DaST
Data·(Systems+Theory)
DEPARTMENT OF INFORMATICS, UNIVERSITY OF ZURICH

Real-time Analytics over Continuously Evolving Databases

Dan Olteanu

March 8, 2023

IfI Research Talk Series

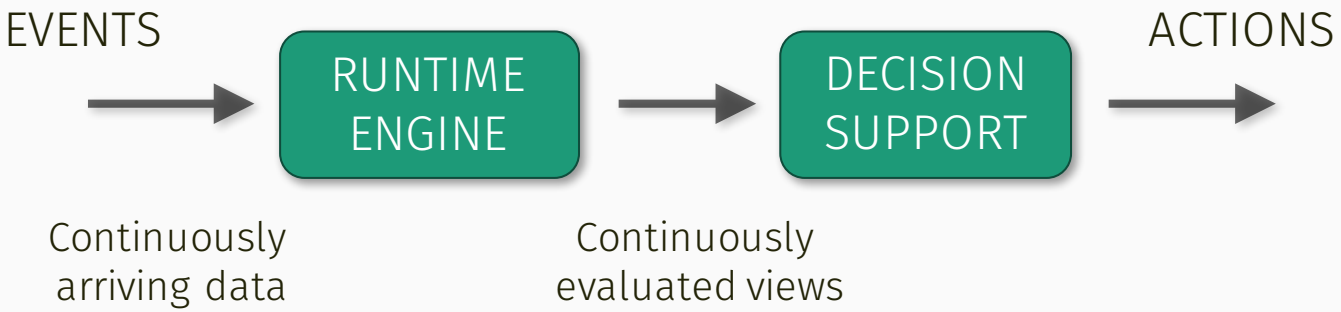
Real-Time Analytics

- Datasets continuously evolve over time
  - E.g.: data streams from sensors, social networks, apps
- Real-time analytics over streaming data
  - Users want fresh up-to-date computation results, e.g., models

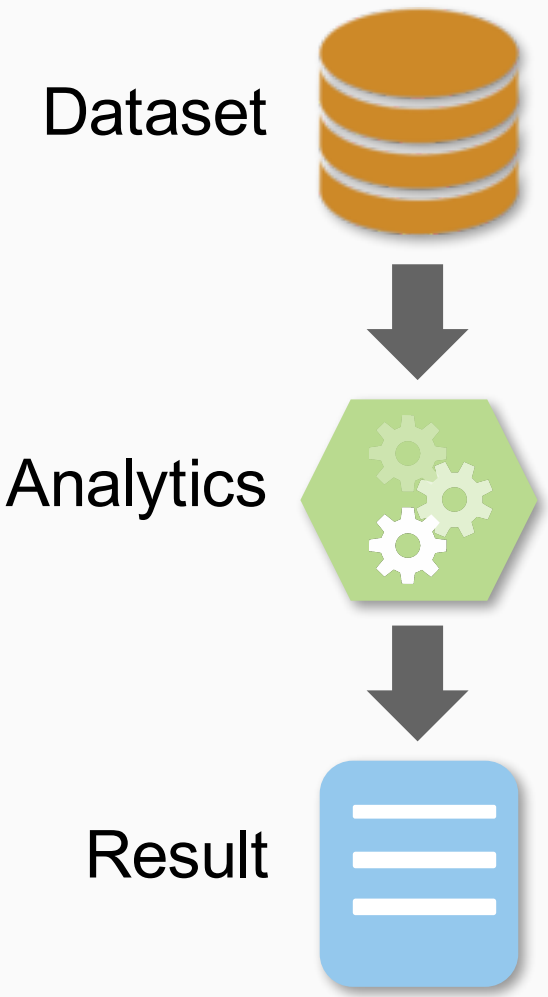
Web Analytics

Sensor Networks

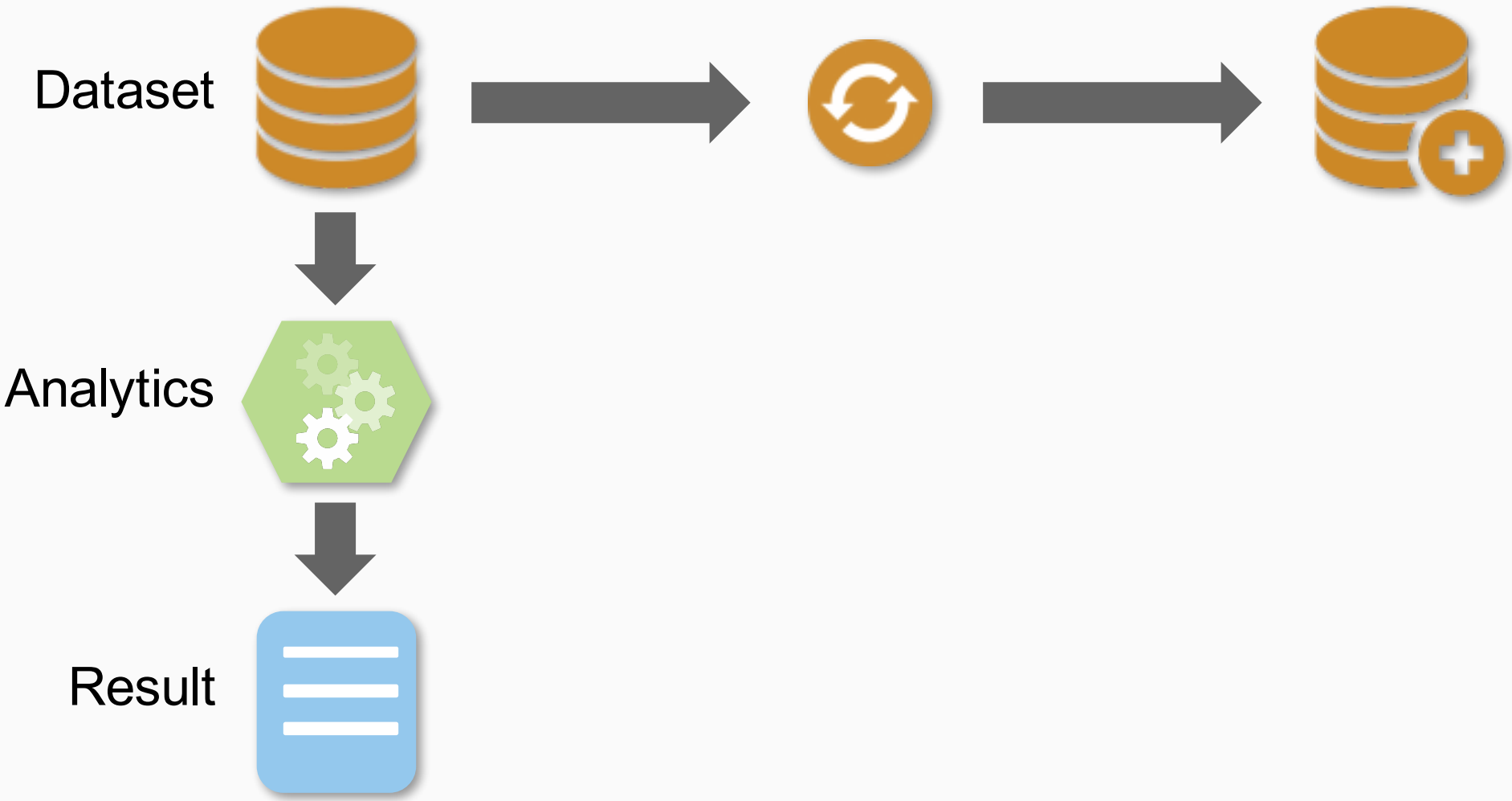
Cloud Monitoring



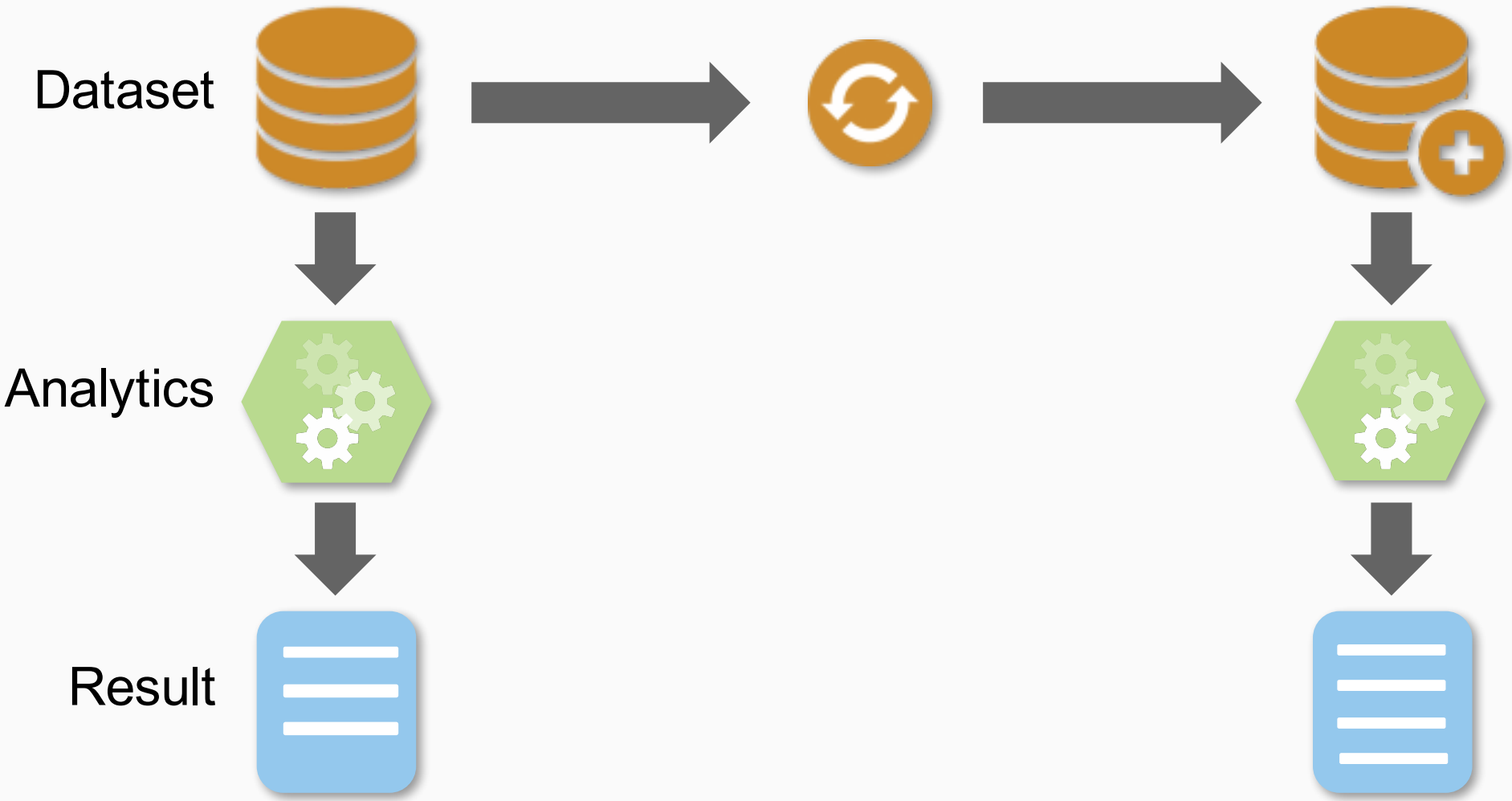
Real-Time Analytics via Incremental Maintenance



