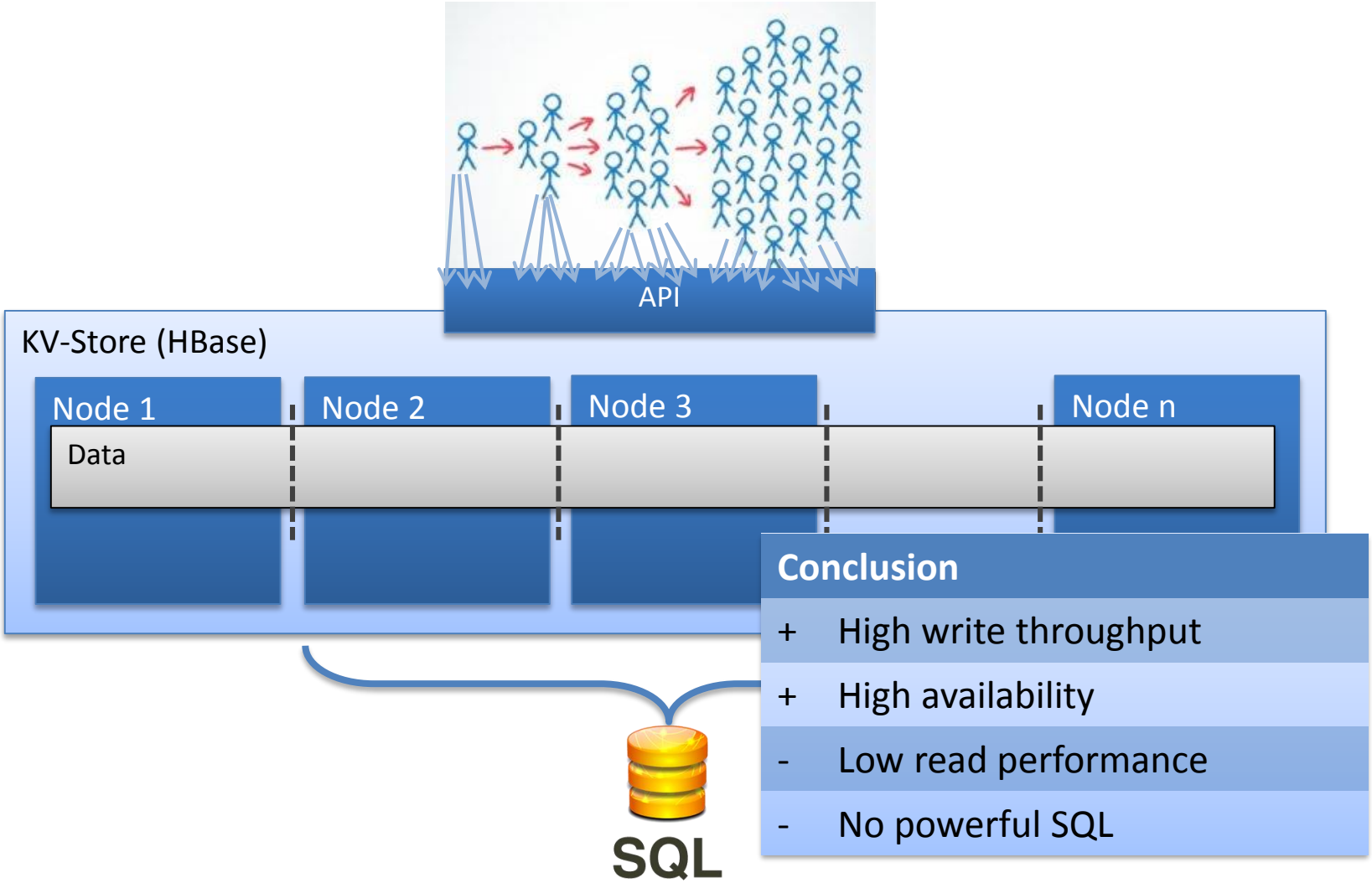
Dynamic Scalable View Maintenance

Jan Adler, Martin Jergler, Hans-Arno Jacobsen

Application & Middleware Systems Research Group
Prof. Hans-Arno Jacobsen

KV-Stores



Conclusion	
+	High write throughput
+	High availability
-	Low read performance
-	No powerful SQL

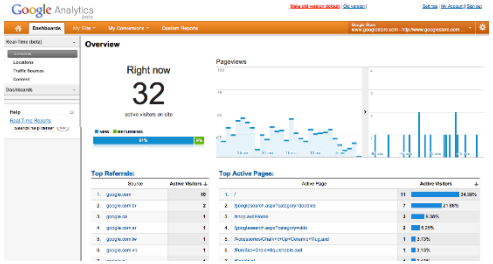
Known approaches

- External computation → load overhead
  - Parallel / from disk (Map Reduce, Hive, Pig)
  - Parallel / in memory (Spark)
- Internal computation → implementation bound
  - Parallel / server-side (Coprocessors, Apache Phoenix)

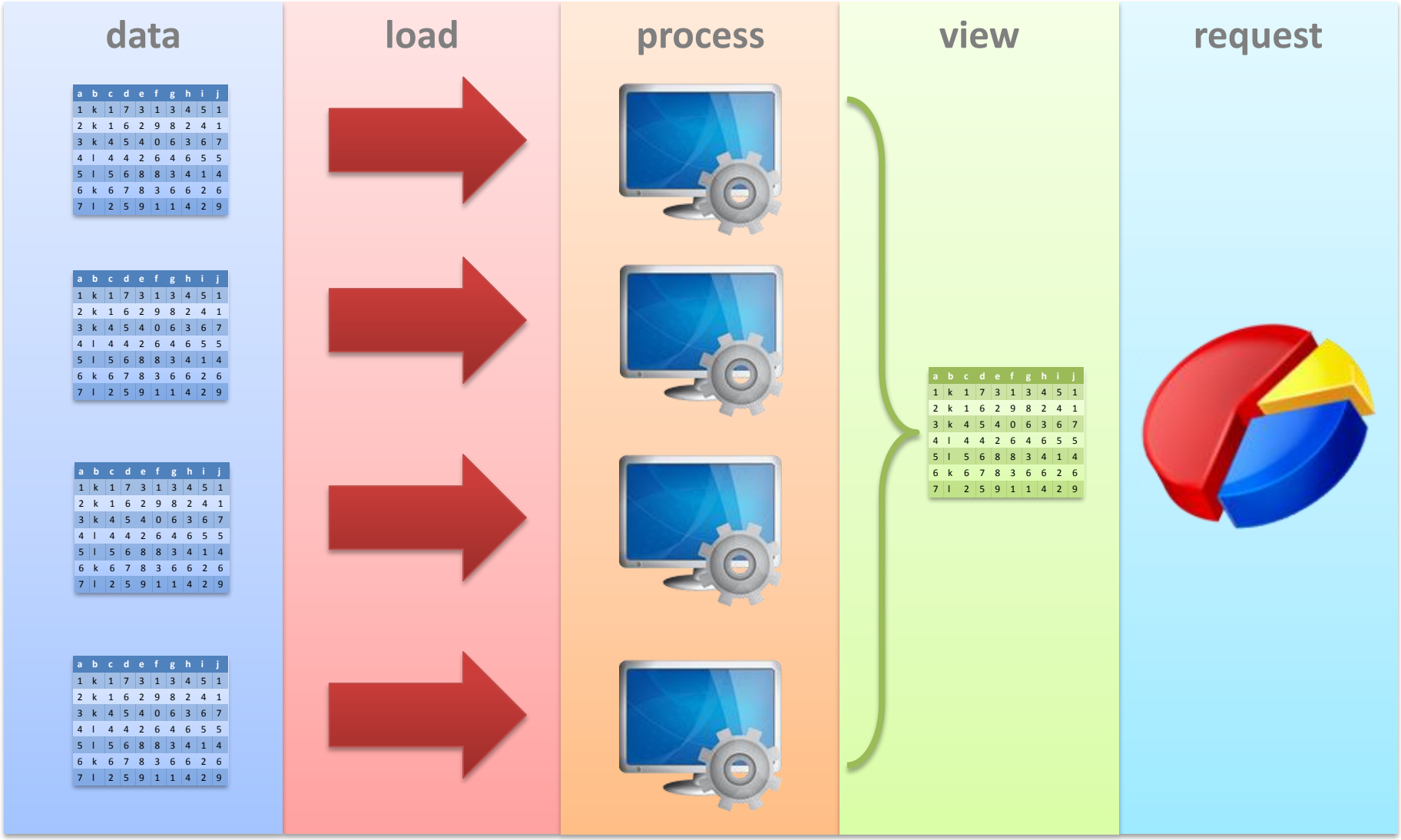
Our approach

- Combine large-scale **data storage** and **materialized views**
  - Views…
    - …bring back SQL-semantics
    - …are highly available through materialization
    - …can be maintained efficiently
  - View maintenance should be…
    - …universal
    - …scalable (exactly as the KV-Store)
    - …consistent
    - …and fault tolerant

