Query Optimizations For Partitioned Tables

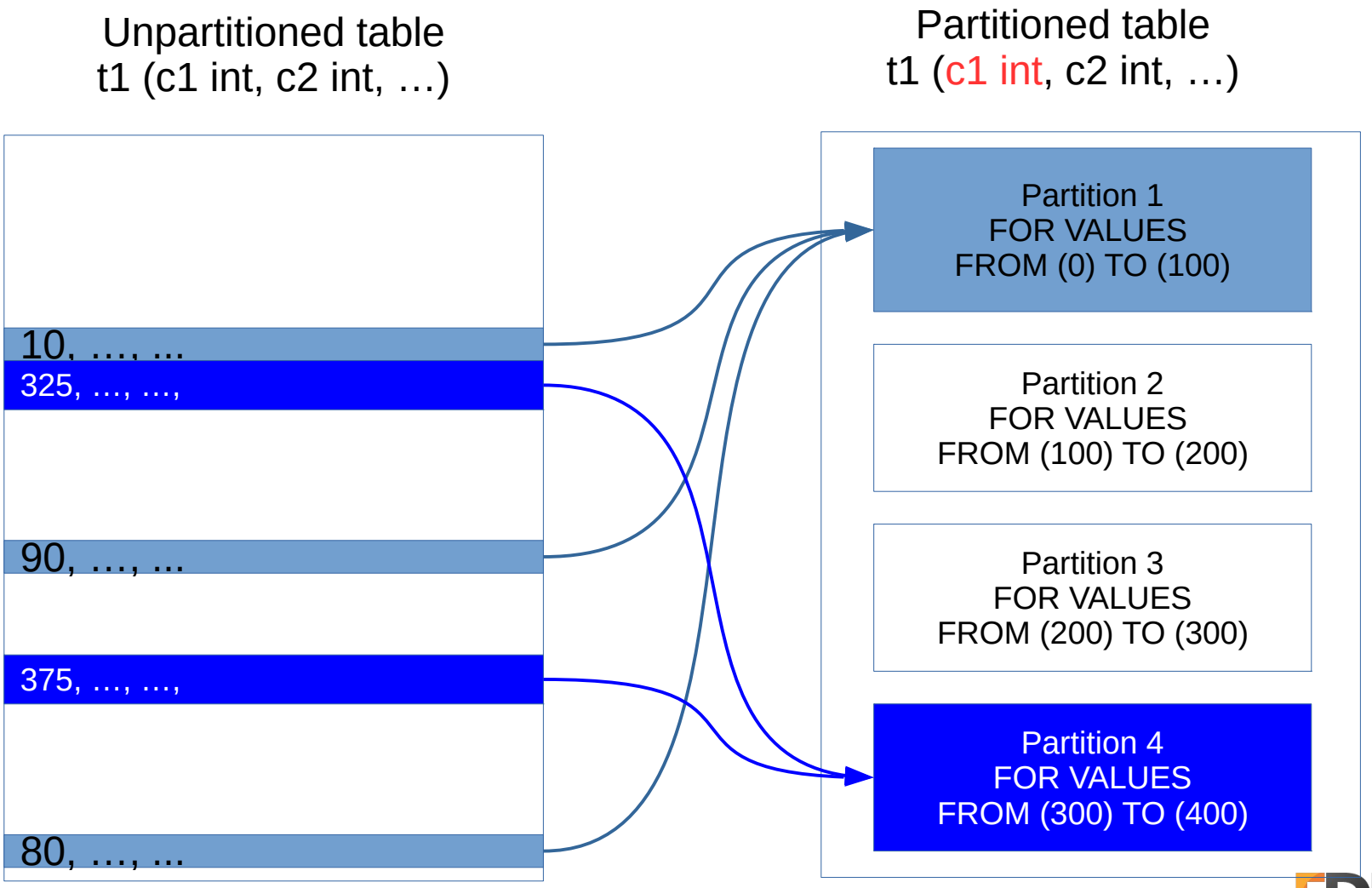
Ashutosh Bapat

@PGCONF INDIA 2018

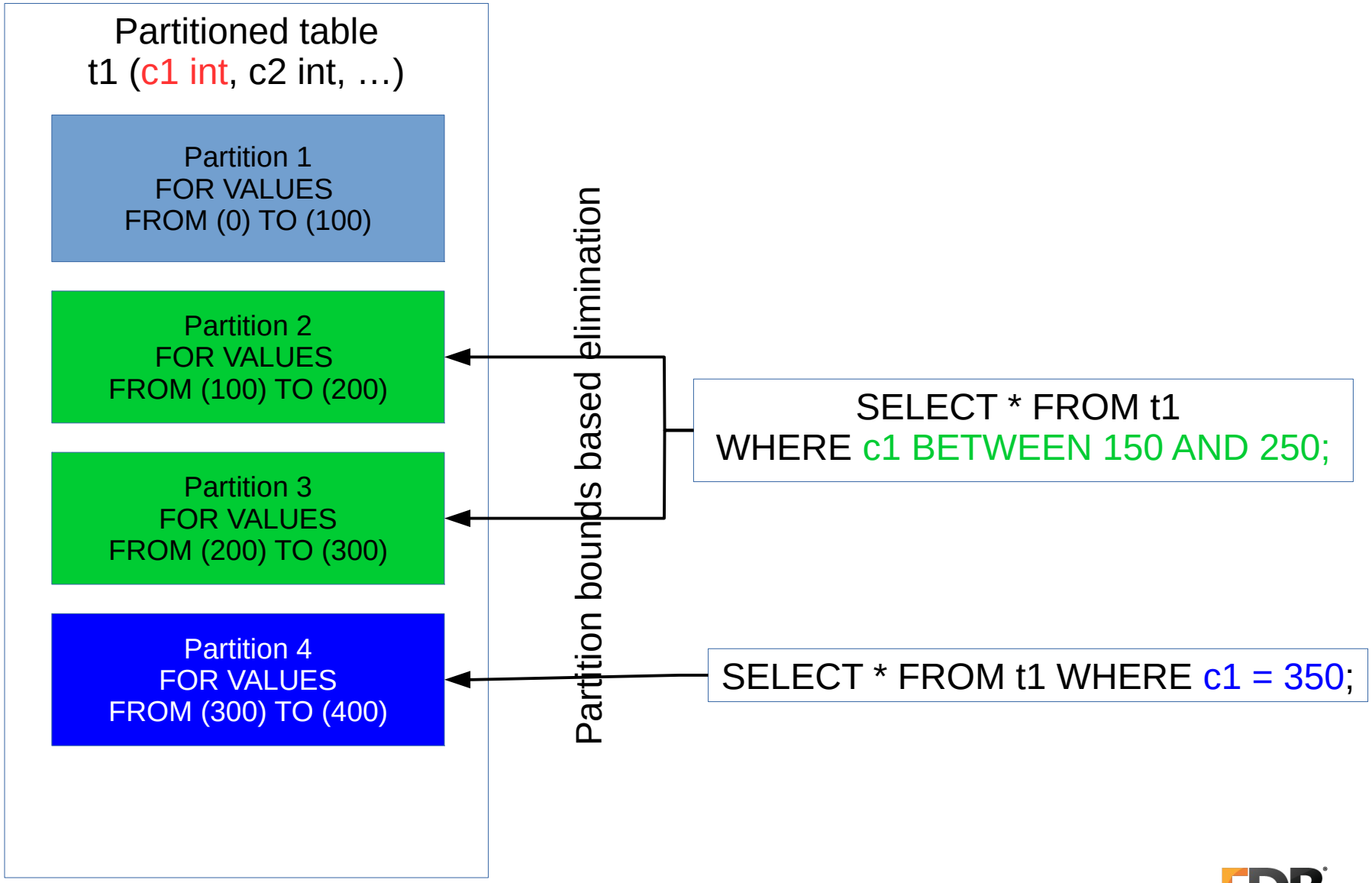
Query Optimization Techniques

- Partition pruning
- Run-time partition pruning
- Partition-wise join
- Partition-wise aggregation
- Partition-wise sorting/ordering

