PostgreSQL: Decoding Partition

EDB POSTGRES

Beena Emerson

February 14, 2019

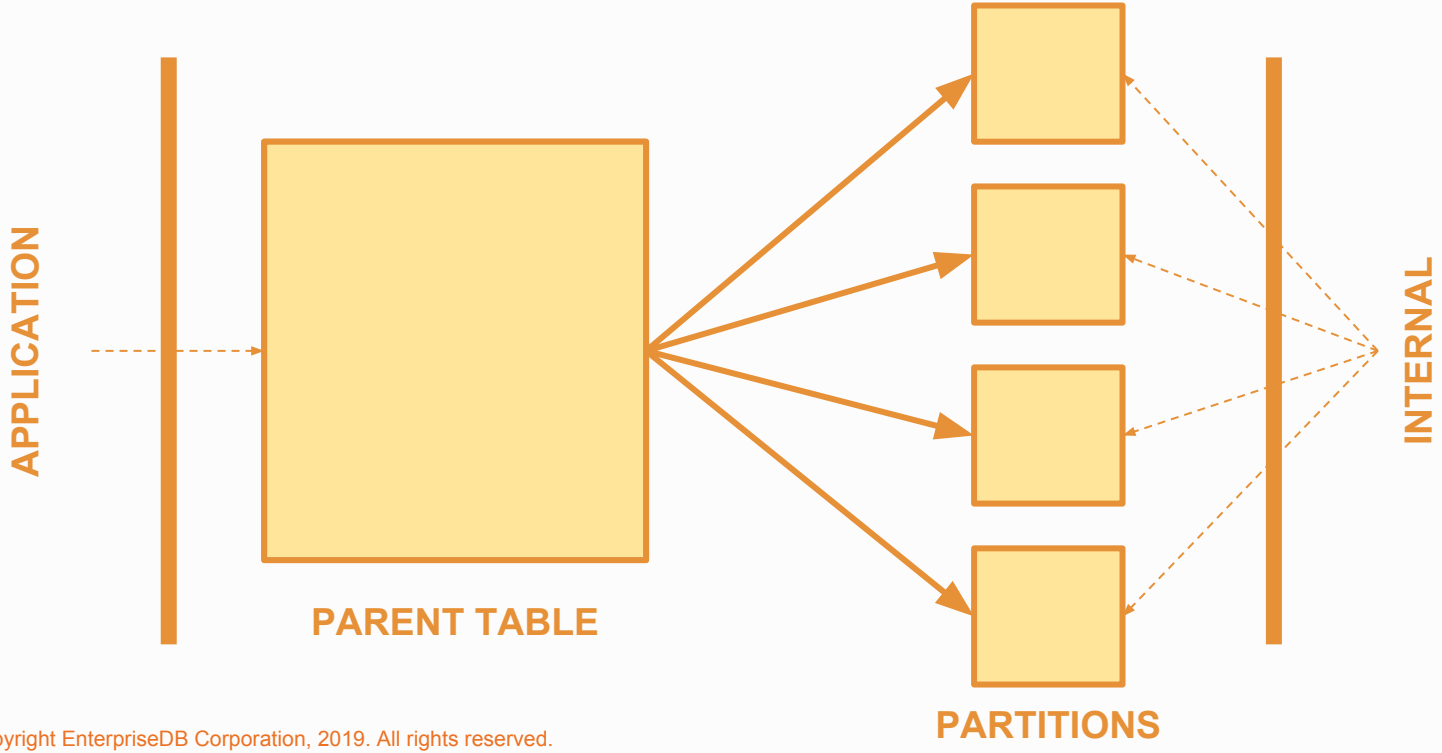
PGConf India

INTRODUCTION

- **What** is Partitioning?
- **Why** partition?
- **When** to Partition?

