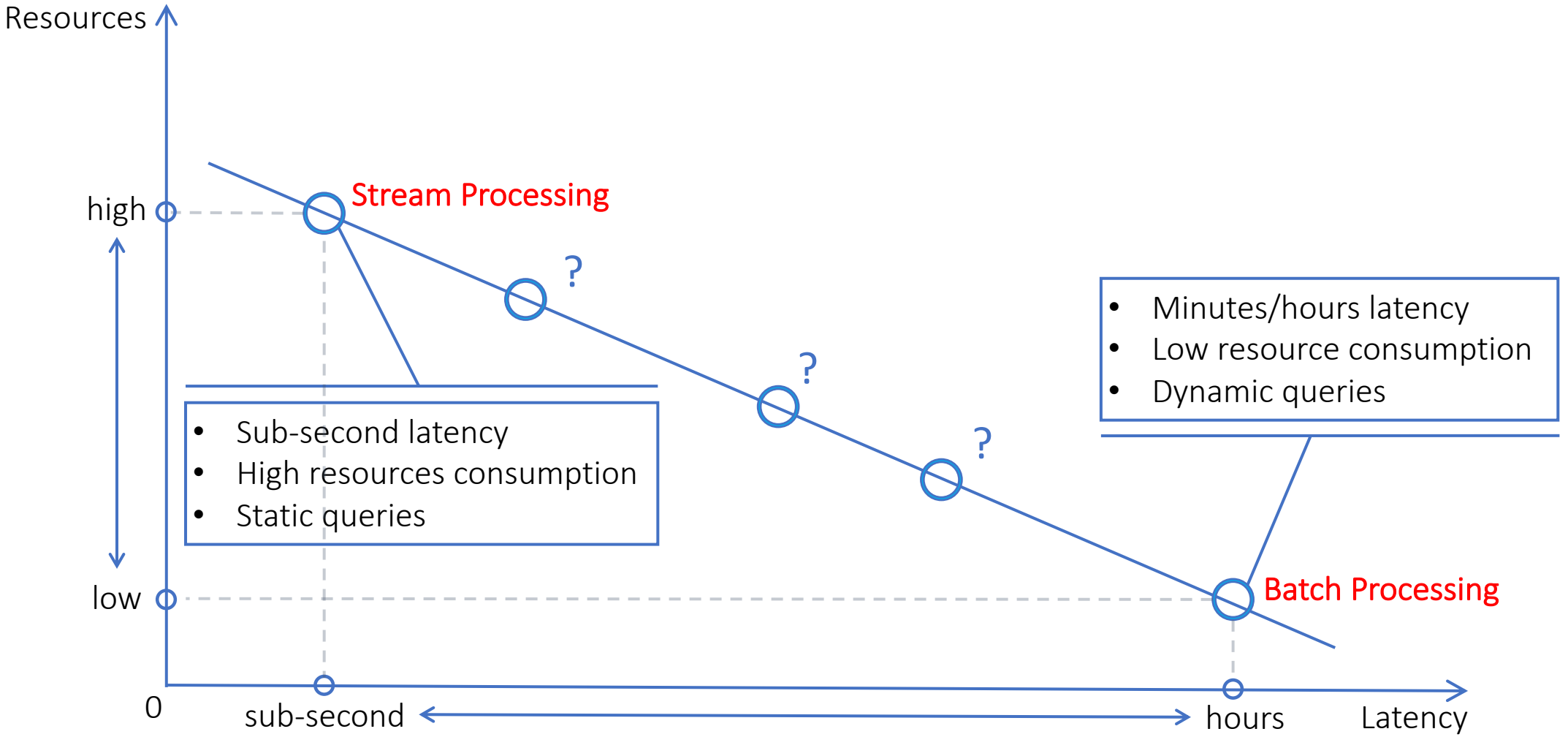
Tempura: A General Cost-Based Optimizer Framework for Incremental Data Processing

Zuozhi Wang[2], Kai Zeng[1], ***Botong Huang***[1], Wei Chen[1], Xiaozong Cui[1], Bo Wang[1], Ji Liu[1], Liya Fan[1], Dachuan Qu[1], Zhenyu Hou[1], Tao Guan[1], Chen Li[2], Jingren Zhou[1]

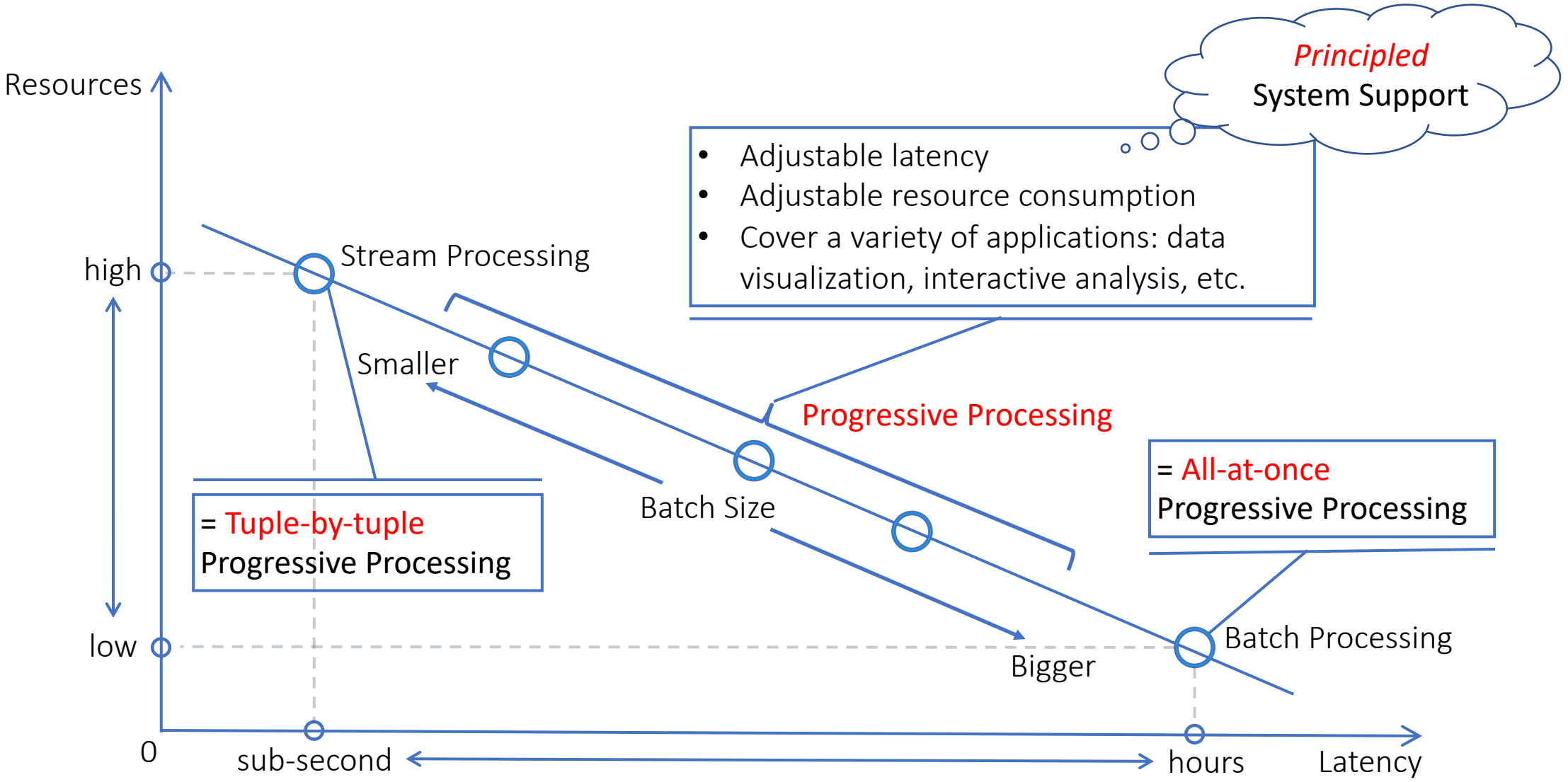
1. Alibaba Group 2. UC Irvine

Execution Paradigms

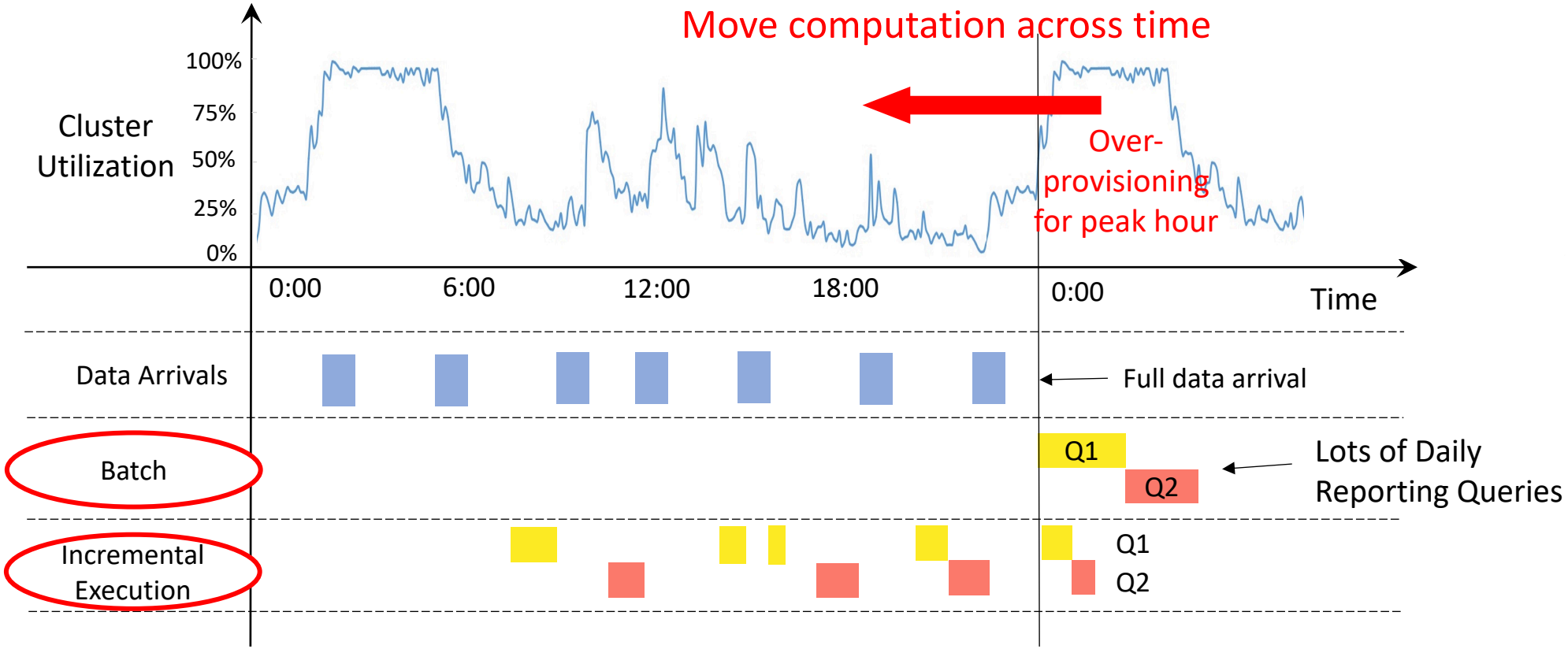


Progressive Execution: A Unified Model

Alibaba Group 阿里巴巴集团

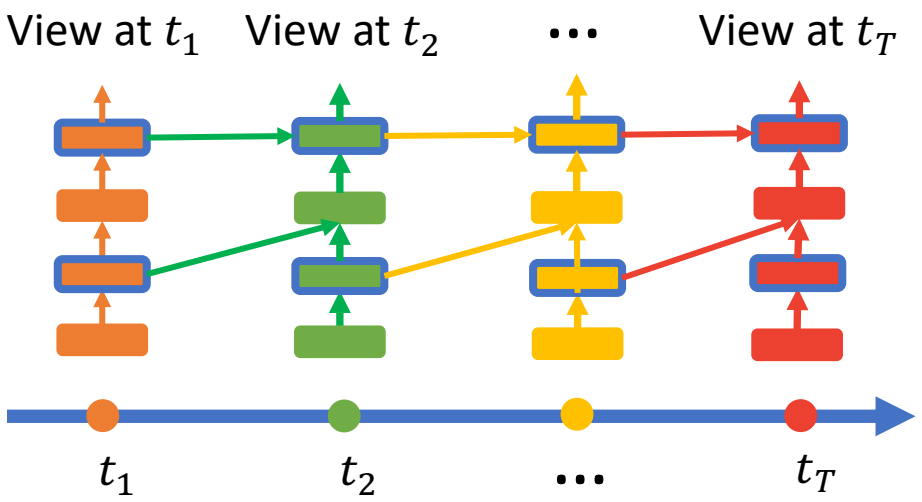


Scenario One:
Resource Skewness in Data Warehouses



Other Scenarios…

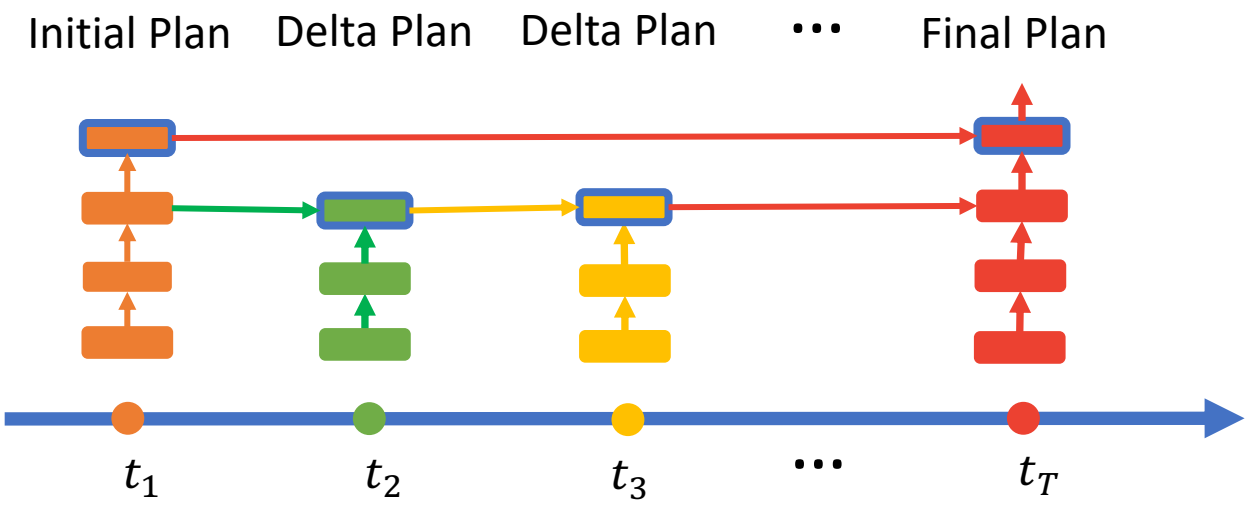
- Incremental view maintenance (IVM)
  - Need results on a standing query at 6:00,12:00,18:00,24:00 every day



Other Scenarios…

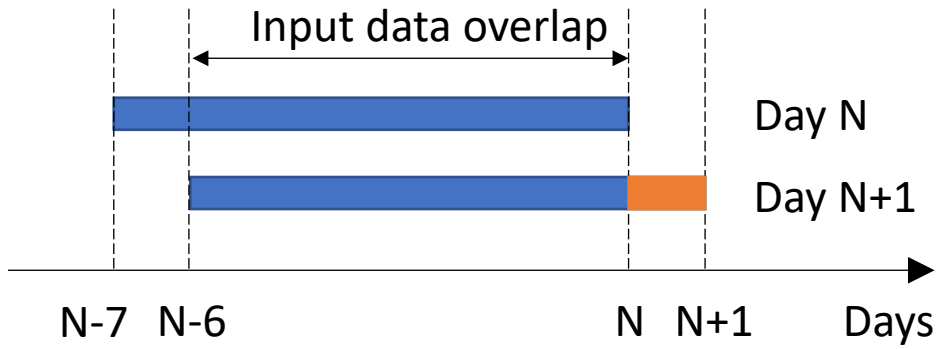
- Incremental view maintenance (IVM)
  - Need results on a standing query at 6:00,12:00,18:00,24:00 every day