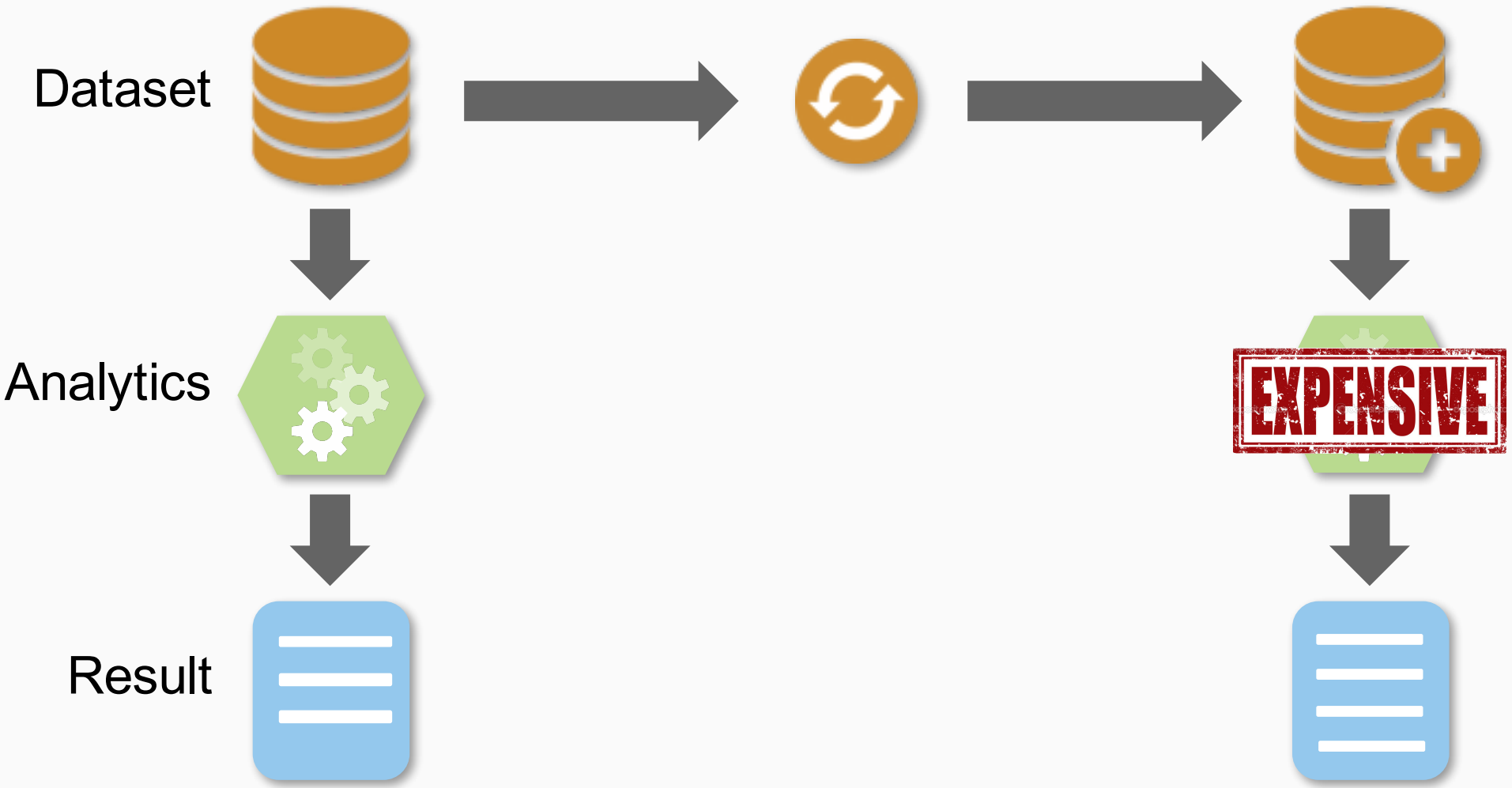
Real-Time Analytics via Incremental Maintenance



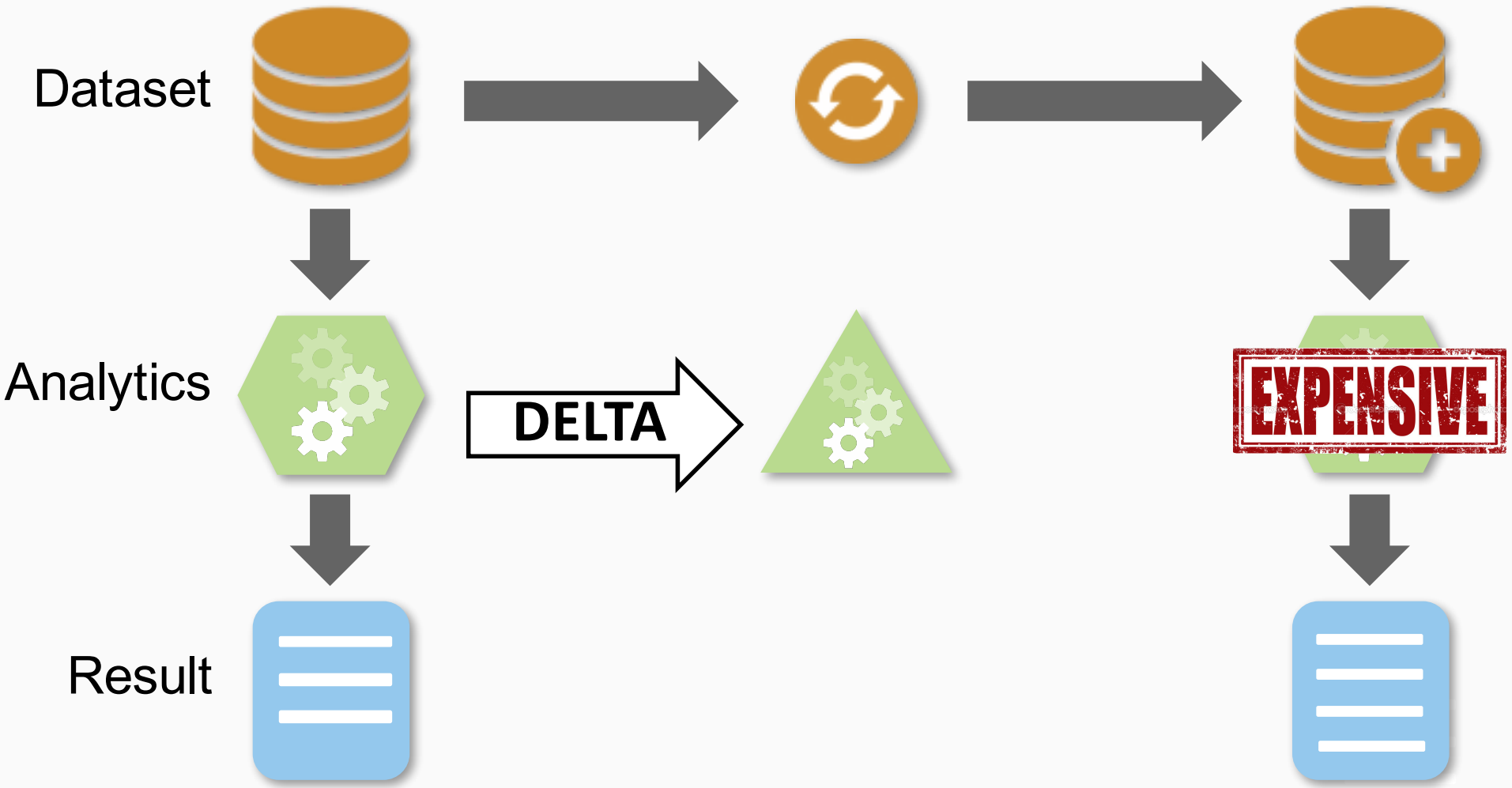
Real-Time Analytics via Incremental Maintenance



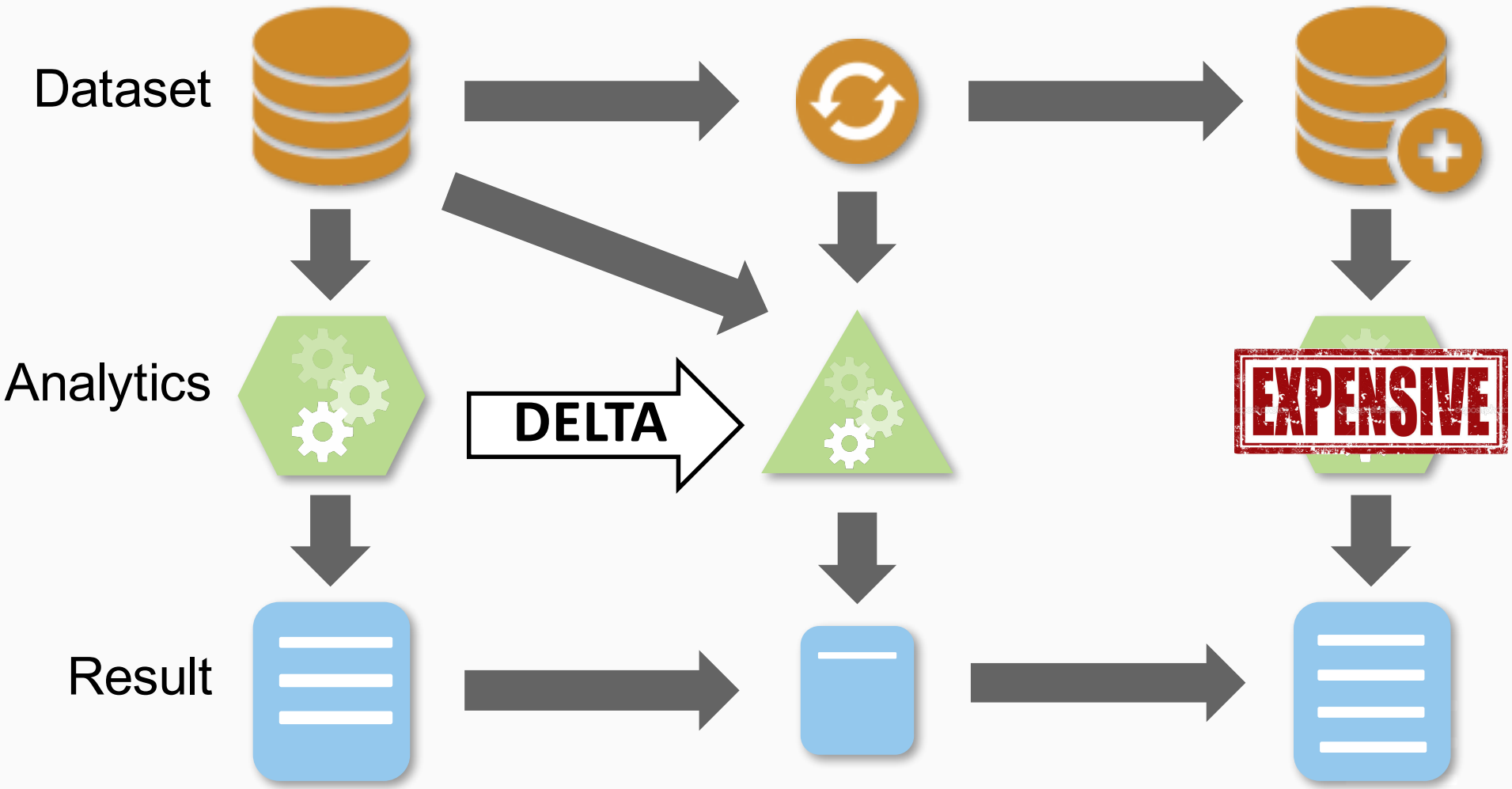
Real-Time Analytics via Incremental Maintenance