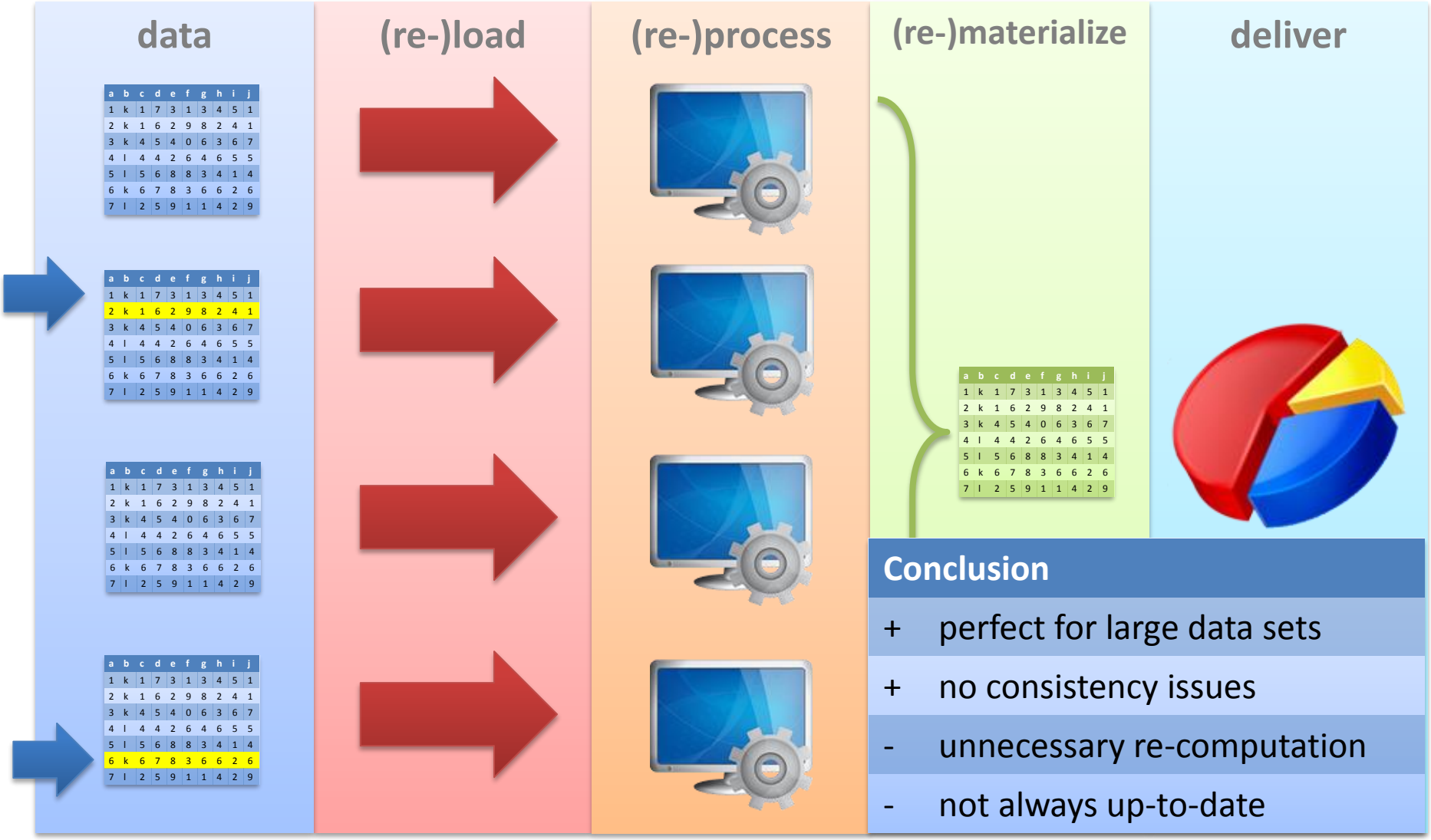
```
<aggregationView>
    <name>sum1</name>
    <type>sum</type>
    <basetable>bt1</basetable>
    <aggregationKey>aggKey</aggregationKey>
    <aggregationValue>aggVal</aggregationValue>
    <numOfRegions>2</numOfRegions>
    <controlTable>true</controlTable>
</aggregationView>
```



Batch processing



Batch processing



a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

Conclusion

+ perfect for large data sets
+ no consistency issues
- unnecessary re-computation
- not always up-to-date

Incremental processing

data

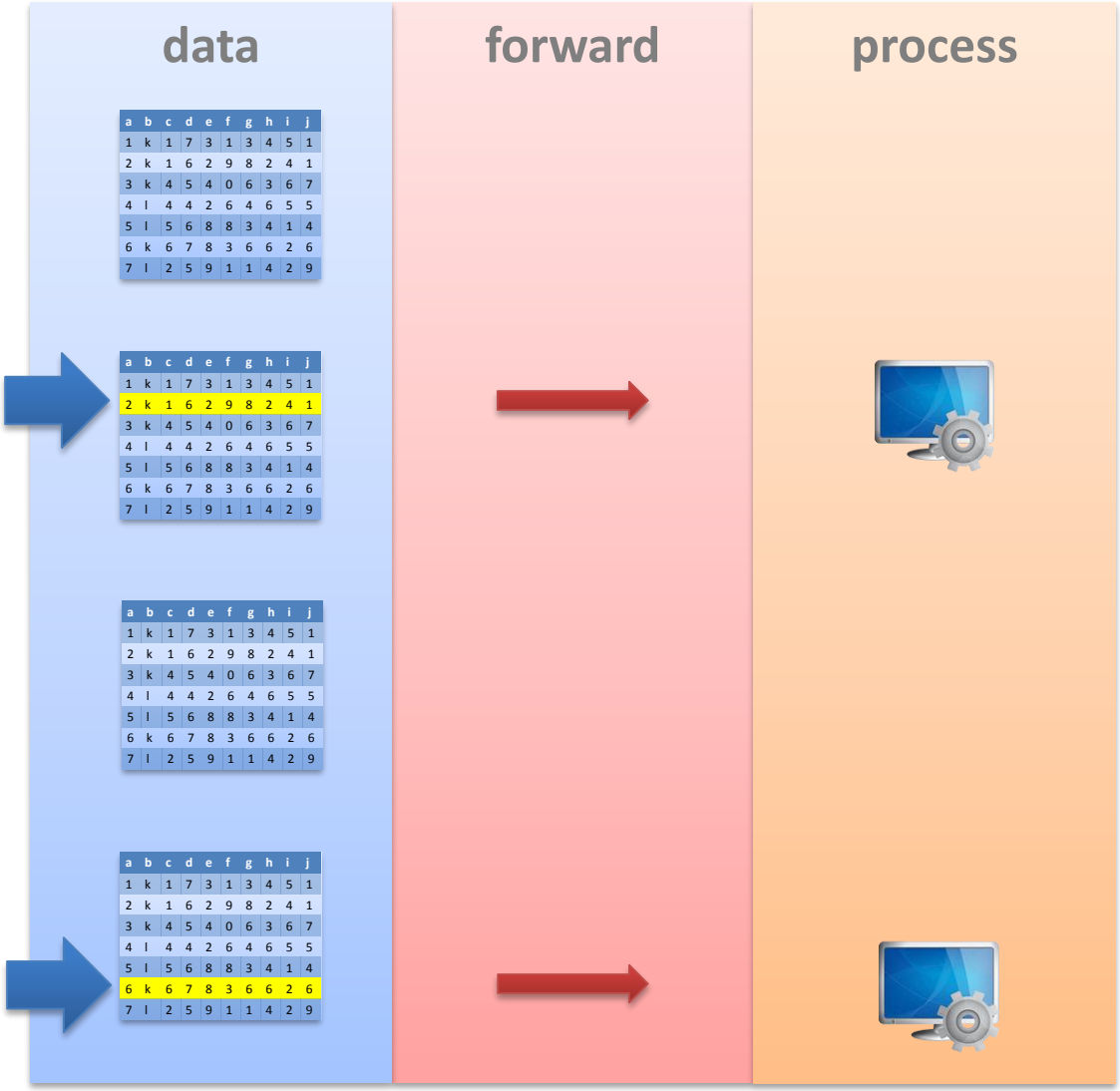
a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