- Streaming, Intermittent Query Processing



Other Scenarios…

- Incremental view maintenance (IVM)
  - Need results on a standing query at 6:00,12:00,18:00,24:00 every day
- Streaming, Intermittent Query Processing
- Periodic range query
  - Daily report over the last week's data



Incremental Query Processing (IQP) Methods

1. Incremental view maintenance (IVM)
2. Streaming: no output retractions

```
summary =
  with sales_status as (
    SELECT sales.o_id, category, price, cost
    FROM sales LEFT OUTER JOIN returns
      ON sales.o_id = returns.o_id )
  SELECT category,
    SUM(IF(cost IS NULL, price, -cost)) AS gross
  FROM sales_status
  GROUP BY category
```

sale_status = sales $\bowtie^{lo}$ *returns*

sales

o_id	cat	price	
o_1	c_1	100	t_1
o_2	c_2	150	t_1
o_3	c_1	120	t_1
o_4	c_1	170	t_1
o_5	c_2	300	t_2
o_6	c_1	150	t_2
o_7	c_2	220	t_2

returns

o_id	cost	
o_1	10	t_1
o_2	20	t_2
o_6	15	t_2

Input tables

Arrival time of records

sales_status

o_id	cat	price	cost
o_1	c_1	100	10
o_2	c_2	150	20
o_3	c_1	120	null
o_4	c_1	170	null
o_5	c_2	300	null
o_6	c_1	150	15
o_7	c_2	220	null

Final output

sale_status at t_1

o_id	cat	price	cost
o_1	c_1	100	10
o_2	c_2	150	null
o_3	c_1	120	null
o_4	c_1	170	null

Changes to sale_status at t_2

o_id	cat	price	cost	#
o_2	c_2	150	null	−1
o_2	c_2	150	20	+1
o_5	c_2	300	null	+1
o_6	c_1	150	15	+1
o_7	c_2	220	null	+1

Method 1: IVM

sale_status at t_1

o_id	cat	price	cost
o_1	c_1	100	10

Changes to sale_status at t_2

o_id	cat	price	cost	#
o_2	c_2	150	20	+1
o_3	c_1	120	null	+1
o_4	c_1	170	null	+1
o_5	c_2	300	null	+1
o_6	c_1	150	15	+1
o_7	c_2	220	null	+1

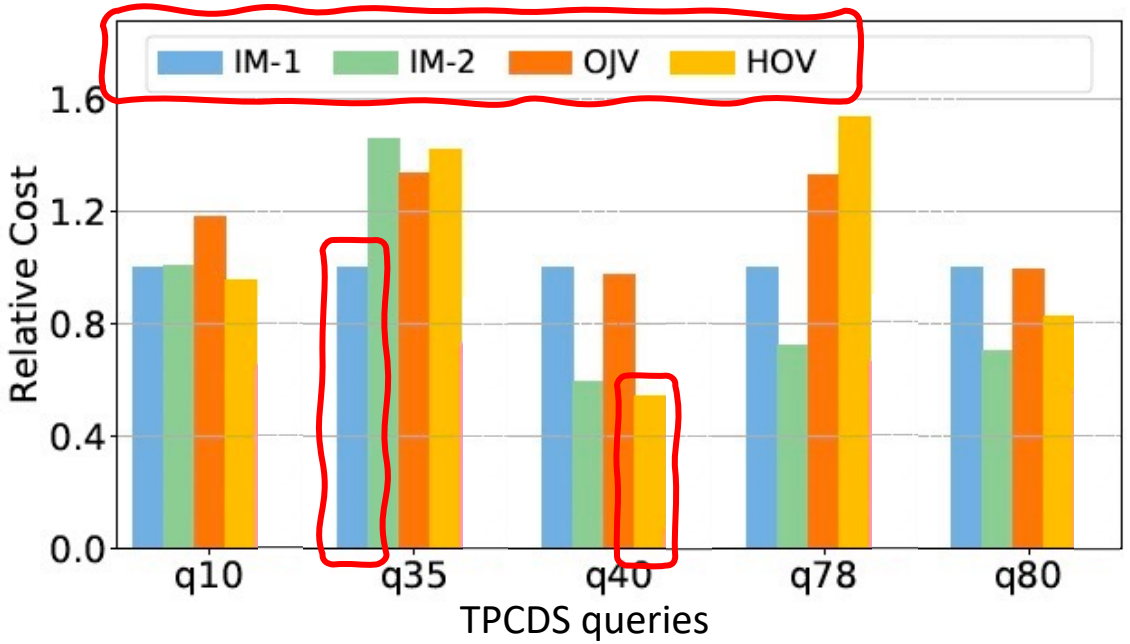
Method 2: Streaming

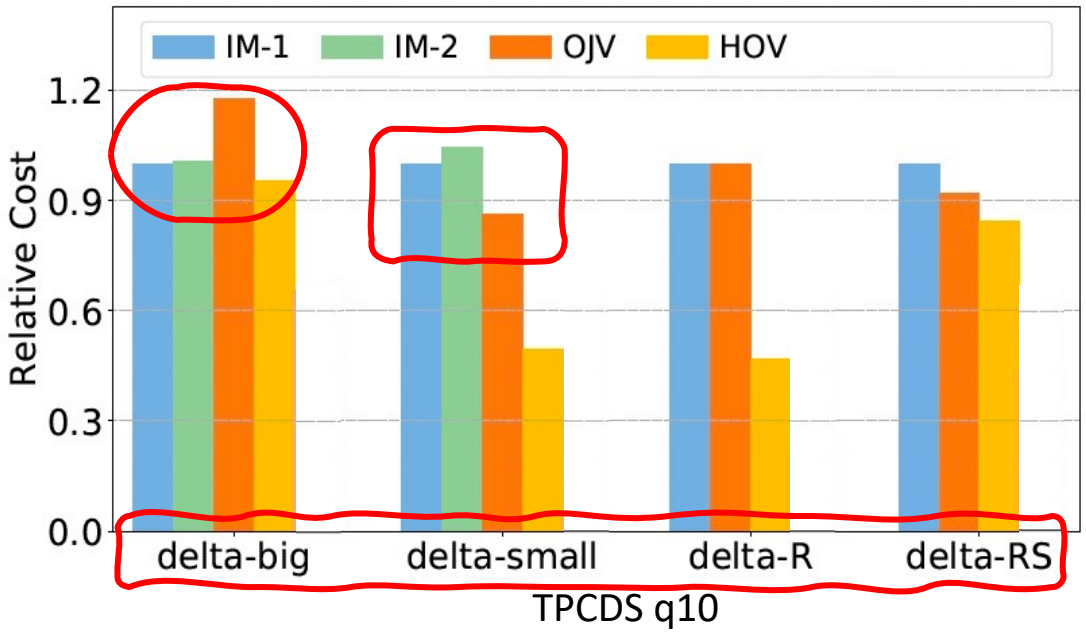
IVM does more computation at t_1,
but can be less efficient with retractions

Optimizing Incremental Plan is Non-trivial

Finding the optimal plan is really challenging…

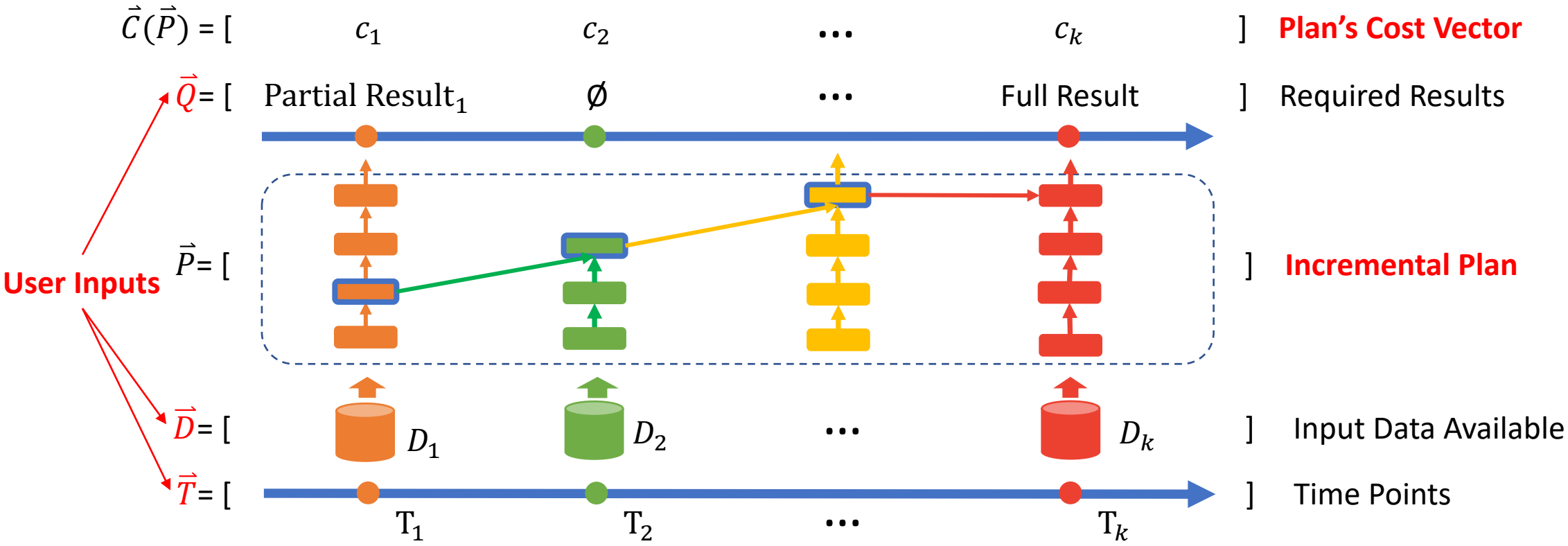
- Many incremental methods
- Different queries and data arrival patterns
- Different user preferences over time





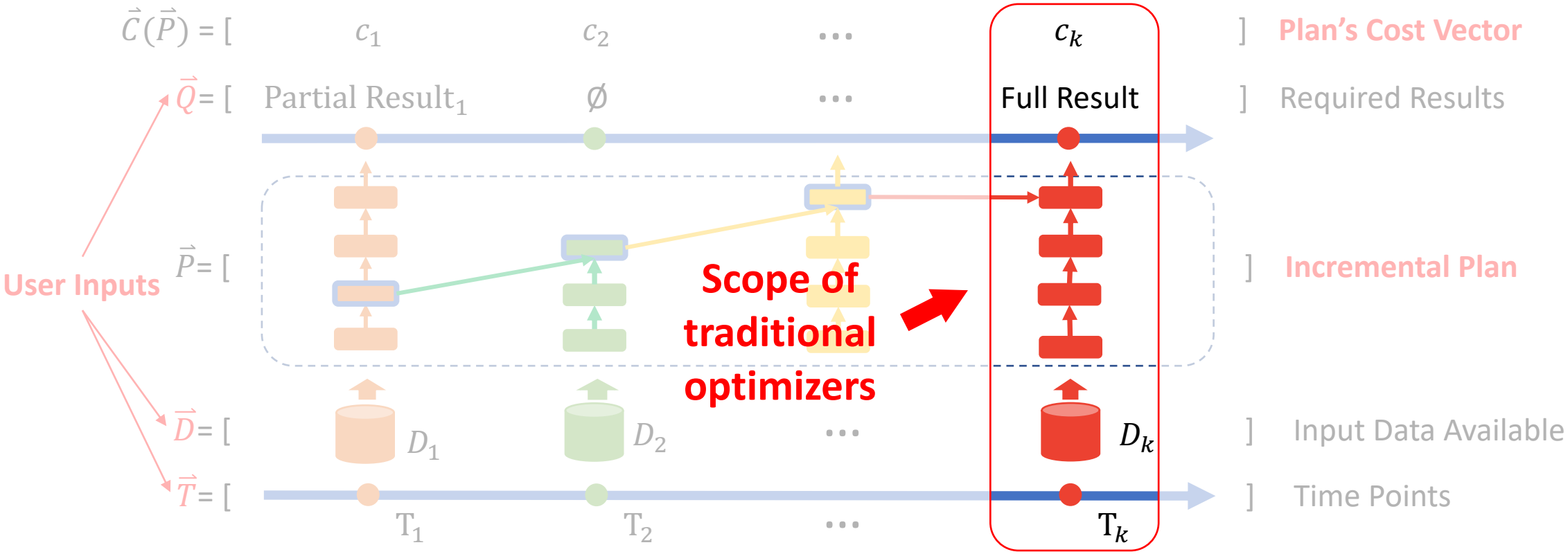
Tempura: A General Incremental Query Optimizer

- Applicable to general incremental scenarios
  - Best plan that minimizes user-defined $\tilde{c}(\vec{C})$, e.g. $\tilde{c}_w(\vec{C}) = \sum_{i=1}^{k} w_i c_i$



Tempura: A General Incremental Query Optimizer

- Applicable to general incremental scenarios
  - Best plan that minimizes user-defined $\tilde{c}(\vec{C})$, e.g. $\tilde{c}_w(\vec{C}) = \sum_{i=1}^{k} w_i c_i$



Tempura: A General Incremental Query Optimizer

- Applicable to general incremental scenarios

- The *first* volcano-style cost-based optimization framework for IQP
  - *The TIP model* that describes IQP in its most general form
  - Explore various incremental methods in one plan space
  - Cost-based search for best multi-time-point plan
  - Use multi-query optimization technique to materialize states

TVR-based Incremental Query Planning (TIP) Model

- Time Varying Relation (TVR):
  - A relation that changes over time
  - TVR *R* + query *Q* defines TVR *Q(R)*
  - Snapshots and Deltas
- Basic transformations
  - Merge operator $+^{\#}$ and $+^{sum}$