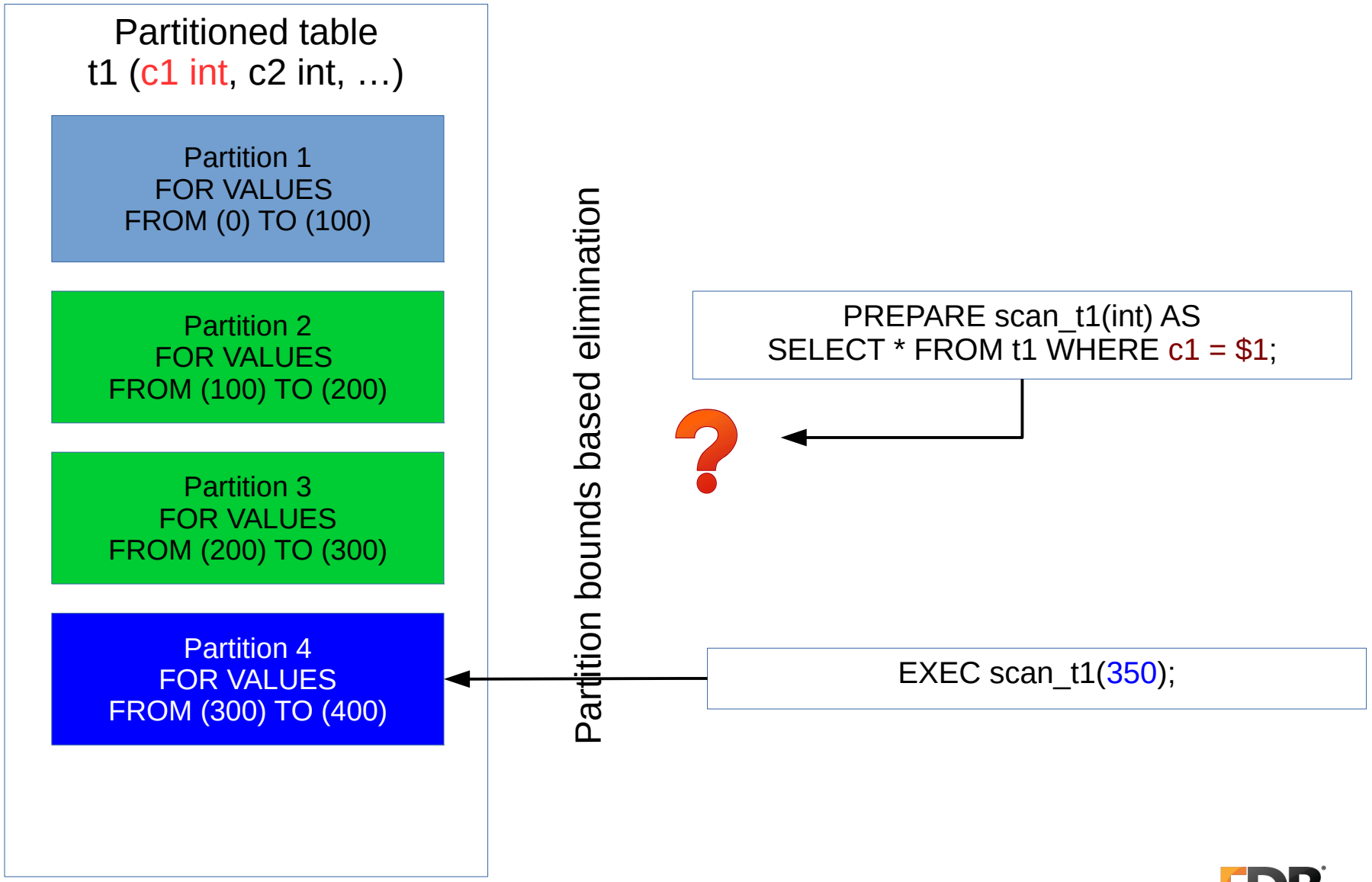
Partitioned Table



Partition Pruning

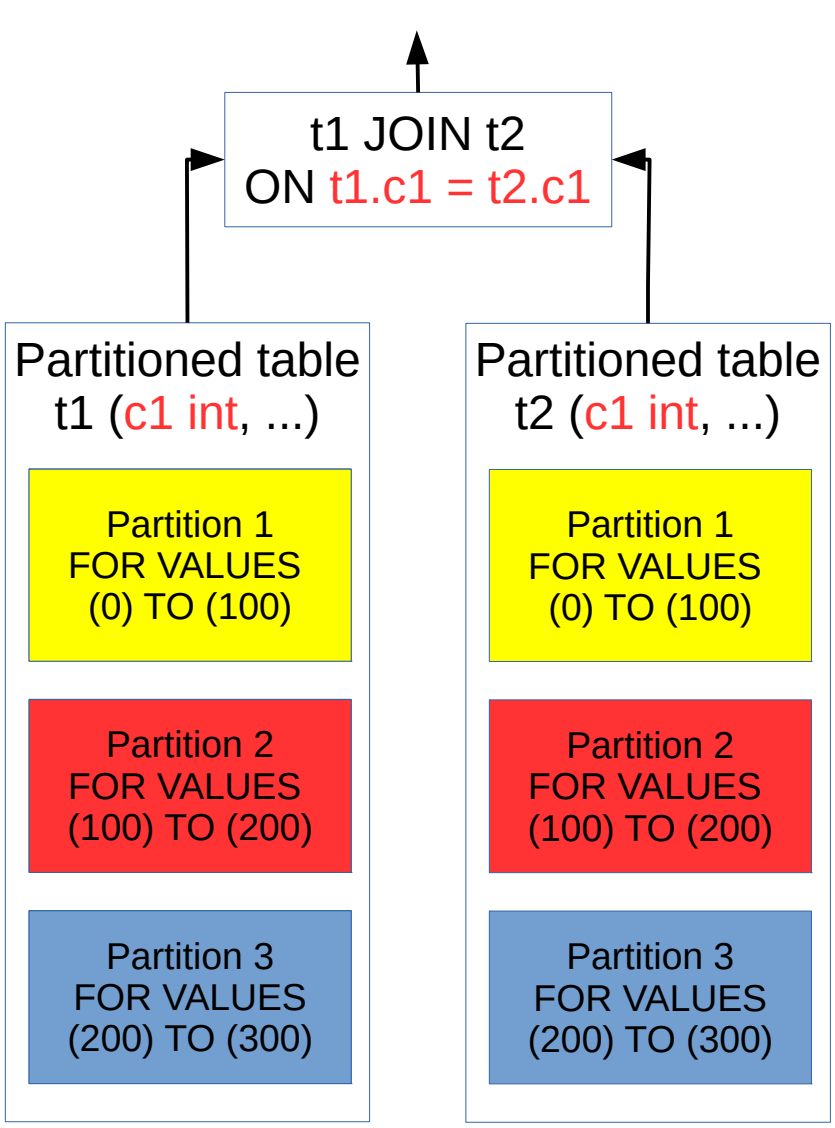


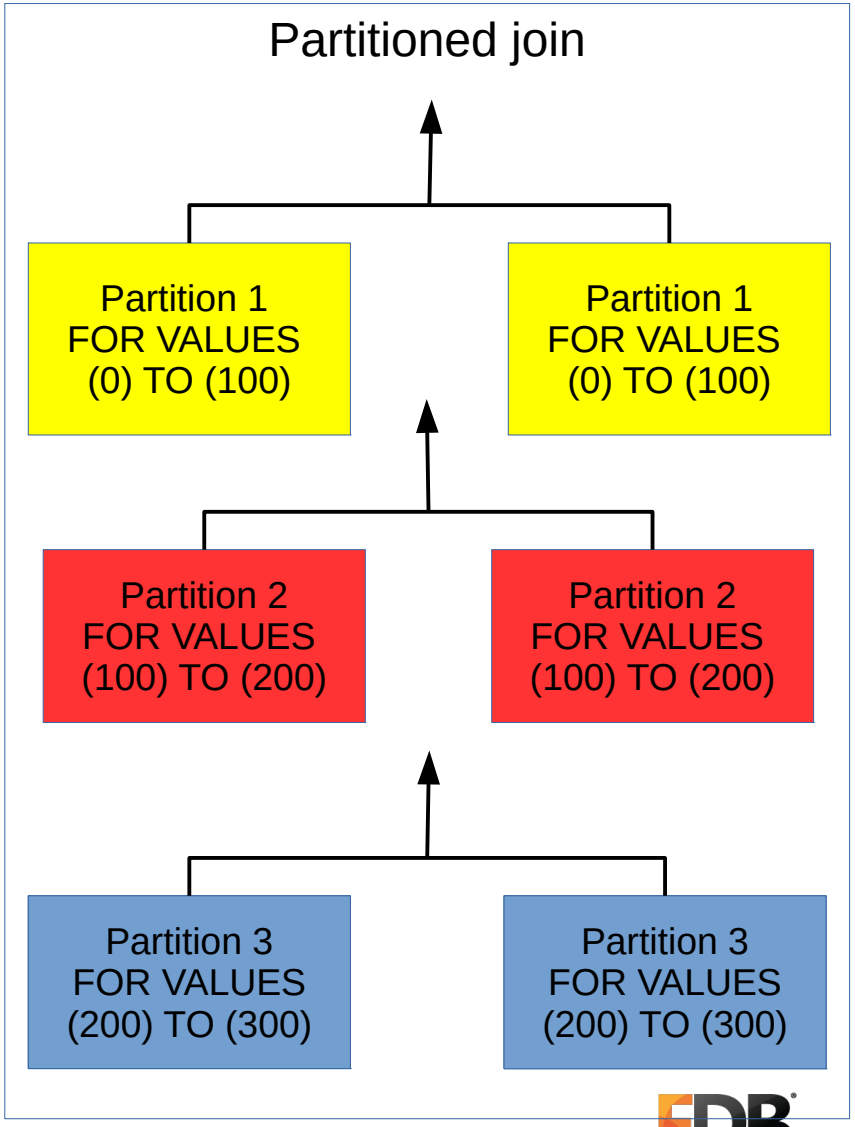
Run-time Partition Pruning