Incremental processing



data

forward

process

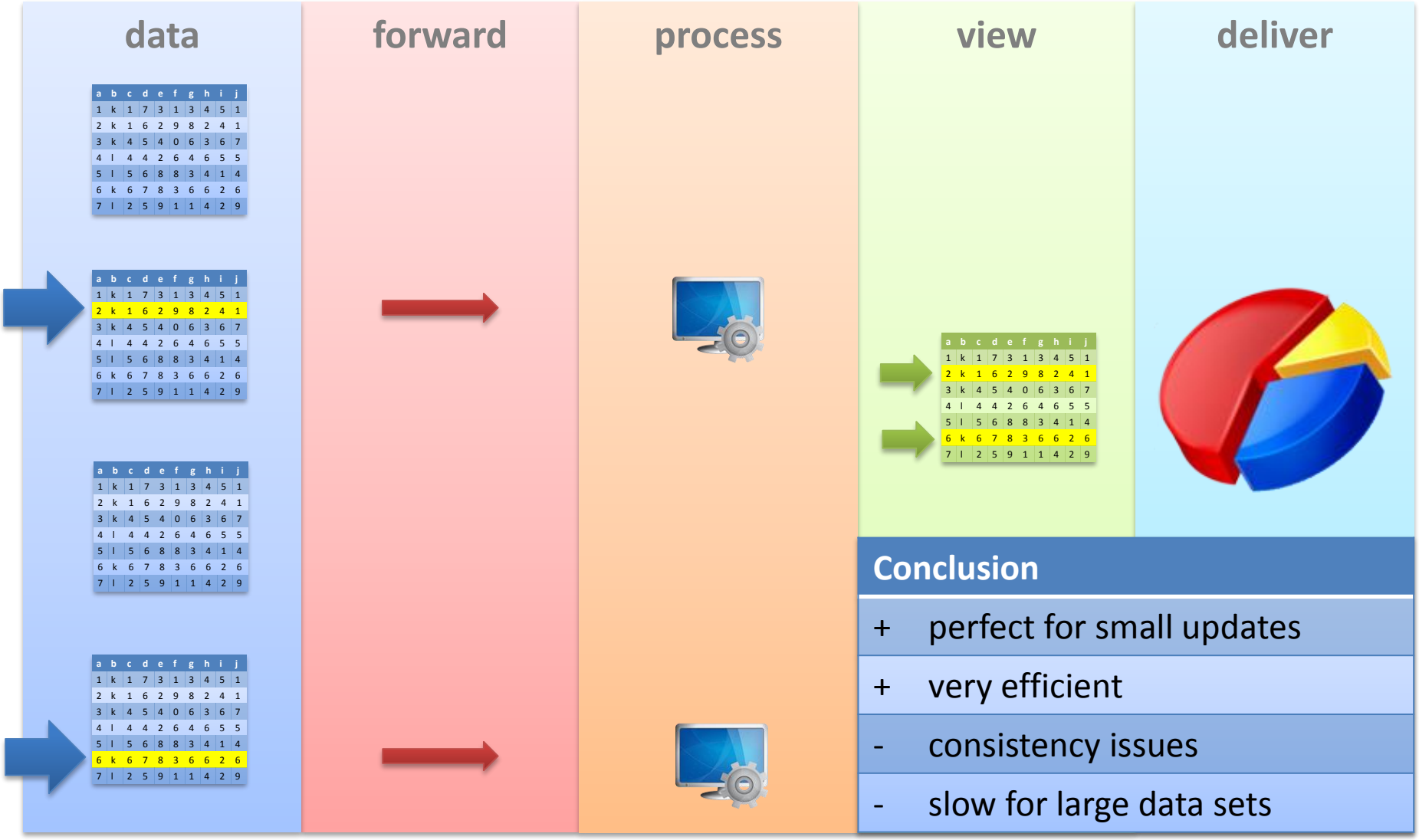
a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

a	b	c	d	e	f	g	h	i	j
1	k	1	7	3	1	3	4	5	1
2	k	1	6	2	9	8	2	4	1
3	k	4	5	4	0	6	3	6	7
4	l	4	4	2	6	4	6	5	5
5	l	5	6	8	8	3	4	1	4
6	k	6	7	8	3	6	6	2	6
7	l	2	5	9	1	1	4	2	9

Incremental processing



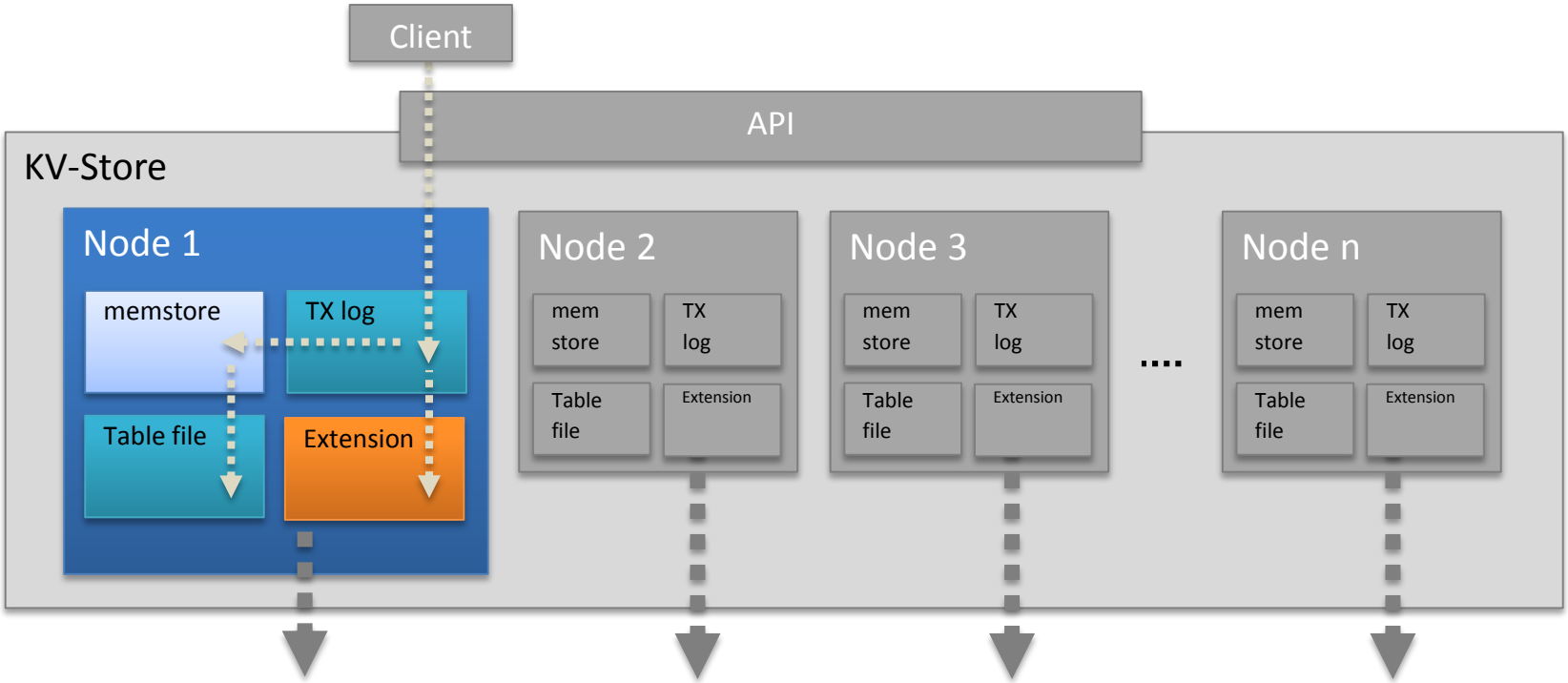
Integration batch + incremental

- Why not combine both approaches?
  1. new query is requested
  2. process batch and **compute** result…
  3. …**switch** and **maintain** result incrementally