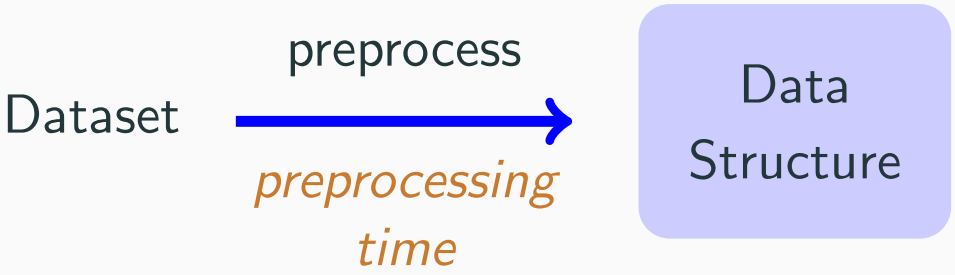
Real-Time Analytics via Incremental Maintenance



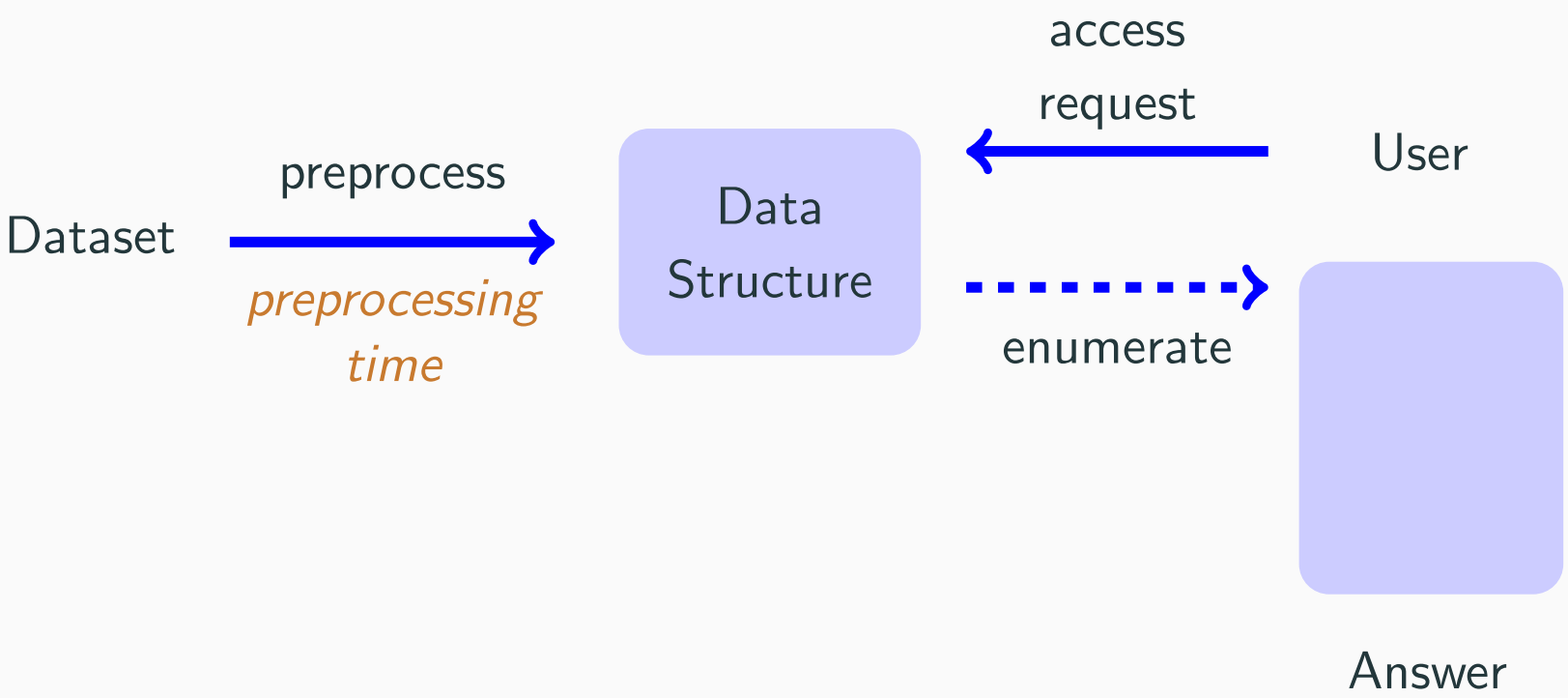
Real-Time Analytics via Incremental Maintenance



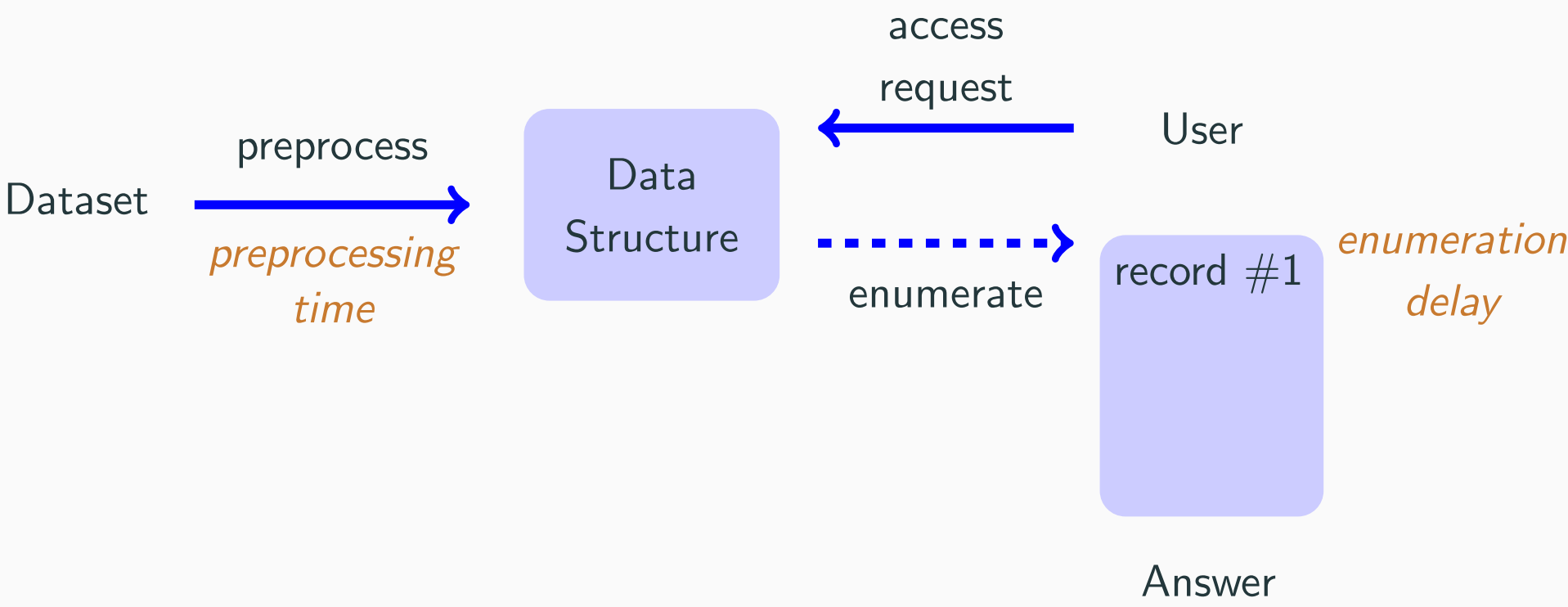
Incremental Maintenance: Refined Formal Analysis