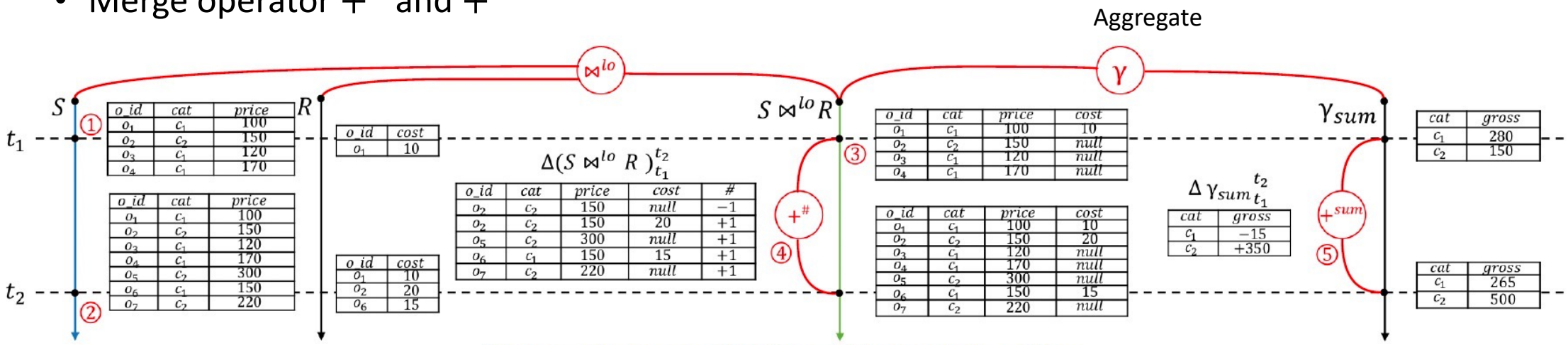


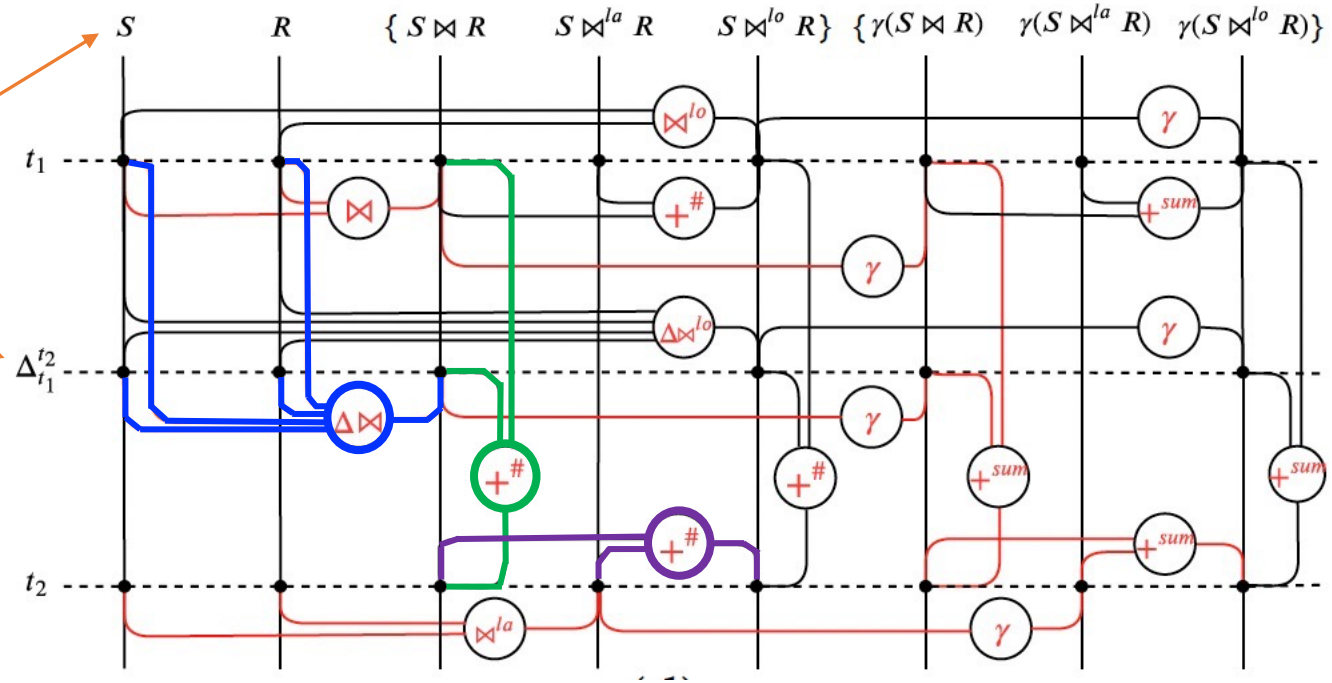
Figure 2: Example TVR's and their relationships.

Plan Space and TVR Rewrite Rules

- TVR generating rules: snapshot and delta compute rules
- Intra-TVR rules: conversions within a TVR
- Inter-TVR rules: advanced conversions between TVRs

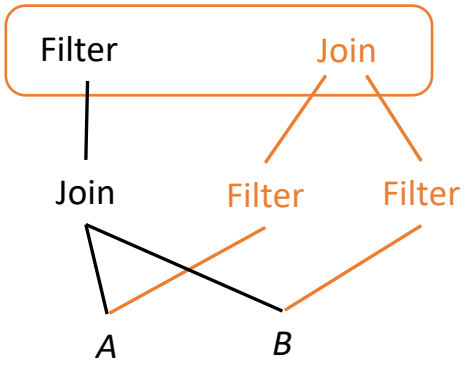
Extend CBO's plan space with time and TVR information

$$\Delta(S \bowtie R)_t^{t'} = \Delta S \bowtie R_t + (S_t + \Delta S) \bowtie \Delta R_t^{t'}$$

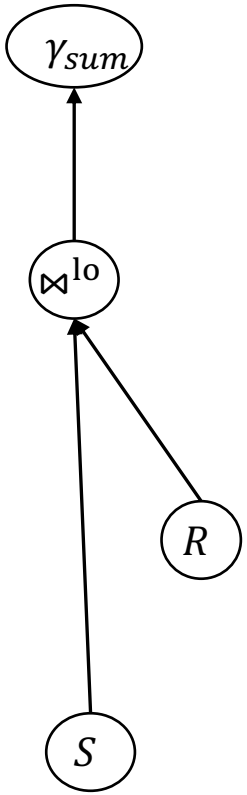


Traditional Memo (Apache Calcite)

- Operators (RelNode)
- Logical equivalence group (RelSet)
- Physical equivalence group (RelSubset)
- Match rules on changed part of memo



Match pattern in memo
New to register into memo



Traditional rule: filter-push-down

Memo Expansion in Tempura

Assume:

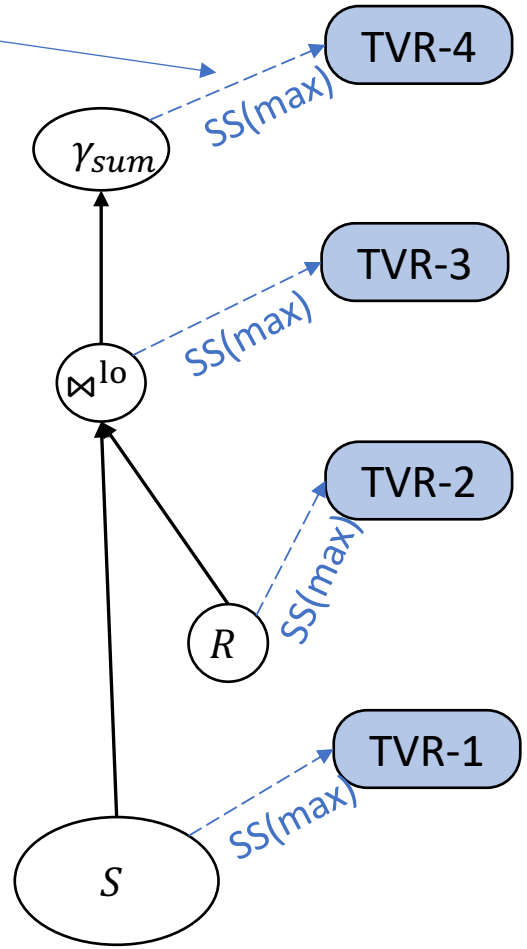
- $\vec{T} = [1, max]$
- $\vec{Q} = [\emptyset, Q]$

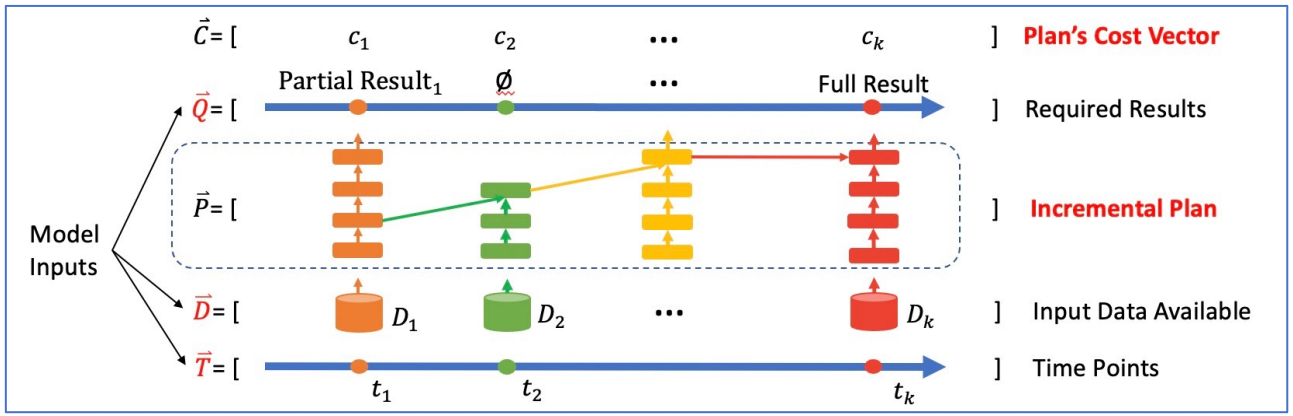
Intra-TVR relationship

$+^{\#}$ SS: Set Snapshot
SD: Set Delta

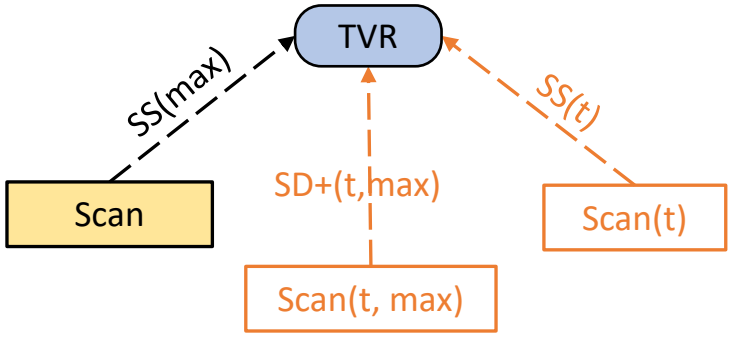
$+^{sum}$ VS: Value Snapshot
VD: Value Delta

Generalization from relations to TVRs



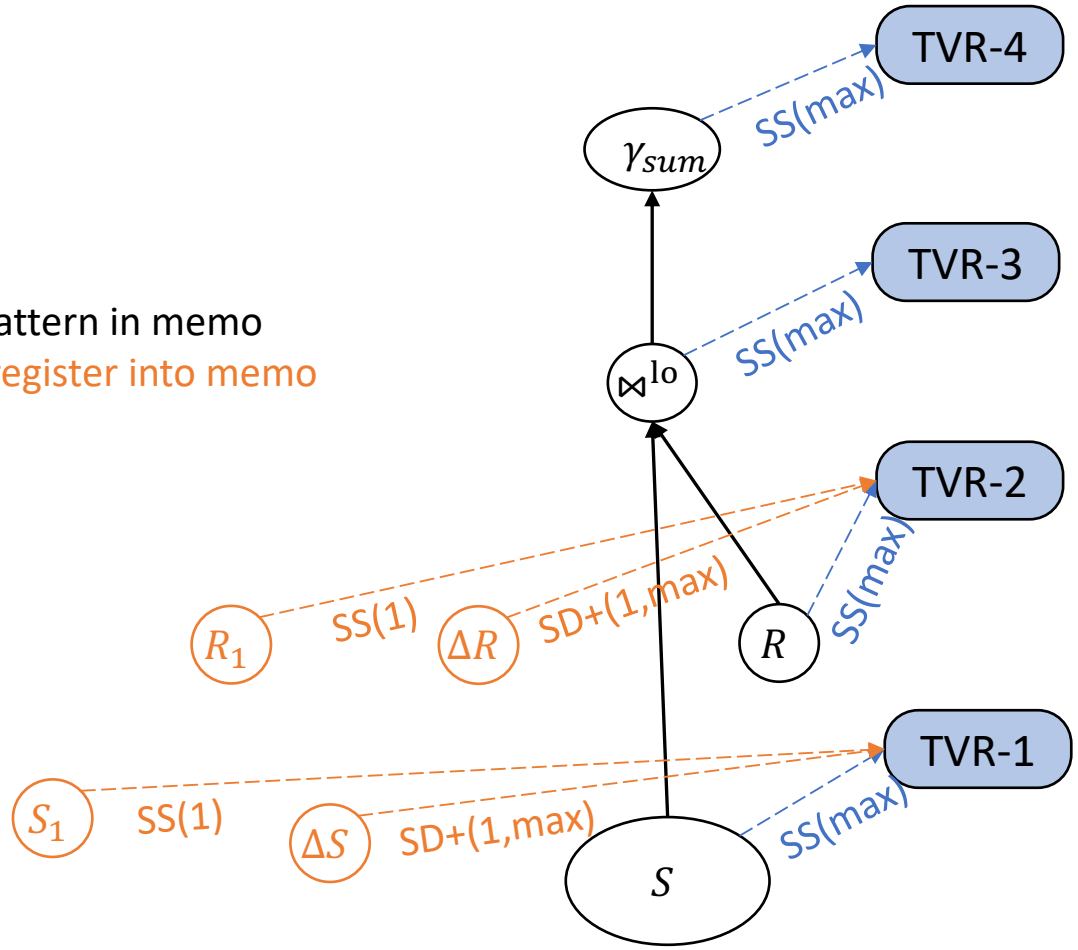


Memo Expansion in Tempura

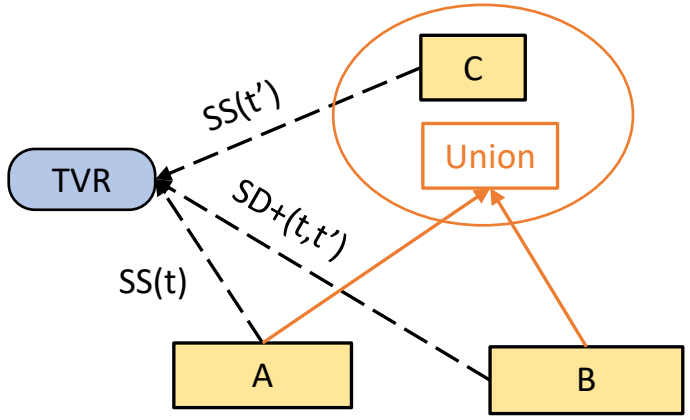


TVR generating rule:
TvrTableScanRule

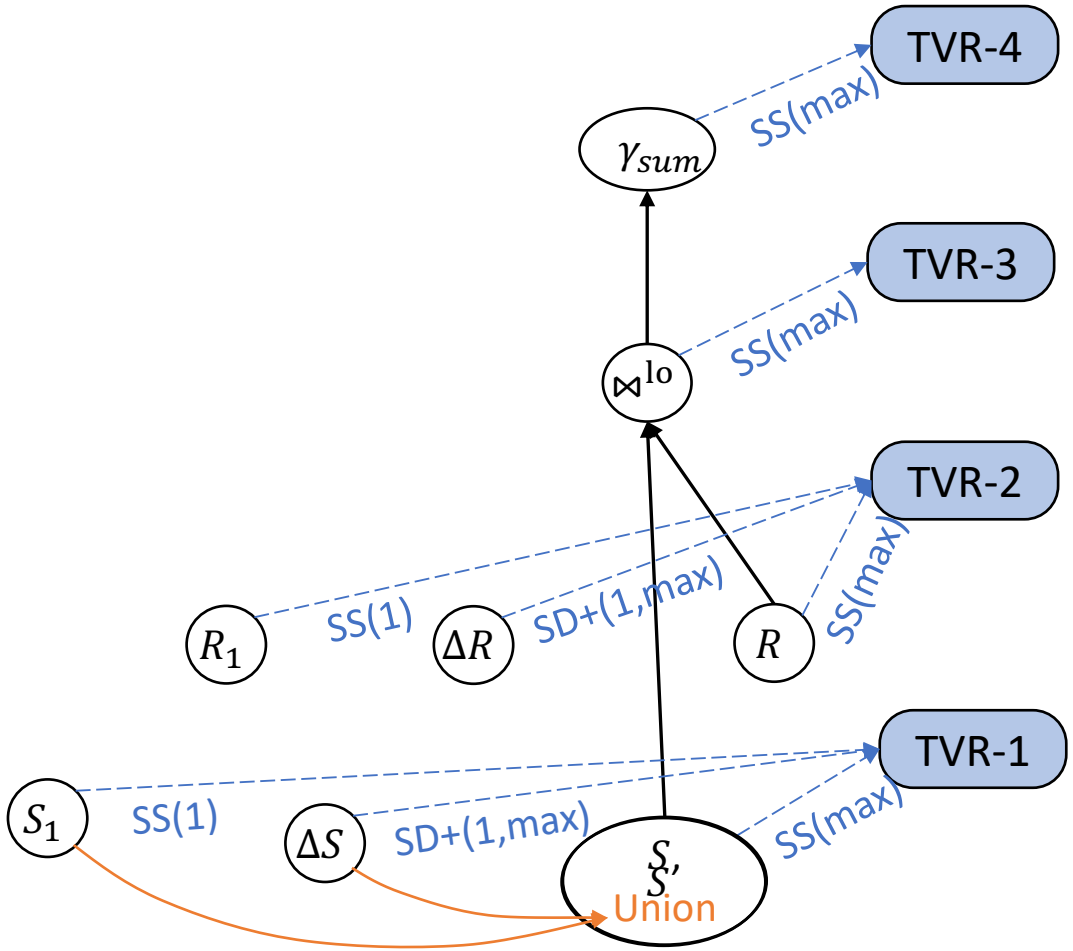
Match pattern in memo
New to register into memo



