/\, \|q^{A}\|$ *Combined sibling detailed result size divided by the all-encompassing query detailed result size*	**Definition:** $(\|q^{s1} - q^{s2}\|) \,/\, (\|q^{s1} + q^{s2}\|)$ *Absolute difference between sibling detailed result sizes divided by their sum*
Interpretation: Compare the result size of both siblings with the result size of the all-encompassing query to determine whether the siblings cover a significant part of the all-encompassing query	**Interpretation:** Compute the overlapping degree of both siblings

A Decision Rule for Practitioners

The combination of two cheap-to-compute metrics predicts the winner reliably.



Rationale: Max-MQO saves time only when sizable siblings overlap (low imbalance). In all other cases, Mid-MQO is the safe, robust choice.

IV

Conclusions

I The ANALYZE Operator

II Multi-Query Optimization

III Evaluation

ANALYZE Proves That Operators Can Self-Optimize

Formal semantics + MQO at operator level — without touching the DBMS optimizer

Contributions

✓ Novel ANALYZE intentional operator — formal, 360° semantics.

✓ Three MQO optimization strategies.

✓ Mid-MQO is robust.

✓ Areas of competence guide the choice for the optimal algorithm provided.

Open Problems

→ Integrate with cost-based DBMS optimizers.

→ Extend to other intentional operators.

→ Relax constraints: more groupers, richer siblings.

Thank you

Questions & discussion welcome

Marios Iakovidis · miakovidis@cs.uoi.gr

Panos Vassiliadis · pvassil@cs.uoi.gr

University of Ioannina, Greece

Resources

- **Website** https://www.cse.uoi.gr/~pvassil/projects/datastory/publications.html
- **Material** github.com/DAINTINESS-Group/DelianCubeEngine
- **Long version** arXiv:2602.08546

Contributions

✓ Novel ANALYZE intentional operator — formal, 360° semantics.

✓ Three MQO optimization strategies.

✓ Mid-MQO is robust.

✓ Areas of competence guide the choice for the optimal algorithm provided.

Acknowledgments
The research project is implemented in the framework of H.F.R.I. call "3rd Call for H.F.R.I. Research Projects to Support Faculty Members & Researchers" (H.F.R.I. Project Number: 23640)

V

Auxiliary Material

Experimental Setup - Datasets

Delian Cubes engine · MySQL 8.0 · Windows 11, 32GB, 14-core · Code on GitHub

Dataset	# Dimension Tables	# Rows	Domain
Northwind	4	~2K	Food import/export
Foodmart	5	~288K	Retail supermarket
pkdd99+	3	~100M	Financial / Loans
TPC-DS ×3	6	~2M / 10M / 100M	Analytics benchmark

Experimental Setup - Workloads

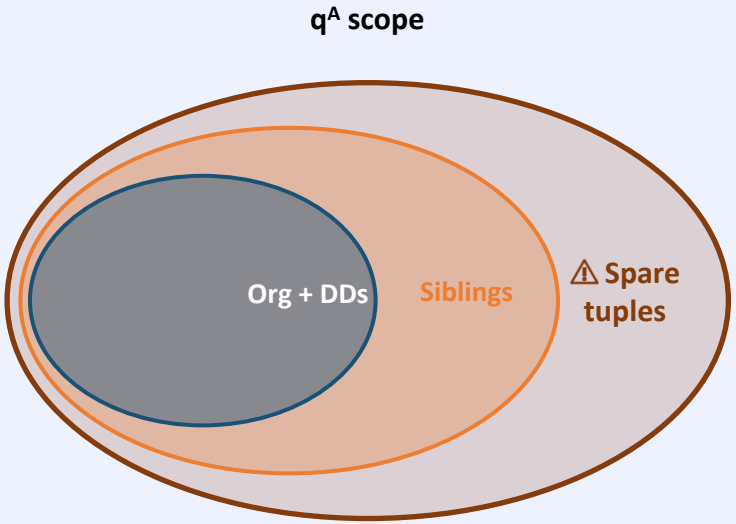
Delian Cubes engine · MySQL 8.0 · Windows 11, 32GB, 14-core · Code on GitHub

Workload	# Queries	# Dimension 1 Size	# Dimension 2 Size	Key Feature
Northwind	10	~90	~9	Two small dimension tables.
Foodmart	10	~10K	~1K	Unbalanced dimensions.
pkdd99+	10	~30K	~5K	One large + one mid dimension.
TPC-DS Item	10	~80K	~20K	Large dimension + Mid dimension.
TPC-DS Time	10	~80K	~70K	Two large dimensions.

Metric: Total Execution Time per ANALYZE query · **Selectivity** increases as QueryID increases (1→10)

Max MQO Trade-off

The spare tuple problem: q^A retrieves more than it needs



① q^A uses sibling selection conditions (broader than original).

② Retrieves tuples irrelevant to any facilitator query.

③ Spare tuples increase as query selectivity increases.