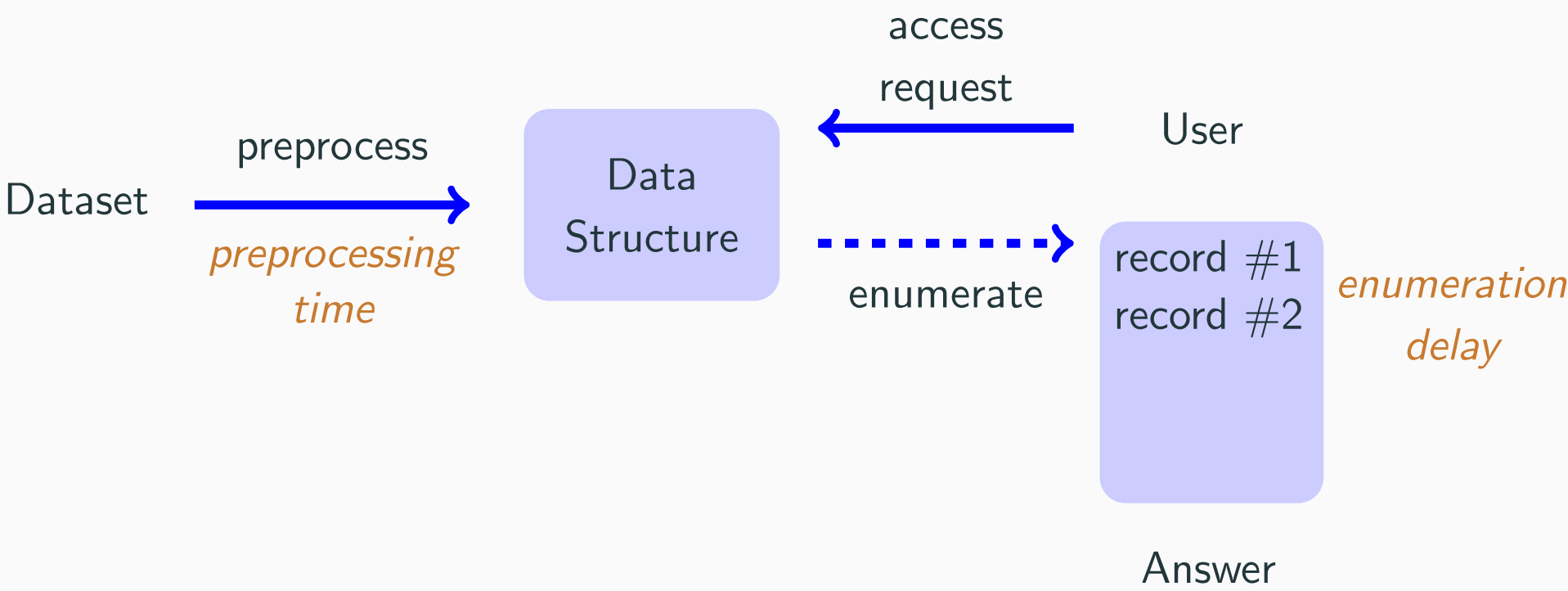
Incremental Maintenance: Refined Formal Analysis



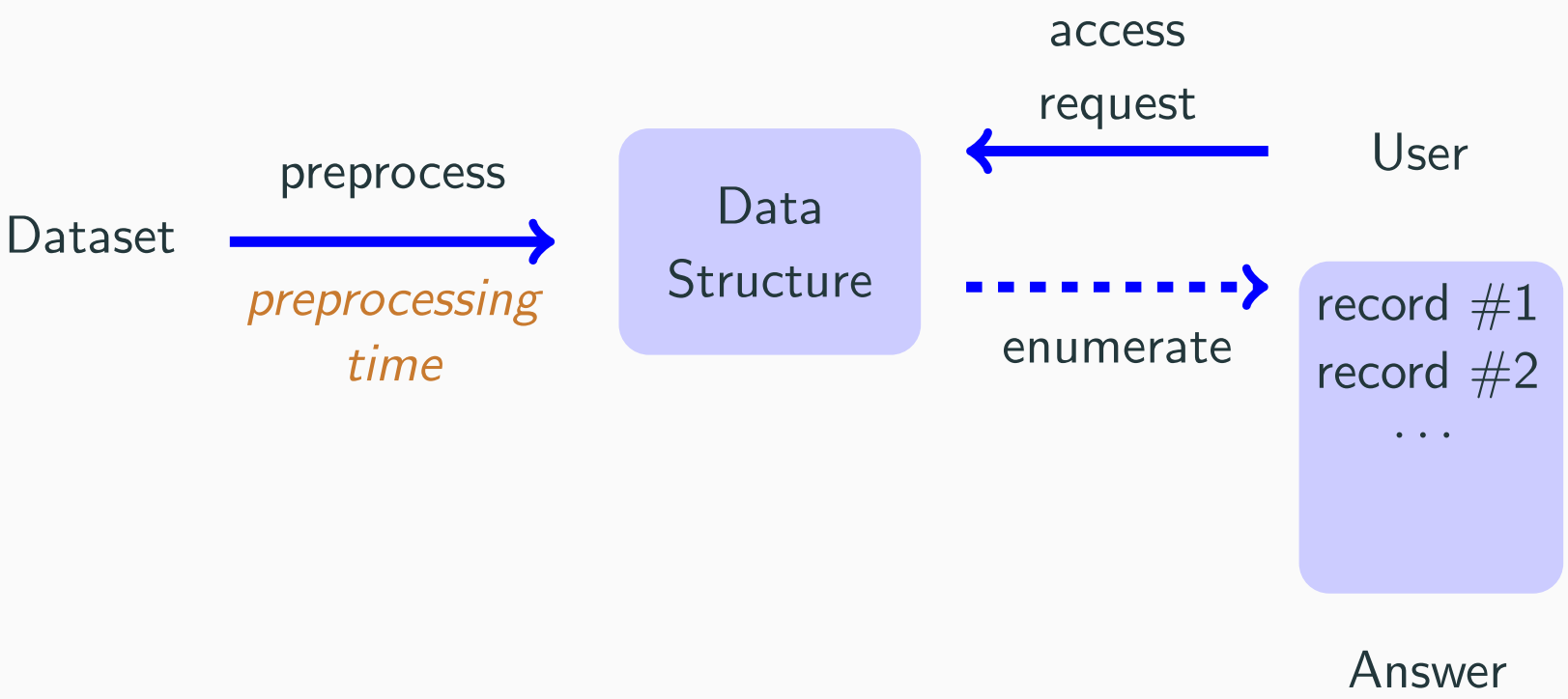
Incremental Maintenance: Refined Formal Analysis



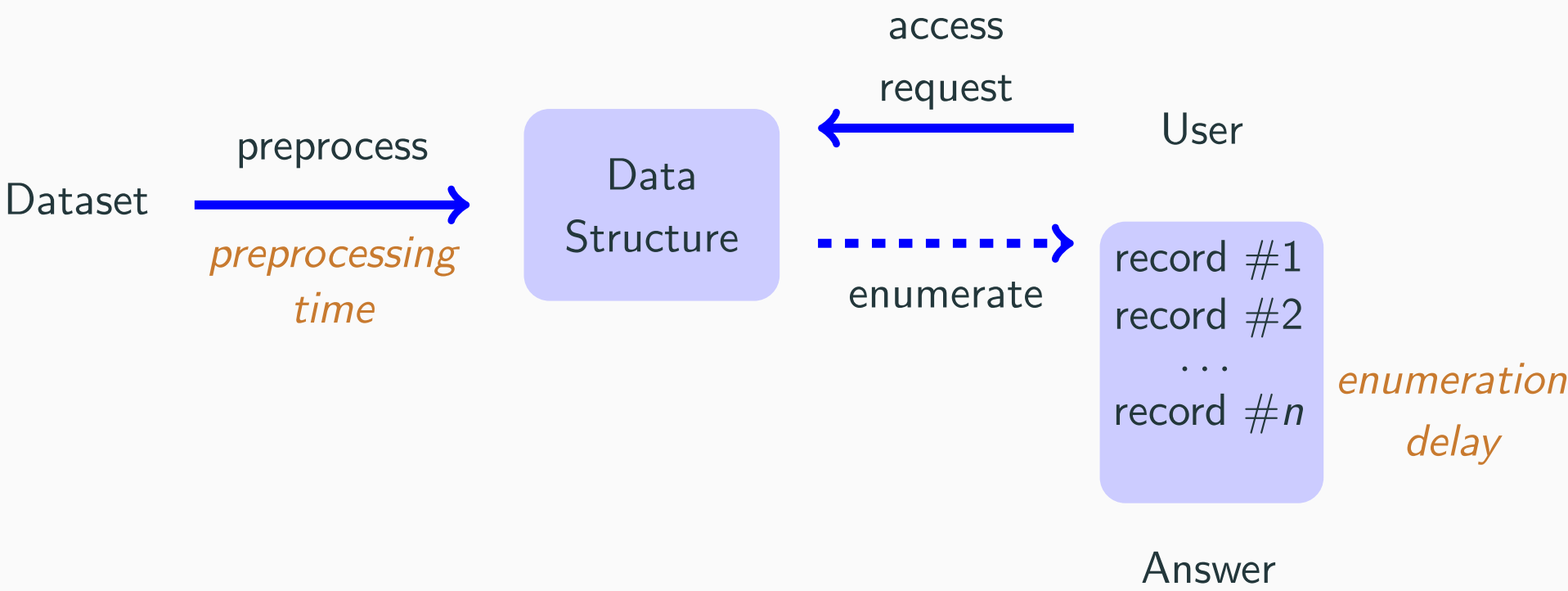
Incremental Maintenance: Refined Formal Analysis



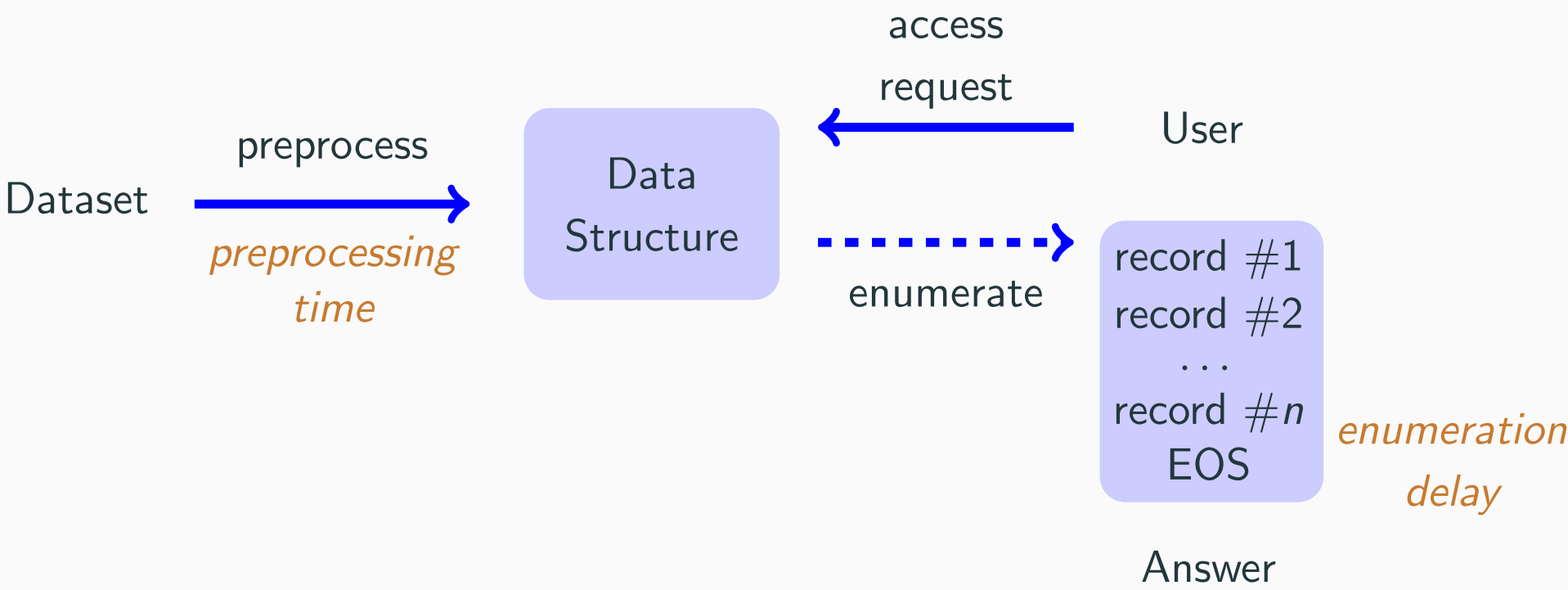
Incremental Maintenance: Refined Formal Analysis