Memo Expansion in Tempura

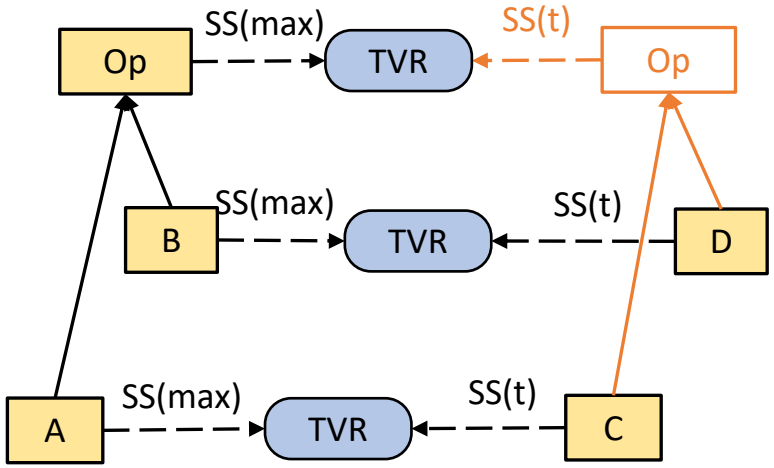


Intra-TVR rule:
TvrSetDeltaMergeRule

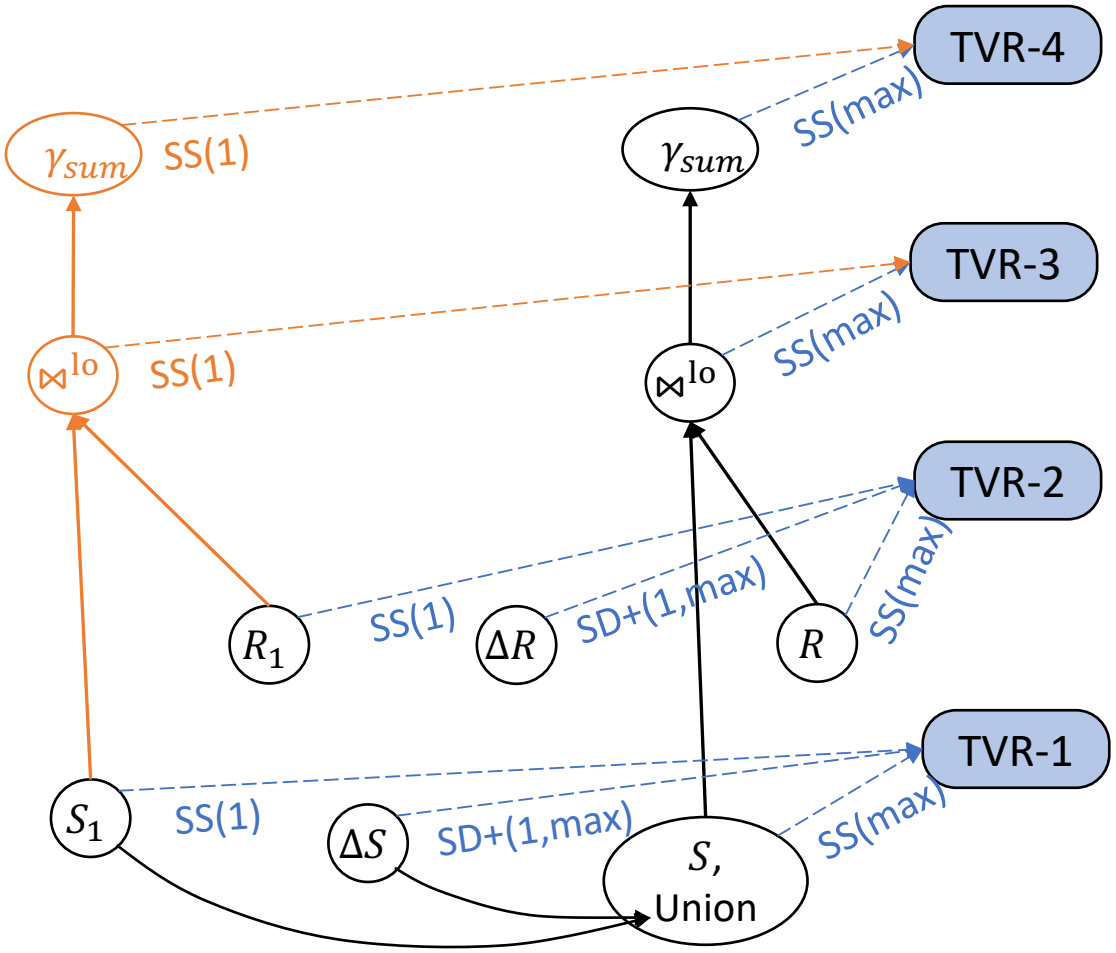


Memo Expansion in Tempura

How to compute a snapshot

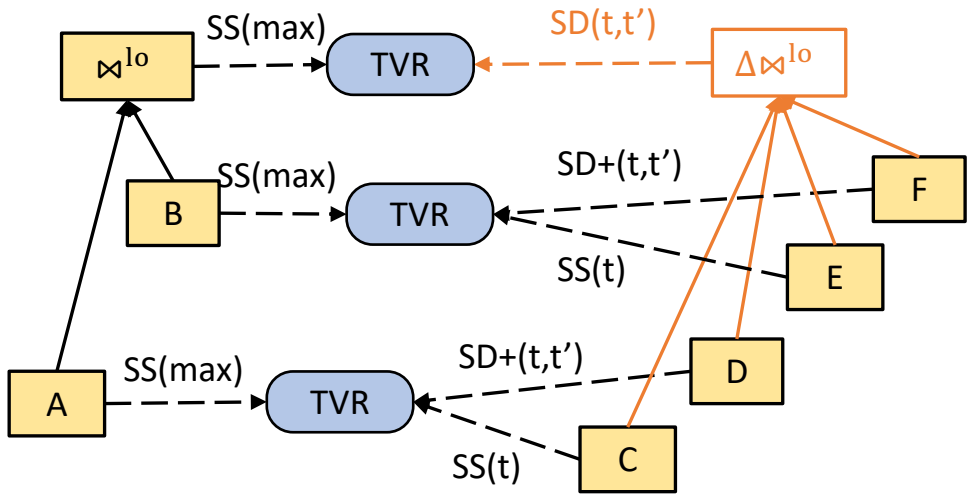


TVR generating rule:
TvrAnyToSetSnapshotRule (two-input op)



Memo Expansion in Tempura

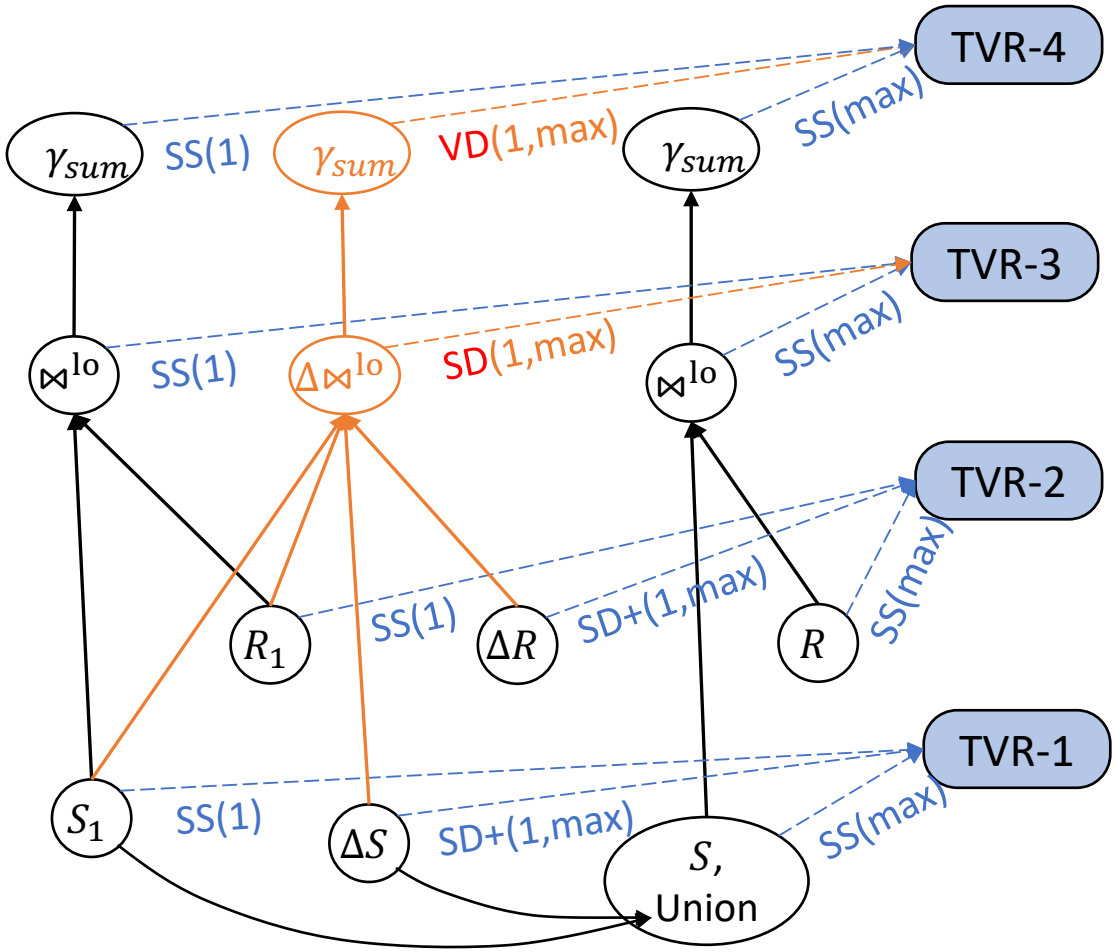
How to compute a delta



TVR generating rule:
TvrJoinRuleOneSideMultiMatch

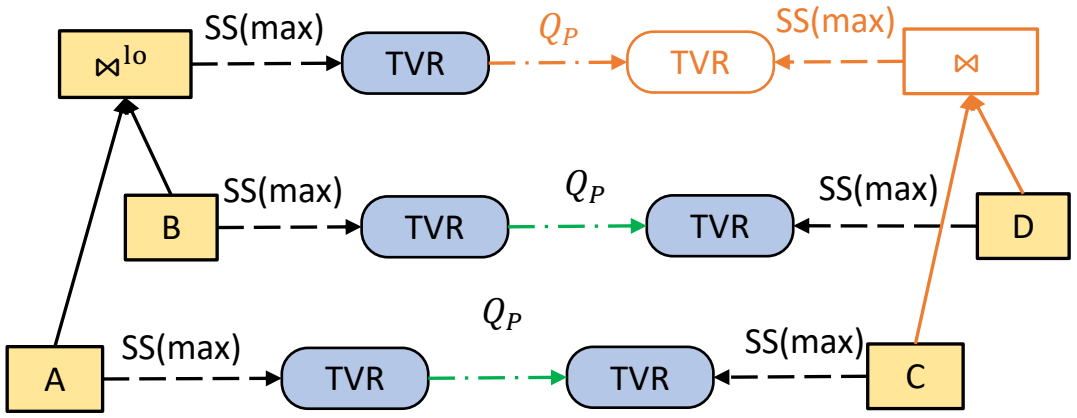
(1) *At deleting* $\Delta S^-, \Delta R^-$ *and inserting* $\Delta S^+, \Delta R^+$:

$$\Delta(S \bowtie^{lo} R)_{t_1}^{t_2} =$$
$$\Delta S^+ \bowtie^{lo} R_{t_2} + S_{t_2} \bowtie \Delta R^+ + (S_{t_1} - \Delta S^-) \bowtie^{ls} (\Delta R^- \bowtie^{la} R_{t_2})$$
$$-\Delta S^- \bowtie^{lo} R_{t_1} - S_{t_1} \bowtie \Delta R^- - (S_{t_1} - \Delta S^-) \bowtie^{ls} (\Delta R^+ \bowtie^{la} R_{t_1})$$

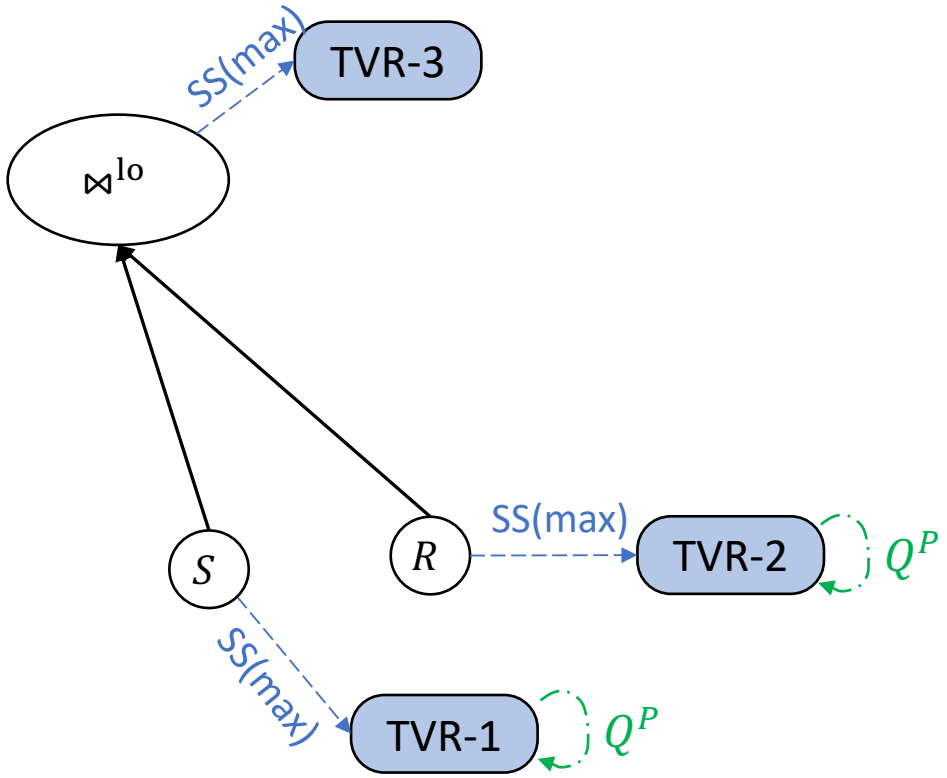


Inter-TVR Rules: Streaming

- Split each TVR into two TVR's:
  - Q^P a non-retractable part that can be outputted over time
  - Q^N the rest part that can only be outputted at the last time point



