- Ashutosh Bapat | 2017.02.03 @ PGConf India

Partition and Conquer large data in PostgreSQL 10

Why to partition tables?

- Manageability
  - Smaller data is easier to manage
  - Partition-wise utility commands
  - Easy bulk load and deletes
- Query/DML Performance
  - Partition elimination (pruning)
  - Partition-wise joins, aggregate
  - Distribute DMLs across partitions
- Data storages based on partition properties
  - “hot” and “cold” partitions
  - Foreign partitions

Why declarative partitioning?
(instead of inheritance based partitioning)

- Maintenance
  - No triggers, rules or constraints to maintain
  - Adding/deleting partition does not require more than a single command
- Faster queries and DMLs
  - No trigger execution overhead
    - DMLs are 10-20 times faster
  - Simpler partition bounds representation
  - Faster partition pruning
  - Partition-wise join, aggregation

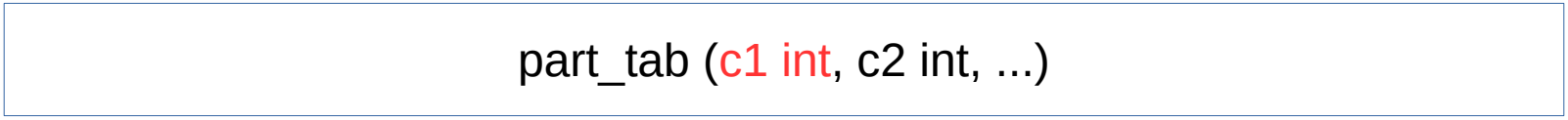
Declarative partitioning in PostgreSQL 10

PostgreSQL Declarative partitioning

- Partitioning methods: List, range, (WIP. hash)
- Partition key: Single or multiple columns, expressions
- Sub-partitioning
  - Partitioned partitions
  - Mixed sub-partitioning
- Development efforts
  - Several previous attempts by many people
  - Amit Langote proposed first patch in August 2015
  - Got committed in Dec 2016, bug fixes, doc changes continue ...