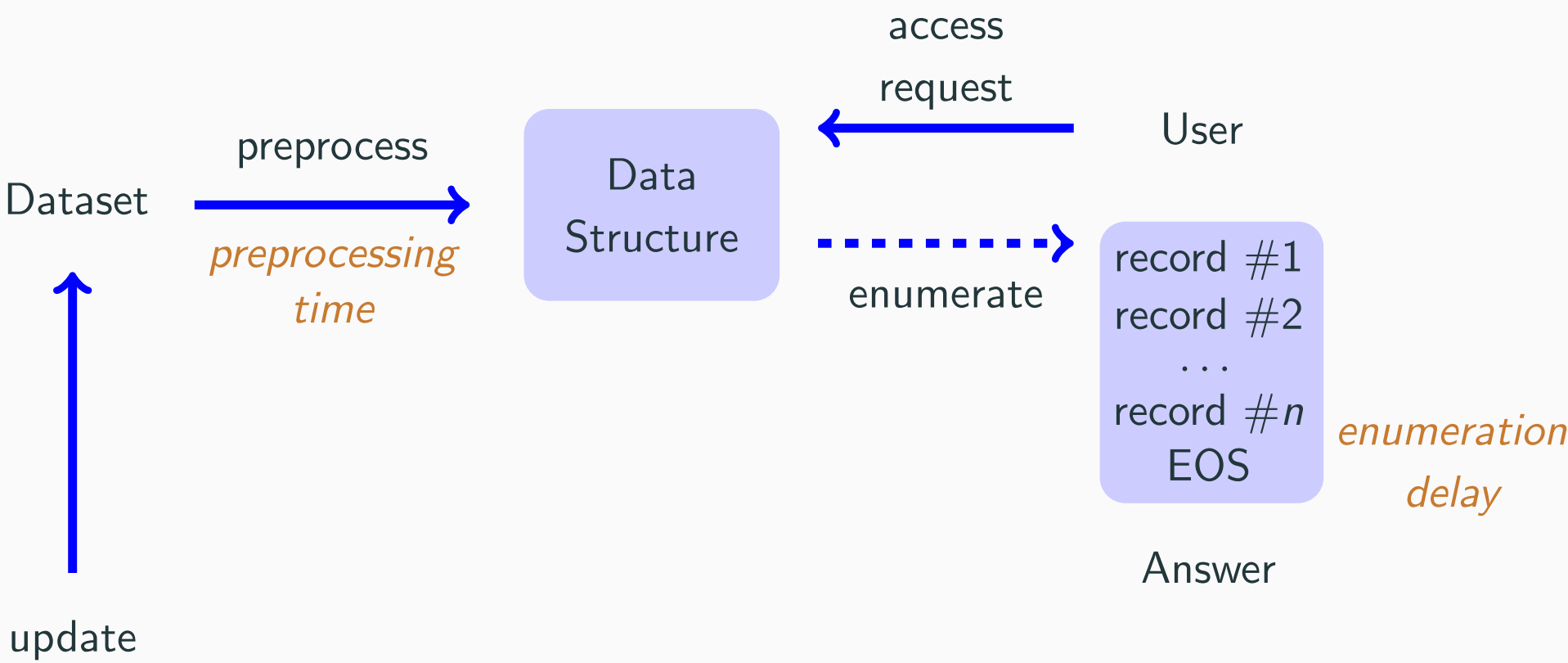
Incremental Maintenance: Refined Formal Analysis



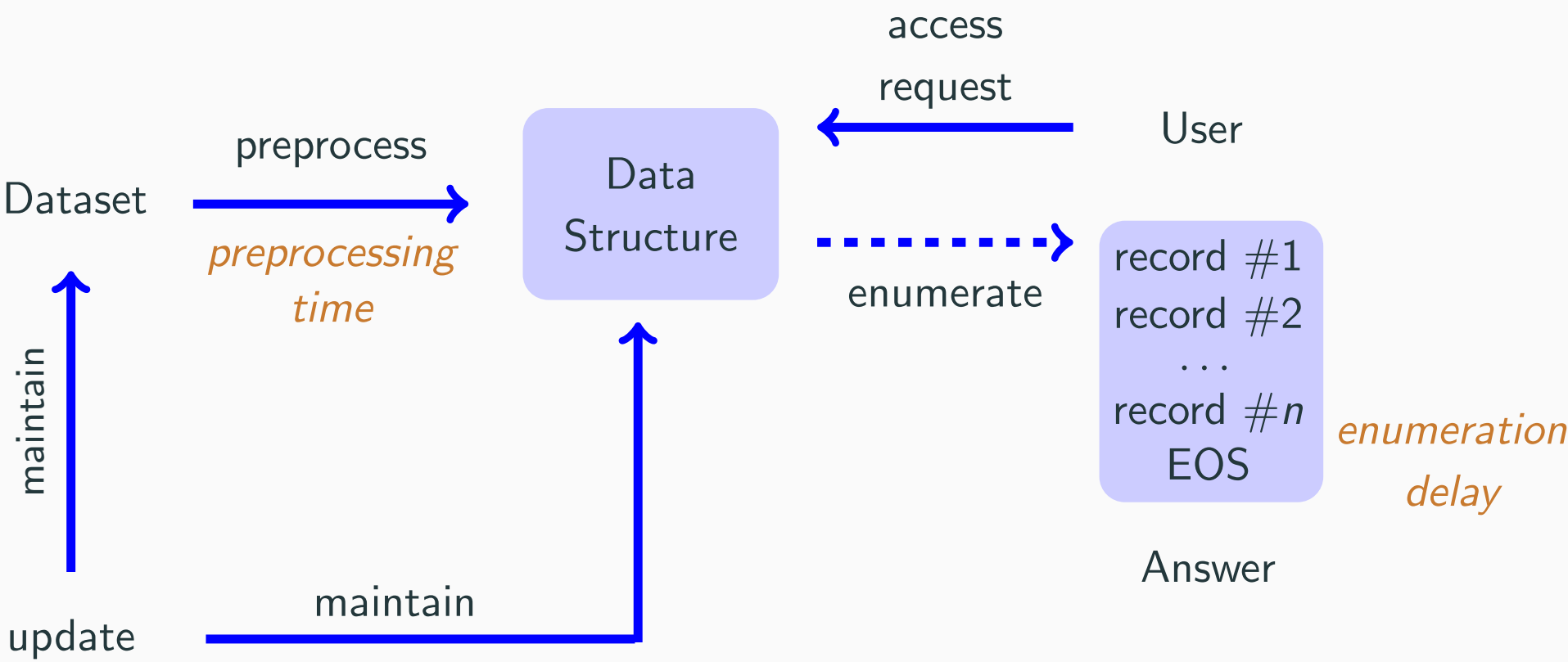
Incremental Maintenance: Refined Formal Analysis



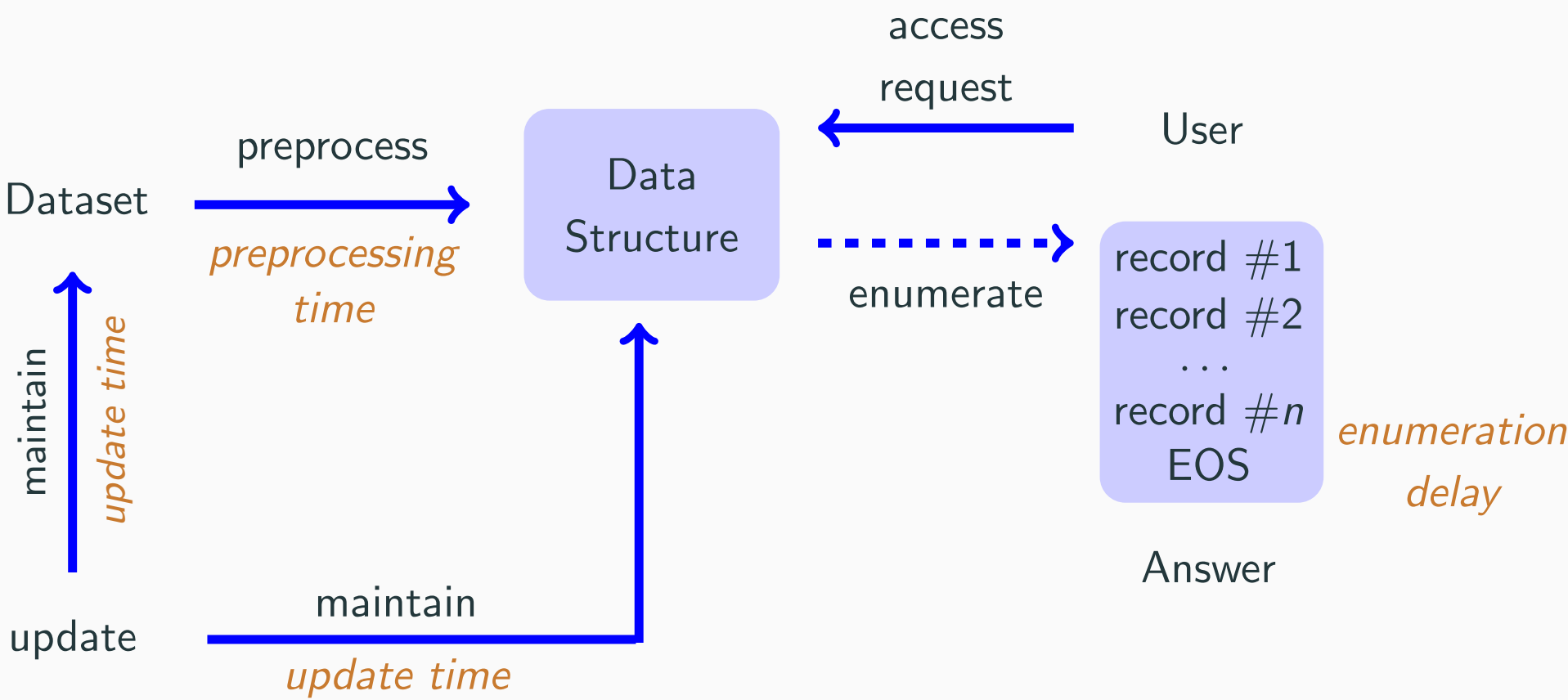
Incremental Maintenance: Refined Formal Analysis



Incremental Maintenance: Refined Formal Analysis



Incremental Maintenance: Refined Formal Analysis



Fundamental Questions

- Can we trade off update time for enumeration delay?