Partition-wise Join

Partition-wise Join

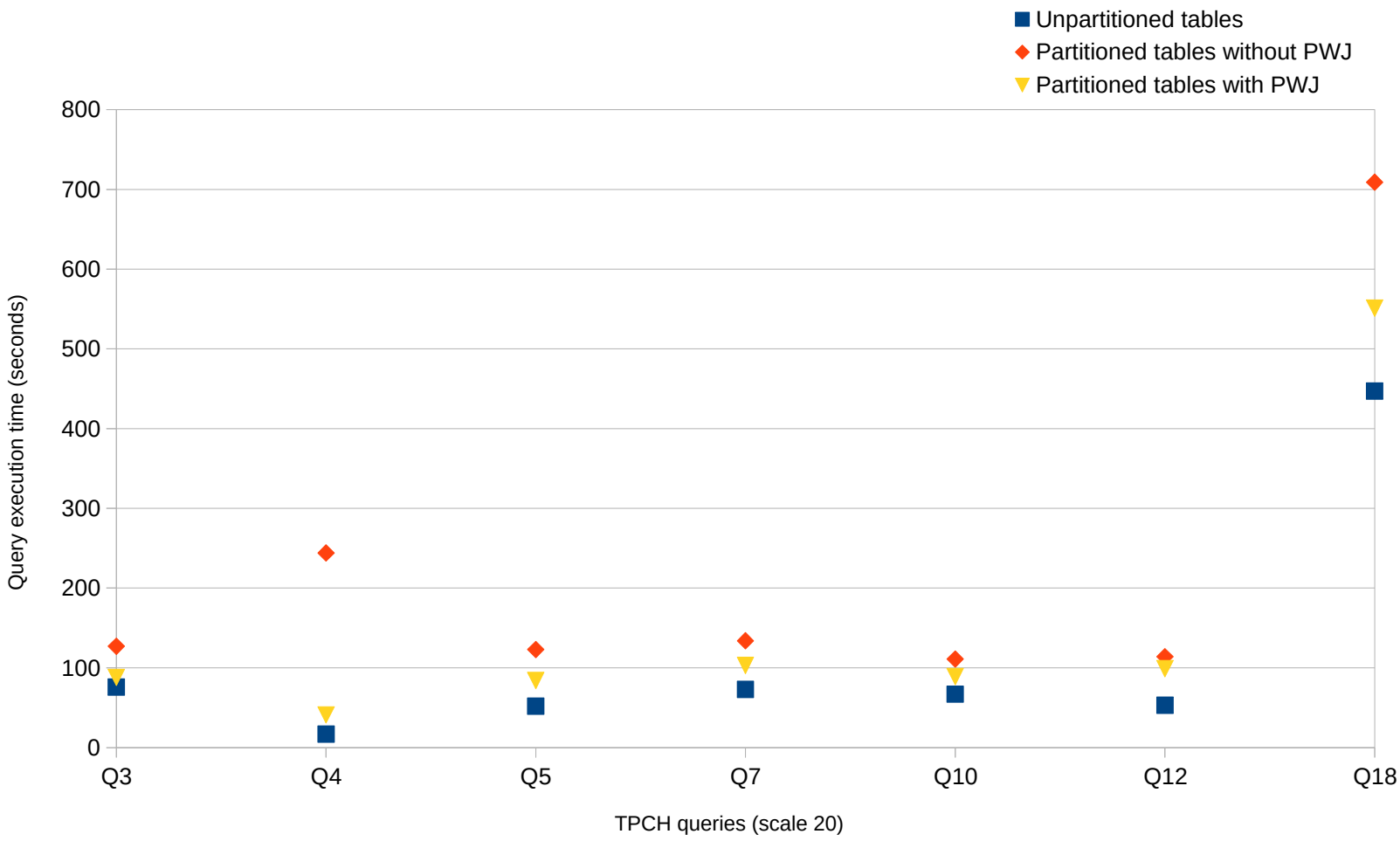




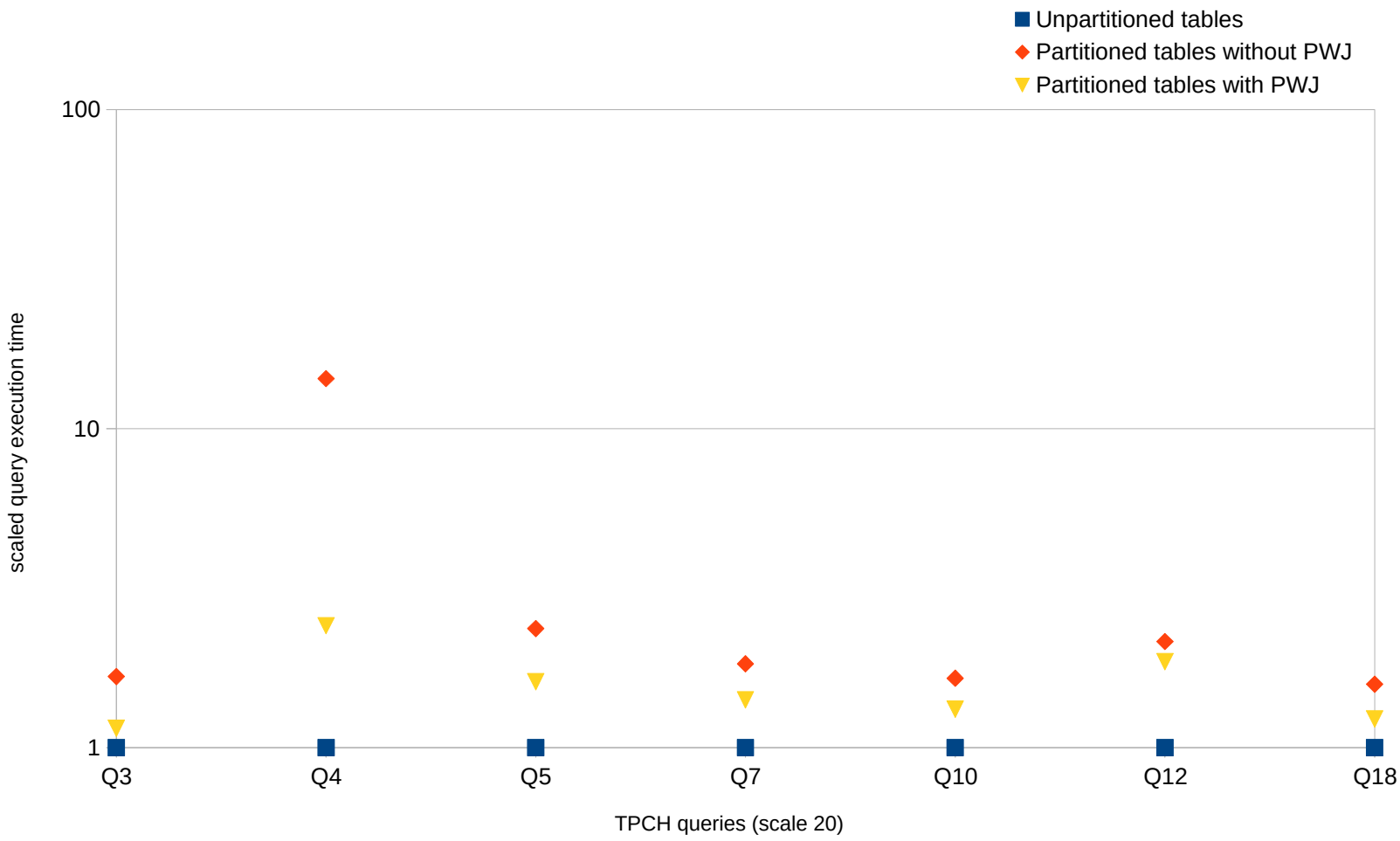
TPCH Vs. Partition-wise Join (Scale 20)