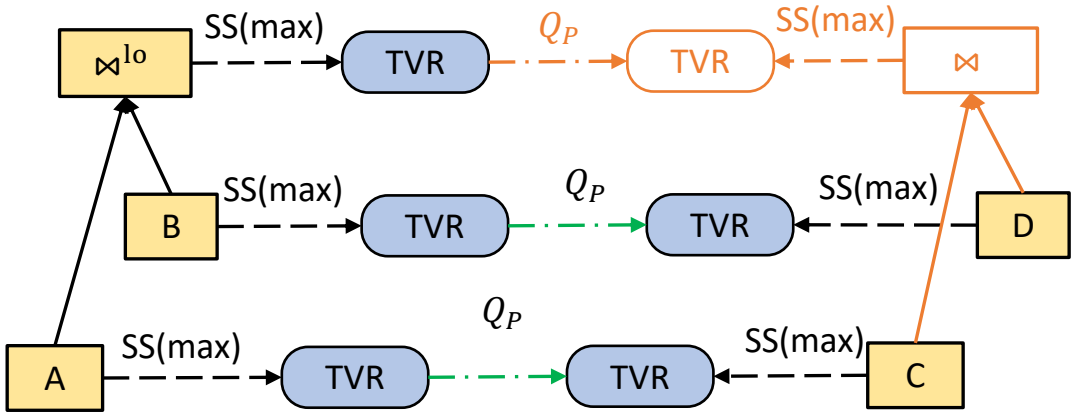
TvrStreamingJoinPropRule (Q^P)



Inter-TVR relationship

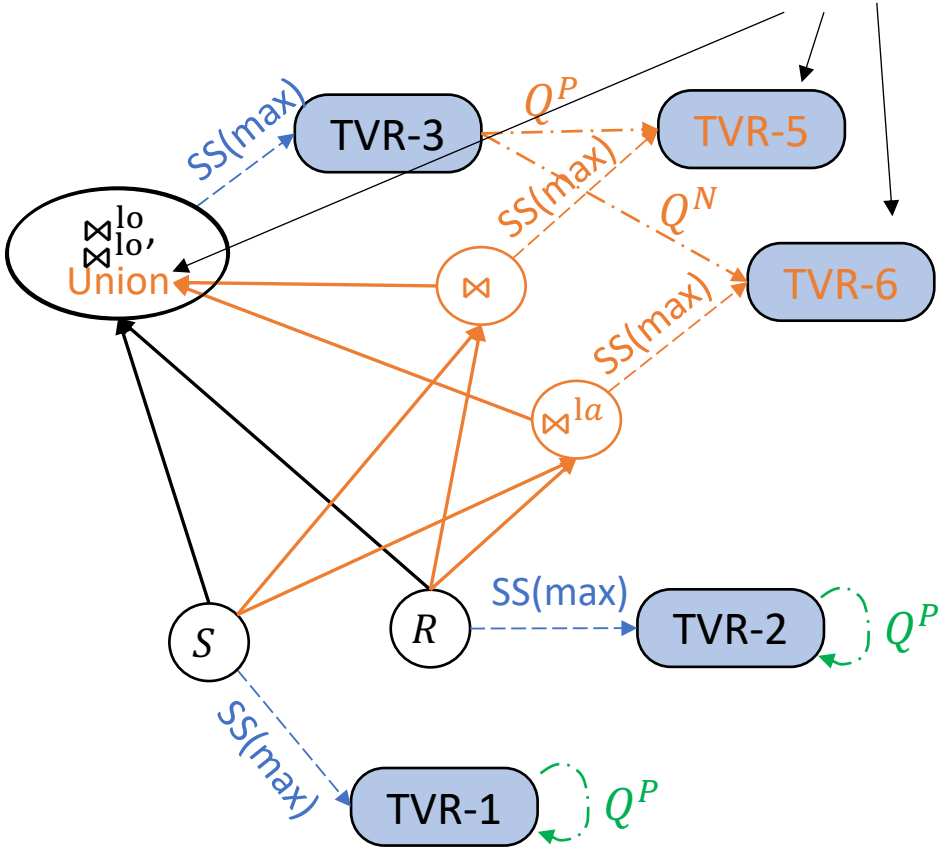
Inter-TVR Rules: Streaming

- Split each TVR into two TVR's:
  - Q^P a non-retractable part that can be outputted over time
  - Q^N the rest part that can only be outputted at the last time point



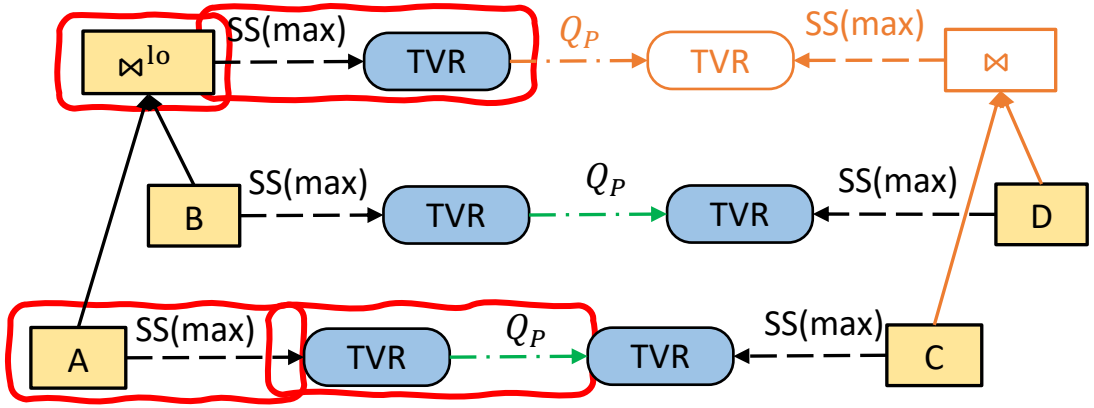
TvrStreamingJoinPropRule (Q^P)

To be further expanded by TVR-generating rules and intra-TVR rules



Tempura Rule Engine

- Operands in rule's match pattern:
  - Operator operand
  - TVR operand
  - Operator edge
  - Intra-TVR edge
  - Inter-TVR edge



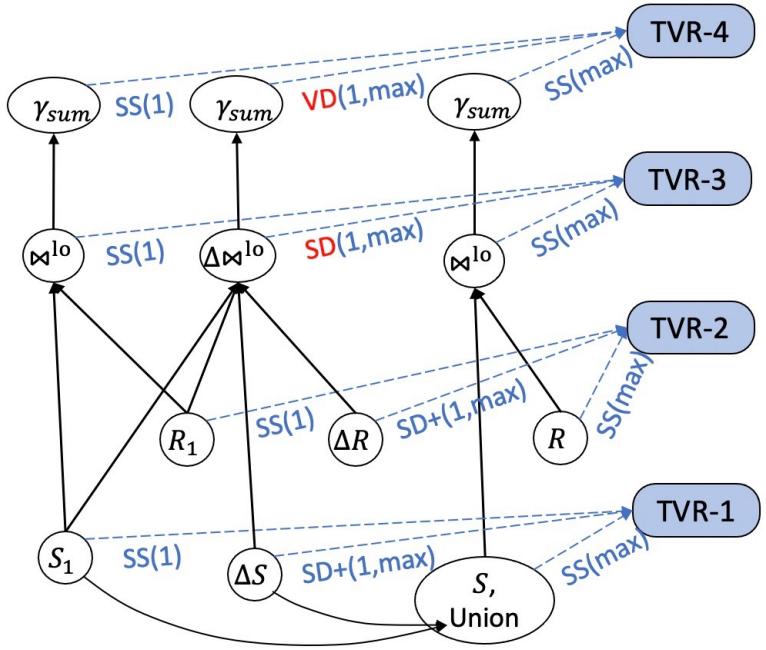
TvrStreamingJoinPropRule (Q^P)

```
private TvrStreamingJoinPropRule() {
  super(operand(OdpsMultiJoin.class, tvrEdgeSSMax(tvr()),
      operand(RelSubset.class, tvrEdgeSSMax(
          tvr(tvrProperty(TvrStreamingPropertyQP.class,
              tvr(tvrEdgeSSMax(logicalSubset()))))), any()),
      operand(RelSubset.class, tvrEdgeSSMax(
          tvr(tvrProperty(TvrStreamingPropertyQP.class,
              tvr(tvrEdgeSSMax(logicalSubset()))))), any()));
}
```

Tempura Rule Engine

- Operands in rule's match pattern:
  - Operator operand
  - TVR operand
  - Operator edge
  - Intra-TVR edge
  - Inter-TVR edge
- Allow rules to register new operators as well as TVR nodes and Intra/Inter-TVR edges

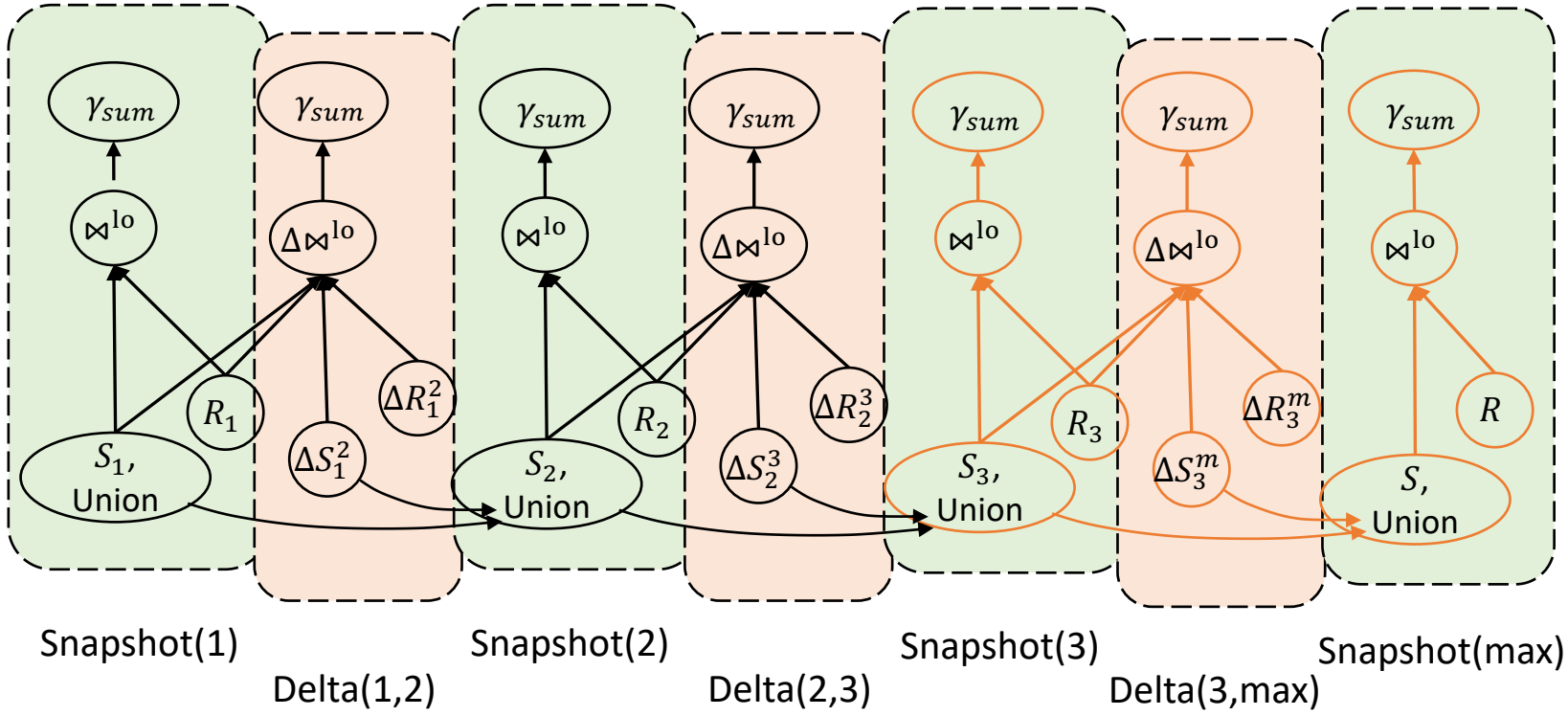
Extensions



- Fully compatible with traditional rules as is
  - For a traditional rule, if all matches rels connects to a TVR (via a SS(max) link), then all newly generated rels will automatically be connected to TVRs

Speed Up with Translational Symmetry

- Similar structure across time
- Repetitive rule match & fire
- Memo copy to speed up