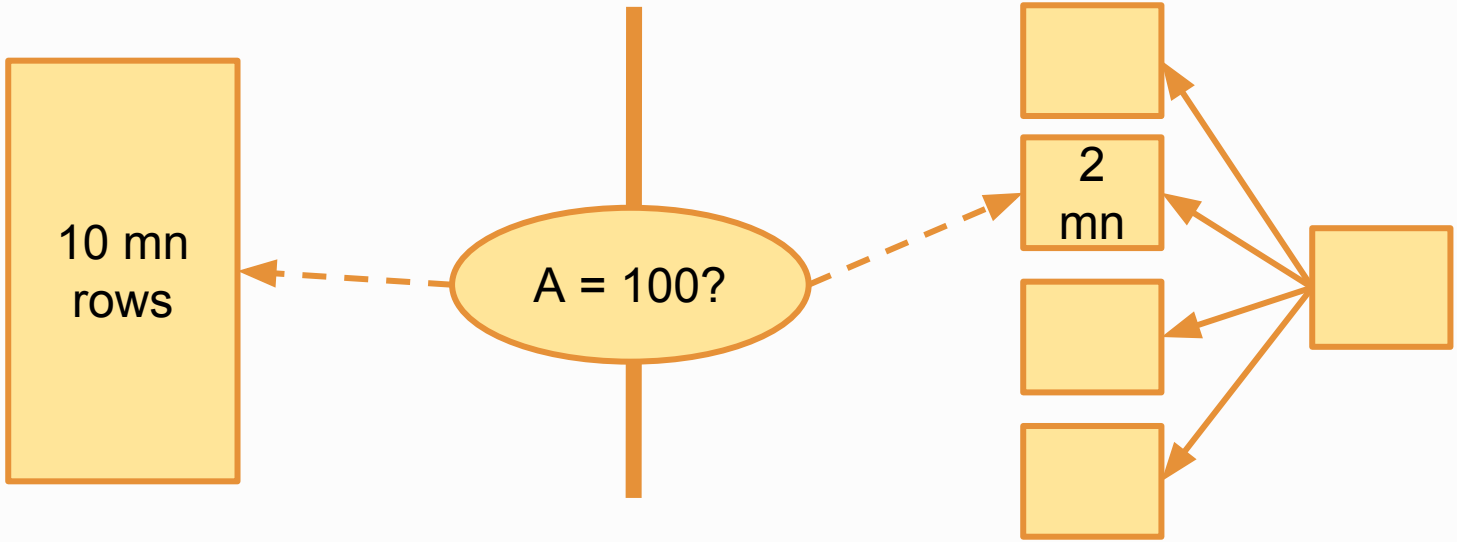
What is Partitioning?

Subdividing a database table into smaller parts.



Why Partition?