- Scale 20
- Schema changes
  - lineitems PARTITION BY RANGE(l_orderkey)
  - orders PARTITION BY RANGE(o_orderkey)
  - Each with 17 partitions
- GUCs
  - work_mem - 1GB
  - effective_cache_size - 8GB
  - shared_buffers - 8GB
  - enable_partition_wise_join = on

TPCH Vs. Partition-wise Join (Scale 20)
(linear time scale)



TPCH Vs. Partition-wise Join (Scale 20)
(scaled execution time)



TPCH Vs. partition-wise join: observations

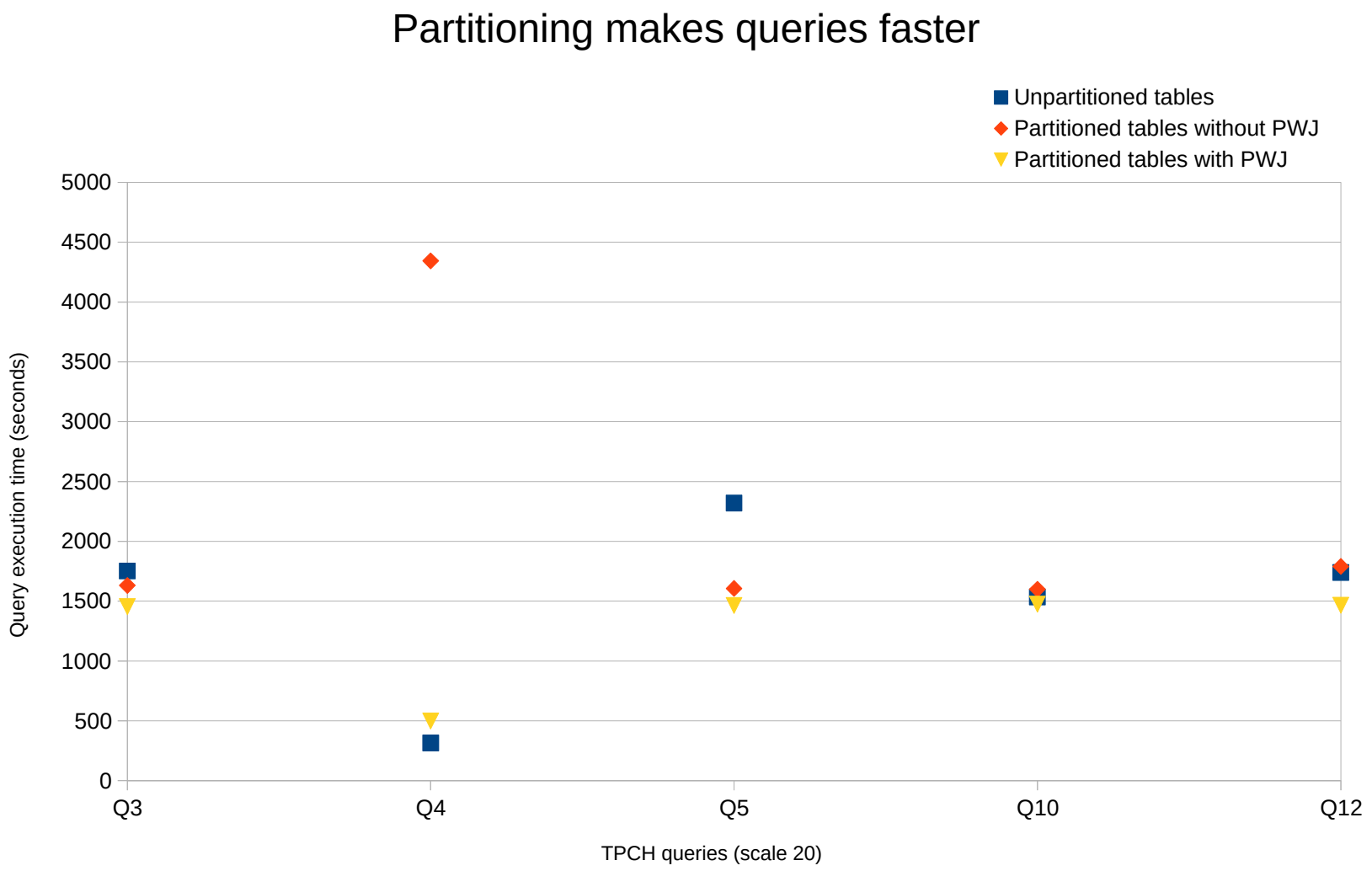
- Join strategy change from join between partitioned tables to join between partitions
  - MergeJoin to HashJoin
    - Q3, Q5, Q10, Q12, Q18
  - HashJoin to parameterized NestLoop join
    - Q4
- Different join strategies for different partition-joins
    - Q7
- Change in order of joining partitioned tables
  - Q3, Q5, Q10, Q12, Q18, Q7

TPCH Vs. Partition-wise Join (Scale 300)

Reported by Rafia Sabih

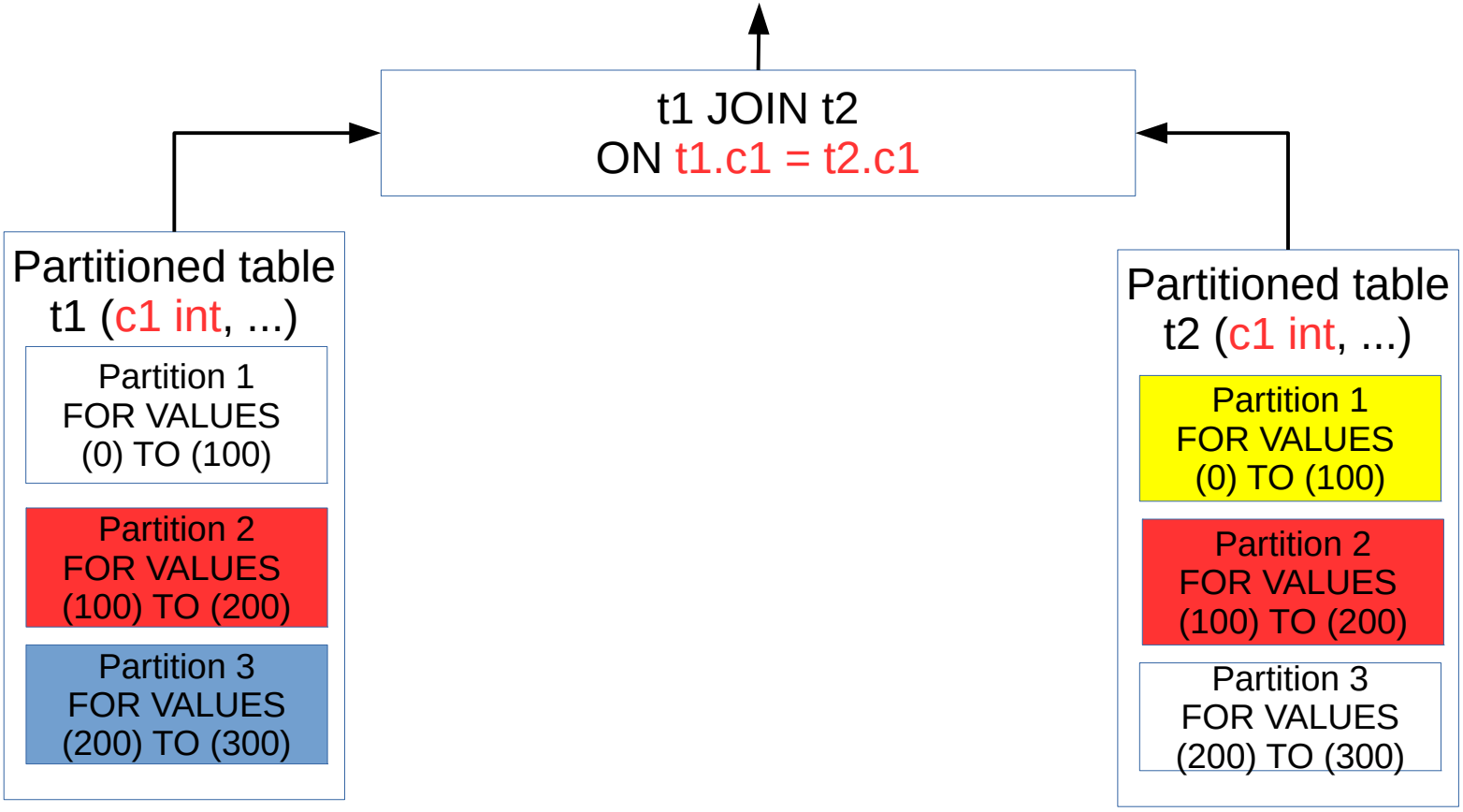
- Scale 300
- Schema changes
  - lineitems PARTITION BY RANGE(l_orderkey)
  - orders PARTITION BY RANGE(o_orderkey)
  - Each with 106 partitions
- GUCs
  - work_mem - 1GB
  - effective_cache_size - 10GB
  - shared_buffers - 10GB
  - enable_partition_wise_join = on
  - max_parallel_workers_per_gather = 4