Integration batch + incremental

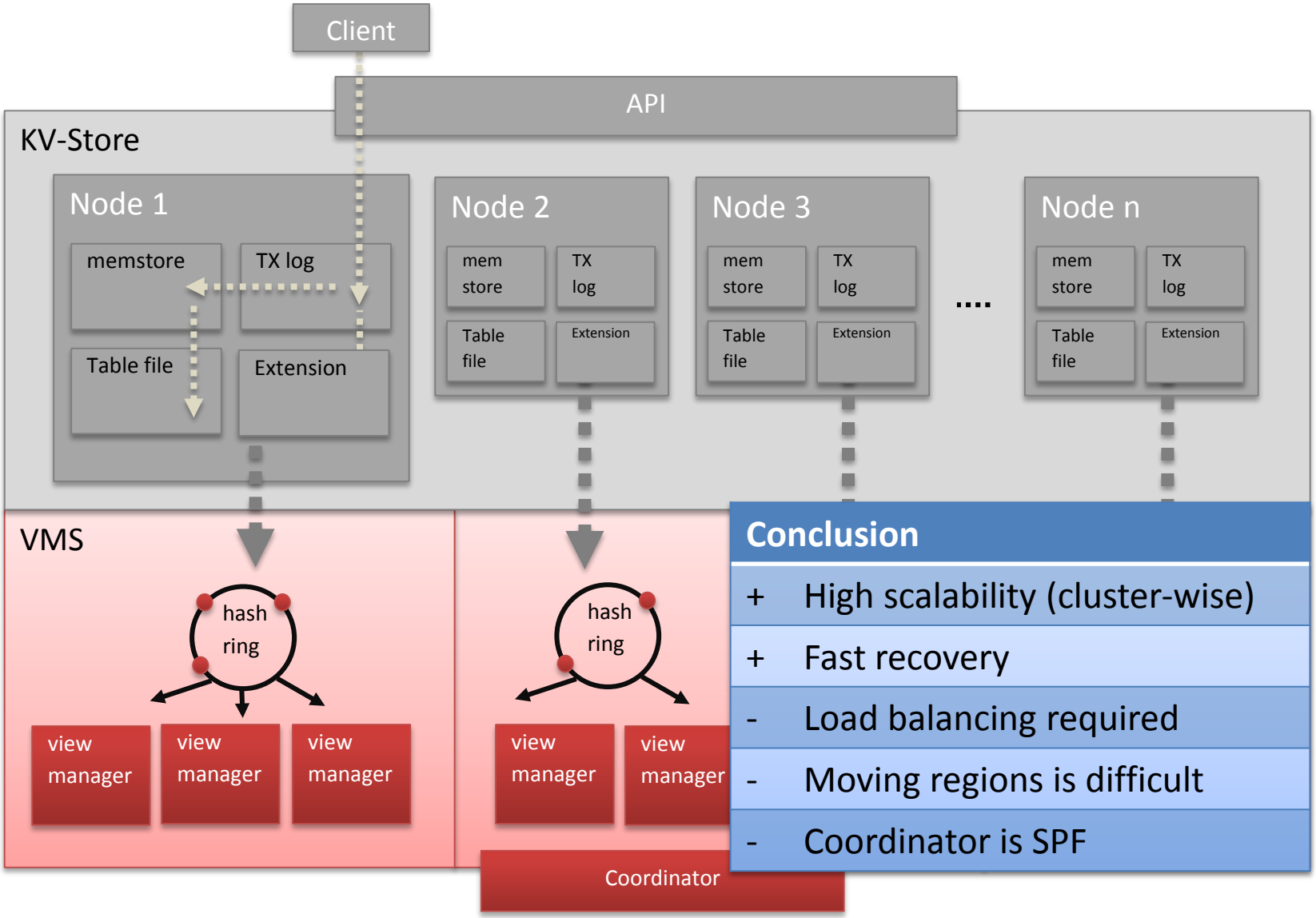
- Problems to solve
  - set-up table structure for incremental processing
  - ensure consistency during switch operation
    → signatures

Generality of KV-model



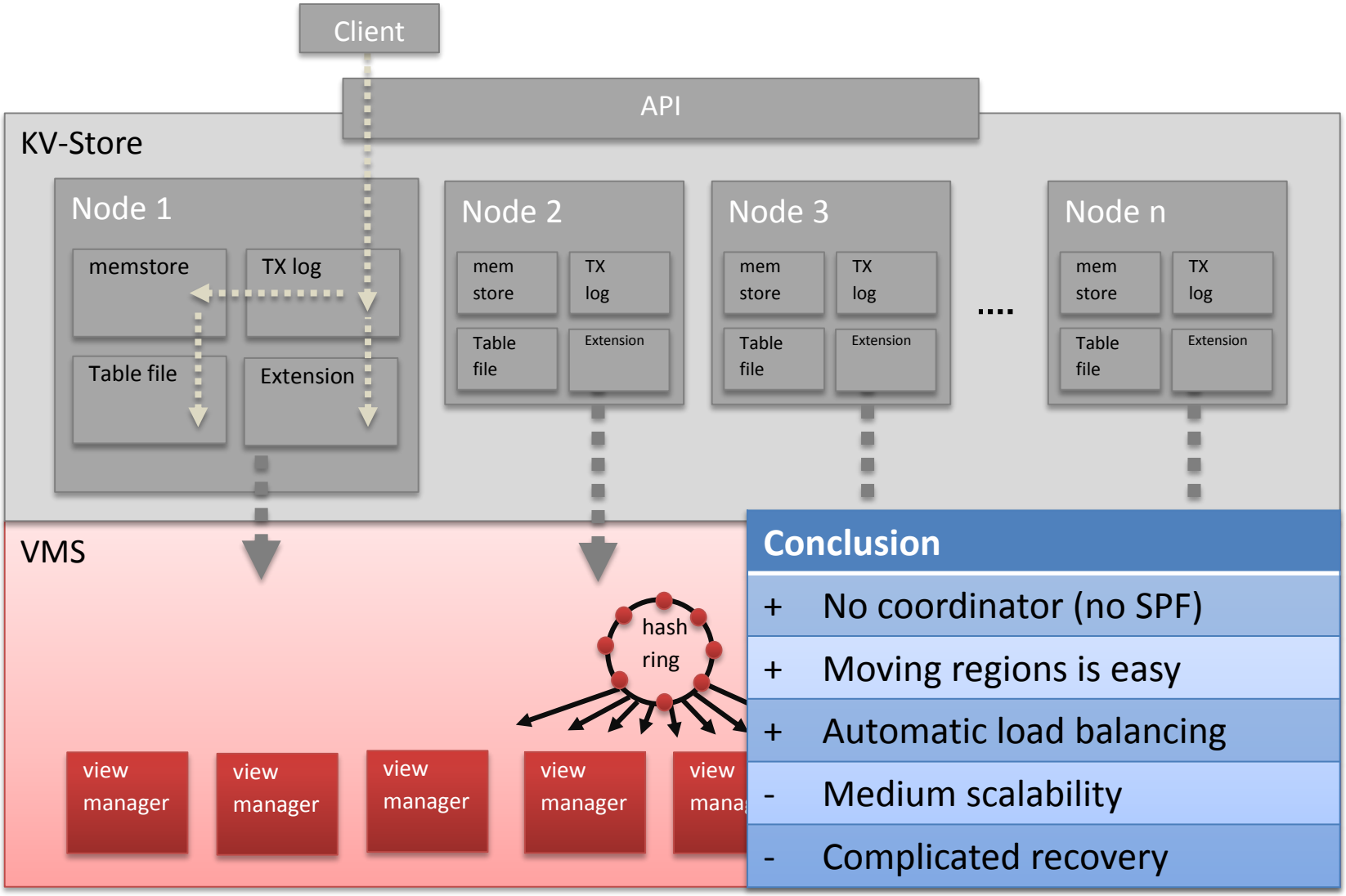
- Model for KV-Stores: HBase, Cassandra, Bigtable, Dynamo
- Ongoing master thesis: Implementation in Cassandra

VMS 1: Local hash-ring



Conclusion	
+	High scalability (cluster-wise)
+	Fast recovery
-	Load balancing required
-	Moving regions is difficult
-	Coordinator is SPF