Fundamental Questions

- Can we trade off update time for enumeration delay?
- What is the best possible (worst-case optimal) maintenance strategy for the answer of a given query?

Fundamental Questions

- Can we trade off update time for enumeration delay?
- What is the best possible (worst-case optimal) maintenance strategy for the answer of a given query?
- How can we maintain efficiently models trained over changing relational data?

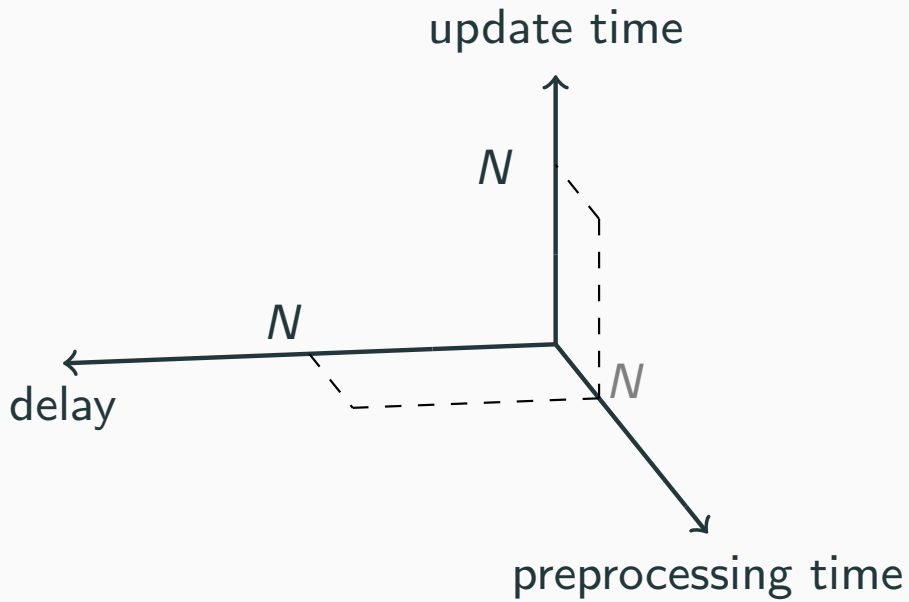
Update - Enumeration Trade-Off & Optimality

Incremental Maintenance for a Simple Query

Matrix-vector multiplication query: $Q(A) = R(A, B) \wedge S(B)$

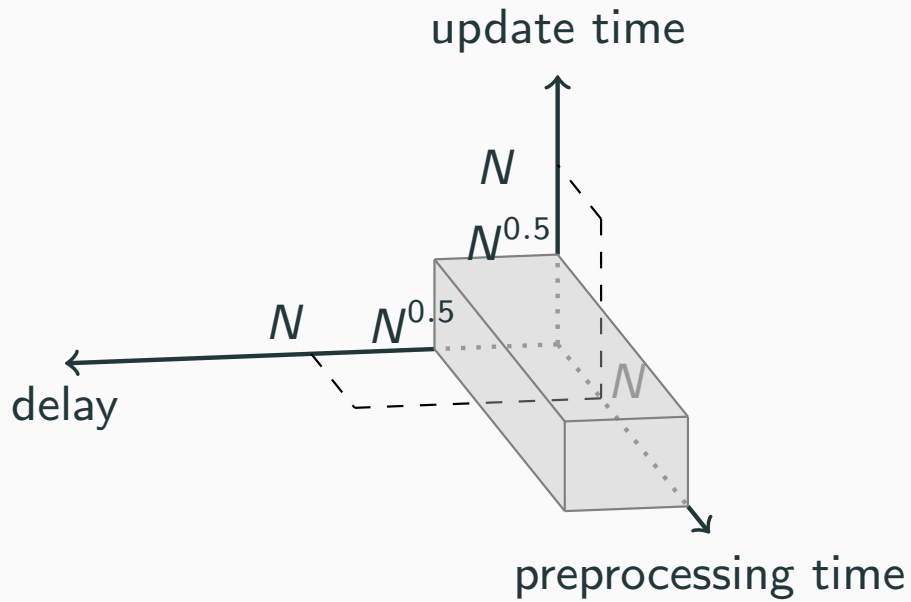
Incremental Maintenance for a Simple Query

Matrix-vector multiplication query: $Q(A) = R(A, B) \wedge S(B)$



Incremental Maintenance for a Simple Query

Matrix-vector multiplication query: $Q(A) = R(A, B) \wedge S(B)$



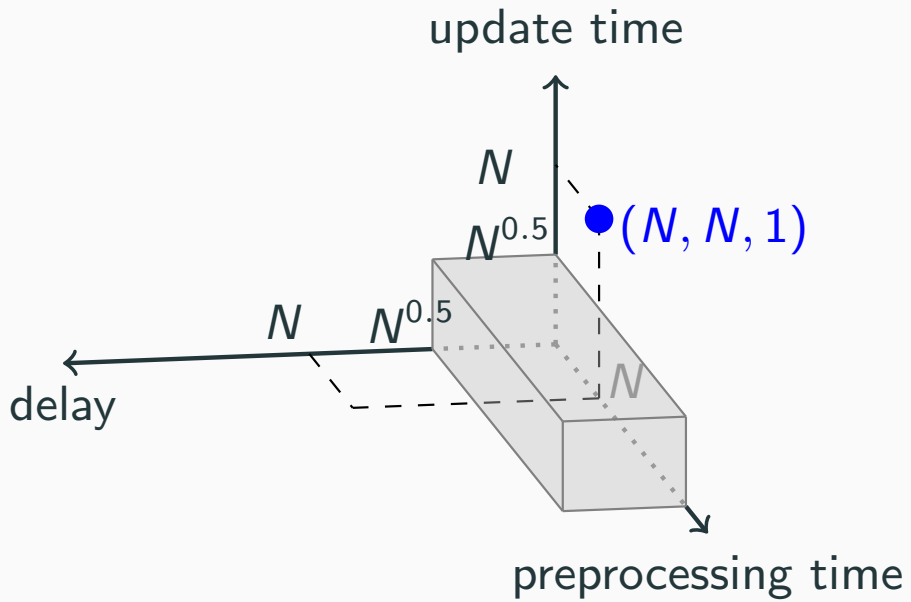
For this query, there is **no algorithm** that admits

preprocessing time	update time	enumeration delay
arbitrary	$\mathcal{O}(N^{0.5-\gamma})$	$\mathcal{O}(N^{0.5-\gamma})$

for any $\gamma > 0$, unless the OMv Conjecture fails

Incremental Maintenance for a Simple Query

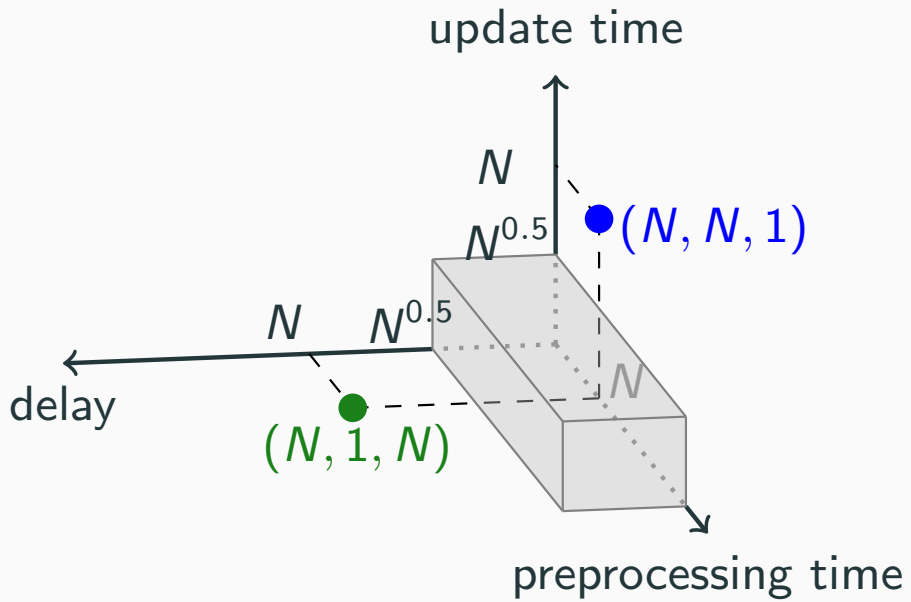
Matrix-vector multiplication query: $Q(A) = R(A, B) \wedge S(B)$



Eager approach: Expensive update, fast enumeration

Incremental Maintenance for a Simple Query

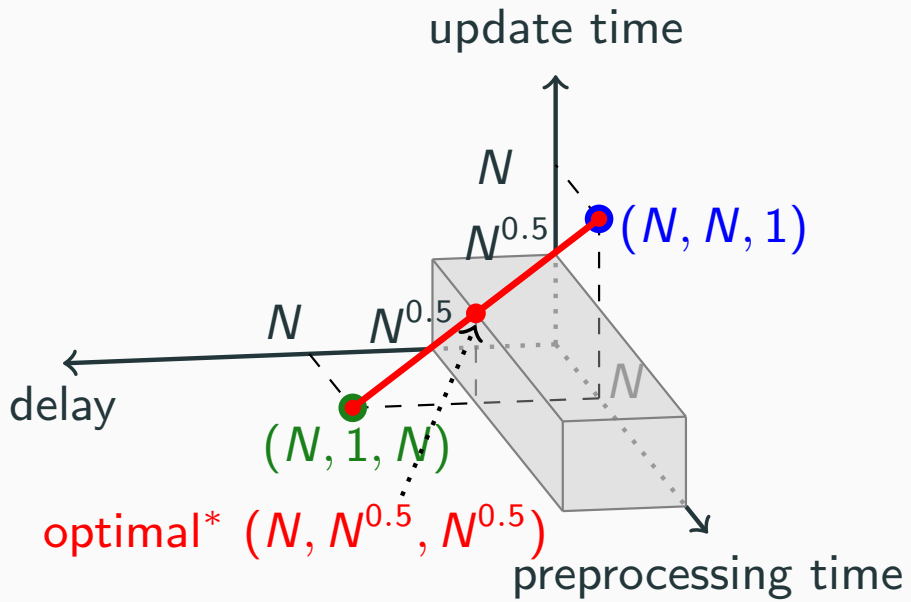
Matrix-vector multiplication query: $Q(A) = R(A, B) \wedge S(B)$