TPCH Vs. Partition-wise Join (Scale 300)

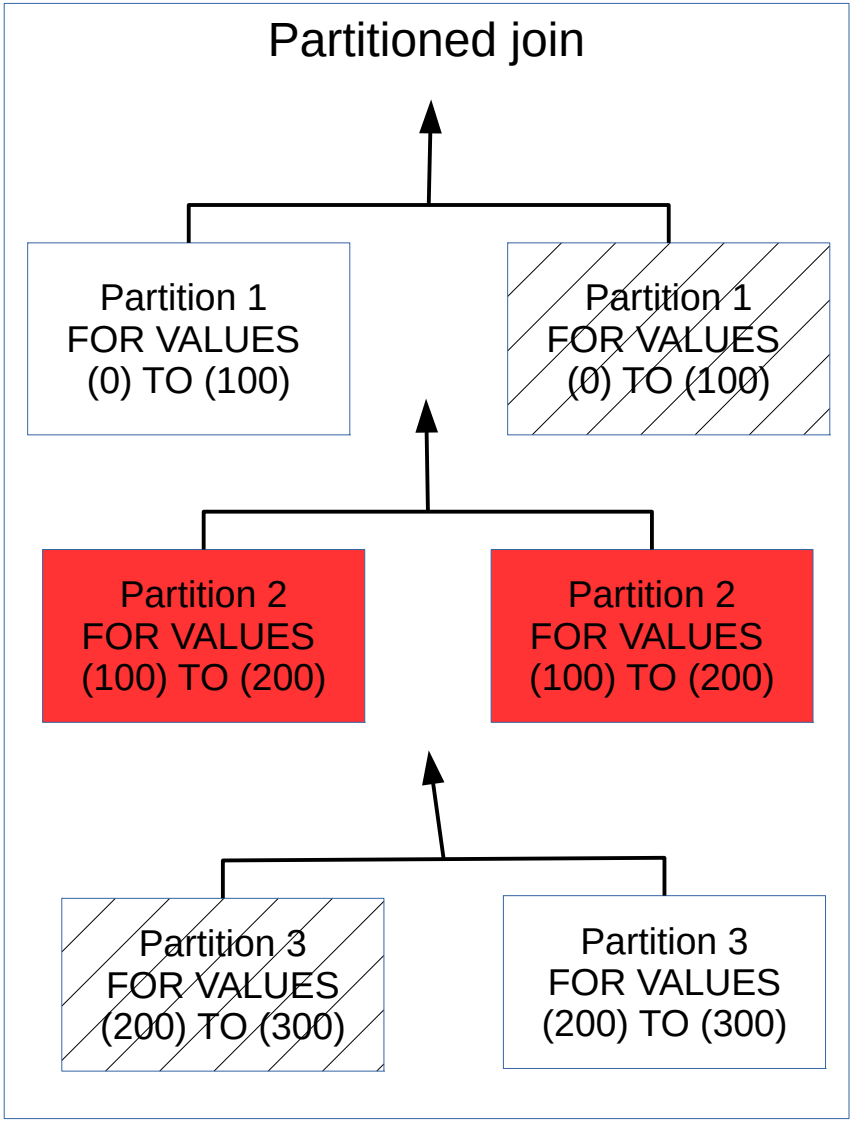


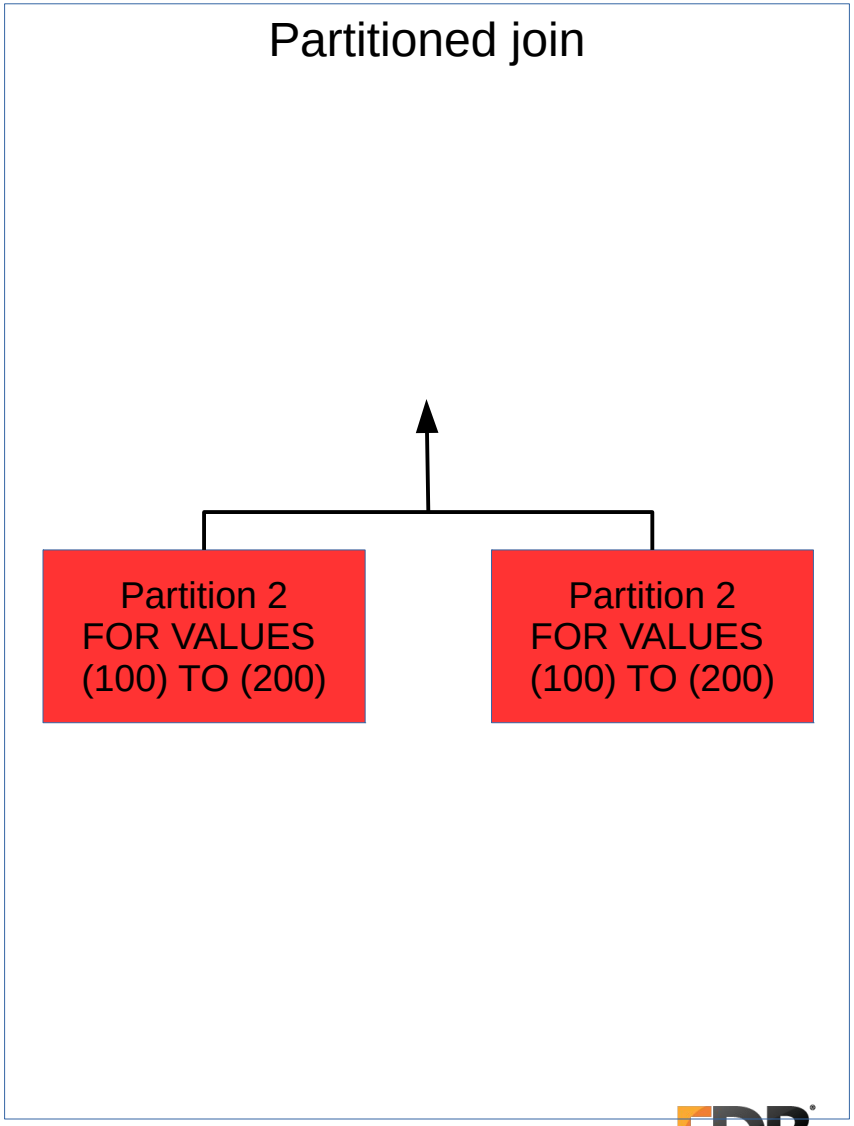
Partition pruning and Partition-wise Join