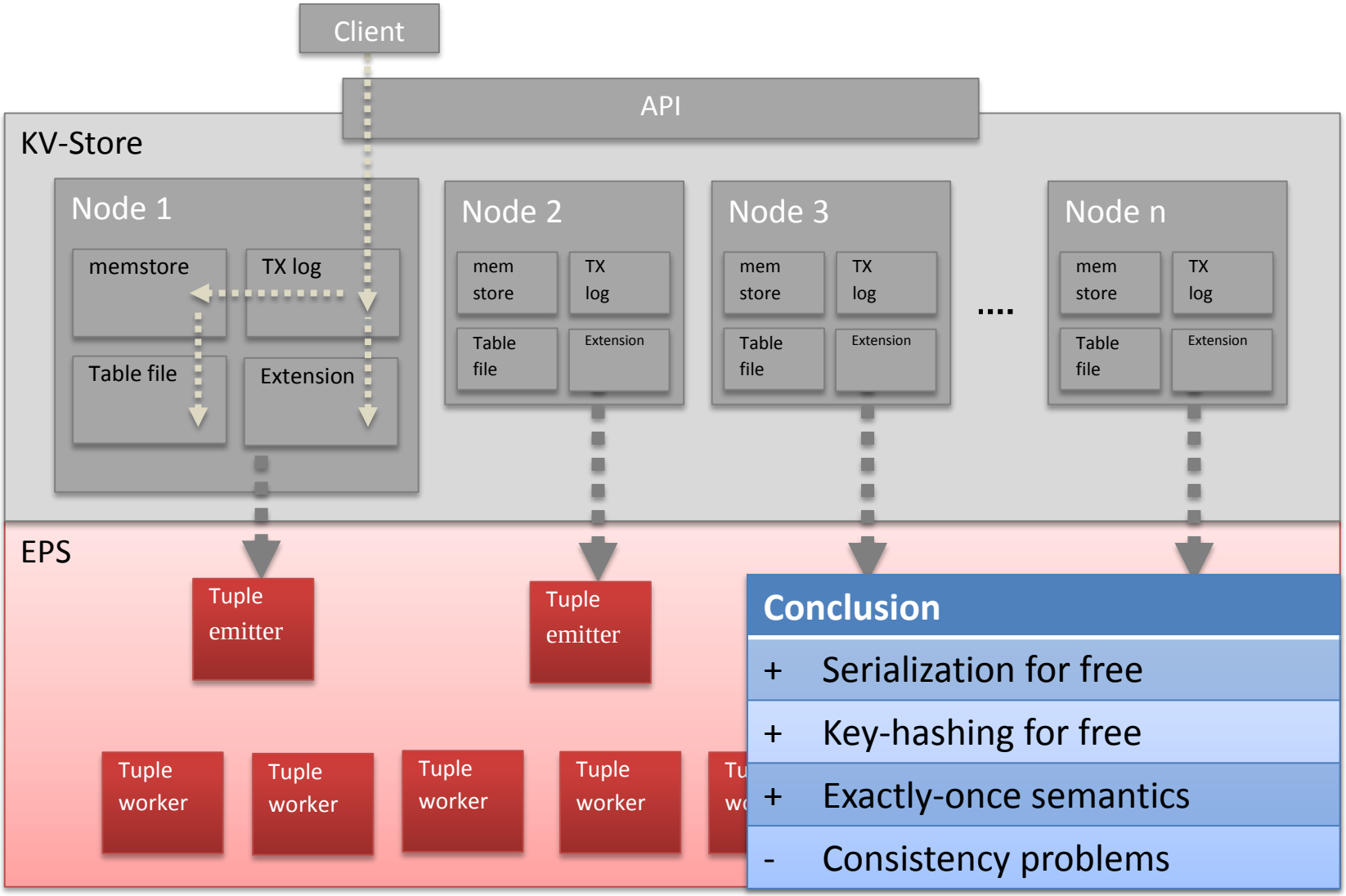
VMS 2: Global hash-ring



Conclusion

+	No coordinator (no SPF)
+	Moving regions is easy
+	Automatic load balancing
-	Medium scalability
-	Complicated recovery

VMS 3: Event processing

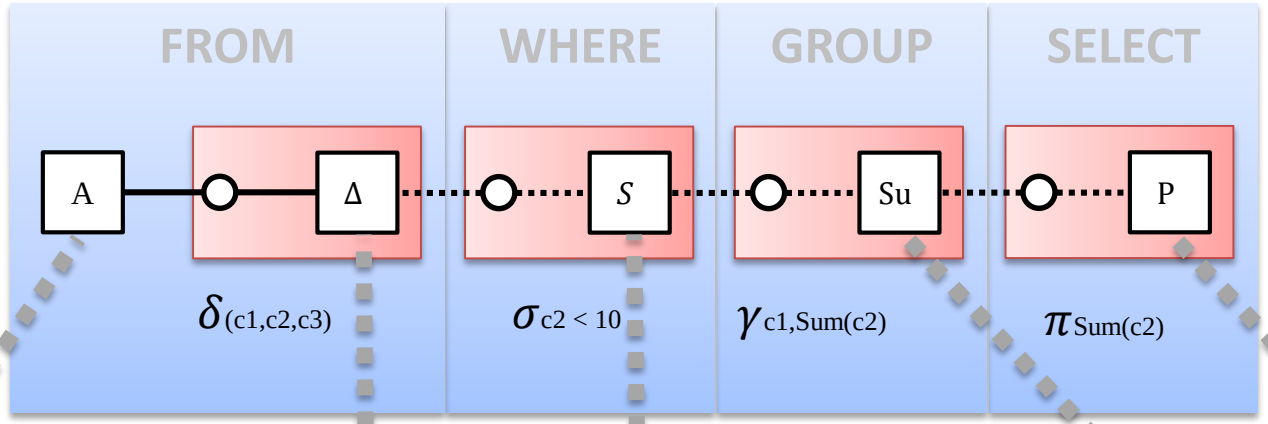


Conclusion

+	Serialization for free
+	Key-hashing for free
+	Exactly-once semantics
-	Consistency problems

Construct a maintenance plan

SELECT $Sum(c2)$ **FROM** A **GROUP BY** $c1$ **WHERE** $c2 < 10$



Rk	c1	c2	c3
k1	x1	6	5
k2	x1	1	12
k3	x2	10	7
k4	x3	3	8
k5	x2	9	5
k6	x4	10	21

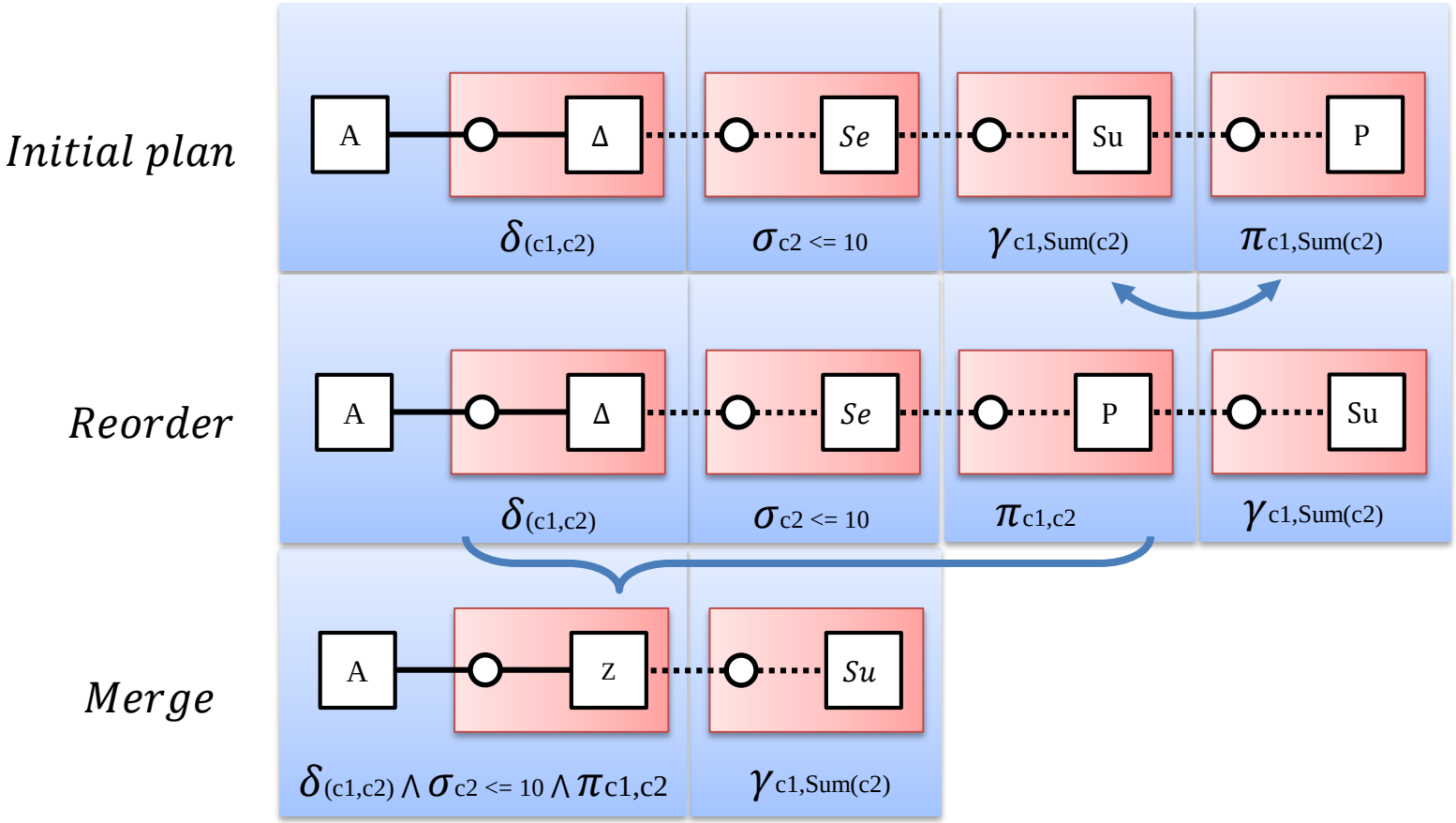
Rk	c1o	c1n	c2o	c2n	c3o	c3n
k1	null	x1	null	6	null	5
k2	null	x1	null	1	null	12
k3	null	x2	null	10	null	7
k4	null	x3	null	3	null	8
k5	null	x2	null	9	null	5
k6	null	x4	null	10	null	21