Creating partitioned table

CREATE TABLE part_tab (c1 int, c2 int, …) **PARTITION BY RANGE (c1);**

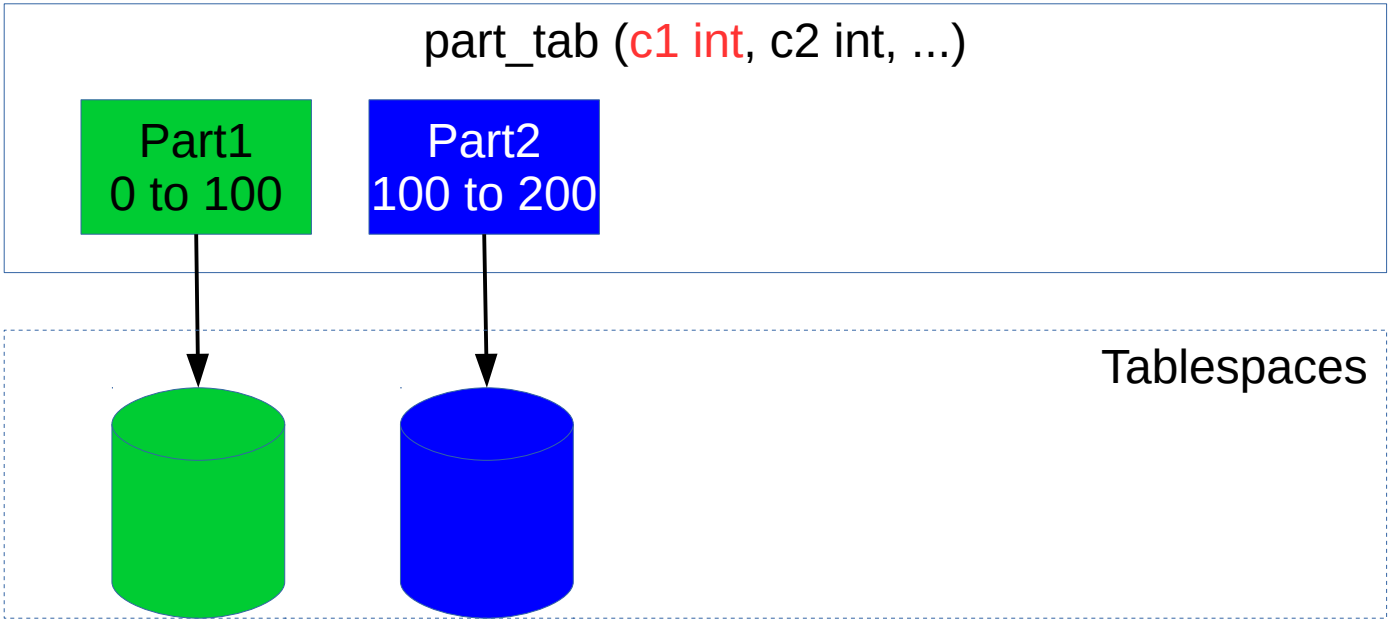


Creating partitioned table

CREATE TABLE part_tab (c1 int, c2 int, …) **PARTITION BY RANGE (c1);**

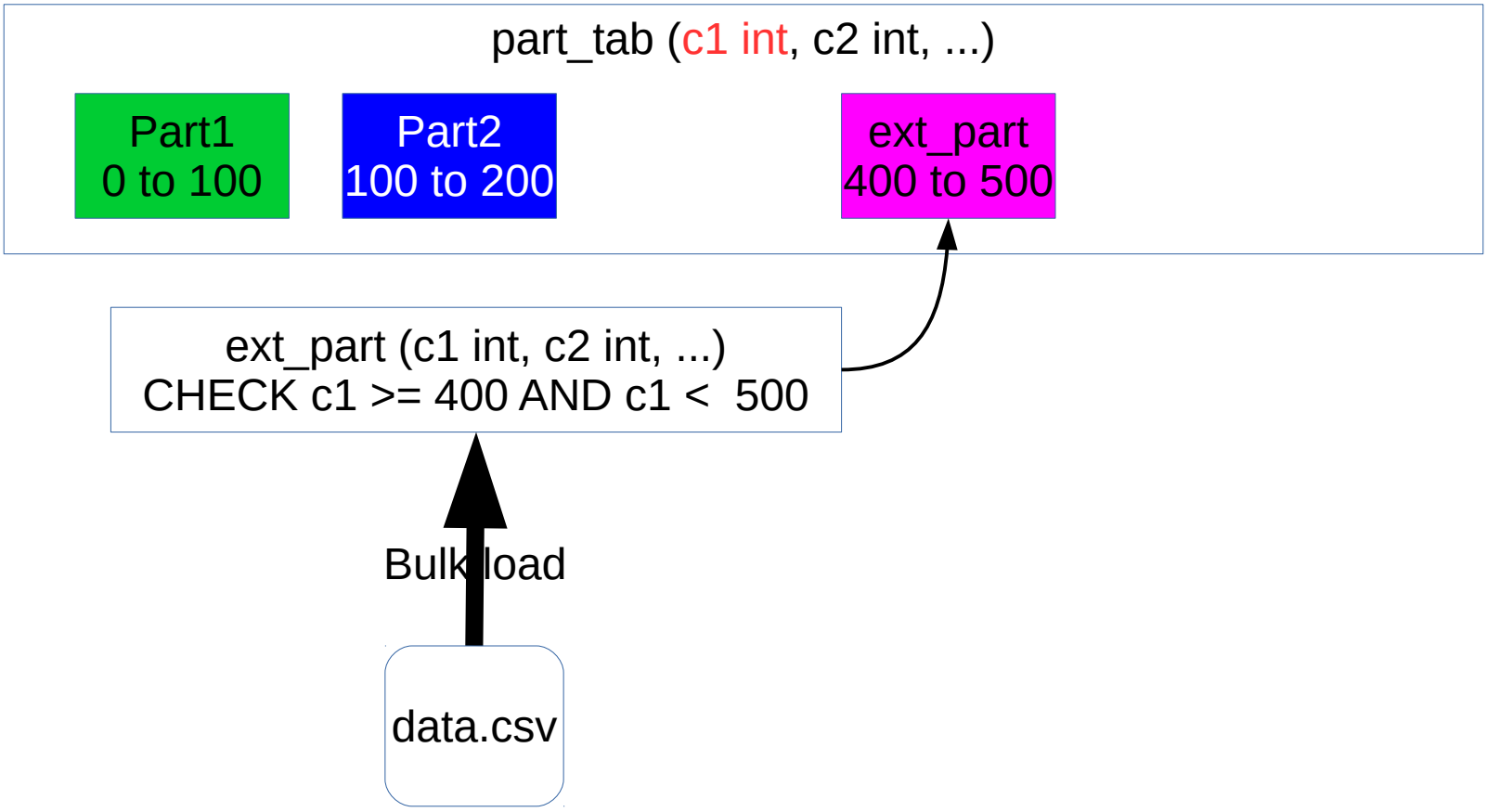
CREATE TABLE part1 **PARTITION OF** part_tab **FOR VALUES FROM (0) TO (100);**