SELECT … FROM t1 JOIN t2 ON t1.c1 = t2.c1
WHERE t1.c1 > 100 AND t2.c1 < 200



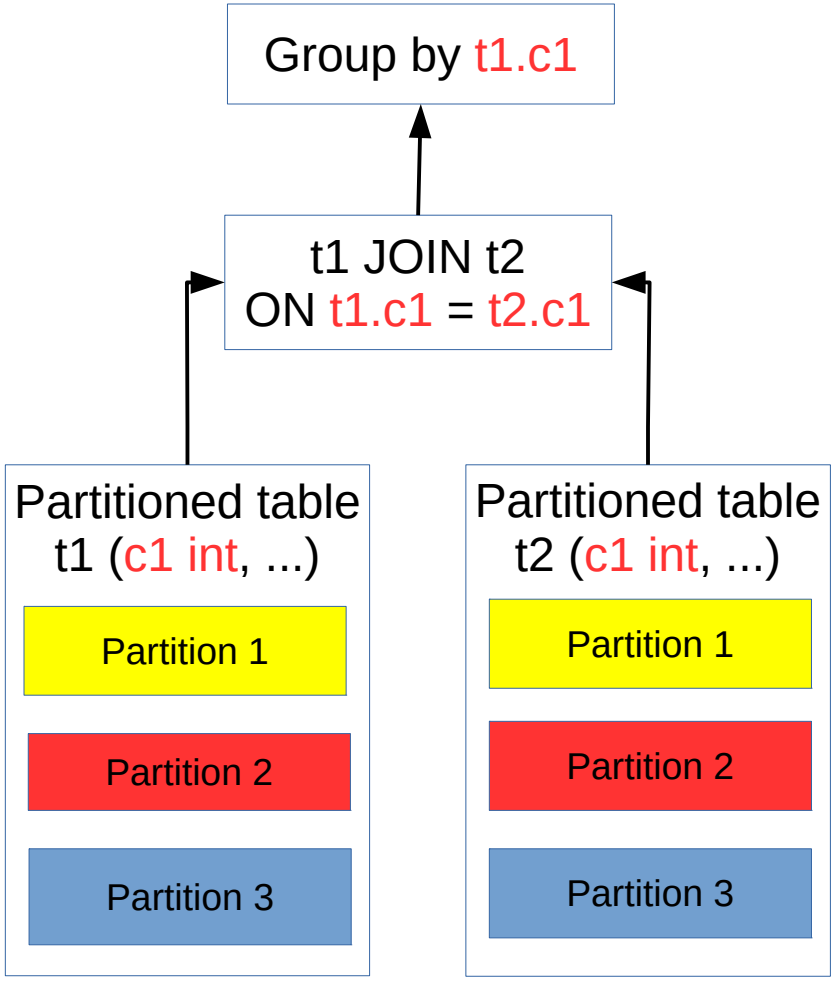
Partition pruning and Partition-wise Join