Rk	c1o	c1n	c2o	c2n	c3o	c3n
k1	null	x1	null	6	null	5
k2	null	x1	null	1	null	12
k4	null	x3	null	3	null	8
k5	null	x2	null	9	null	5

rk	cSum
x1	7
x2	9
x3	3

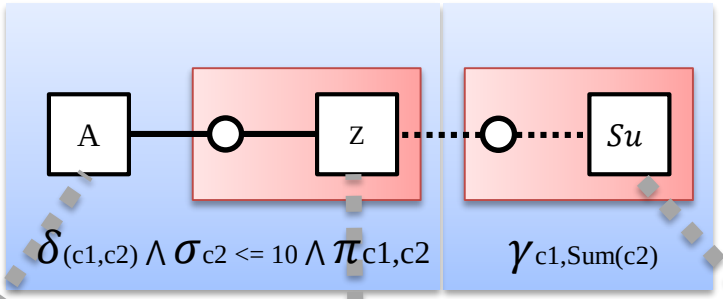
rk	cSum
x1	7
x2	9
x3	3

Optimize the maintenance plan



Result

SELECT $Sum(c2)$ **FROM** A **GROUP BY** $c1$ **WHERE** $c2 < 10$



Rk	c1	c2	c3
k1	x1	6	5
k2	x1	1	12
k3	x2	10	7
k4	x3	3	8
k5	x2	9	5
k6	x4	10	21

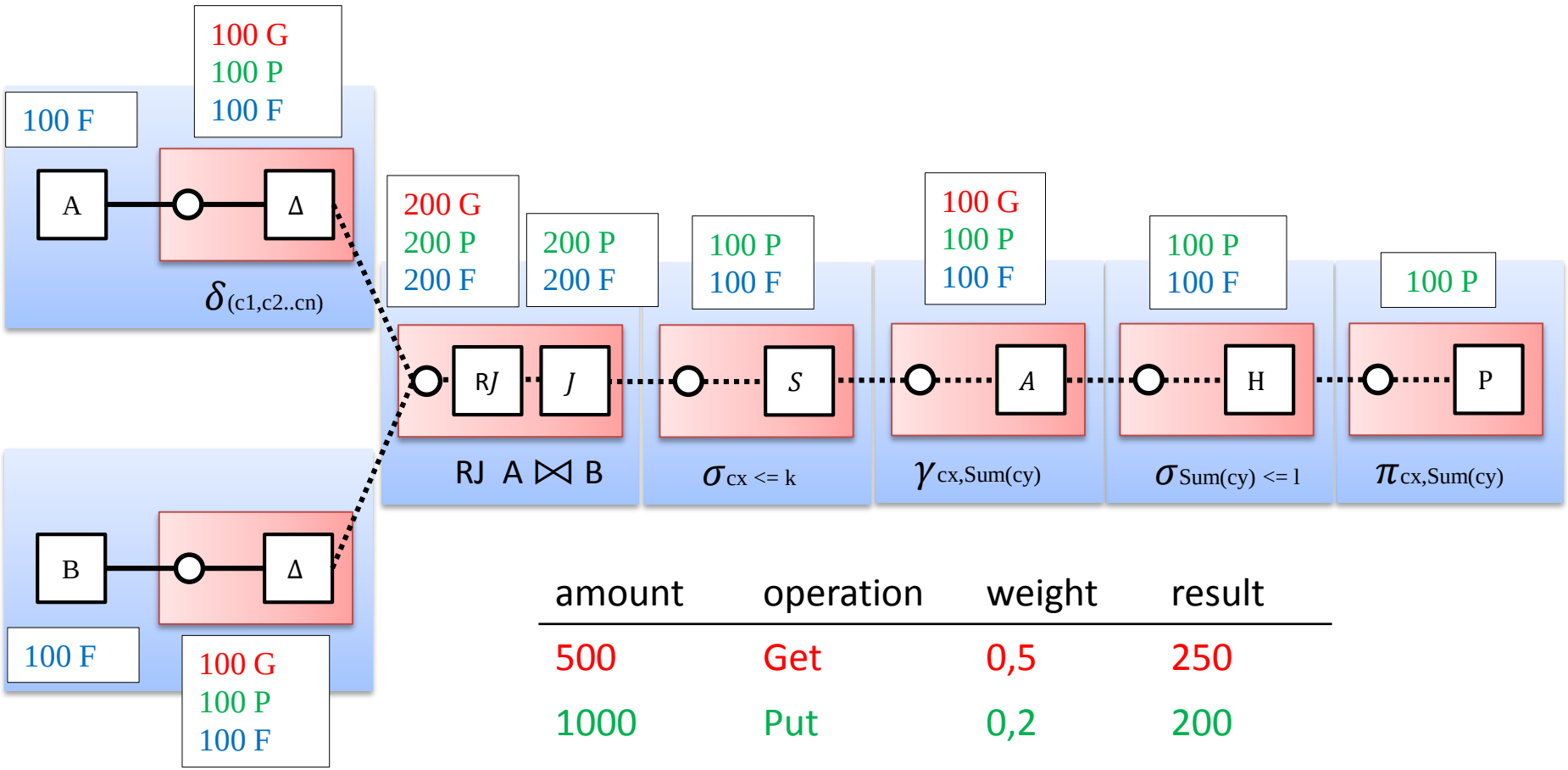
Rk	c1o	c1n	c2o	c2n
k1	null	x1	null	6
k2	null	x1	null	1
k4	null	x3	null	3
k5	null	x2	null	9

rk	cSum
x1	7
x2	9
x3	3

The SQL-Pattern

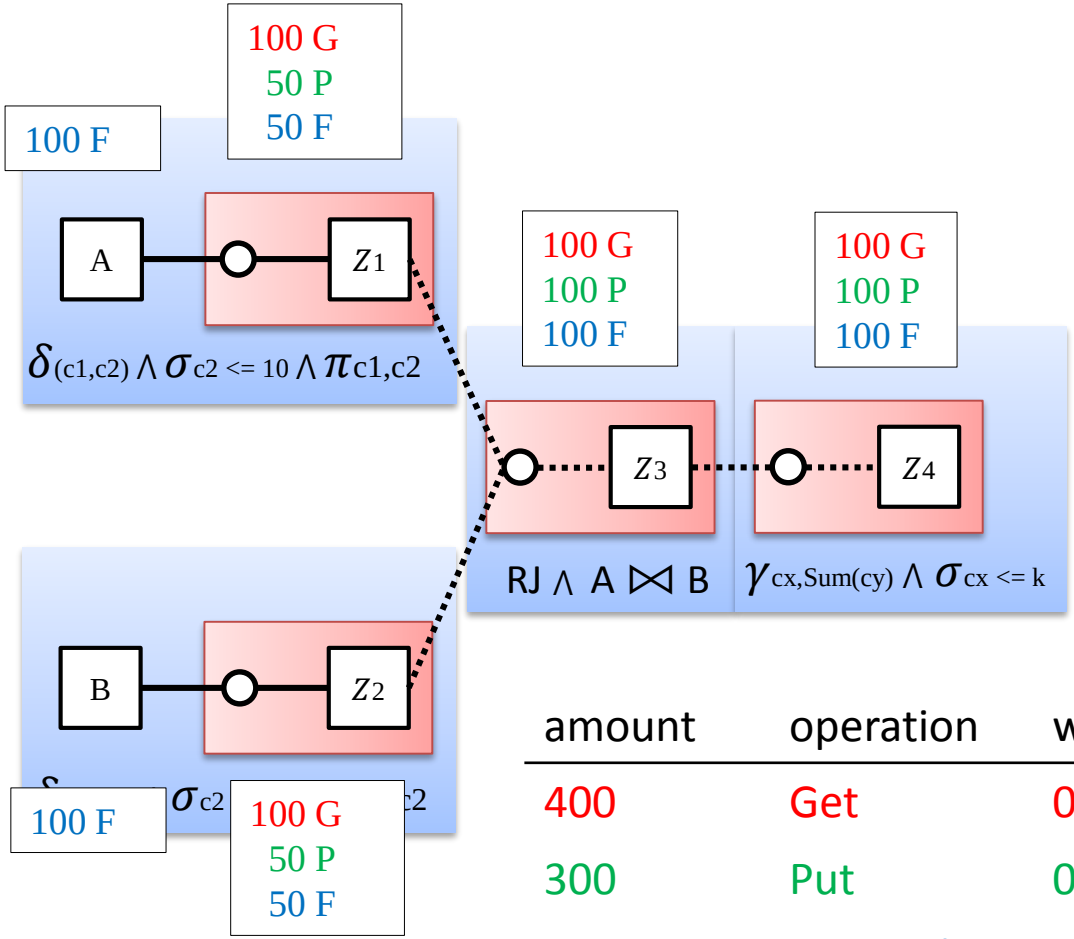
SQL clause	view
1. FROM	base table, delta view, join view
2. WHERE	selection view
3. GROUP BY	count, sum, min, max, avg, index view
4. HAVING	selection view
5. SELECT	projection view
6. ORDER BY	not implemented yet