CREATE TABLE part2 **PARTITION OF** part_tab **FOR VALUES FROM (100) TO (200);**



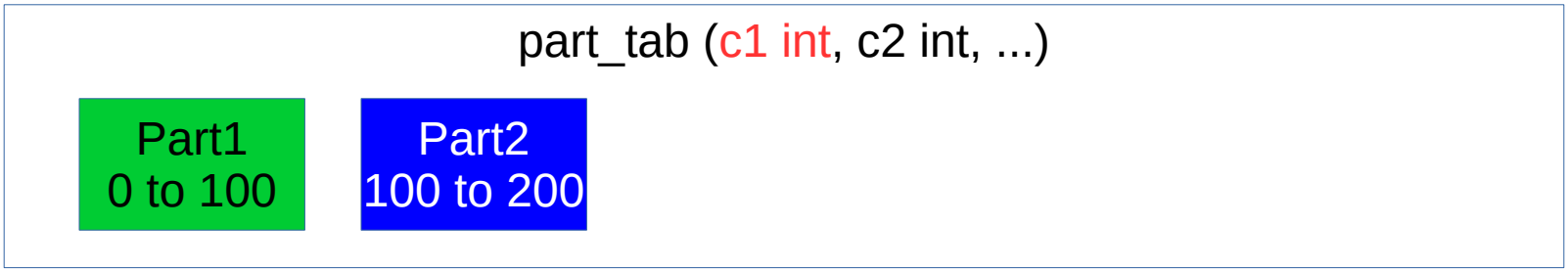
ATTACH partition

ALTER TABLE part_tab **ATTACH PARTITION** ext_part FOR VALUES FROM (400) to (500);



DETACH partition

ALTER TABLE part_tab DETACH PARTITION ext_part;



ext_part (c1 int, c2 int, ...)
CHECK c1 >= 400 AND c1 < 500

Partitions are tables

- Same columns as parent table
- Partition specific constraints, defaults, indexes, storage parameters
- Tablespace separate from that of parent
  - “hot” and “cold” partitions
- VACUUM, ANALYZE, CLUSTER can be run separately
  - Utilities don't block the whole table
  - Work where it's required