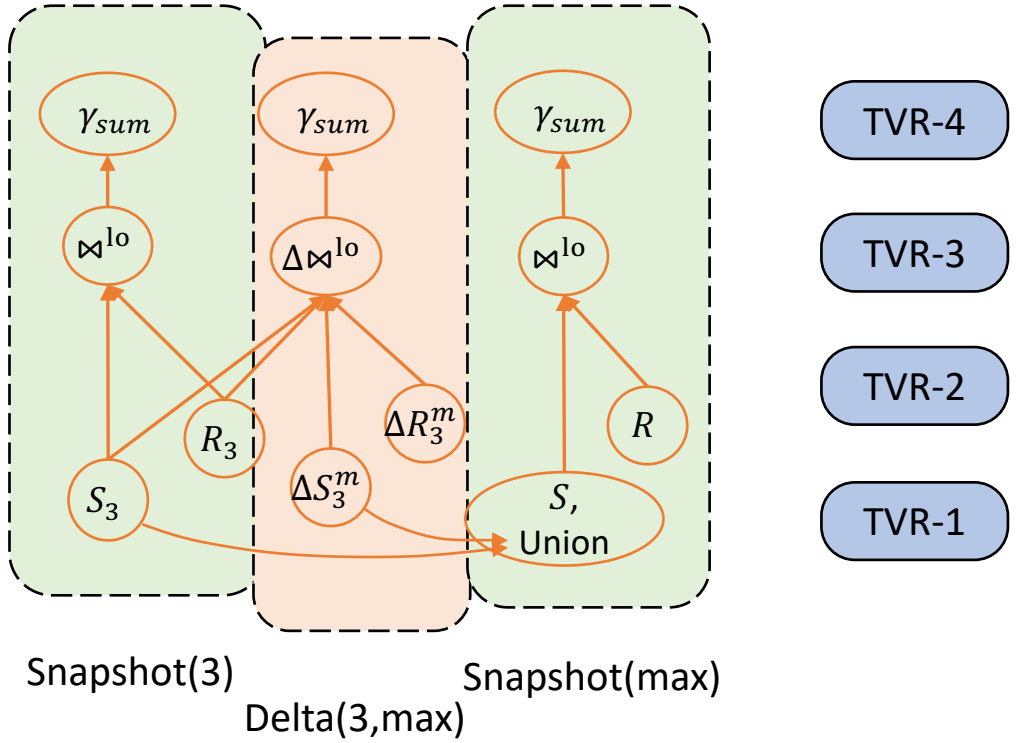


Speed Up with Translational Symmetry

2. Copy with minimal rule matching

1. Fully expand one snapshot and one delta

w. physical rules w/o prune empty rules



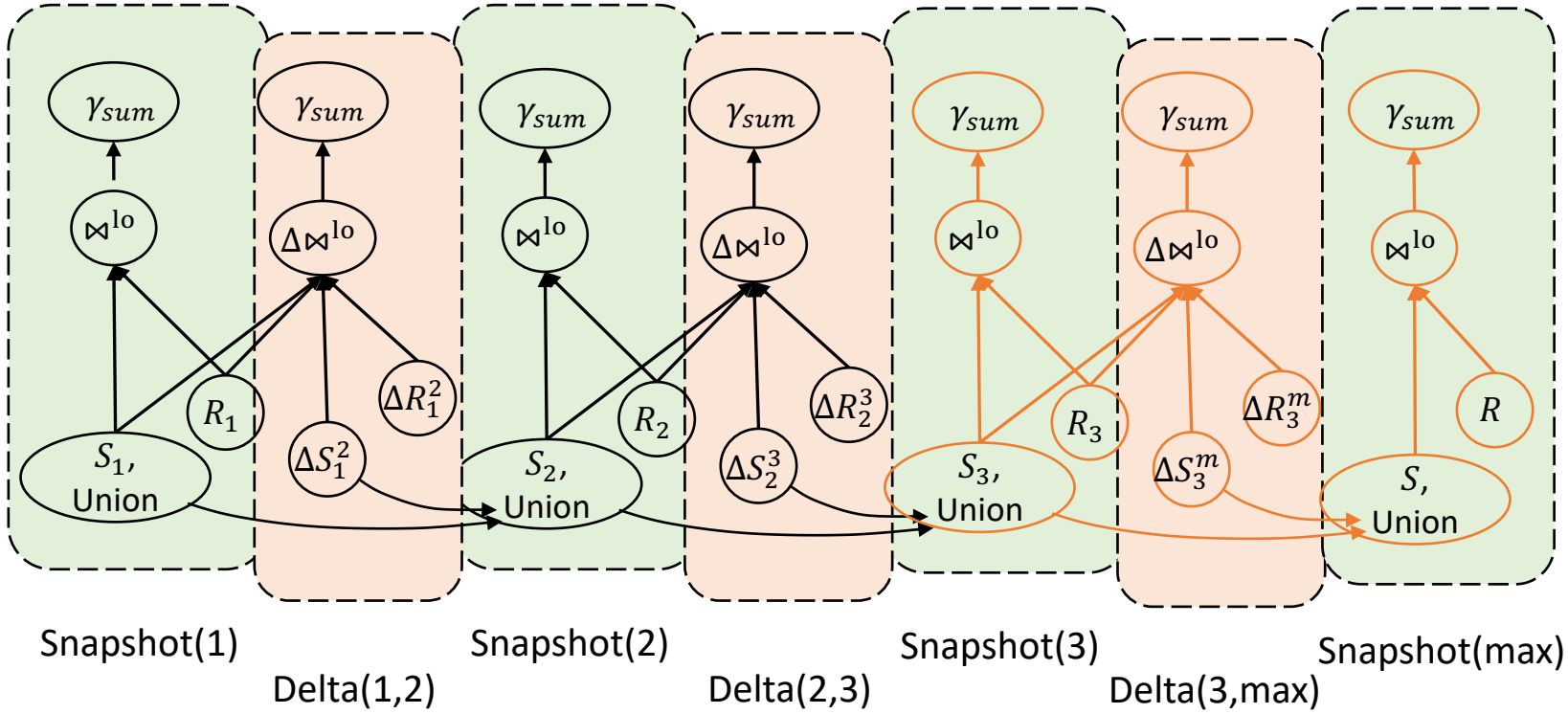
Speed Up with Translational Symmetry

2. Copy with minimal rule matching

3. Additional rule fire (e.g. prune empty)

1. Fully expand one snapshot and one delta

w. physical rules w/o prune empty rules



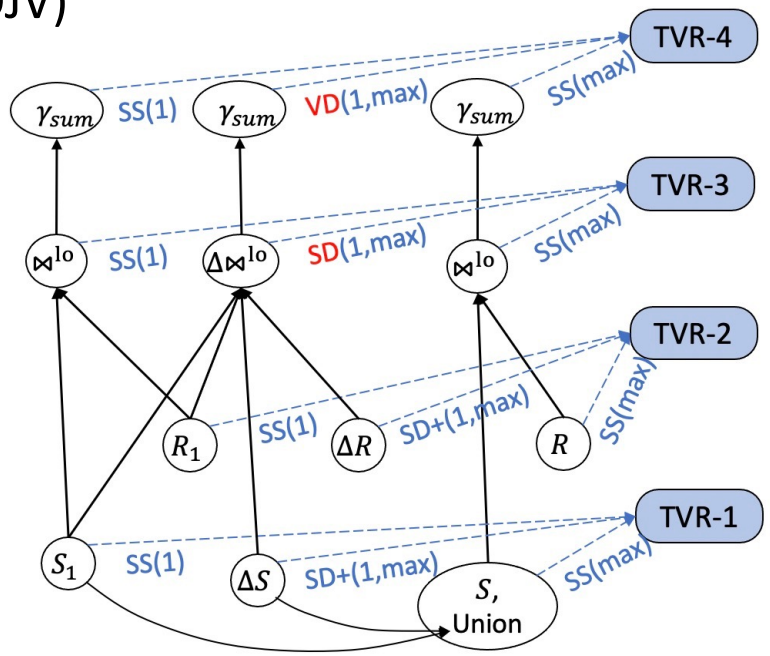
The TIP Model Summary

- TVR Algebra + TVR Rewrite Rules

- Expressiveness
  - Various families of existing incremental methods:
    IVM, Streaming, DBToaster, Outer Join View Maintenance (OJV)
  - Everything in one plan space -> cost based search!

- Building on top of traditional volcano-style optimizer
  - Mark the plan space with time and TVR info
  - Extend rule engine to support TVR Rewrite Rules

- Speeding things up
  - Copy memo along time dimension
  - Left-deep delta merge



Tempura: A General Incremental Query Optimizer

- Applicable to general incremental scenarios

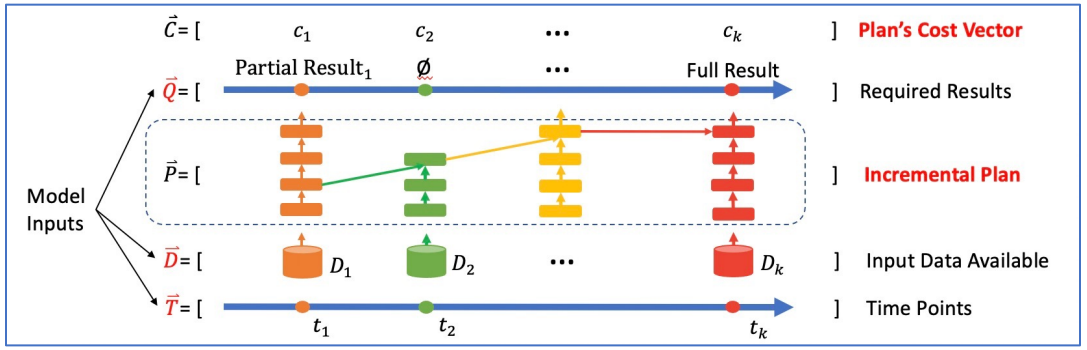
- The *first* volcano-style cost-based optimization framework for IQP
  - *The TIP model* that describes IQP in its most general form
  - Express existing incremental methods in one plan space
  - Cost-based search for best multi-time-point plan
  - Use multi-query optimization technique to materialize states

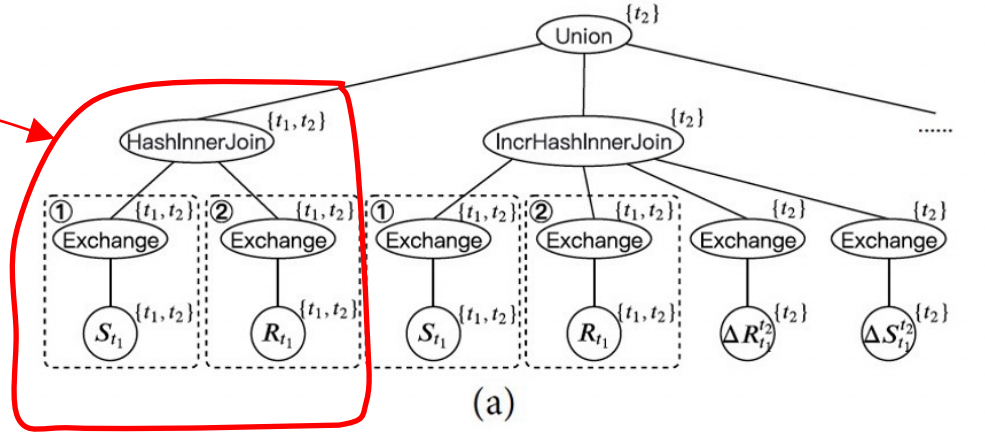
Finding Best Plan in a Space Involving Time

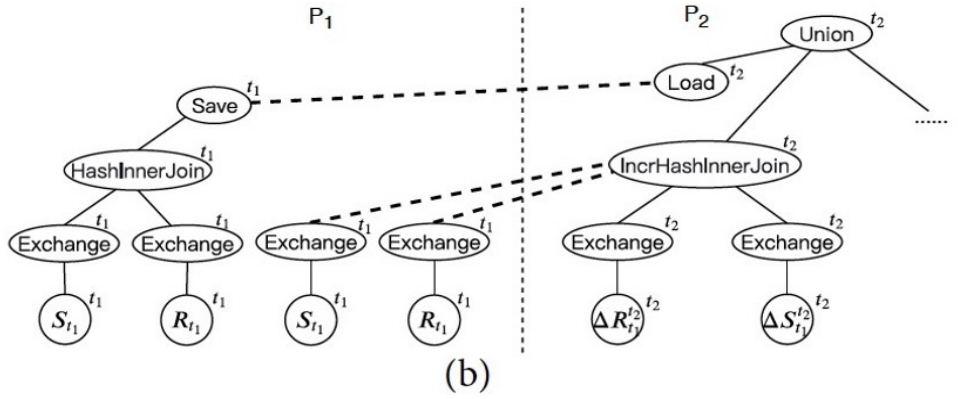
- Execution time assignment e.g. $\{t_1, t_2\}$
- Cost at different times

To use this at t_2, we have two options:
Assume cost of compute = 10, save = 5, load = 4
(1) Compute and save at t_1, load at t_2. Cost = [15, 4]
(2) Compute at t_2. Cost = [0, 10]

$$Cost\ Function\colon\ \tilde{c}_w(\vec{C}) = \sum_{i=1}^{k} w_i c_i$$



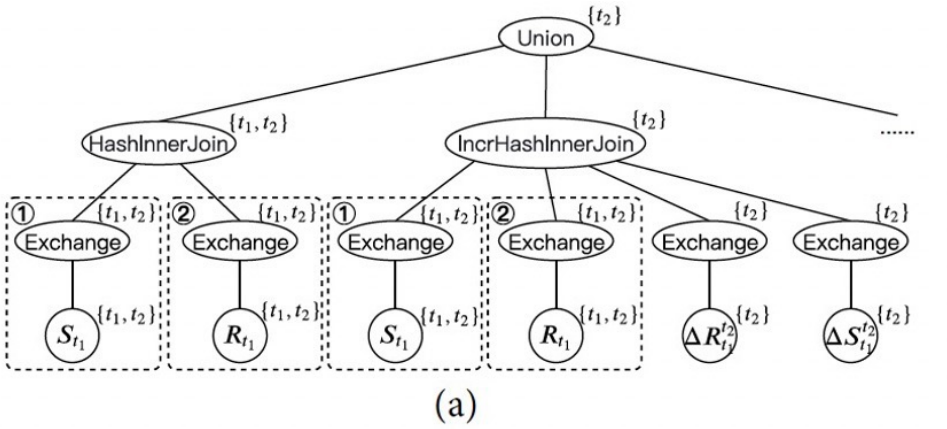




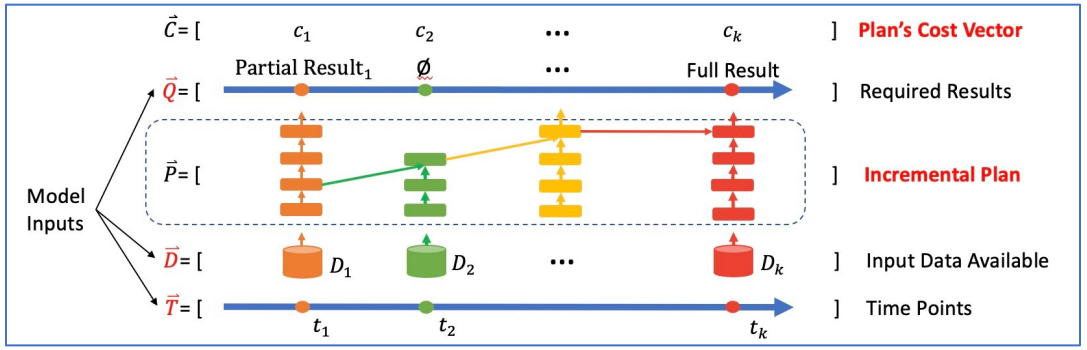
Finding Best Plan in a Space Involving Time

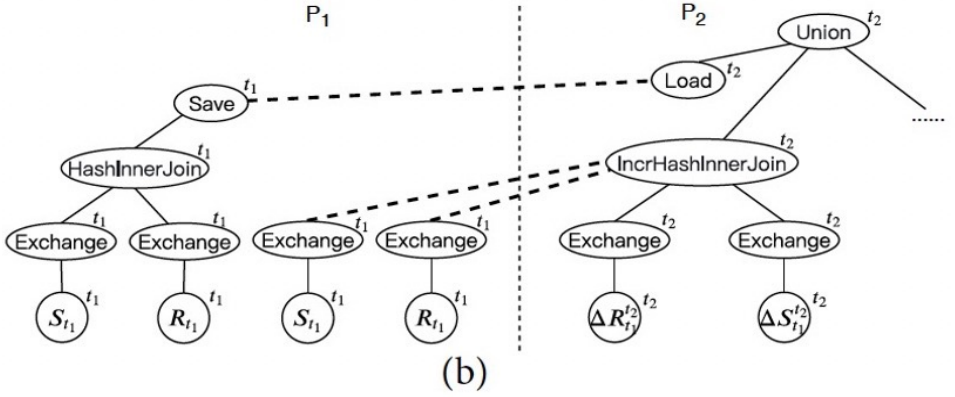
- Execution time assignment
- Cost at different times
- DP without sharing sub-plan
  - DP only works for certain $\tilde{c}(\vec{C})$

$\tilde{c}_w(\vec{C}) = \sum_{i=1}^{k} w_i c_i$ *YES* $\tilde{c}_w(\vec{C}) = \prod_{i=1}^{k} c_i$ *NO*



(a)