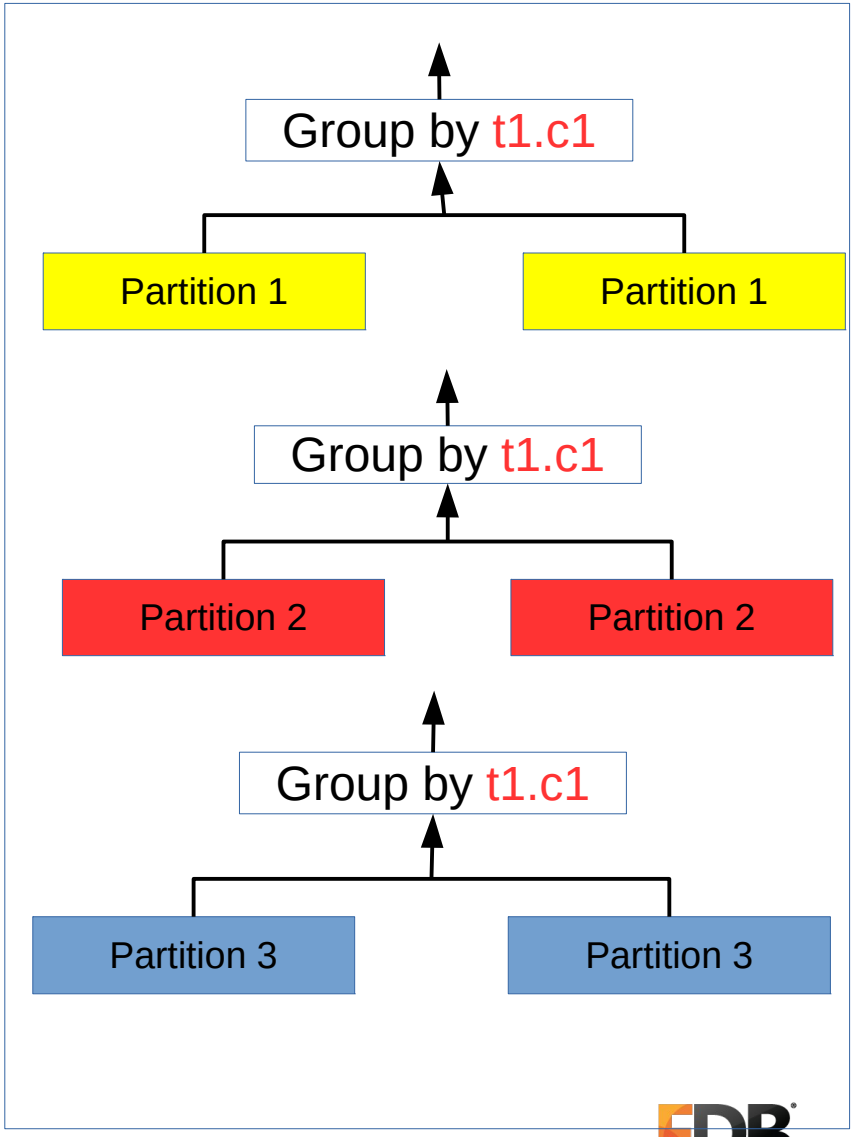




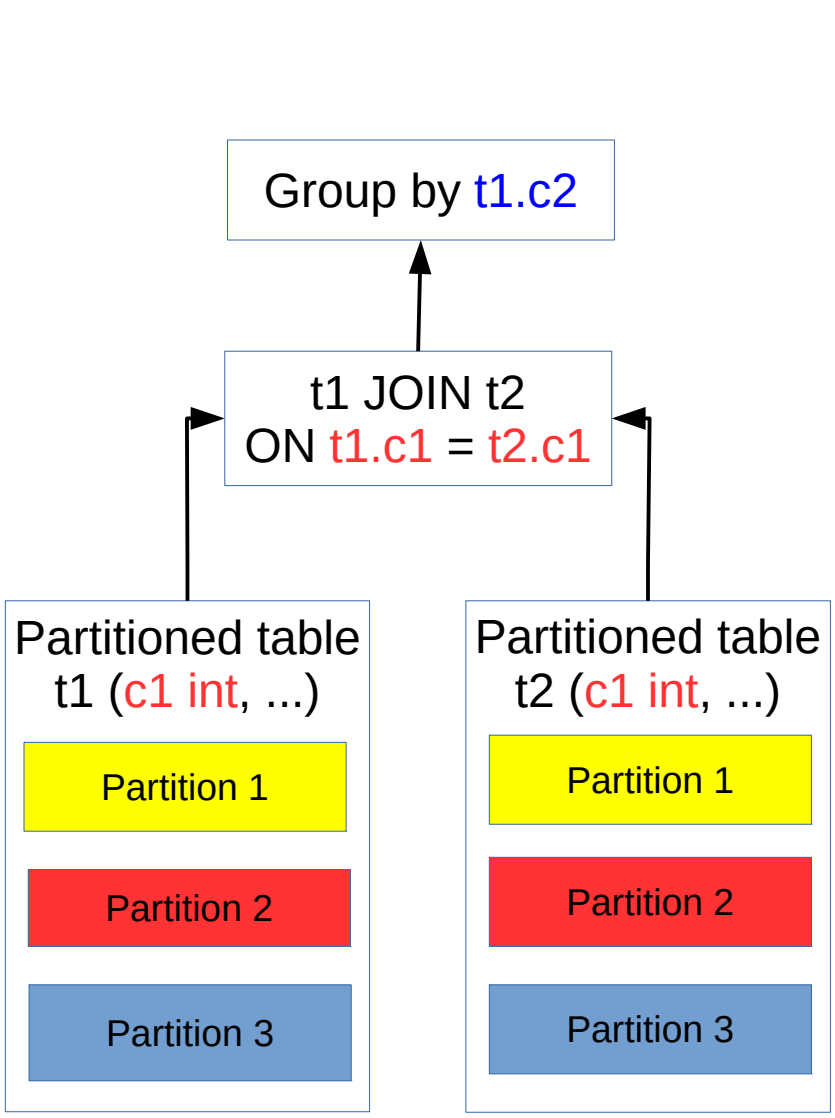
Partition-wise aggregation and grouping

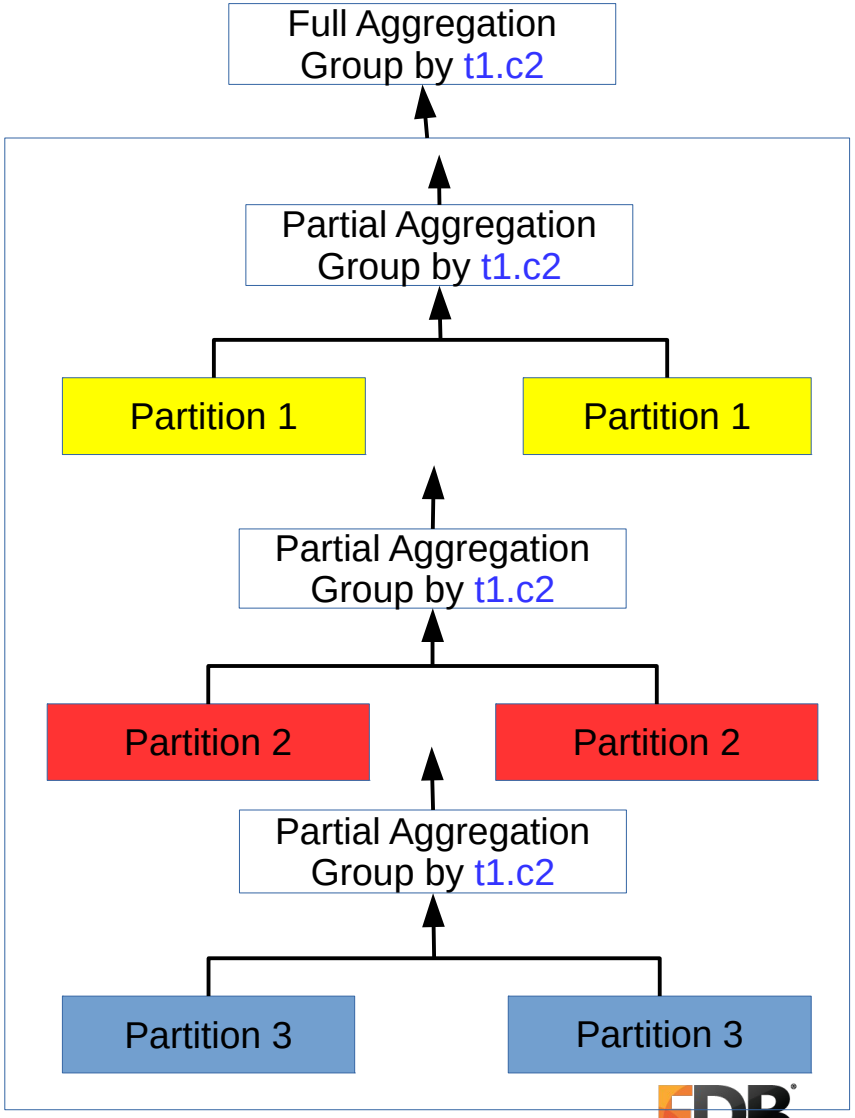
Full partition-wise aggregation





Partial partition-wise aggregation



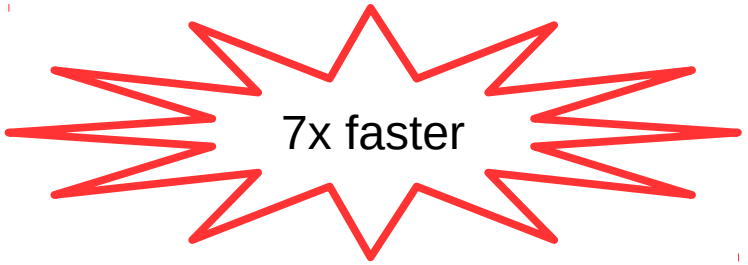


Example

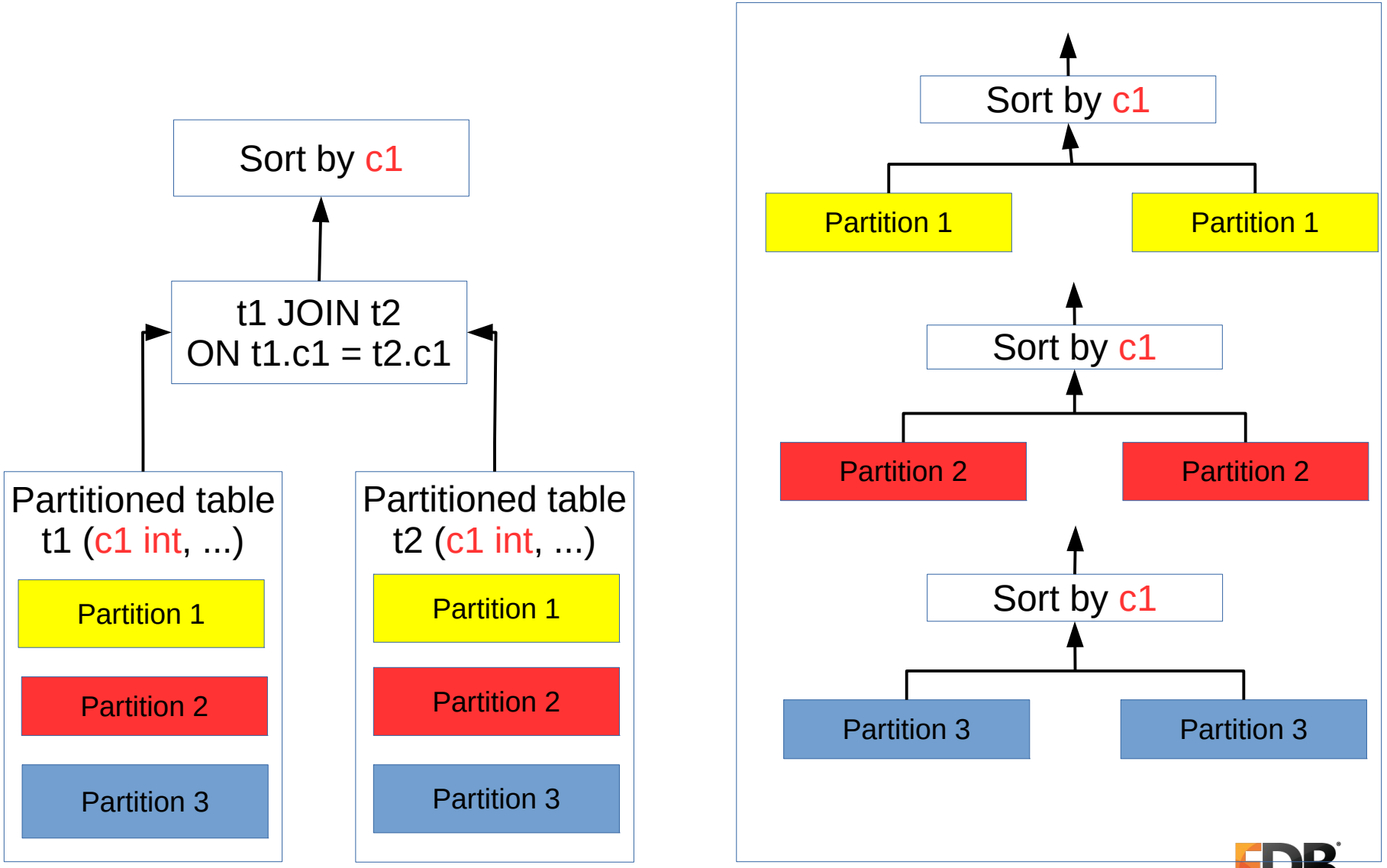
```
Source: Jeevan Chalke's partition-wise aggregate proposal
Query: SELECT a, count(*) FROM plt1 GROUP BY a;
plt1: partitioned table with 3 foreign partitions, each with 1M rows

Query returns 30 rows, 10 rows per partition
enable_partition_wise_agg to false
QUERY PLAN
------------------------------------
 HashAggregate
   Group Key: plt1.a
   ->  Append
         ->  Foreign Scan on fplt1_p1
         ->  Foreign Scan on fplt1_p2
         ->  Foreign Scan on fplt1_p3
 Planning time: 0.251 ms
 Execution time: 6499.018ms ~ 6.5s

enable_partition_wise_agg to true
QUERY PLAN
-----------------------------------------------------
 Append
   ->  Foreign Scan: Aggregate on (public.fplt1_p1 plt1)
   ->  Foreign Scan: Aggregate on (public.fplt1_p2 plt1)
   ->  Foreign Scan: Aggregate on (public.fplt1_p3 plt1)
 Planning time: 0.370ms
 Execution time: 945.384ms ~ .9s
```