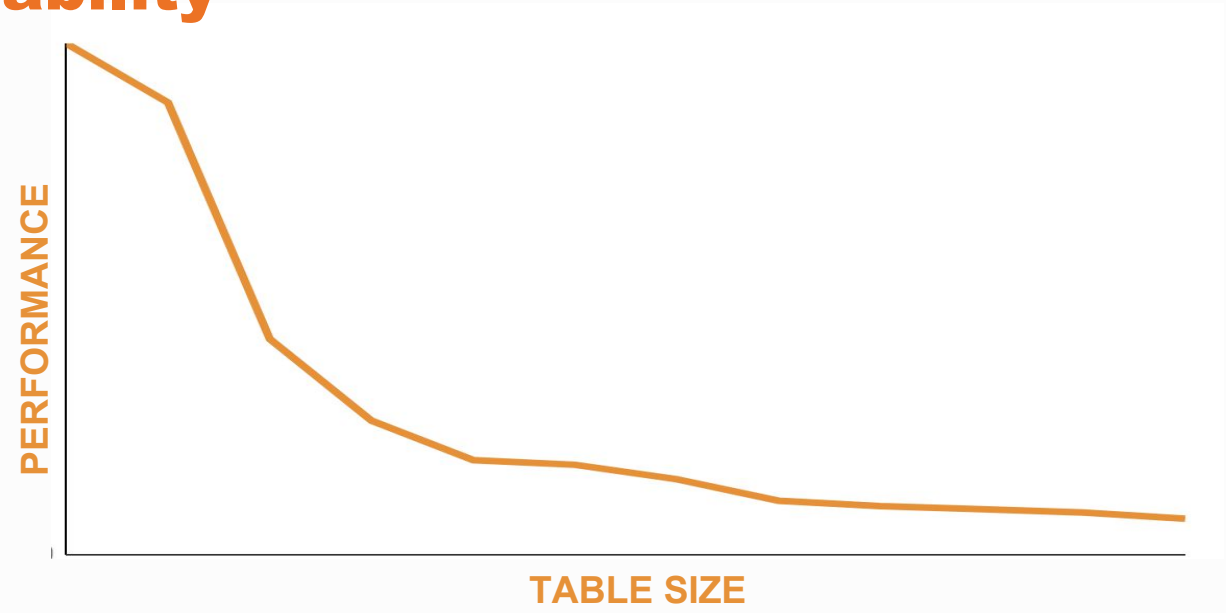
Faster Data Access



Scanning smaller tables takes less time.

Why Partition?

Scalability

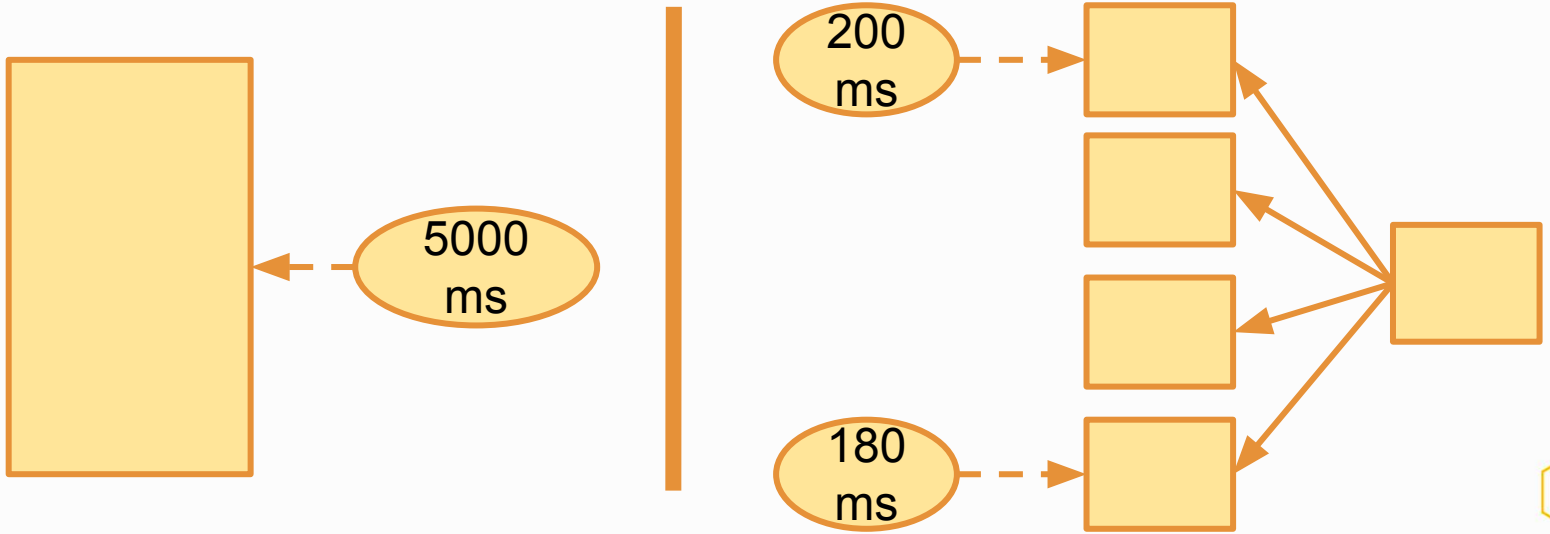


With partition, we would be dealing with multiple tables of smaller sizes.

EDB POSTGRES

Why Partition?

Easy Maintenance



Commands are faster and can be run in parallel

EDB POSTGRES

When to Partition?