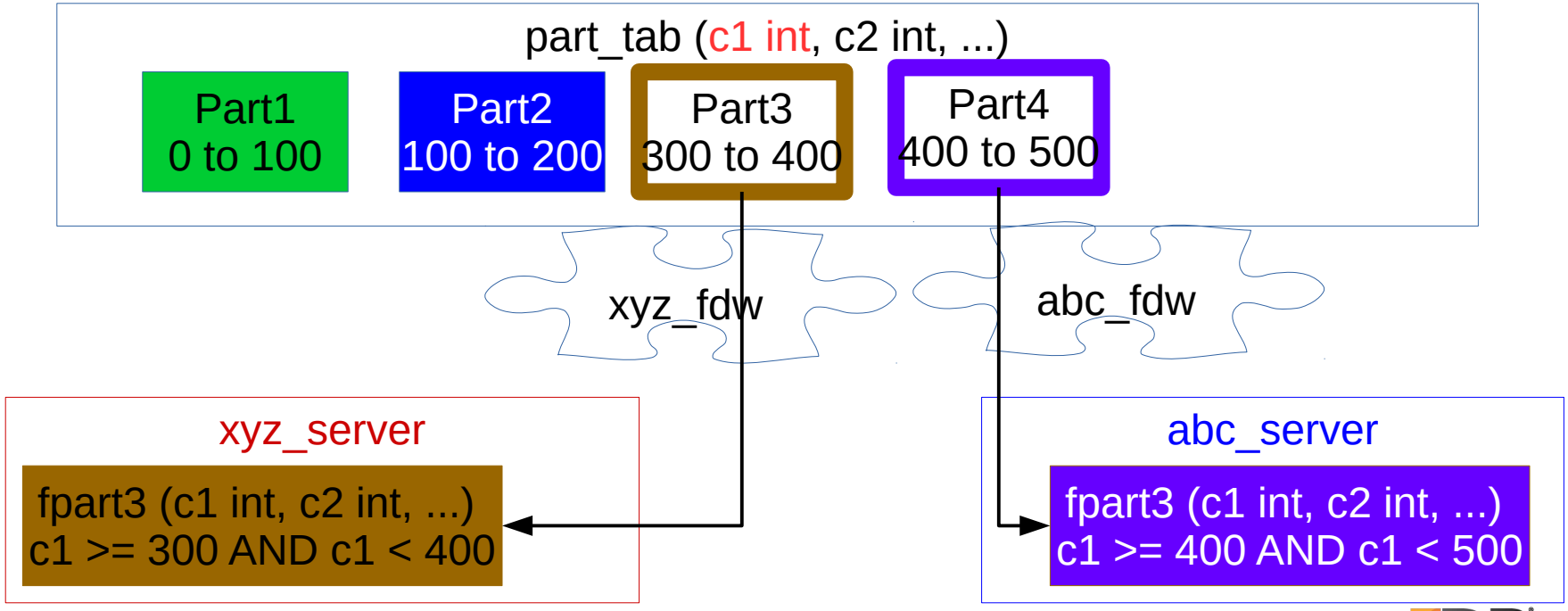
Foreign partitions

CREATE TABLE part_tab (c1 int, c2 int, …) **PARTITION BY RANGE (c1)**;

...

CREATE **FOREIGN** TABLE part3 **PARTITION OF** part_tab **FOR