MQO Trade-offs

Min-MQO: baseline execution, Max-MQO: full merging of the queries: Mid-MQO: balanced approach

MQO Method	# Executable Queries	Post-Processing in Memory	Spare Tuples	Key Feature
Min-MQO	5	No	No	Executes all five queries without post-processing.
Mid-MQO	3	Yes	No	Reduces the number of queries executed but maintains siblings to avoid retrieving a large number spare tuples. Post-processing required to distribute results.
Max-MQO	1	Yes	Yes	Executes a single all-encompassing query, retrieves a significant number of spare tuples. Post-processing required to distribute results.

Mid-MQO is the preferable method because it strikes a balance between reducing the number of executable queries and not producing a lot of spare tuples.

Motivation - The Analyst's Problem

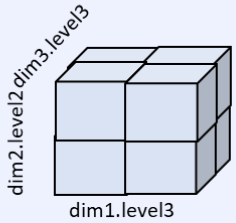
Analysts hunt for highlights manually.

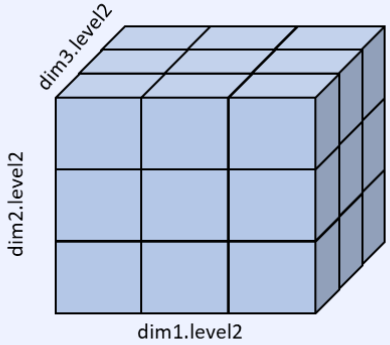
What analysts do today:	What they actually want:
Issue query for target subspace.	Express intent once.
Write sibling queries manually.	Get all perspectives automatically.
Drill-down by hand.	Formal, reproducible semantics.
Stitch results in a spreadsheet.	Efficient execution under the hood.
Repeat for next subspace…	Focus on highlights, not plumbing.

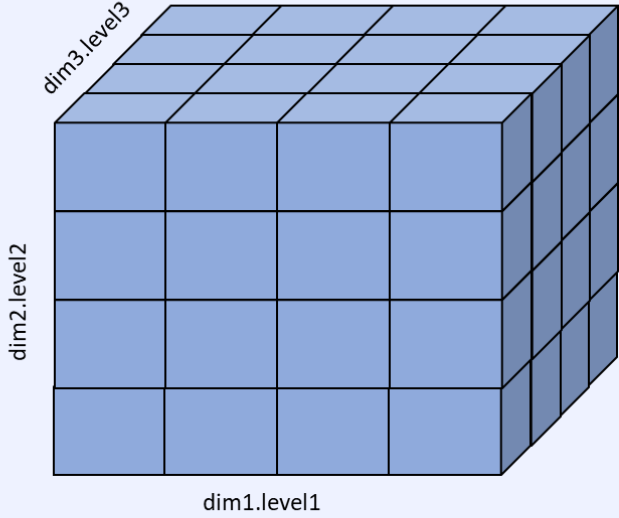
Motivation: *formalize the intentions of data analysts for 360° view of the data in a user-friendly way.*