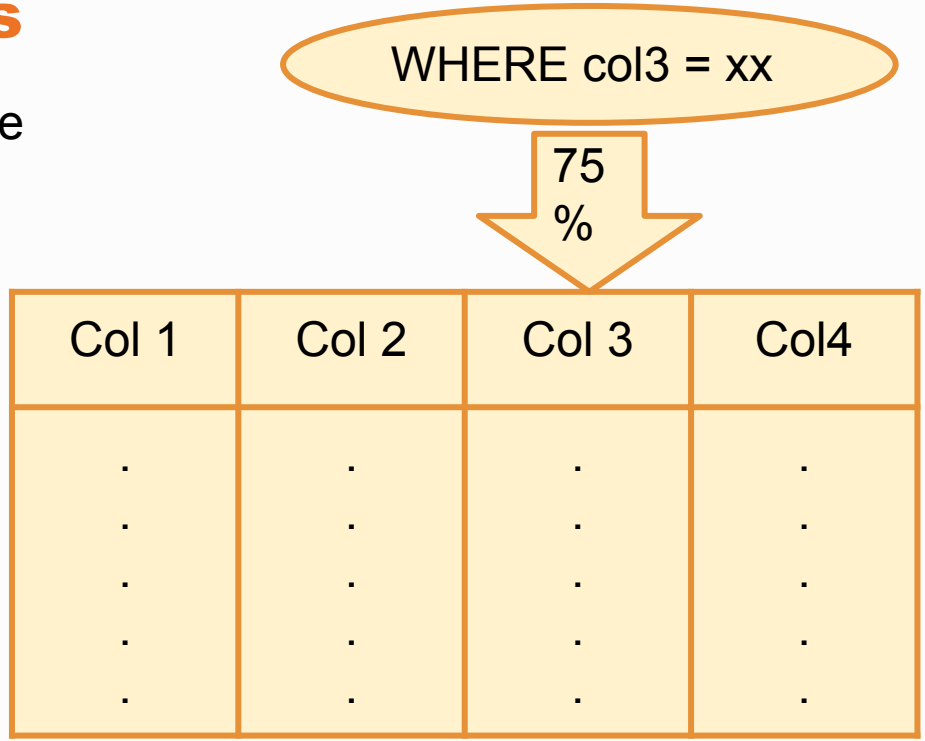
Table Size

- Rule of thumb: size of the table exceeds physical memory of the database server.
- Extensive DML operations

When to Partition?