VALUES FROM (300) TO (400) SERVER xyz_server**;

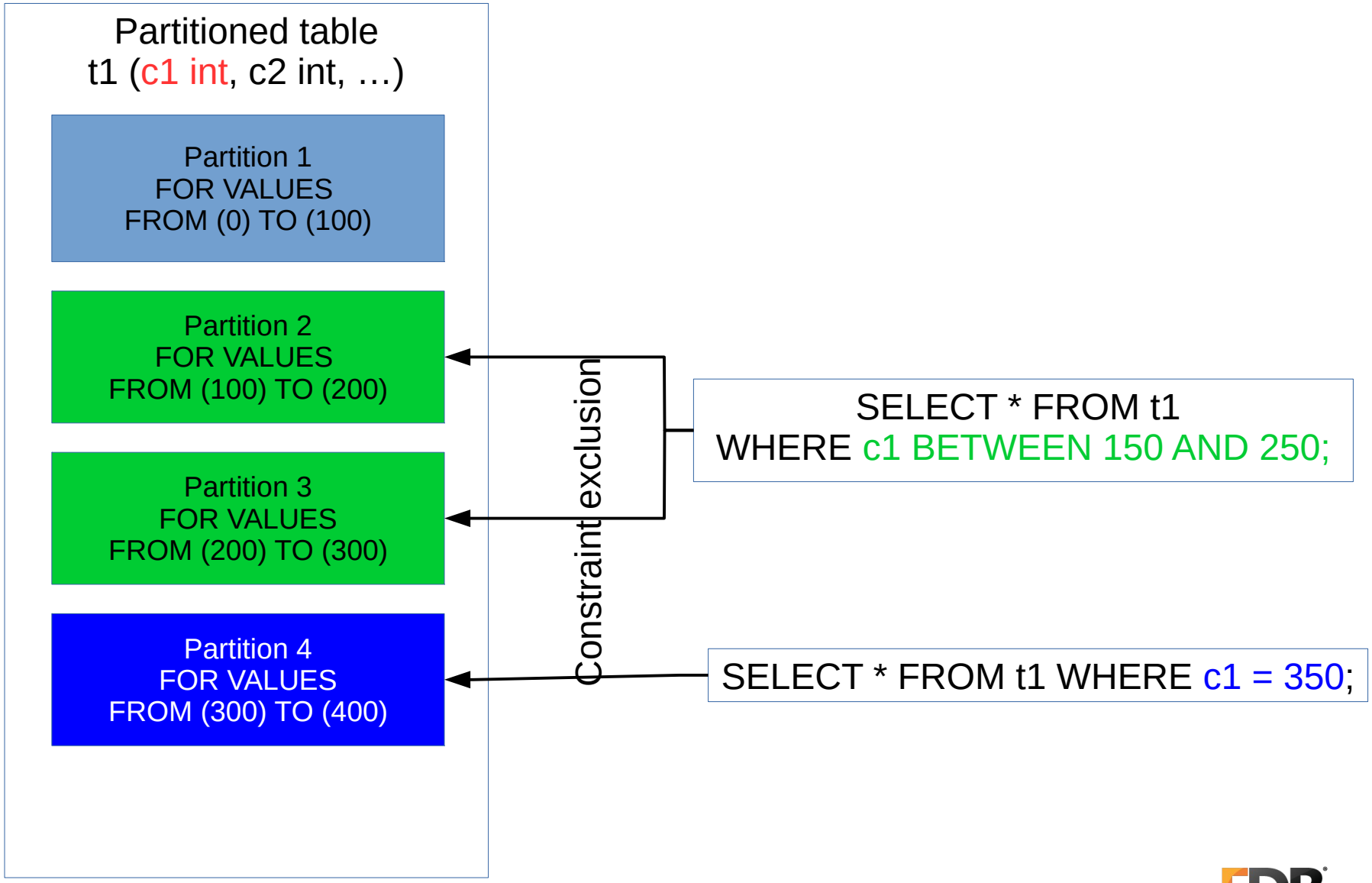
CREATE **FOREIGN** TABLE part4 **PARTITION OF** part_tab **FOR VALUES FROM (400) TO (500) SERVER abc_server**;