Preliminaries – The multidimensional model

Data Cube

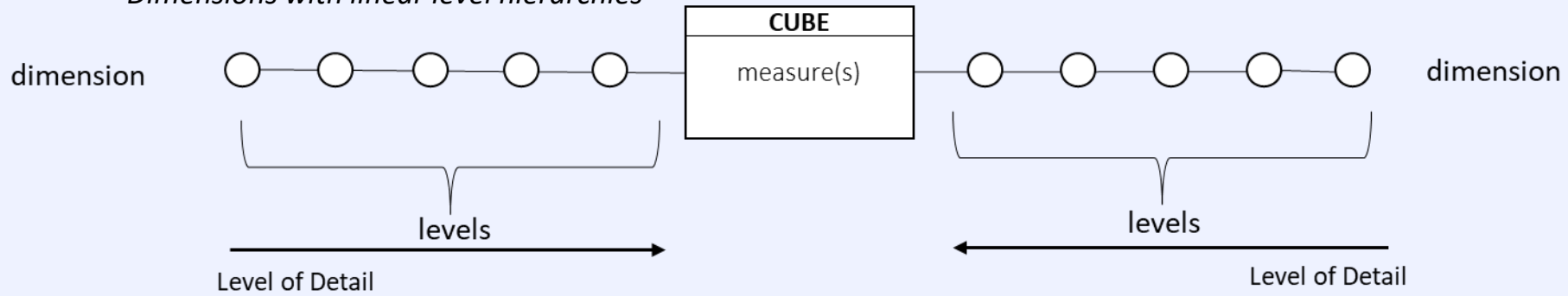






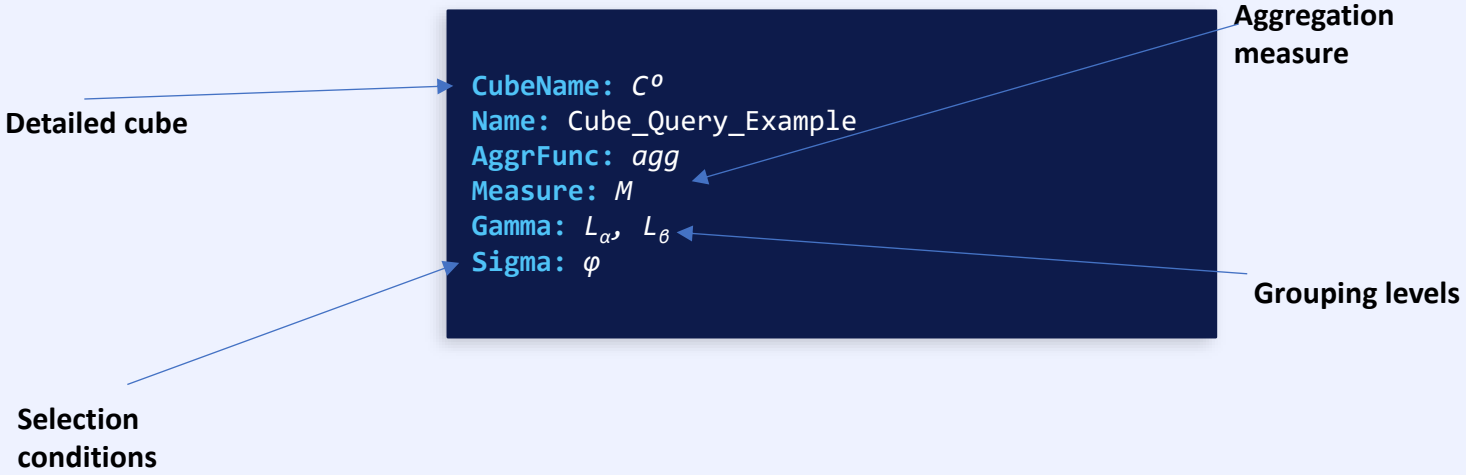
Level of Detail

Dimensions with linear level hierarchies



Preliminaries – Cube Queries

Cube Query Syntax



$$q = \langle C^o, \varphi, [L_\alpha, L_\beta, M], agg(M^o) \rangle$$

A cube query is considered a cube

Preliminaries - Cube Query Constraints

Three simplifying assumptions that define the working model [1].

C1 Single Measure

$q = \langle C^o, \varphi, [L_\alpha, L_\beta, M], agg(M^o) \rangle$

Each ANALYZE query operates on exactly one measure. Multi-measure queries are outside the current scope.

C2 Exactly Two Groupers

$GROUP\ BY\ D_\alpha.L_\alpha, D_\beta.L_\beta$

The result is always grouped by precisely two-dimension levels - one per grouper dimension α and β.

C3 Filter Level ≥ Grouper Level

$If\ dim\ has\ filter\ \varphi_i\ and\ grouper\ L_i,$
$then\ dim.level(\varphi_i) \geq dim.level(L_i)$

Any atomic filter on a dimension that is also a grouper must be expressed at a level no finer than the grouper level.

We assume that the selection conditions are conjunctions of atomic filters of the form L = value.

Why these constraints? *They maximise clarity without sacrificing the key semantics – ensure perfect rollability.*

[1] M.Iakovidis, P.Vassiliadis, Semantics and multi-query optimization algorithms for the analyze operator, CoRR abs/2602.08546 (2026).

02

The ANALYZE Operator

Syntax, semantics & constraints

Reference Example - Hierarchy Structure

Example dimensions to be used for the remaining of the discussion

Date Dimension	
ALL	*single value 'All'*
Year	*e.g. 1997*
Quarter	*e.g. Q3-1997*
Month	*e.g. July*
Day	*e.g. Monday*

Customer Dimension	
ALL	*single value 'All'*
Country	*e.g. USA*
State	*e.g. CA*
Region	*e.g. San Francisco*
CustID	*e.g. 99*

Level of Detail

Meet the ANALYZE Operator - Syntax

Assuming the setup discussed in Preliminaries section

SQL-like Definition

```
ANALYZE agg(M^θ) as M
FROM C^θ
FOR φ
GROUP BY L_α, L_β
AS queryName
```

Algebraic Representation

$$analyzeOP = \langle C^{\theta}, \varphi, [D_{\alpha}.L_{\alpha}, D_{\beta}.L_{\beta}, M], agg(M^{\theta}) \rangle$$

Meet the ANALYZE Operator - Semantics

One invocation. Five perspectives. 360° view.