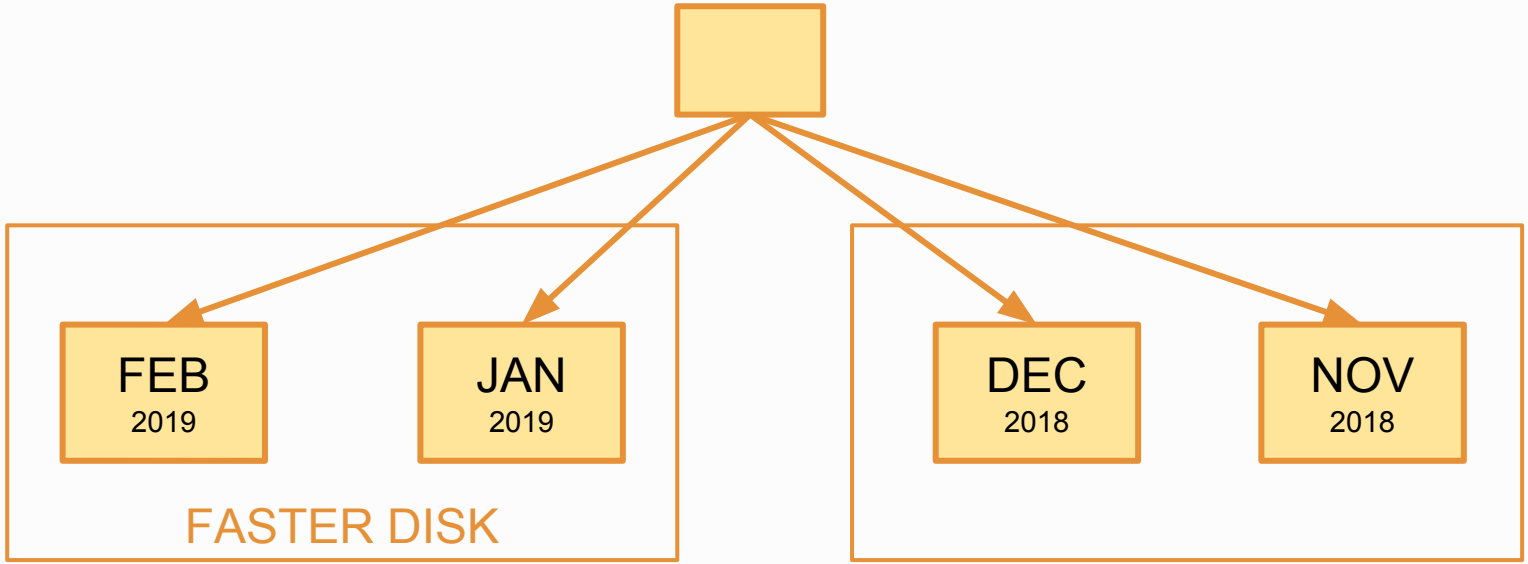
Targeted Columns

Good partition key candidate



When to Partition?

Hot and Cold Data



CAUTION!

Bad partitioning is worse than no partitioning!

Choose partitioning keys, strategy and no of partitions after thorough analysis and testing.

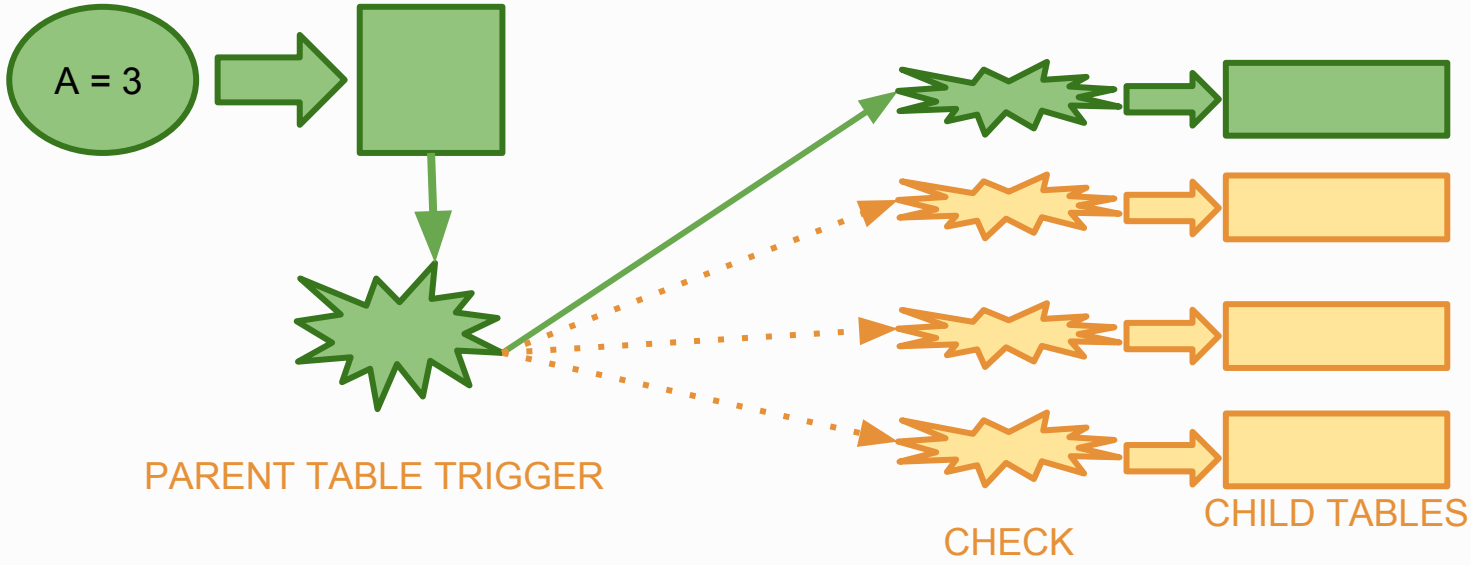
EDB POSTGRES

PARTITIONING IN POSTGRESQL

- Inheritance (Trigger- based) Partition
- Declarative Partitioning

Inheritance (Trigger- based) Partition

- Since PostgreSQL 8.1
- Table inheritance simulating partitioning



Inheritance (Trigger- based) Partition

Problems

- Partitions tracking is manual and error-prone.
- Slower Performance
- Constraints may not be mutually exclusive