The SQL pattern



amount	operation	weight	result
500	Get	0,5	250
1000	Put	0,2	200
1100	Forward	0,1	110
			560

Optimized SQL pattern

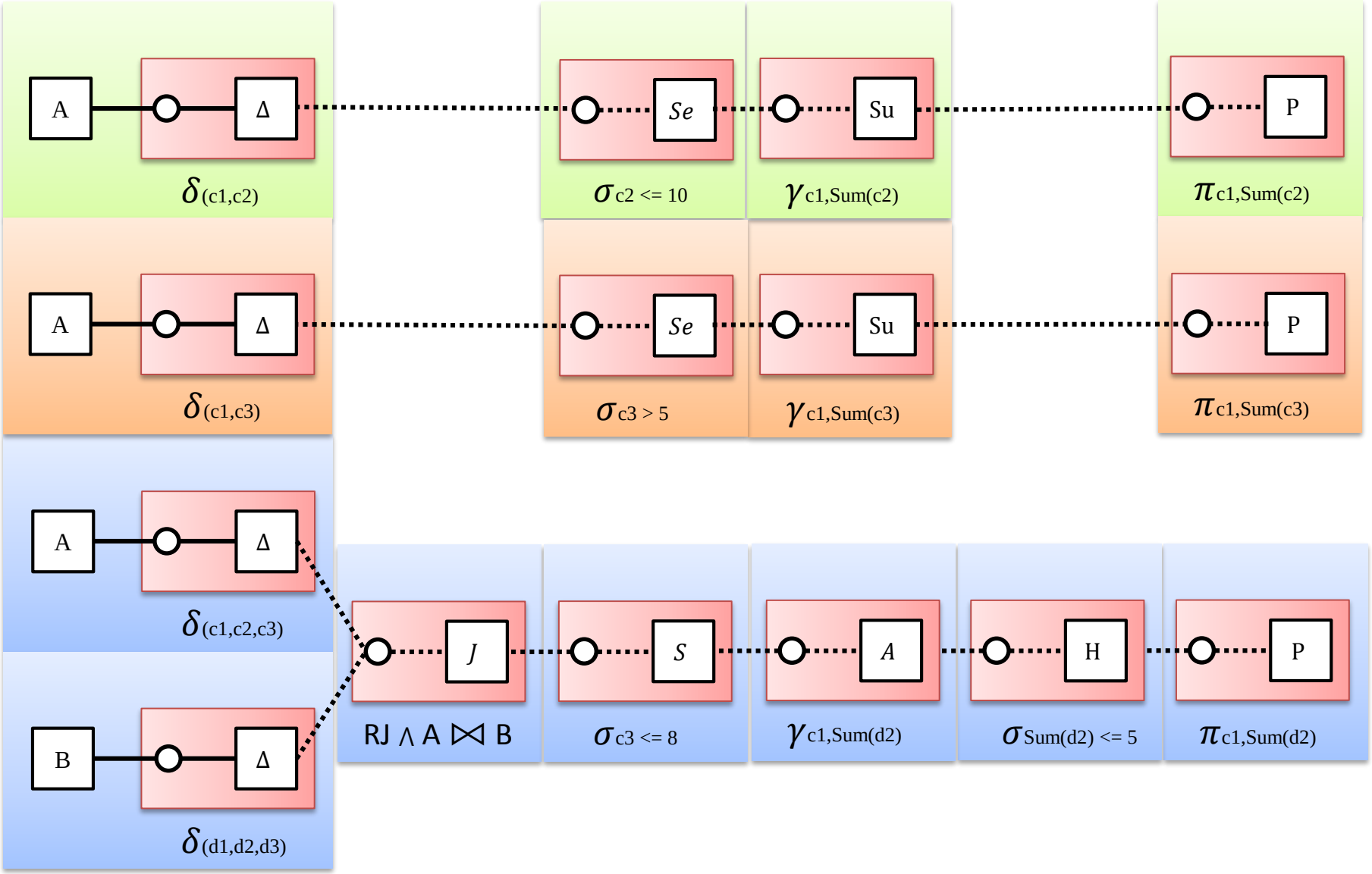


amount	operation	weight	result
400	Get	0,5	200
300	Put	0,2	60
500	Forward	0,1	50
			310

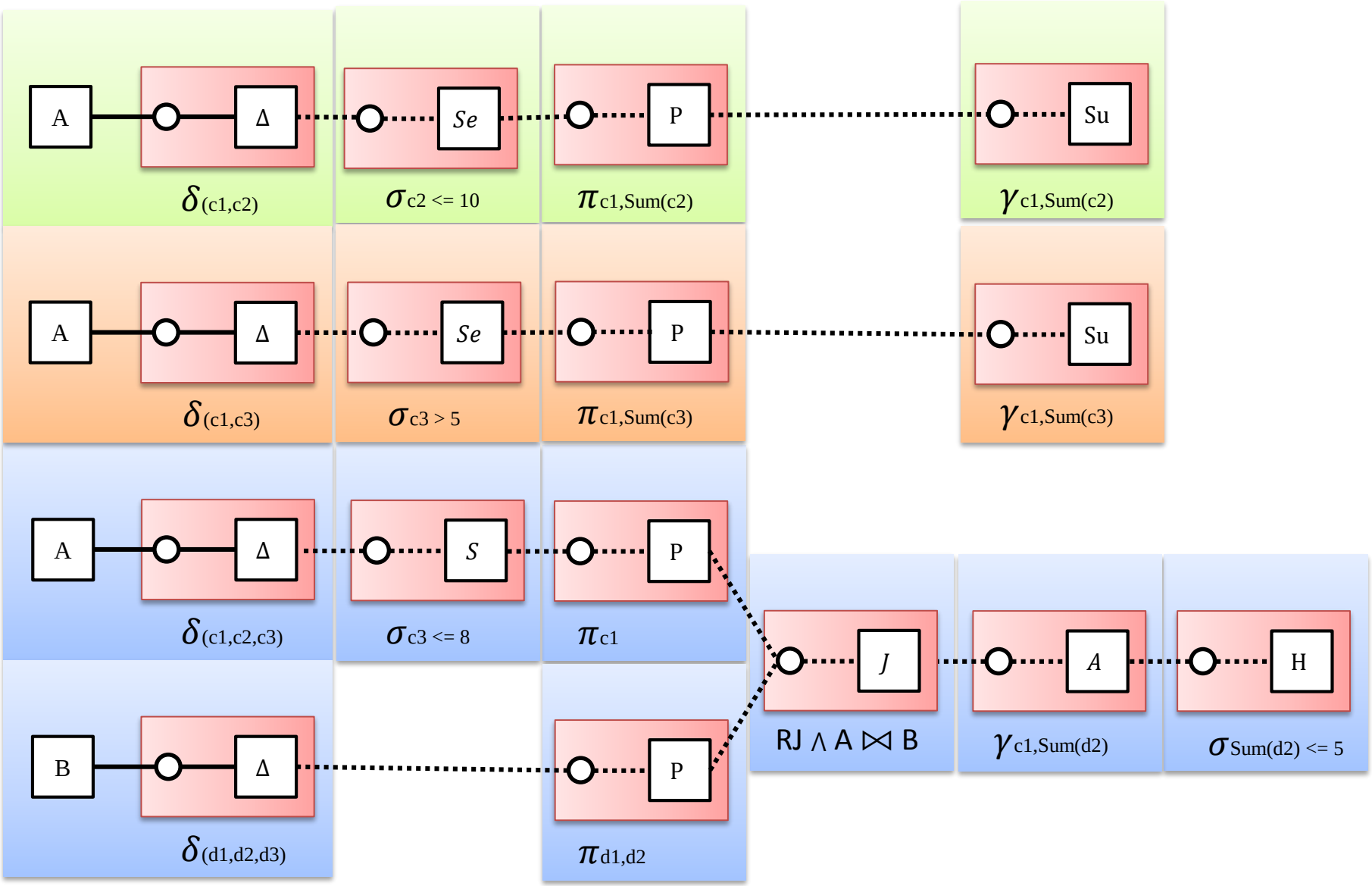
Multi query

1. **SELECT** $Sum(c2)$ **FROM** A **GROUP BY** $c1$ **WHERE** $c2 =< 10$
2. **SELECT** $Sum(c3)$ **FROM** A **GROUP BY** $c1$ **WHERE** $c3 > 10$
3. **SELECT** $Count(d2)$ **FROM** A **INNER JOIN** B **ON** $A.c1 = B.d1$ **GROUP BY** $A.c1$ **WHERE** $A.c3 \leq 8$ **HAVING** $Count(d2) > 1$