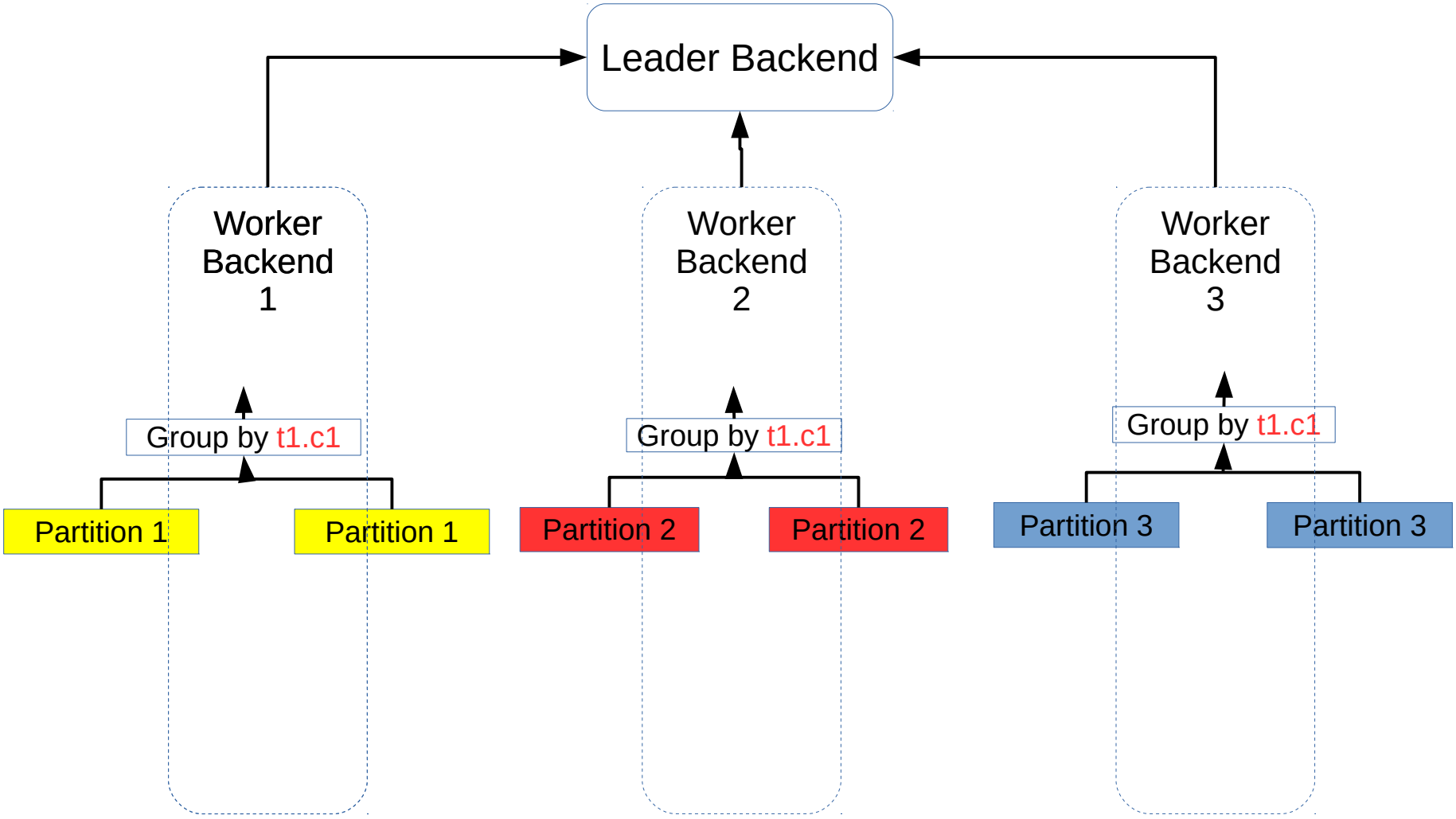


Partition-wise sorting

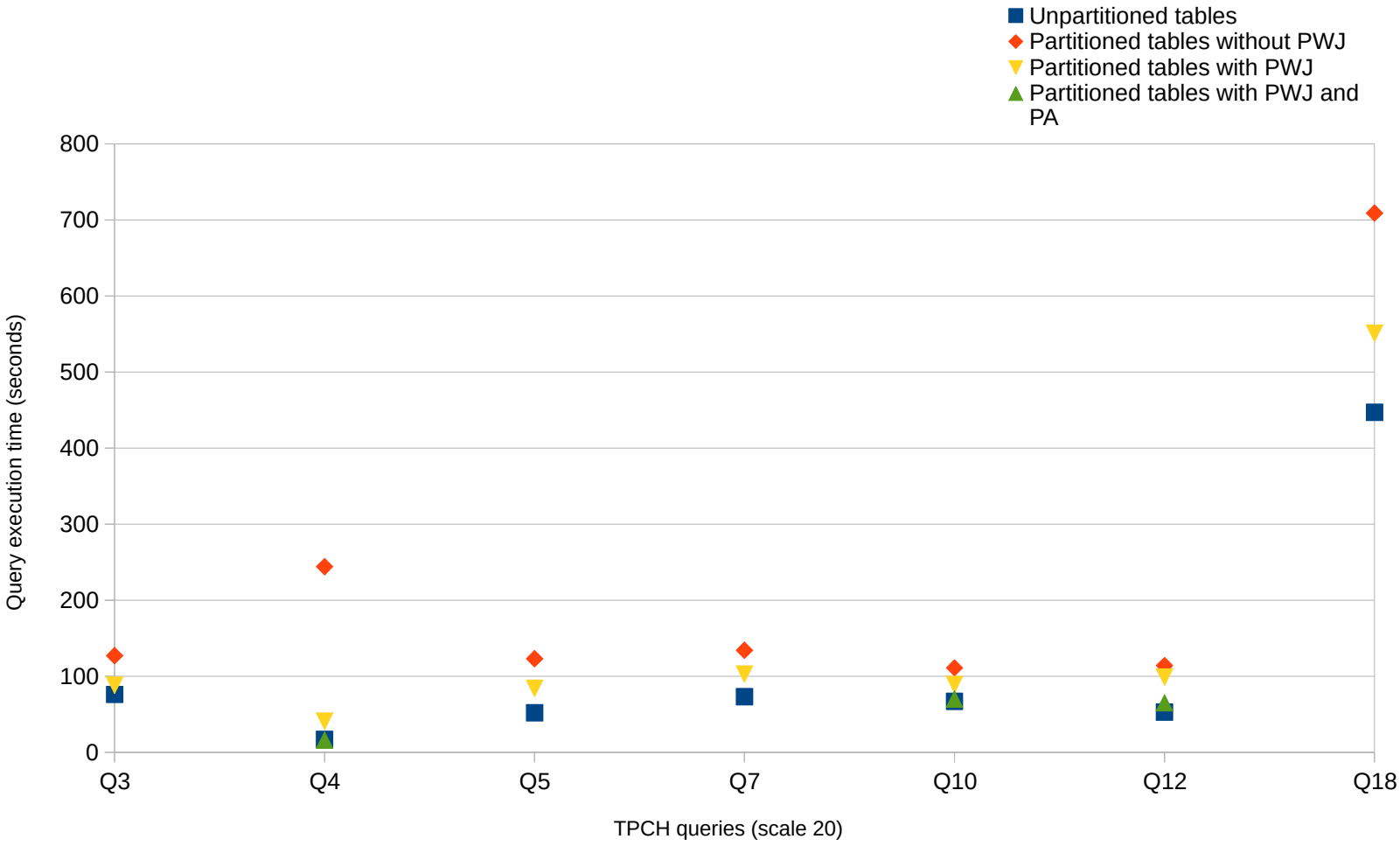


Parallel Append and Partitioning

Parallel Append and Partitioning



TPCH Vs. PWJ + PA



Query Optimization Techniques - patches

- Committed patches
  - Basic partition-wise join - Ashutosh Bapat (EDB)
  - Parallel append – Amit Khandekar (EDB)
- Patches submitted on hackers and being reviewed
  - Partition pruning – Amit Langote (NTT)
  - Run-time partition pruning – Beena Emerson (EDB)
  - Partition-wise aggregation – Jeevan Chalke (EDB)
  - Partition-wise sorting/ordering – Ronan Dunklau, Julien Rouhaud (Dalibo)
- Benchmarking by Rafia Sabih (EDB)


THANK YOU
grazie
merci
spasiba
kam ouen
tak
gratzias
manana
mahalo
kitos
cheers
hvala
welalin
toda
gracias
grassie
thank you
danki
mahalo
danki
gracias
merci
thanks
na gode
mesi
dankon
modupe
talofa
miigwetch
thanks
takk
domo arrigato
danke
kitos
takk
dziekuje
grattitude
takk