Declarative Partitioning

- Introduced in PostgreSQL 10
- Somewhat automates creation of partitions
- Easy partition maintenance
  - No overlapping boundaries
  - Add/Delete partition easily
  - Simpler syntax
- Scope for better performance

Declarative Partitioning

Terms

- Partitioned table - parent table
- Partition key - column on which table is partitioned
- Partitions - the smaller child tables
- Partition bounds - constraints of each partition

Declarative Partitioning

- Create Partitioned Table

```
CREATE TABLE parent ( <col list > )
   PARTITION BY <partition strategy>
    ( <partition key> );
```

- Create Partitions

```
CREATE TABLE child
   PARTITION OF parent <partition bounds>;
```

… repeat for required partitions

Declarative Partitioning

- Existing table can be added as a partition to a partitioned table

```
ALTER TABLE parent ATTACH PARTITION p1 <bounds>;
```

  All entries in the table will be checked.

- A partition can be removed and made standalone

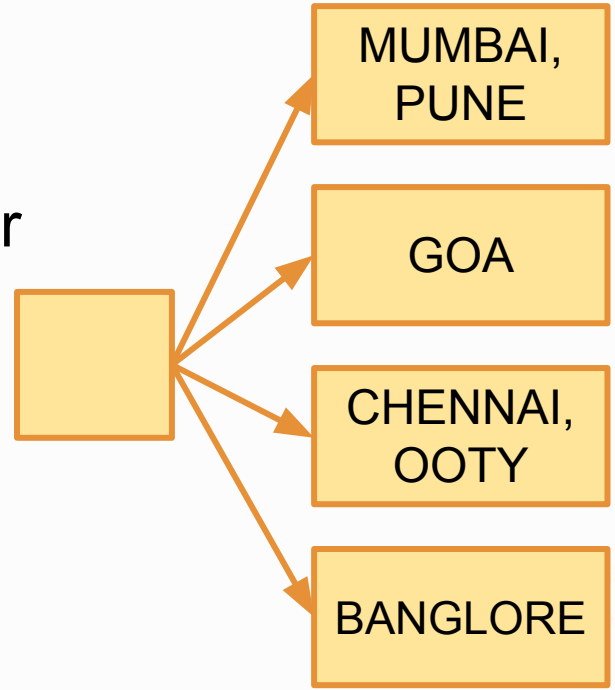
```
ALTER TABLE parent DETACH PARTITION child;
```

PARTITION STRATEGY AND TYPE

- Strategy
  - List
  - Range
  - Hash
- Types
  - Multi-column partitioning
  - Multi-level partitioning

List Strategy

- PostgreSQL 10
- Explicitly specify key values
  - Single or multiple value per partition



List Strategy

- Create Partitioned Table

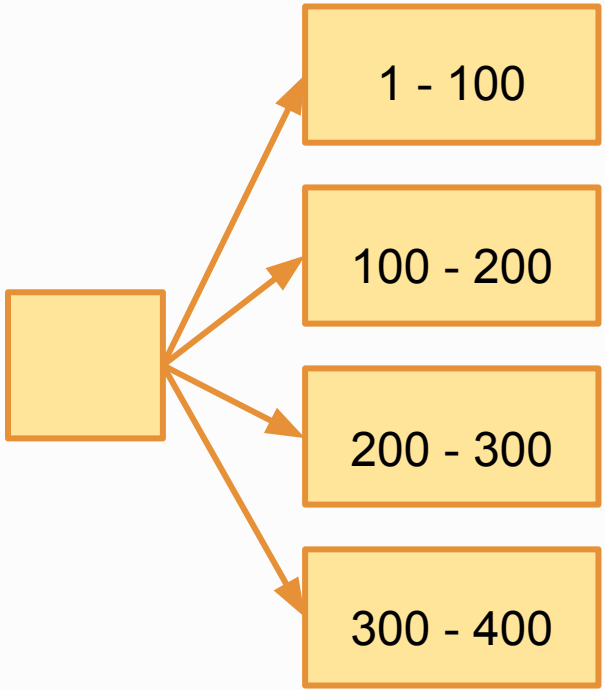
```
CREATE TABLE parent ( <col list > )
   PARTITION BY LIST ( <partition key> );
```