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS AnalyzeQueryExample
```

Concrete example: store sales in CA, Q3-1997, grouped by month × region

Original query - q^{org} *Target subspace*

Sibling 1 (*a* dimension) - q^{sA} *Peer comparison - dim Date*

Sibling 2 (*b* dimension) - q^{sB} *Peer comparison – dim Customer*

Drill-down 1 (*a* dimension) - q^{ddA} *Finer detail - dim Date*

Drill-down 2 (*b* dimension) - q^{ddB} *Finer detail - dim Customer*

Meet the ANALYZE Operator - Semantics

Sibling Queries

Loosen the CA filter → show all US states side by side. CA is now one bar among many peers.

→ Puts your result in context.

Drill-Down Queries

Keep the same filters but go one step finer → months become days (dim *Date*) or regions become customerIDs (dim *Customer*).

→ More detail on demand.

Both are things analysts already do manually — ANALYZE formalizes and automates them.

$q^{org} \rightarrow q^{s^A}, q^{s^B}$

$q^{org} \rightarrow q^{dd^A}, q^{dd^B}$

Meet the ANALYZE Operator – Facilitator Queries

Original query - q^{org}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS OriginalQuery
```

→ SQL Query

Sibling 1 (*a* dimension) - q^{S^A}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.year=1997 ∧ Customer.state=CA
GROUP BY Date.quarter, Customer.region
AS Sibling1
```

→ SQL Query

Sibling 2 (*b* dimension) - q^{S^B}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧
Customer.country=USA
GROUP BY Date.month, Customer.state
AS Sibling2
```

→ SQL Query

Drill-down 1 (*a* dimension) - q^{dd^A}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.day, Customer.region
AS DrillDown1
```

→ SQL Query

Drill-down 2 (*b* dimension) - q^{dd^B}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.customerID
AS DrillDown2
```

→ SQL Query

Meet the ANALYZE Operator – Sibling Queries

To reduce the number of generated queries:

- We consider siblings only for atoms that their dimensions are grouping dimensions.
- Merge all sibling slices into one, by adapting the aggregation level

Example: Sibling 1 of store sales in CA, Q3-1997, grouped by month × region

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.year=1997 ∧ Customer.state=CA
GROUP BY Date.quarter, Customer.region
AS Sibling1
```

✓ **Merges all siblings into one query**

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=Q1-1997 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS Sibling11
```

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=Q2-1997 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS Sibling12
```

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=Q4-1997 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS Sibling13
```

✗ To compute siblings exhaustively we need 3 queries

Meet the ANALYZE Operator – Example

Sibling 1 (*a* dimension) - q^{sA}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.year=1997 ∧ Customer.state=CA
GROUP BY quarter, region
AS Sibling1
```

SQL Query

```
SELECT Date.quarter, Customer.region, sum(Store_Sales) as SumSales
FROM Sales,Date,Customer
WHERE Sales.date_id = Date.date_id AND Sales.customer_id =
Customer.customer_id AND Date.year = 1997 AND Customer.state = CA
GROUP BY Date.quarter, Customer.region
```

Max-MQO for the ANALYZE Operator

All five queries can be answered from one single query. Multi-Query Usability Theorem - proven, not assumed[1].

q^A — All-Encompassing Query

Broadest selection · Highest useful groupers

q^{org}	q^{s^a}	q^{s^b}	q^{dd^a}	q^{ddb}
Original Facilitator Query	Sibling 1 Facilitator Query	Sibling 2 Facilitator Query	Drill-down 1 Facilitator Query	Drill-down 2 Facilitator Query

Theorem 5.1: q^A subsumes all five facilitator queries — correctness is guaranteed by re-filtering and re-aggregating from its result set.

[1] M.Iakovidis, P.Vassiliadis, Semantics and multi-query optimization algorithms for the analyze operator, CoRR abs/2602.08546 (2026).

Max MQO - All encompassing query

Original query - q^{org}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS AnalyzeQueryExample
```

All-encompassing query

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.year=1997 ∧ Customer.country=USA
GROUP BY Date.month, Date.quarter, Date.year, Customer.customerID,
Customer.region, Customer.state
AS AllEncompassingQuery
```

Result distribution is necessary

Mid MQO - Merging of q^{org} And q^{dds}

Original query - q^{org}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS OriginalQuery
```

Drill-down 1 (*a* dimension) - q^{ddA}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.day, Customer.region
AS DrillDown1
```

Drill-down 2 (*b* dimension) - q^{ddB}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.customerID
AS DrillDown2
```