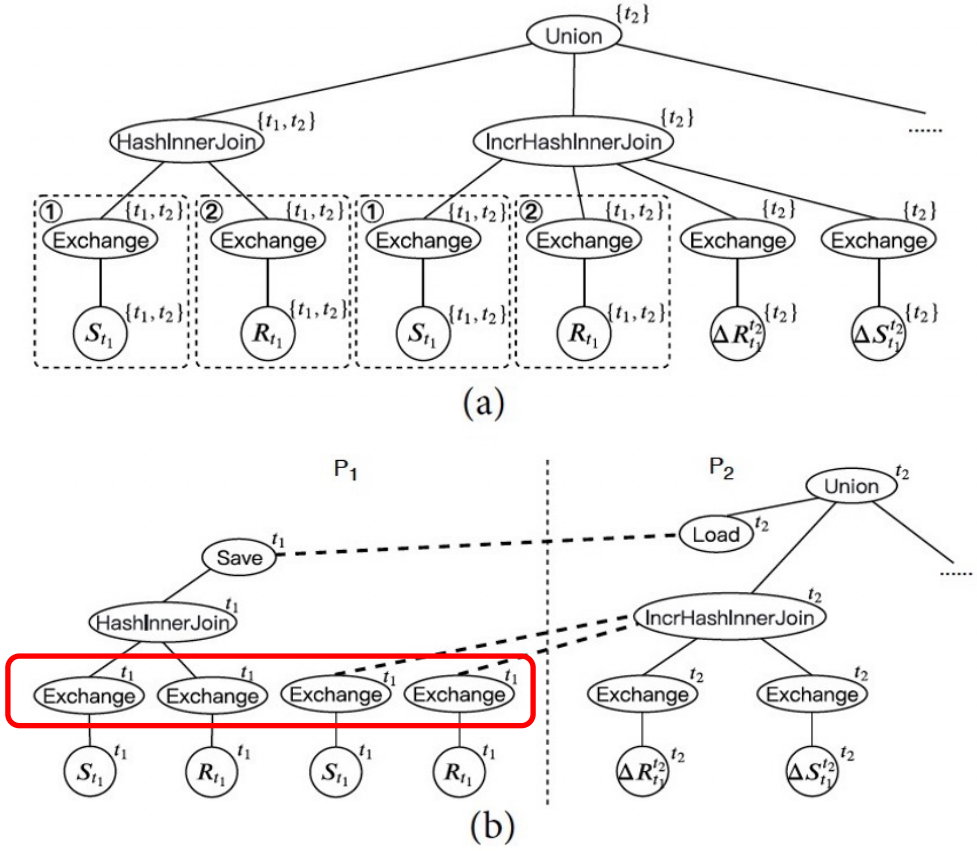




(b)

Finding Best Plan in a Space Involving Time

- Execution time assignment
- Cost at different times
- DP without sharing sub-plan
- DP with sharing sub-plans:
  - Multi-query optimization

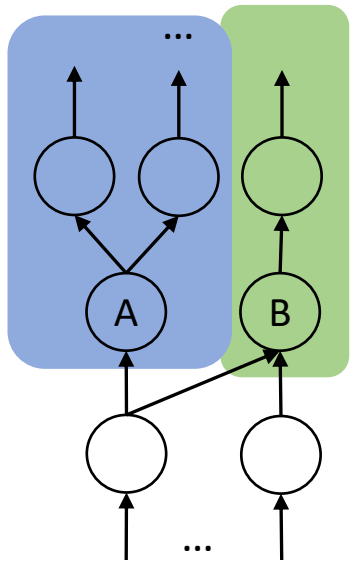


Greedy MQO

- Candidate: (subset, exec time) pairs
- Back and forth cost propagations
  - What if we materialize subset A?
  - What if we materialize subset B?
- Transaction map: subset → best
  - Staging, rollback, commit
  - Quick switch between versions

Algorithm 1 Greedy Algorithm for MQO

```
1: 𝕊 = ∅
2: ℂ = shareable candidate set consiting of all shareable nodes and their
   potential execution times {⟨s, τ(s)⟩}
3: while ℂ ≠ ∅ do
4:     Pick ⟨s, τ(s)⟩ ∈ ℂ that minimizes c̃(bestPlan(𝕊′)) where 𝕊′ =
       {⟨s, τ(s)⟩} ∪ 𝕊
5:     if c̃(bestPlan(𝕊′)) < c̃(bestPlan(𝕊)) then
6:         ℂ = ℂ − {⟨s, τ(s)⟩}
7:         𝕊 = 𝕊′
8:     else
9:         ℂ = ∅
10:    end if
11: end while
12: return 𝕊
```

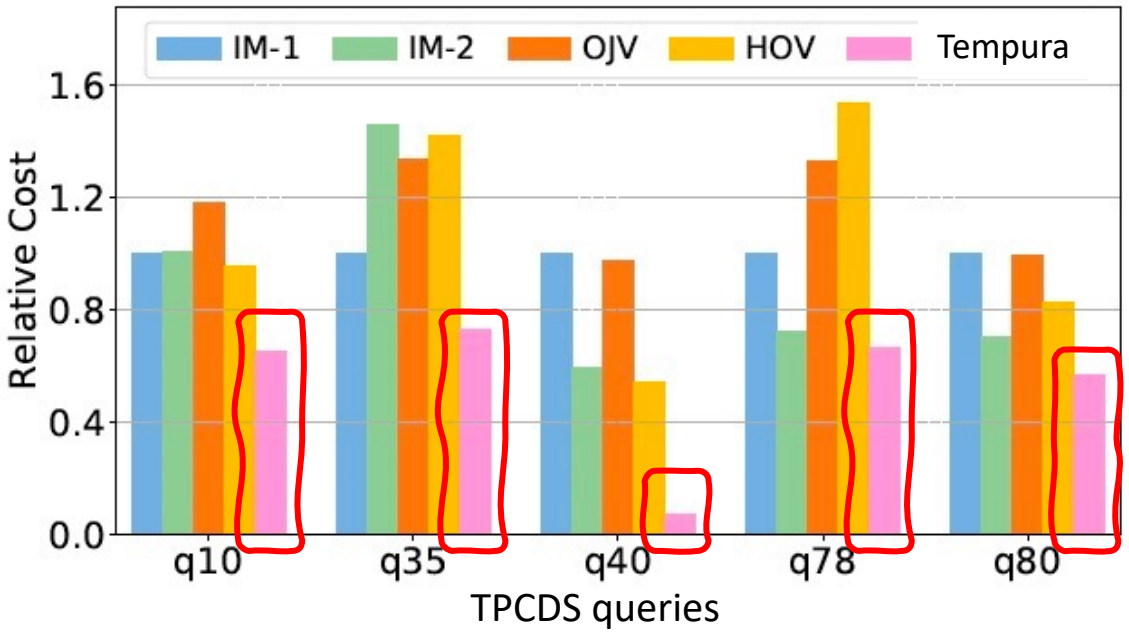


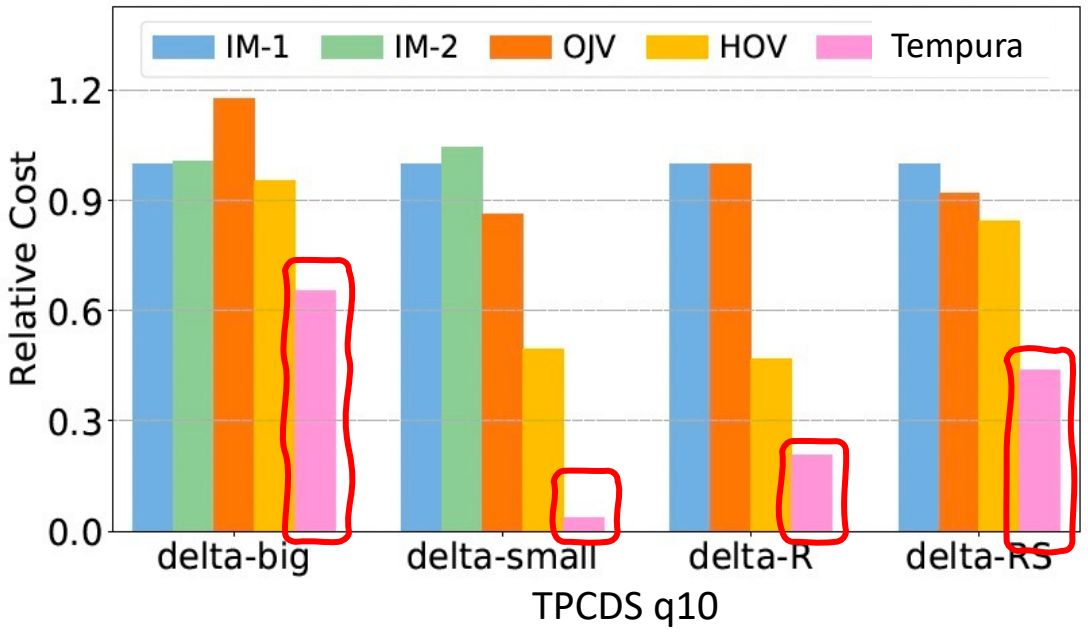
Cost propagations in the memo

Effectiveness Study of Tempura

- Guaranteed optimal plans
- Combined benefits of multiple incremental methods
- MQO's smart decisions

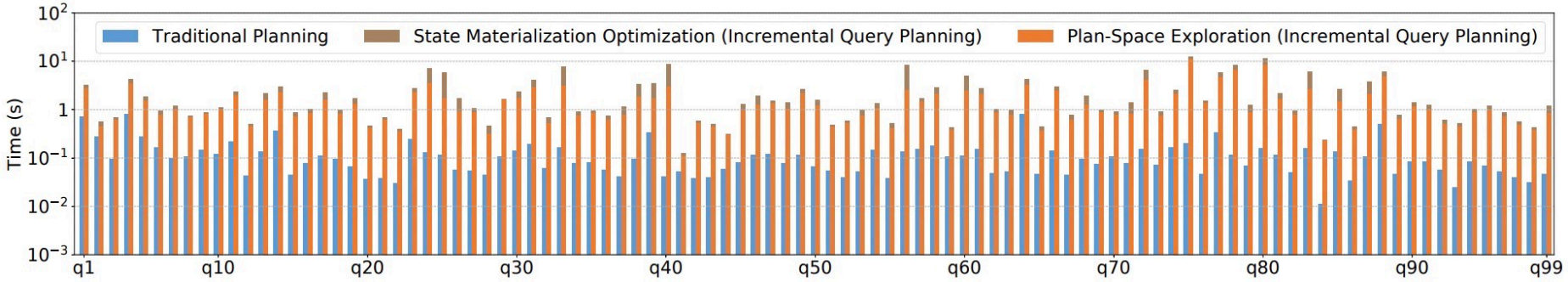
IM-1: Standard IVM
IM-2: Streaming
OJV: Outer join view maintenance
HOV: DBToaster
Tempura: all above combined



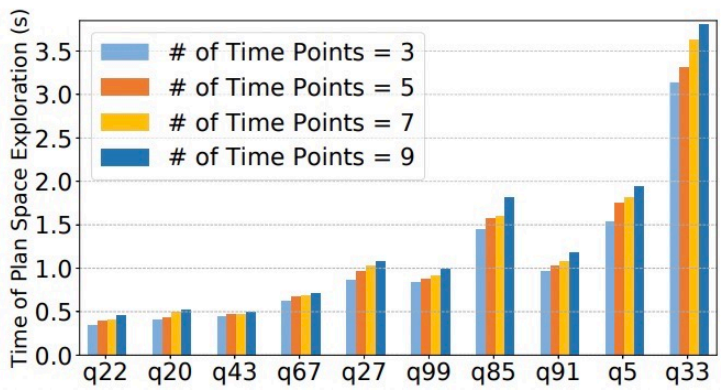


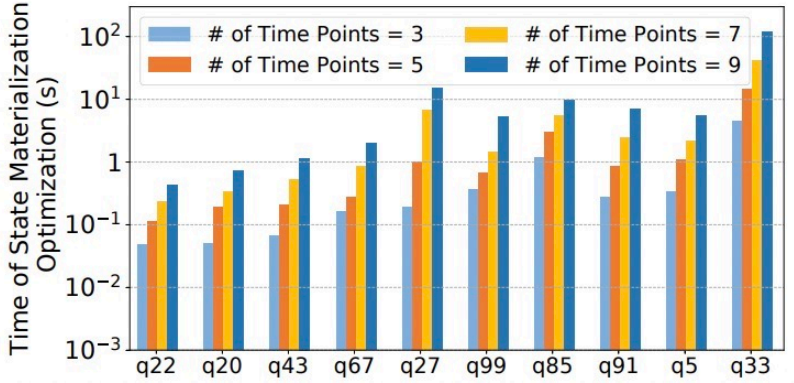
Optimization Time

- Traditional vs 3-time-point Tempura



- Plan Exploration vs MQO
  - Memo copy





Summary of Tempura

- Applicable to general incremental scenarios

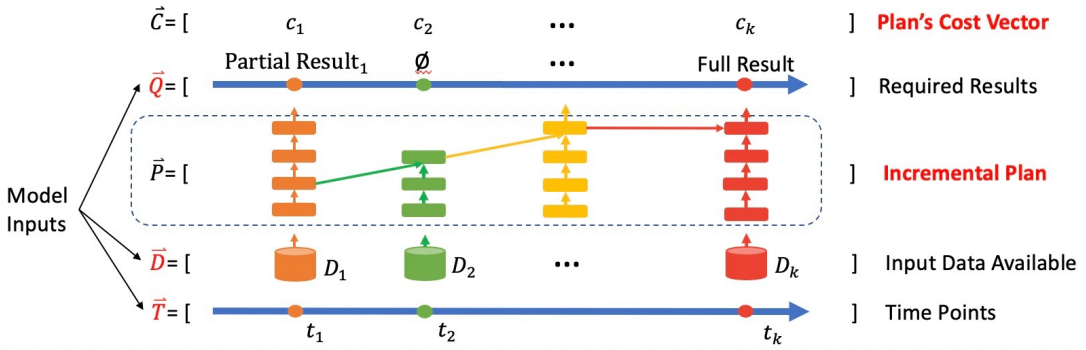
- The *first* volcano-style cost-based optimization framework for IQP
  - *The TIP model* that describes IQP in its most general form
  - Explore various incremental methods in one plan space
  - Cost-based search for best multi-time-point plan
  - Use multi-query optimization technique to materialize states

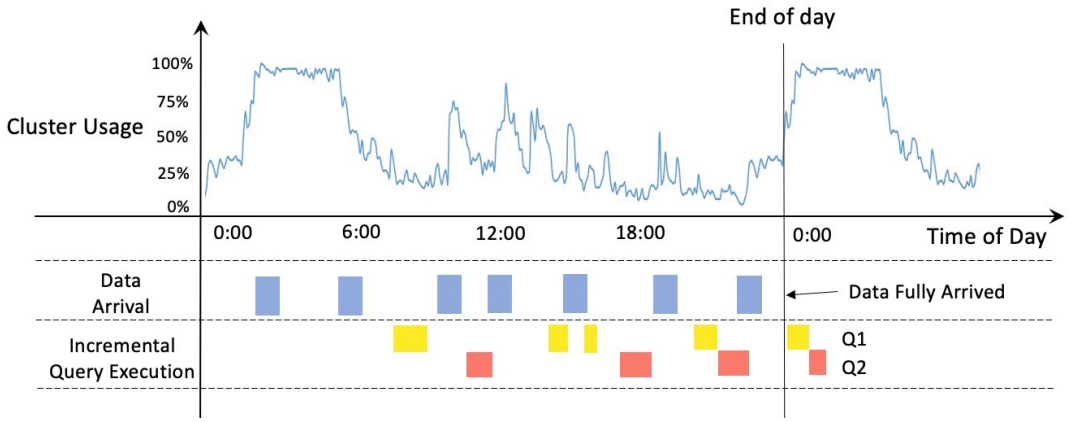
Thanks!
Q&A

Tempura Continued: Dynamic Re-optimization

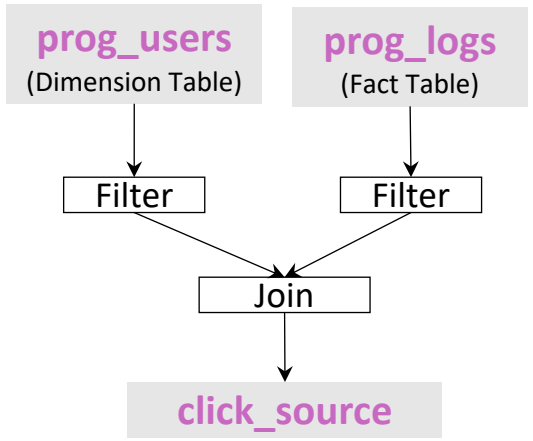
- Problem: hard to decide future time points beforehand
- Idea: add time point on the fly?
- Solution: dynamic re-optimization
  - Old plan $\vec{T}$= [t_1, … , t_{i-1}, t_i,…, t_k]