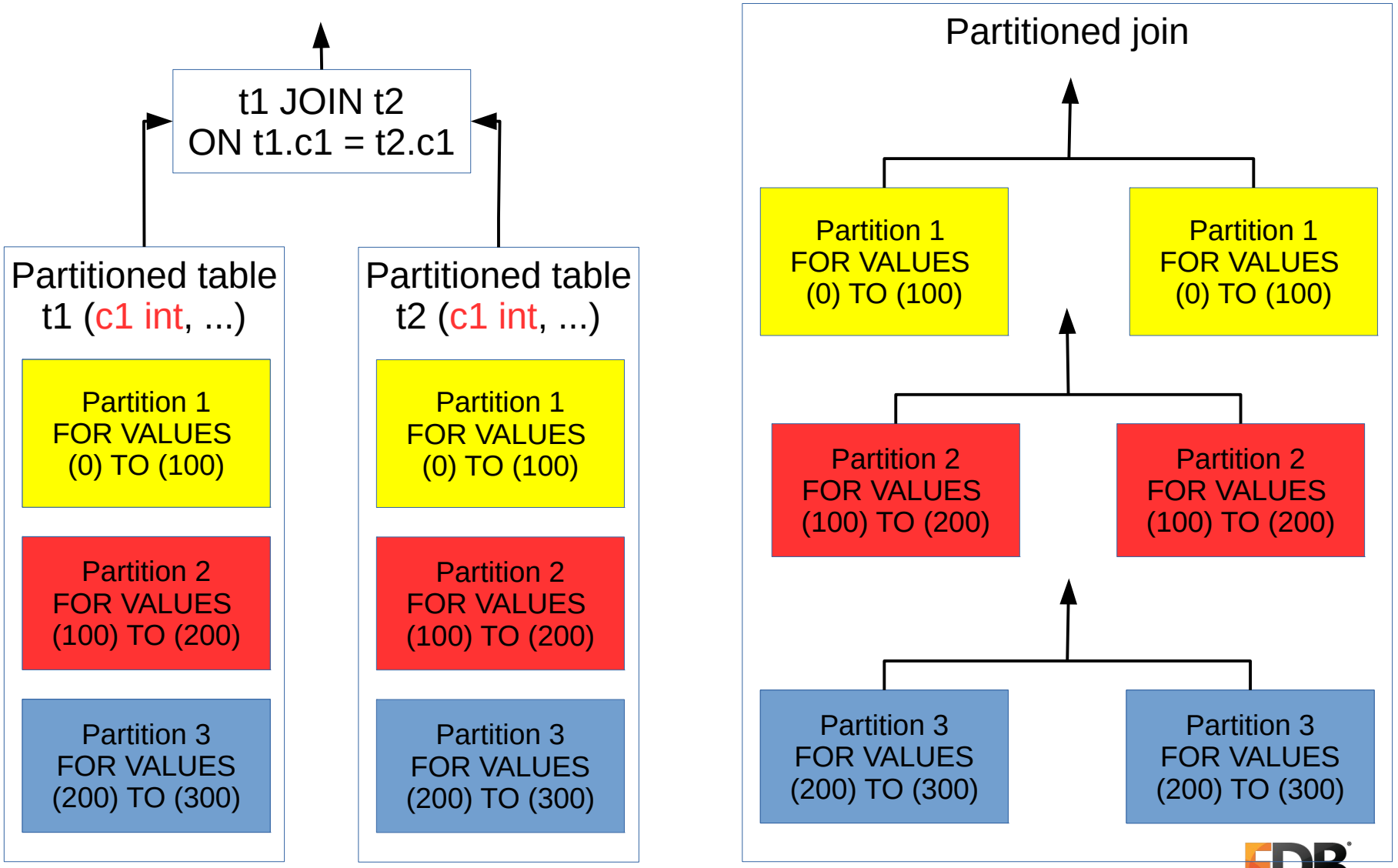
Query optimizations

- partition pruning
- partition-wise join
- partition-wise aggregation

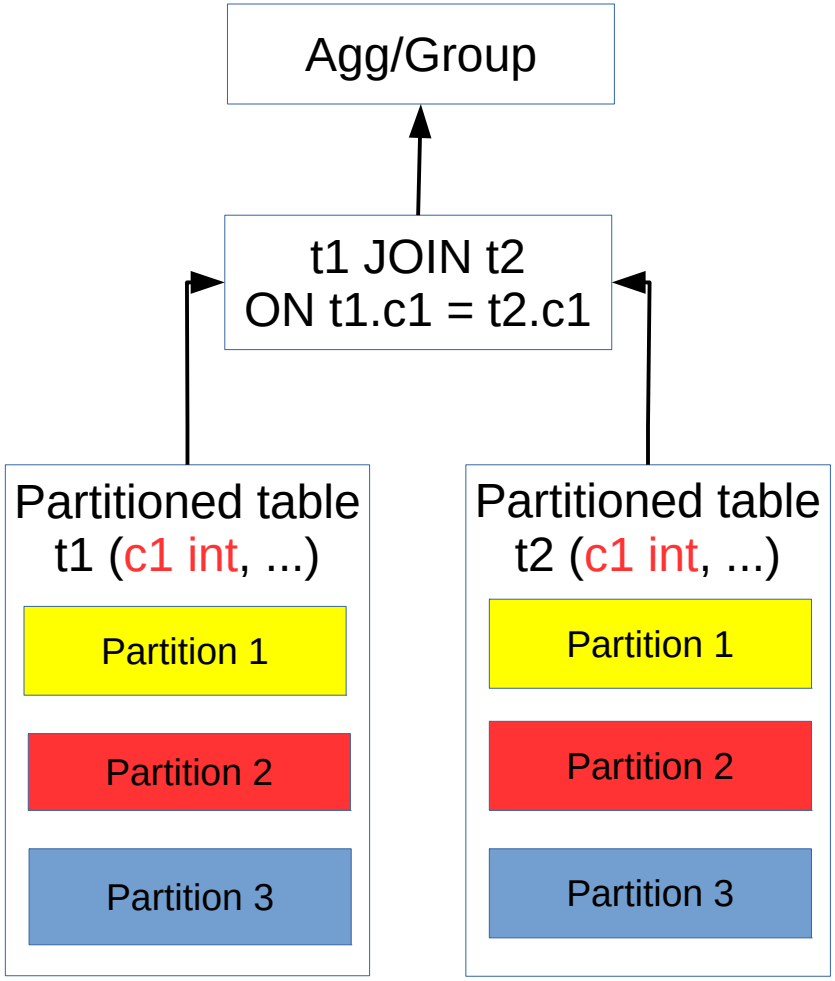
Query optimization: partition pruning

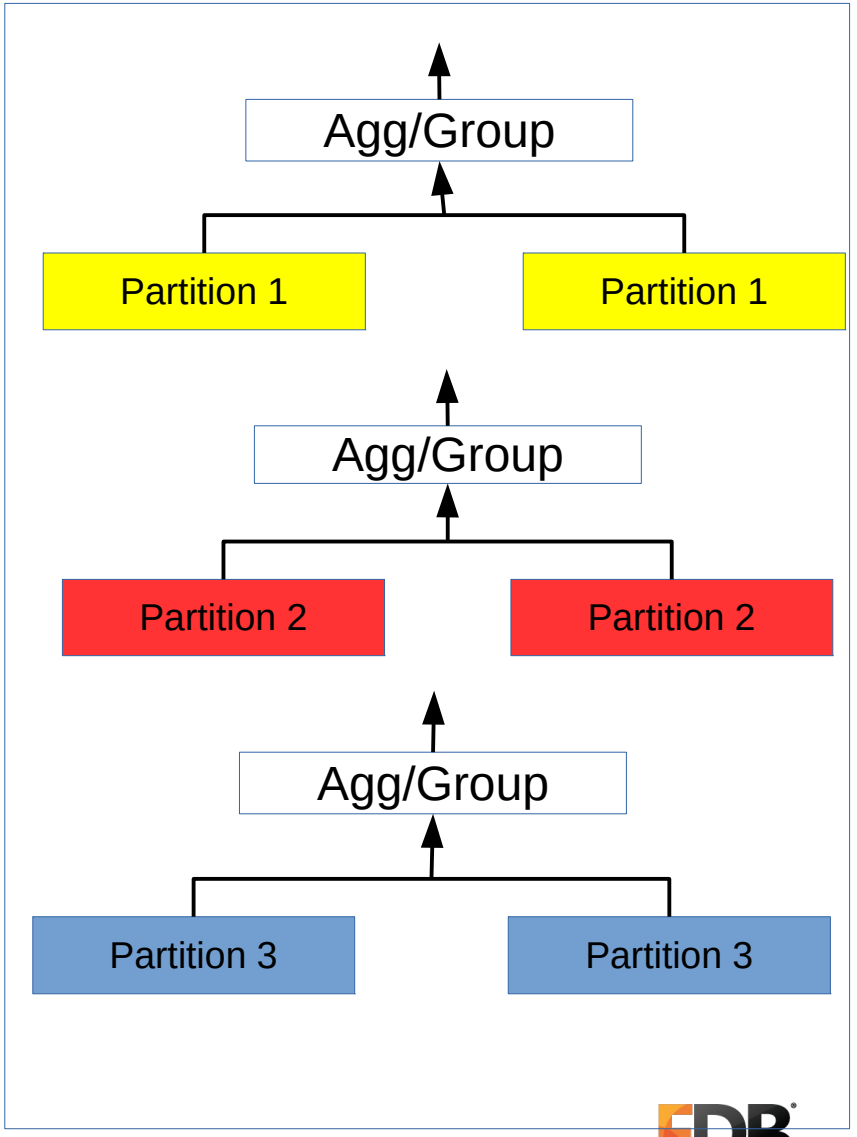


Query optimization: partition-wise join



Query optimization: partition-wise aggregation





Choosing partitioning scheme

- Bad partitioning is worse than no partitioning
- Query optimization
  - Partition key is the key to success
  - Columns in conditions
  - include it in the query
- Storage management
  - Columns that decide the storage
  - Usually timestamp of the row
  - Easy to add and drop partitions
- Keep an eye on current limitations
  - Expected to reduce with next few releases

Limitations

- We have just begun …
- No triggers on partitioned table
  - Need to create those for each of the partitions
- No SPLIT, MERGE, EXCHANGE partition
- No global indexes
  - No primary key, unique constraints
- Foreign partitions
  - INSERTs via parent table not supported
  - No ACID support for transactions involving multiple foreign servers


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