Selection conditions are exactly the same between q^{org}, q^{ddA}, q^{ddB}

Mid MQO - Merging of q^{org} And q^{dds}

Original query - q^{org}

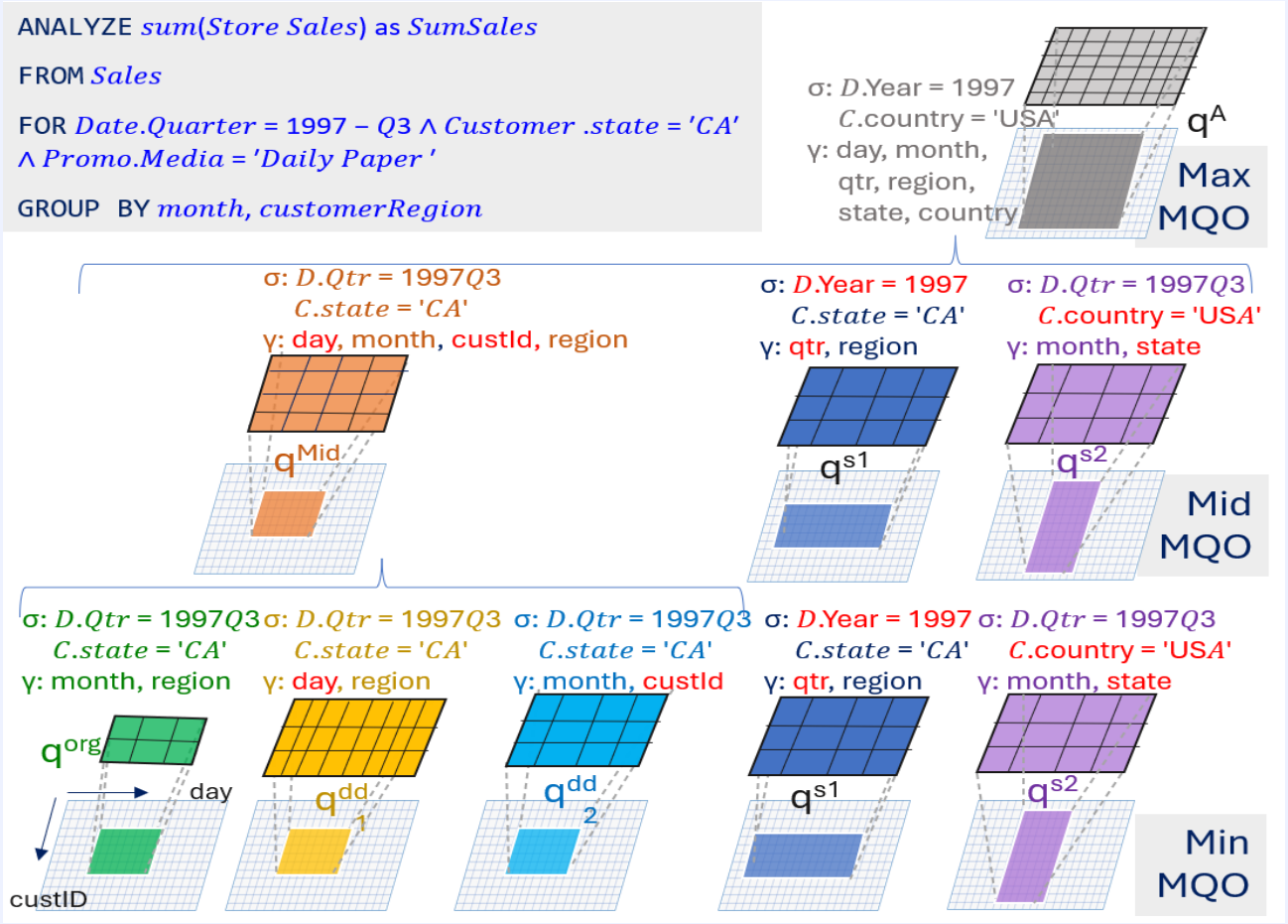
```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.month, Customer.region
AS AnalyzeQueryExample
```

Original&Drill-Downs Query q^{MID}

```
ANALYZE sum(Store_Sales) as SumSales
FROM Sales
FOR Date.quarter=1997Q3 ∧ Customer.state=CA
GROUP BY Date.day, Date.month, Customer.customerID,
Customer.region
AS Original&DrillDownsQuery
```