    $\vec{P}$= [P_1, … , P_{i-1}, P_i,…, P_k]
  - P_{i-1} has finished execution
  - Change future schedule to $\vec{T}'$= [$t_{i'}$,…, $t_{k'}$]
  - Generate a new plan $\vec{P}'$= [$P_{i'}$, … , $P_{k'}$], utilizing the states saved at [P_1, … , P_{i-1}]





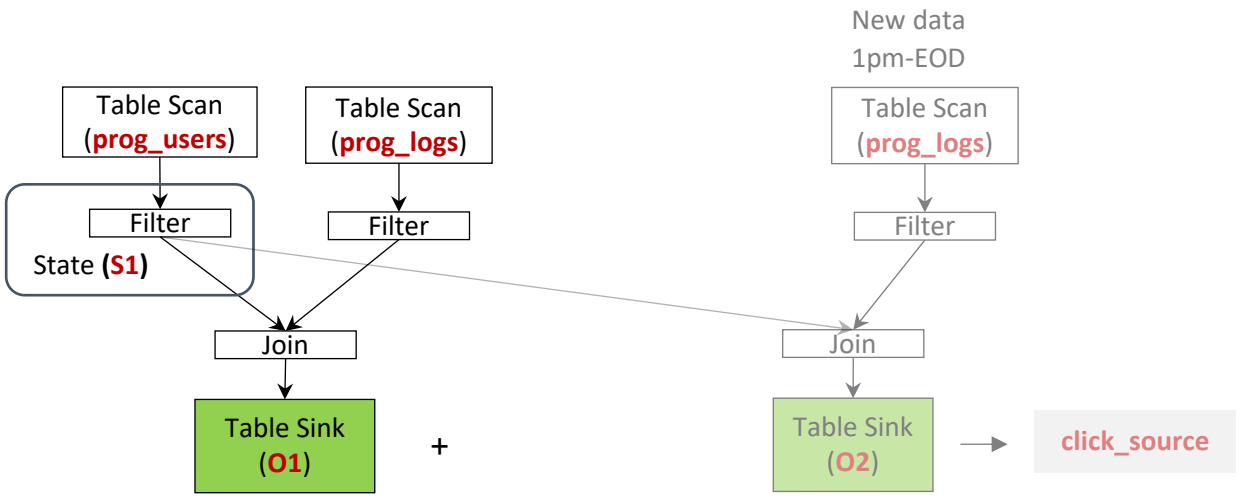
Dynamic Re-Optimization: Example



Plan generated at 10am

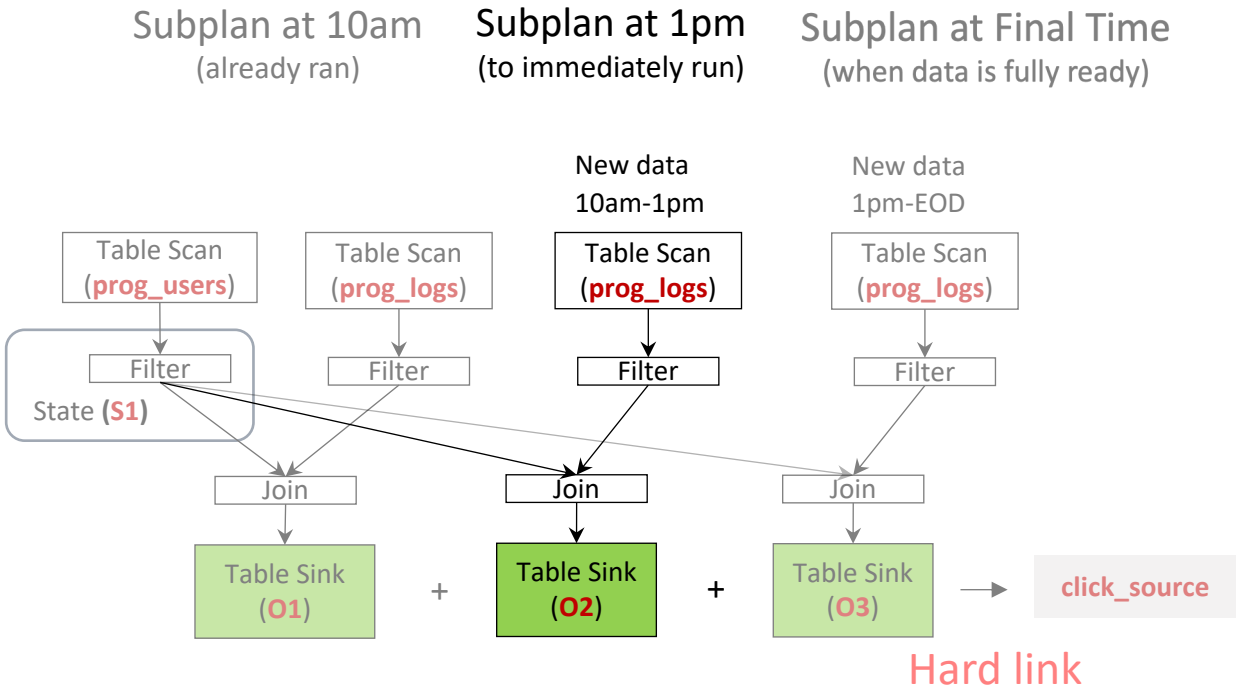
Subplan at 10am
(to immediately run)

Subplan at Final Time
(when data is fully ready)



Dynamic Re-Optimization: Example

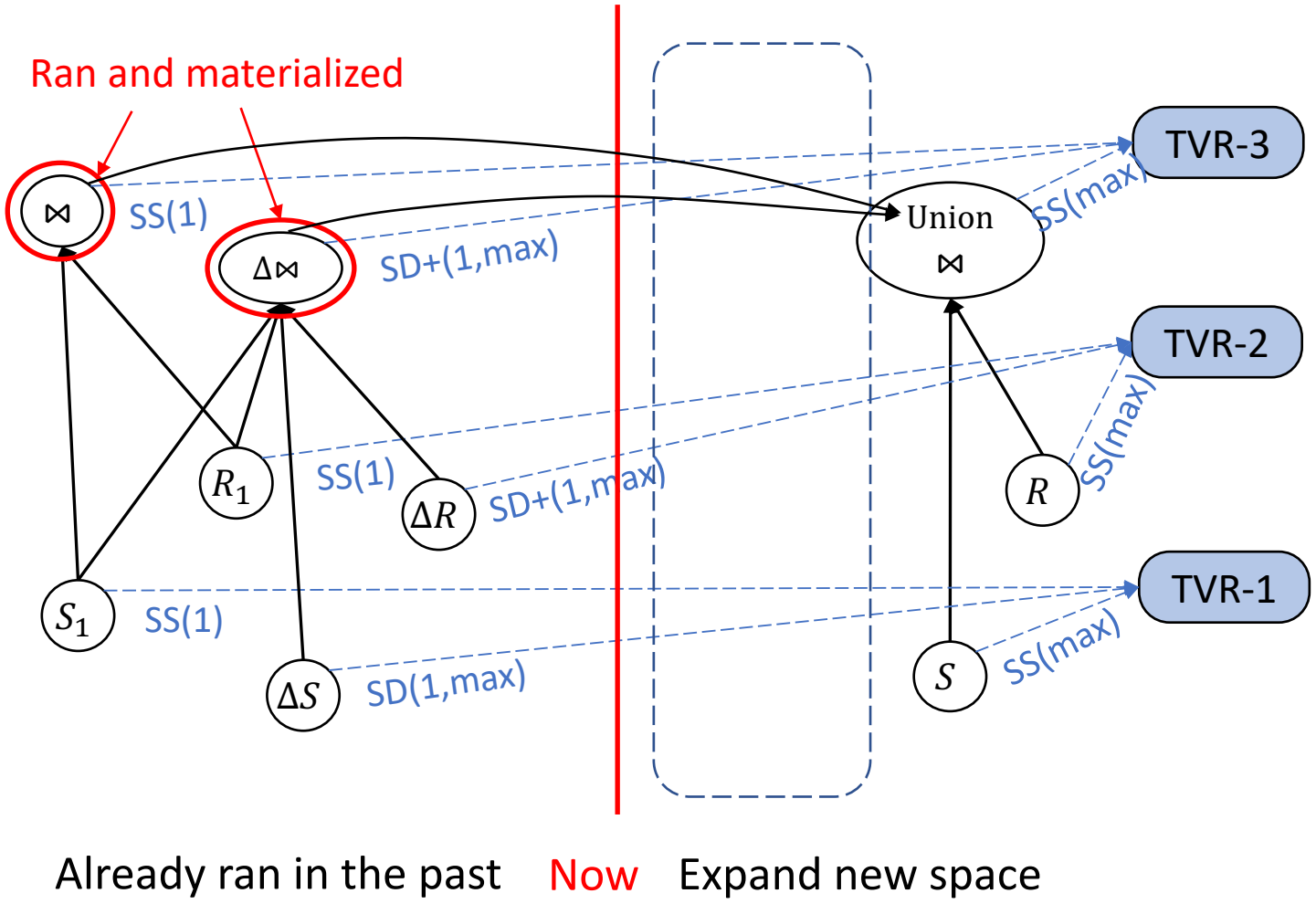
Reoptimized
Plan at 1pm



Re-optimization: Memo's Perspective

- Further expand the old memo with new time points
- Treat results saved earlier as existing view

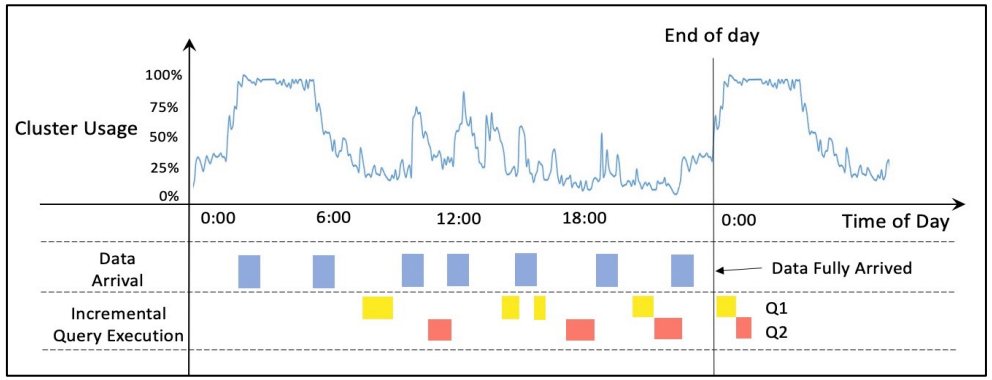
In re-optimization, latest (real) data stats rather than estimated ones will be used



Early Results for Downstream Query Planning

Q2:
select * from ***T***
where id > 10

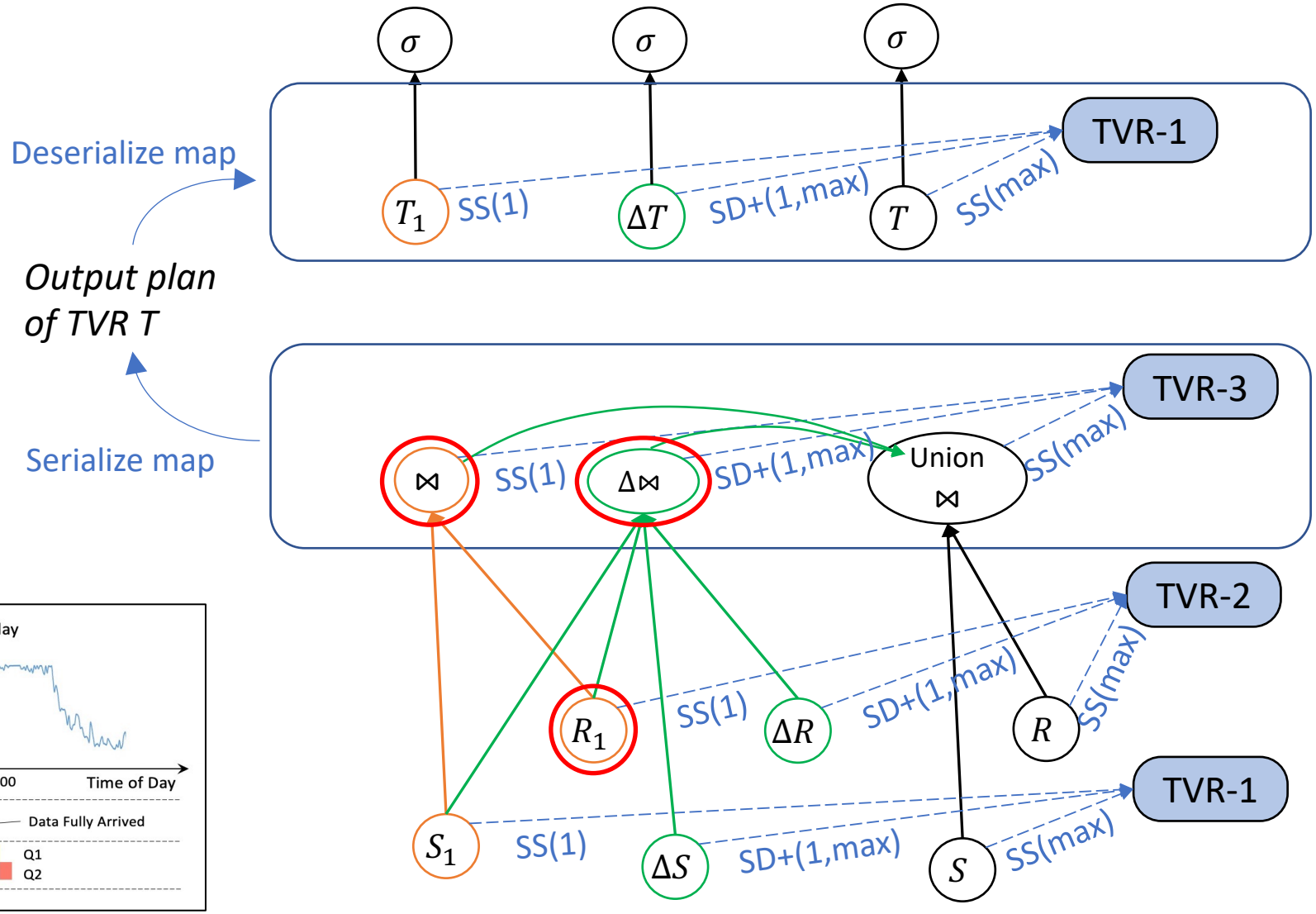
Q1: $T = S \bowtie R$



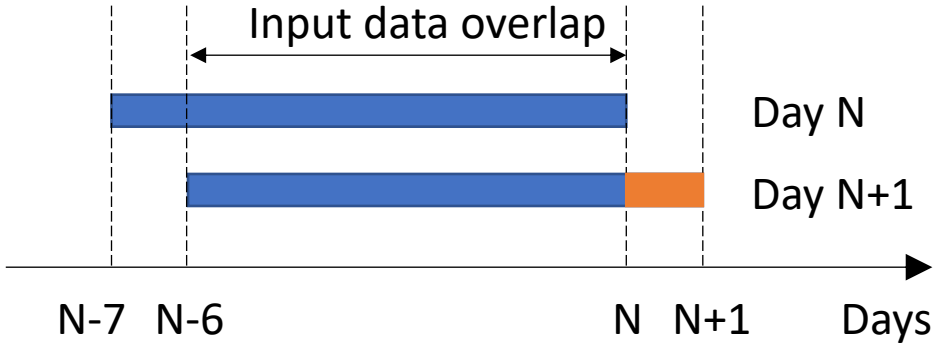
Deserialize map

Output plan of TVR T

Serialize map



Range Query



- Periodic batch job that process data of last few days
- Plan as if we are running progressively, one incremental run per day

Already computed and saved in yesterday's query instance