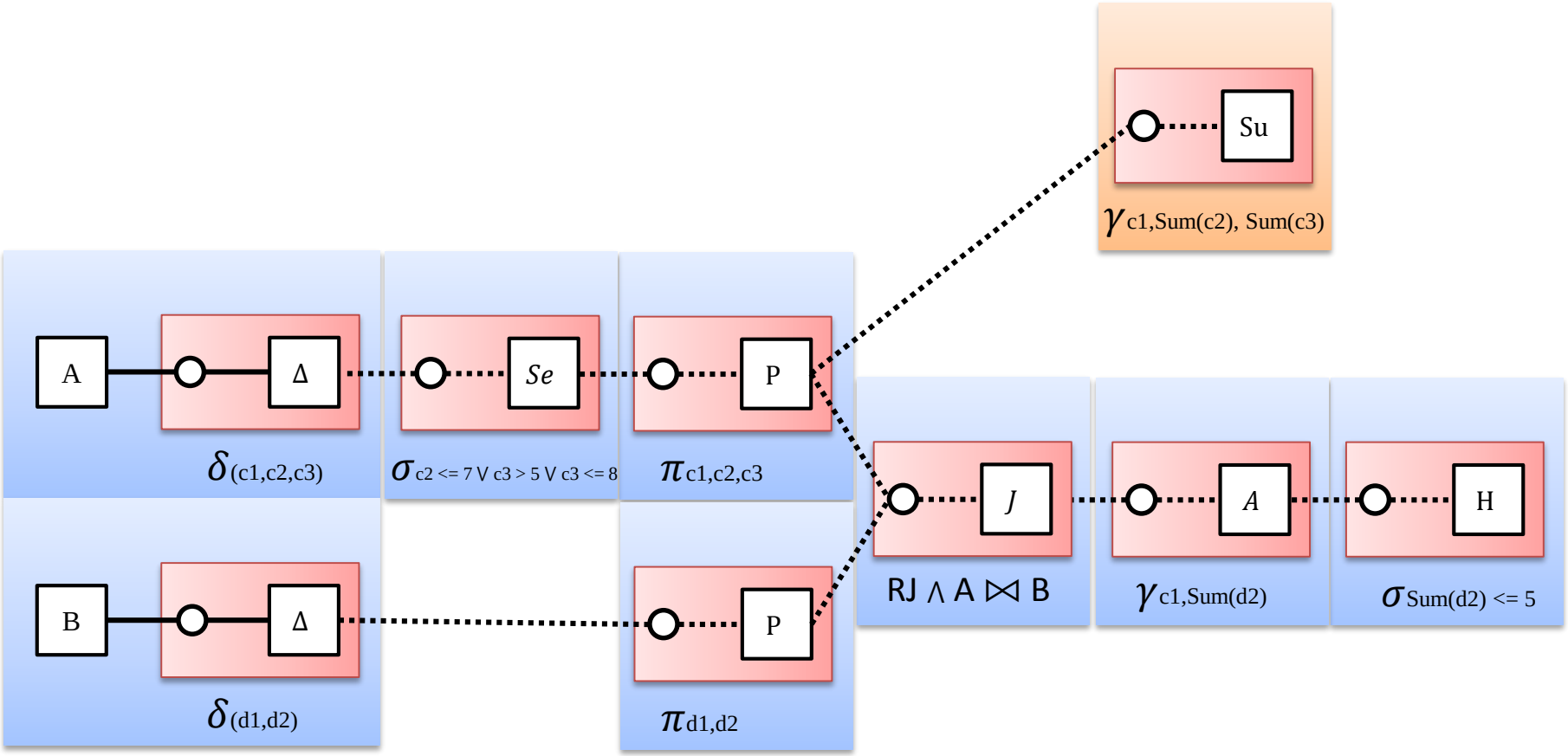
Multi query plan



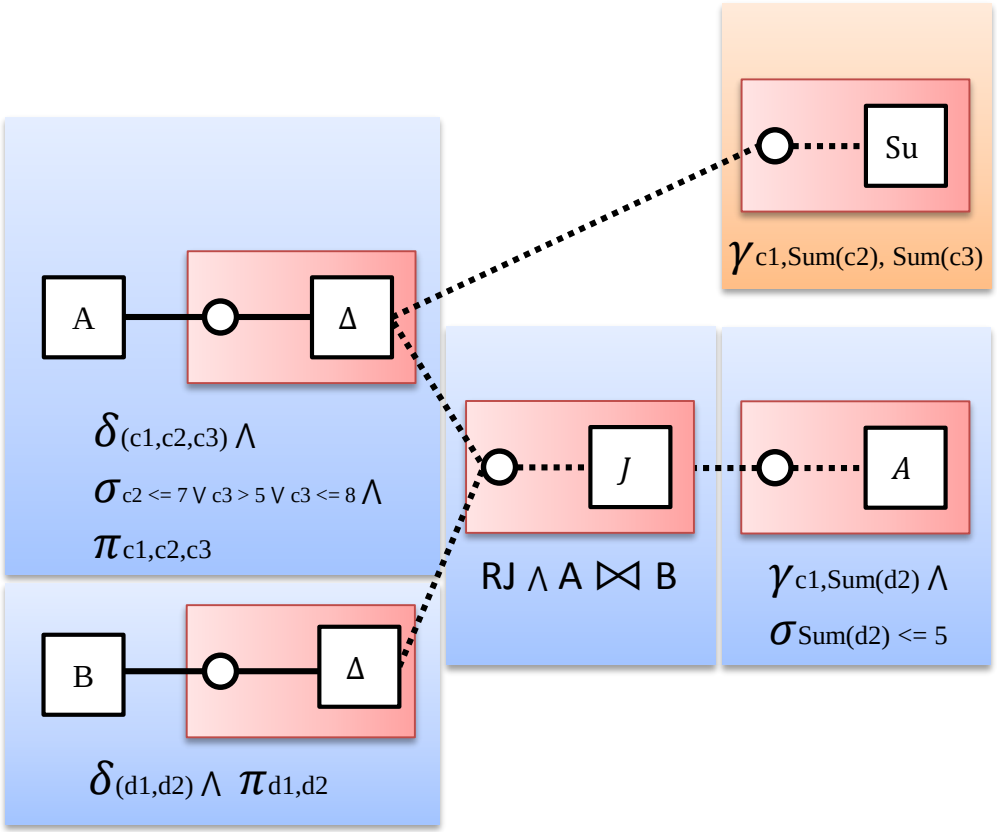
Reorder


A
Δ
δ(c1,c2)
Se
σc2 <= 10
P
πc1,Sum(c2)
Su
γc1,Sum(c2)
A
Δ
δ(c1,c3)
Se
σc3 > 5
P
πc1,Sum(c3)
Su
γc1,Sum(c3)
A
Δ
δ(c1,c2,c3)
S
σc3 <= 8
P
πc1
B
Δ
δ(d1,d2,d3)
P
πd1,d2
J
RJ ∧ A ⋈ B
A
γc1,Sum(d2)
H
σSum(d2) <= 5

Combine


Su
γ c1,Sum(c2), Sum(c3)
A
Δ
δ (c1,c2,c3)
Se
σ c2 <= 7 V c3 > 5 V c3 <= 8
P
π c1,c2,c3
J
RJ ∧ A ⋈ B
A
γ c1,Sum(d2)
H
σ Sum(d2) <= 5
B
Δ
δ (d1,d2)
P
π d1,d2

Merge


A
Δ
δ(c1,c2,c3) ∧
σc2 <= 7 ∨ c3 > 5 ∨ c3 <= 8 ∧
πc1,c2,c3
B
Δ
δ(d1,d2) ∧ πd1,d2
Su
γc1,Sum(c2), Sum(c3)
J
RJ ∧ A ⋈ B
A
γc1,Sum(d2) ∧
σSum(d2) <= 5

Future work

- Implement alternative architectures
- Implement SQL-command line
- Evaluate multi-view optimization
- Get the papers accepted

THE END