Sibling queries remain untouched. Result distribution is necessary

MQO Search Space

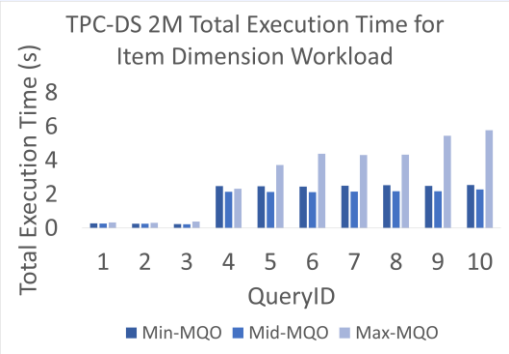


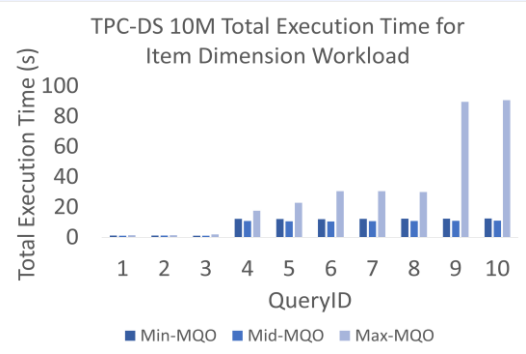
04

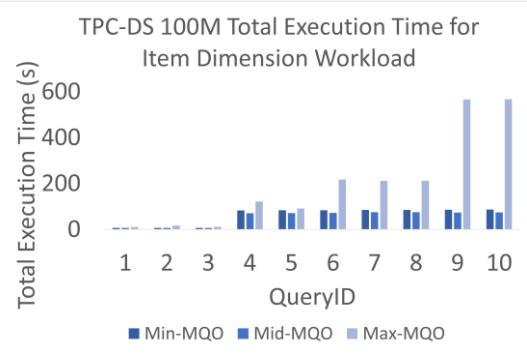
Experiments

Performance across datasets and workloads

Partial Merging Beats Both Extremes - Usually







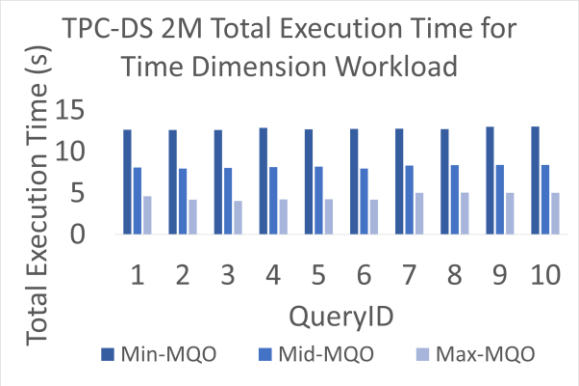
✓ Mid-MQO fastest at all scales

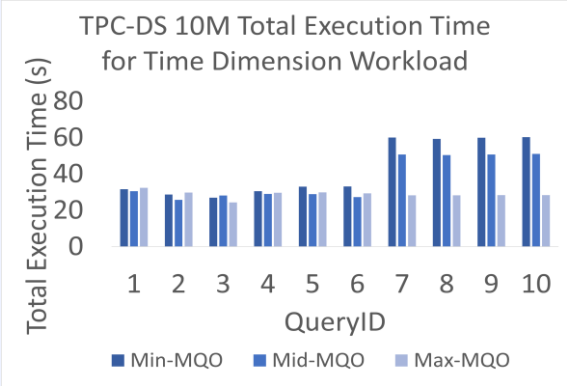
✗ Max-MQO worst — gap widens with selectivity

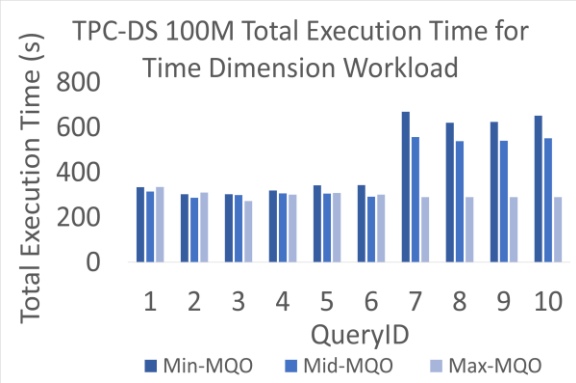
≈ Min and Mid close — DD savings are modest here

Spare tuple overhead compounds with higher selectivity → Max-MQO pays the price.

Sometimes Max-MQO prevails







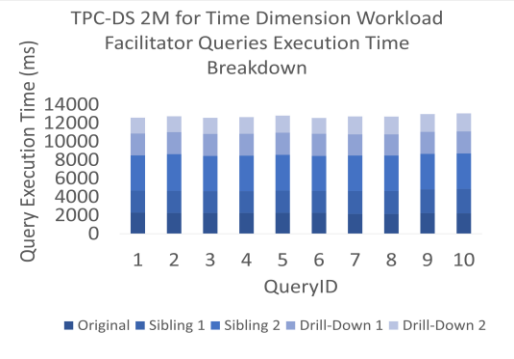
✓ Max-MQO fastest most of the time

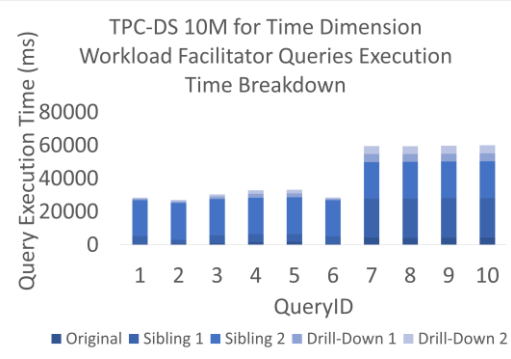
✗ Mid-MQO worst — gap widens with selectivity