Lazy approach: Fast update, expensive enumeration

Incremental Maintenance for a Simple Query

Matrix-vector multiplication query: $Q(A) = R(A, B) \wedge S(B)$



Our approach: Any update-enumeration trade-off on the red line

Preprocessing time: $O(N)$ Update time: $O(N^{\epsilon})$ Delay: $O(N^{1-\epsilon})$ for any $\epsilon \in [0, 1]$

$(*)$: $\epsilon = 0.5$ is weakly Pareto optimal by OMv Conjecture

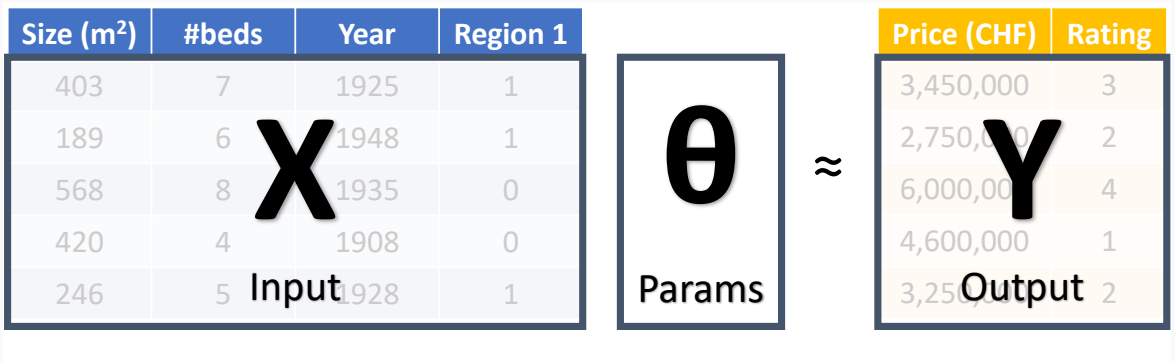
Maintaining Machine Learning Models

over

Evolving Relational Data

Example: Learning Polynomial Regression Models

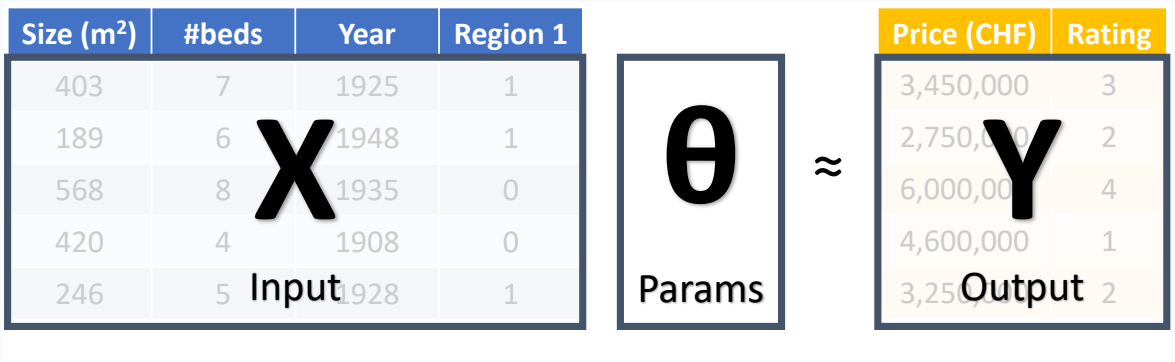
Goal: Find model parameters Θ best satisfying



- Features **X** and labels **Y** are given by database joins

Example: Learning Polynomial Regression Models

Goal: Find model parameters Θ best satisfying



- Features $\mathbf{X}$ and labels $\mathbf{Y}$ are given by database joins
- Solved using iterative gradient computation:

$$\mathbf{\Theta}_{i+1} = \mathbf{\Theta}_i - \alpha\, \mathbf{X}^{\mathbf{T}}(\mathbf{X}\,\mathbf{\Theta}_i - \mathbf{Y}) \quad \text{(repeat until convergence)}$$

- Approach considered here: Compute once for all iterations the Covariance Matrix $[\mathbf{X}\ \mathbf{Y}]^{\mathbf{T}}\,[\mathbf{X}\ \mathbf{Y}]$

Covariance Matrix Defined by Queries

Covariance matrix $[\mathbf{X}\ \mathbf{Y}]^T\ [\mathbf{X}\ \mathbf{Y}]$ can be expressed in SQL

```
Q = SELECT SUM(1 *1), SUM(1 *X_1), ... SUM(1 *X_n), SUM(1 *Y),
           SUM(X_1*1), SUM(X_1*X_1), ... SUM(X_1*X_n), SUM(X_1*Y),
           ...
           SUM(X_n*1), SUM(X_n*X_1), ... SUM(X_n*X_n), SUM(X_n*Y)
           SUM(Y *1), SUM(Y *X_1), ... SUM(Y *X_n), SUM(Y *Y)
      FROM R1 JOIN R2 JOIN ... JOIN Rn
```

Covariance Matrix Defined by Queries

Covariance matrix $[\mathbf{X}\ \mathbf{Y}]^T\ [\mathbf{X}\ \mathbf{Y}]$ can be expressed in SQL

```
Q = SELECT [ SUM(1 *1), SUM(1 *X_1), ... SUM(1 *X_n), SUM(1 *Y),
             SUM(X_1*1), SUM(X_1*X_1), ... SUM(X_1*X_n), SUM(X_1*Y),
             ...
             SUM(X_n*1), SUM(X_n*X_1), ... SUM(X_n*X_n), SUM(X_n*Y),
             SUM(Y *1), SUM(Y *X_1), ... SUM(Y *X_n), SUM(Y *Y) ]
       FROM R1 JOIN R2 JOIN ... JOIN Rn
```

We compute and maintain under data updates:

- COUNT = SUM(1) = database join size
- vector of SUM($\mathbf{X}_i$) for feature/label $\mathbf{X}_i$
- matrix of SUM($\mathbf{X}_i \cdot \mathbf{X}_j$) for features/label $\mathbf{X}_i$ and $\mathbf{X}_j$

Maintaining the Covariance Matrix using the Covariance Ring

Covariance Ring has the support:

- Set of triples $(\mathbb{Z}, \mathbb{R}^m, \mathbb{R}^{m \times m})$

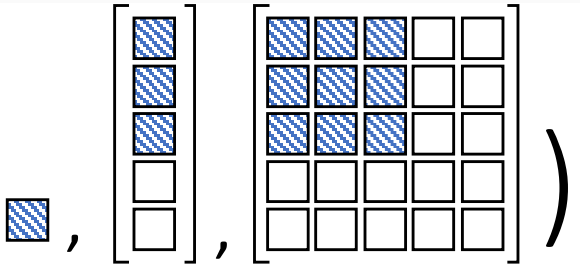
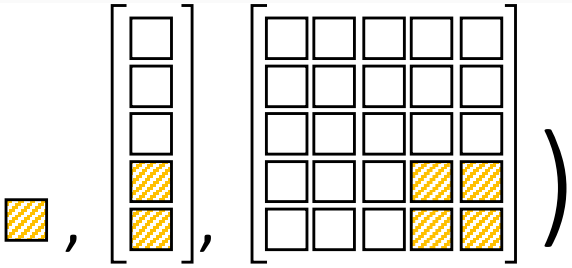
$$\Big(\text{COUNT}, \quad \text{vector of SUM}(\mathbf{X}_i), \quad \text{matrix of SUM}(\mathbf{X}_i \cdot \mathbf{X}_j) \Big)$$

- Neutral elements for sum and product operations:

$$\mathbf{0} = (0, \mathbf{0}_{m \times 1}, \mathbf{0}_{m \times m})$$
$$\mathbf{1} = (1, \mathbf{0}_{m \times 1}, \mathbf{0}_{m \times m})$$

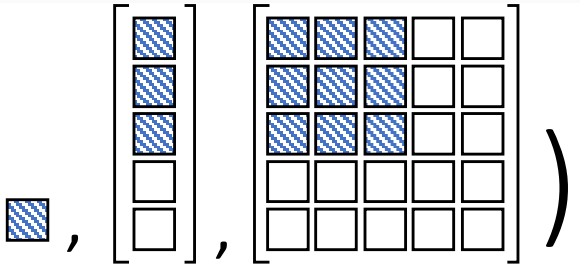
Maintaining the Covariance Matrix using the Covariance Ring

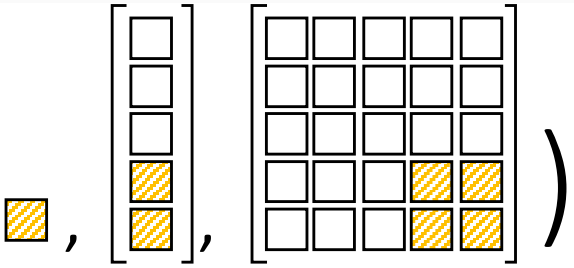
Covariance Ring has the sum and product operations:

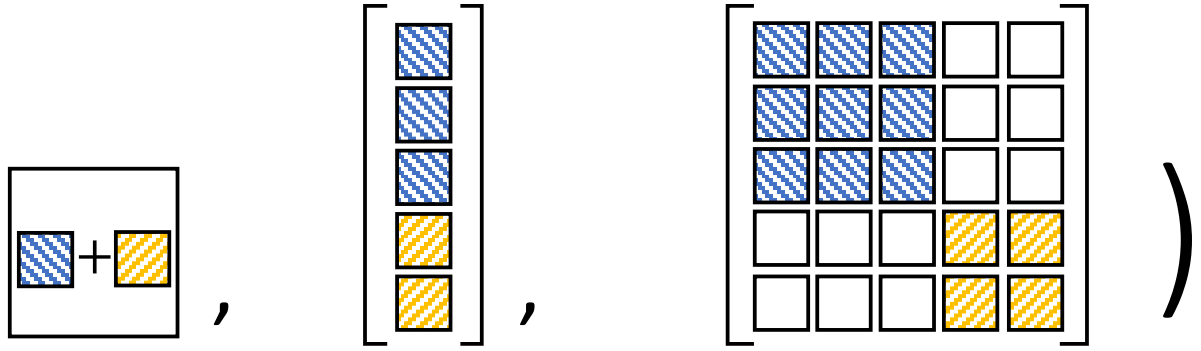
$a =$  $b =$ 

Maintaining the Covariance Matrix using the Covariance Ring

Covariance Ring has the sum and product operations:

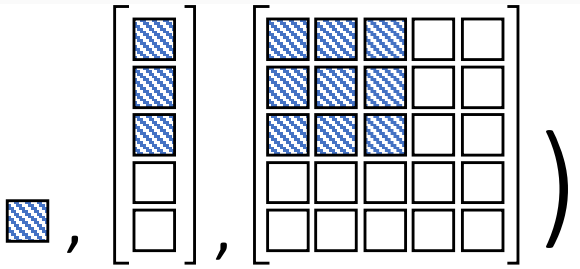
$a =$ 

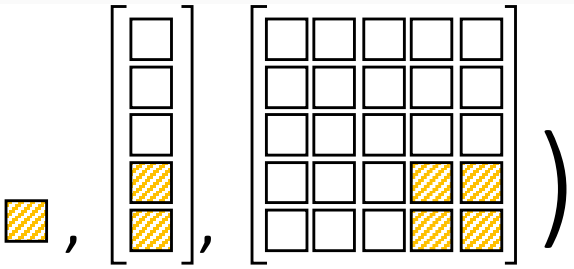
$b =$ 

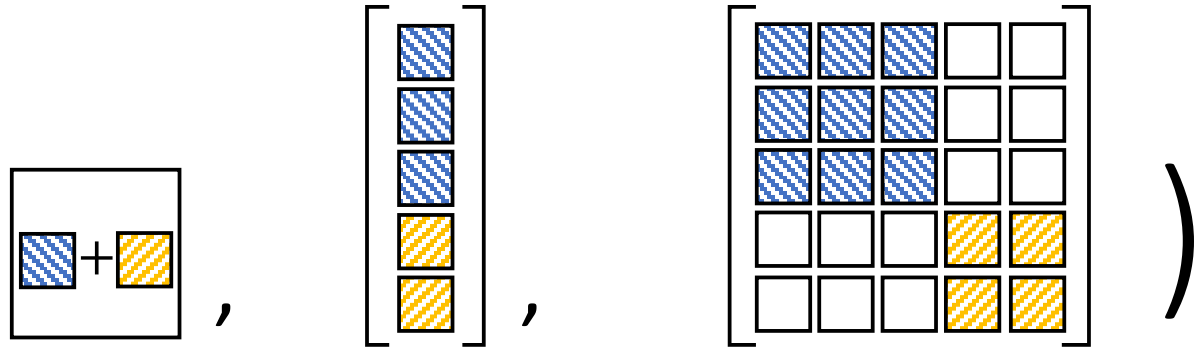
$a + b =$ 

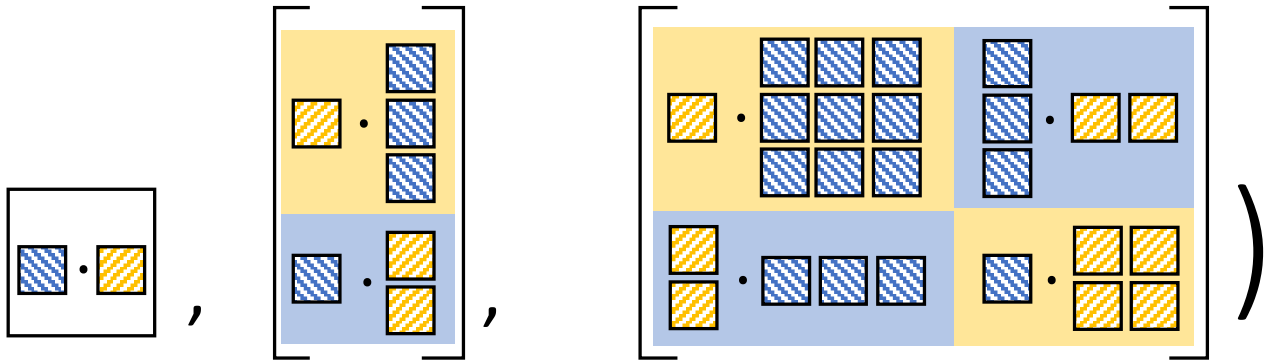
Maintaining the Covariance Matrix using the Covariance Ring

Covariance Ring has the sum and product operations:

$a =$ 

$b =$ 

$a + b =$ 

$a * b =$ 

Does the

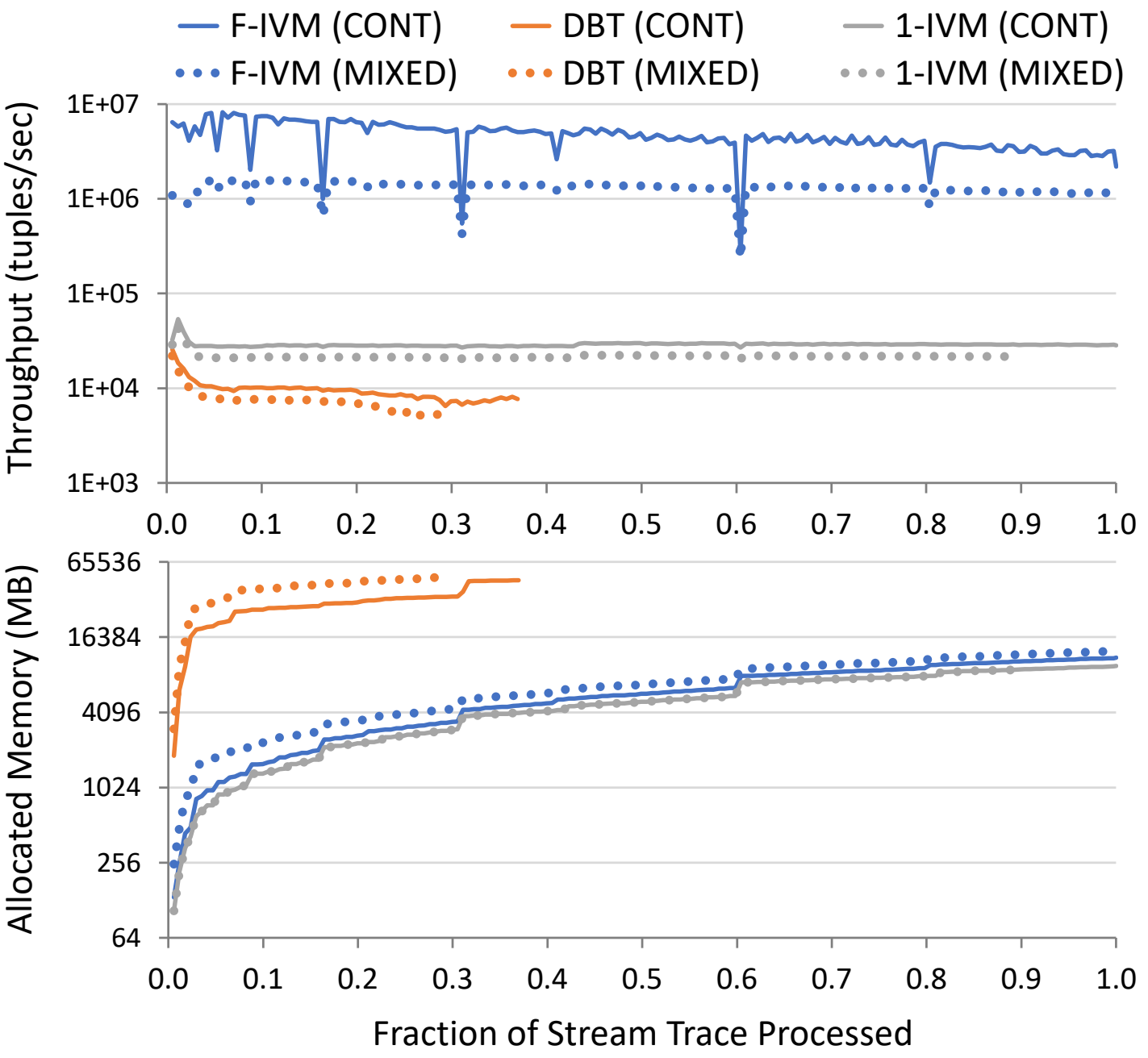
Theory

Make Any Difference in

Practice

?

Refresh Rate for a Linear Regression Model



Systems:

* F-IVM (ours)
* DBToaster
* Classical IVM

Features:

* CONTinuous
* MIXED:
  continuous
  and categorical

125M inserts in batches of 1K records at a time

Knowledge Transfer to Industry

Rk-means: Fast Clustering for Relational Data

Conventional machine learning algorithms cannot be applied until a data matrix is available to process. When the data matrix needs to be obtained from a relational database via a feature extraction query, the computation cost can be prohibitive, as the data matrix may be (much) larger than the total input relation size. This paper introduces Rk-means, or relationalk-means algorithm, for clustering relational data tuples without having to access the full data matrix.

Read More

Learning Models over Relational Data using Sparse Tensors and Functional Dependencies

Integrated solutions for analytics over relational databases are of great practical importance as they avoid the costly repeated loop data scientists have to deal with on a daily basis: select features from data residing in relational databases using feature extraction queries involving joins, projections, and aggregations; export the training dataset defined by such queries; convert this dataset into the format of an external learning tool; and train the desired model using this tool. These integrated solutions are also a fertile ground of theoretically fundamental and challenging problems at the intersection of relational and statistical data models.

Read More

Maintaining Triangle Queries under Updates

We consider the problem of incrementally maintaining the triangle queries with arbitrary free variables under single-tuple updates to the input relations. We introduce an approach called IVM that exhibits a trade-off between the update time, the space, and the delay for the enumeration of the query result, such that the update time ranges from the square root to linear in the database size while the delay ranges from constant to linear time. IVM achieves Pareto worst-case optimality in the update-delay space conditioned on the Online Matrix-Vector Multiplication conjecture.

Read More

Acknowledgments

DaST IVM team

Ahmet

Haozhe

Milos

RelationalAI IVM team

ElSeidy

Henrik

Niko

References

- Incremental View Maintenance with Triple Lock Factorization Benefits.
  ACM SIGMOD 2018
- Counting Triangles under Updates in Worst-Case Optimal Time.
  ICDT 2019 (Best paper award)
- F-IVM: Learning over Fast-Evolving Relational Data.
  ACM SIGMOD 2020
- Trade-offs in Static and Dynamic Evaluation of Hierarchical Queries.
  ACM PODS 2020 & LMCS 2023 (to appear)
- Maintaining Triangle Queries under Updates.
  ACM Trans. Database Syst. 2020
- Machine learning over static and dynamic relational data.
  DEBS 2021
- Conjunctive Queries with Output Access Patterns under Updates.
  ICDT 2023
- Evaluation Trade-Offs for Acyclic Conjunctive Queries. CSL 2023

Thank you!