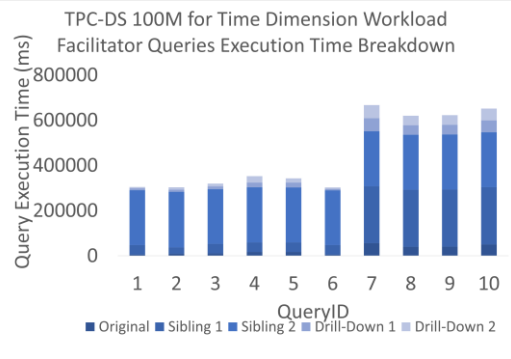
Max has stable performance regardless of selectivity

Large, Overlapping Siblings Flip the Winner

TPC-DS Time workload · Date dim ≈80K tuples · Time dim ≈70K tuples







Why Max wins at 2M

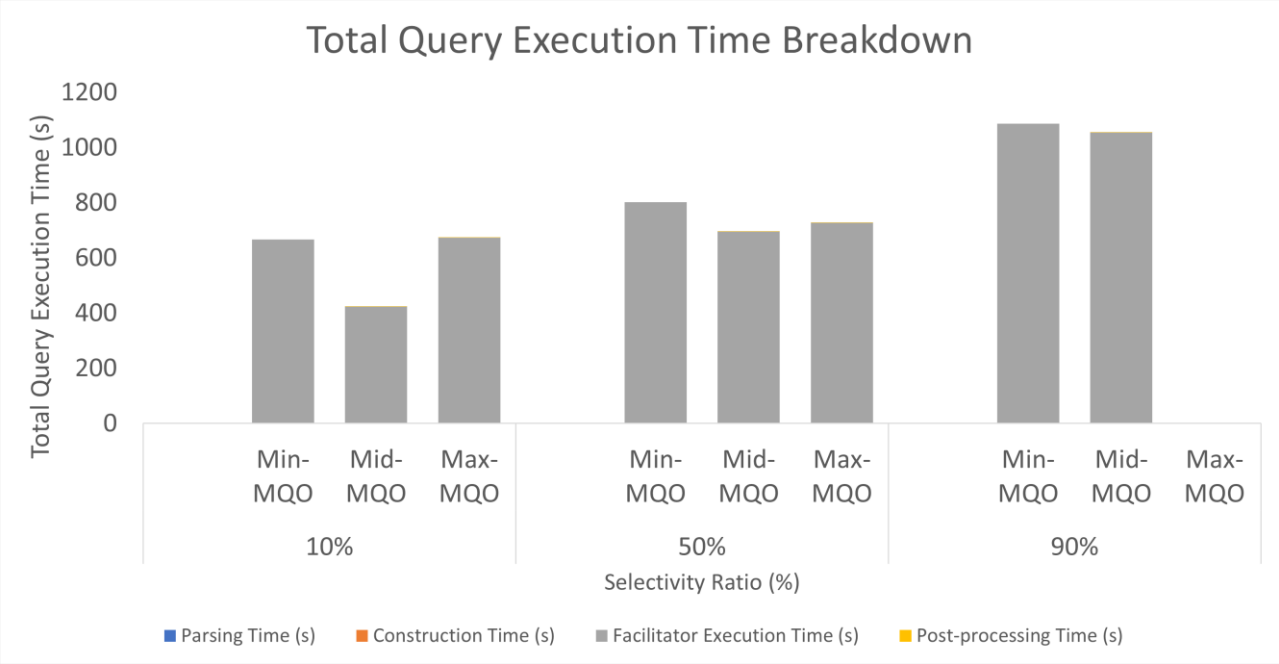
Both dim tables are large → siblings are sizable and overlap → q^A visits the overlap once.

Why Mid recovers at 100M

As data grows, q^A's broader scan cost outweighs the sibling savings at high selectivity.

ANALYZE Query Total Execution Time Breakdown

TPC-DS Time workload 100M Workload



A Decision Rule for Practitioners

We attempt to find decision criteria for choosing the optimal ANALYZE execution method

- We analyze the experimental results of 7 workloads and 70 queries (TPC-DS Item for all 3 sizes, TPC-DS Time for all 3 sizes, and pkdd99+).
- We take into account the number of detailed tuples each facilitator query returns, and the best performing method for each ANALYZE query.
- For each ANALYZE query, we calculate the *imbalance coefficient* of the sibling queries to measure how balanced are the two queries with respect to the number of detailed tuples each query fetches.
- For each ANALYZE query, we calculate the *sibling mega ratio* to compare the number of tuples both siblings fetch with the number of detailed tuples of the all-encompassing query q^A.
- A decision tree is applied to the workloads to assist in determining in which conditions each method wins.

05

Conclusions

Findings, lessons, and open directions

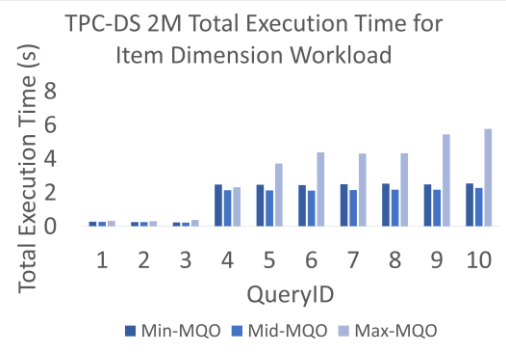
ANALYZE Proves That Operators Can Self-Optimize

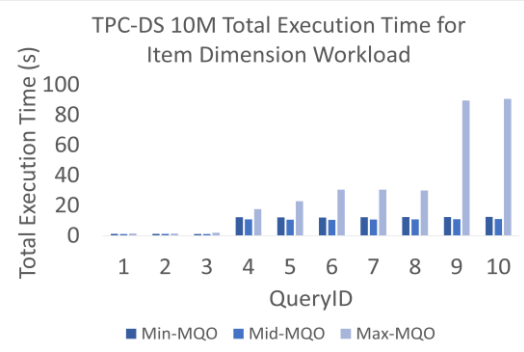
Formal semantics + MQO at operator level — without touching the DBMS optimizer

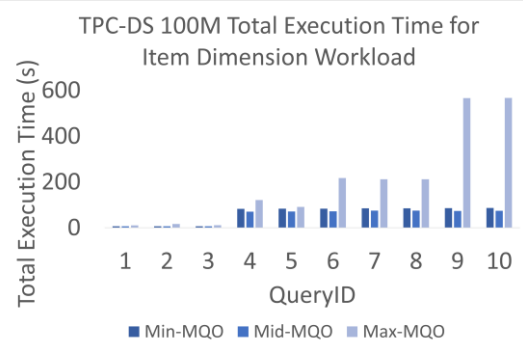
Contributions	Open Problems/Future Work
✓ Novel ANALYZE operator with rigorous formal semantics	→ Integrate with cost-based DBMS optimizers
✓ Multi-Query Usability Theorem — correctness proven	→ Deploy on Apache Spark / distributed engines
✓ Three MQO strategies: Min-MQO, Mid-MQO, Max-MQO	→ Extend to PREDICT, EXPLAIN operators
✓ Mid-MQO: best general-purpose strategy	→ Relax two-grouper constraint
✓ Decision tree to switch to Max-MQO when warranted	→ User-defined / context-aware sibling benchmarks

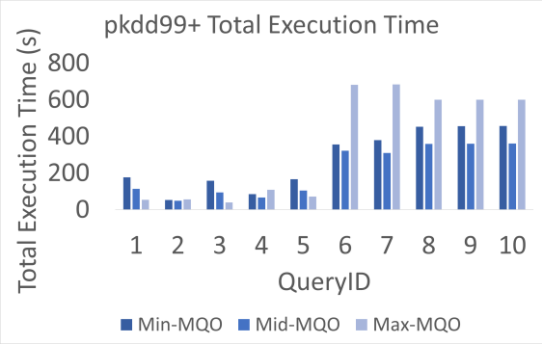
The external analytics engine can optimize query execution outside the DBMS — a lesson for future intentional operators.

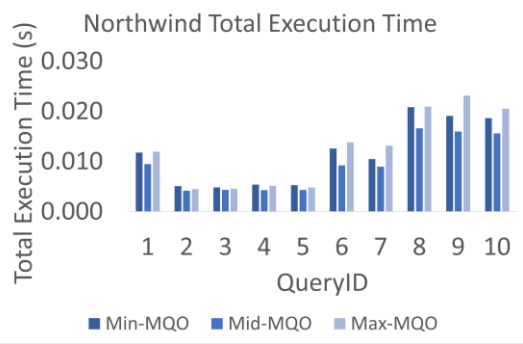
Partial Merging Beats Both Extremes - Usually











✓ Mid-MQO fastest at all scales

✗ Max-MQO worst — gap widens with selectivity

≈ Min and Mid close — DD savings are modest here

Spare tuple overhead compounds with higher selectivity → Max-MQO pays the price.

The Intentional Analytics Model

User expresses analytical intentions – not cube queries [1]

Traditional OLAP: roll-up, drill-down → analysts declare ***what*** data to retrieve and what actions to apply.

IAM: analysts express ***intentions*** about what they want to discover, not what data and actions they need.

ASSESS	DESCRIBE	EXPLAIN	PREDICT	SUGGEST	ANALYZE ★
How does a cube cell compares to a benchmark?	*What does this cell look like?*	*Why does this measure behave this way?*	*What will future values look like?*	*What should I explore next?*	*Give me a 360° view of this subspace*
Cube cells rated vs. expected values.	Annotated cube with clusters / outliers.	Statistical model of cause-effect.	Cube cells extended with forecasts.	Recommended next OLAP operations.	Original + siblings + drill-downs.
Francia et al. 2021	*Francia et al. 2022*	*Francia et al. 2024*	*Francia et al. 2026.*	*Open problem*	*This paper*

[1]: Vassiliadis, Marcel, Rizzi. "Beyond Roll-Up's and Drill-Down's: An Intentional Analytics Model." *Information Systems*, 85, 68–91, 2019.