- Create Partitions

```
CREATE TABLE child PARTITION OF parent
   FOR VALUES
      IN ( <val1> [, <val2>] ) ;
```

Range Strategy

- PostgreSQL 10
- Range boundaries
  - Lower inclusive (>=)
  - Upper exclusive (<)
- Unbounded values
  - MINVALUE
  - MAXVALUE



Range Strategy

- Create Partitioned Table

```
CREATE TABLE parent ( <col list > )
   PARTITION BY RANGE ( <partition key> );
```

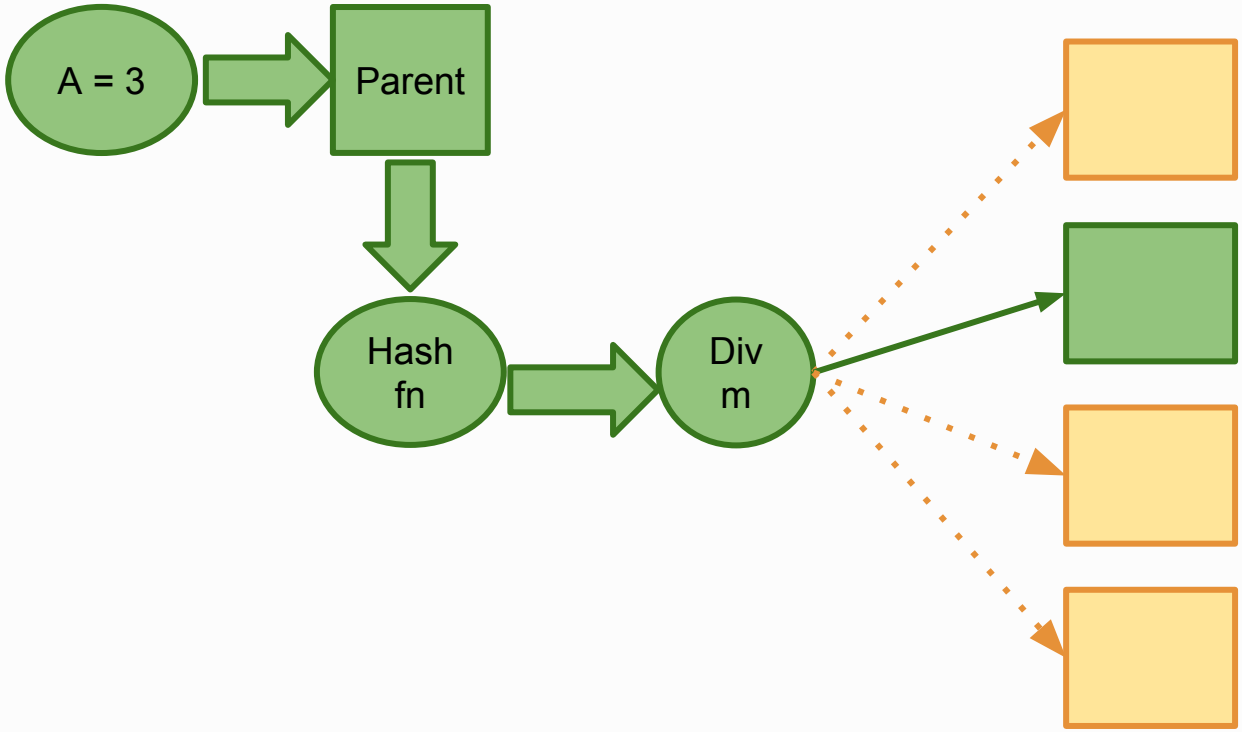
- Create Partitions

```
CREATE TABLE child PARTITION OF parent
   FOR VALUES
      FROM ( <lower> ) TO ( <upper> ) ;
```

Hash Strategy

- PostgreSQL 11
- Data does not have natural boundaries
- Specify modulus and remainder
  - remainder < modulus
  - Remainder - non-negative integer
  - Modulus - positive integer
- Rows spread on hash value of the partition key

Hash Strategy



Hash Strategy

- Create Partitioned Table

```
CREATE TABLE parent ( <col list > )
   PARTITION BY HASH ( <partition key> );
```

- Create Partitions

```
CREATE TABLE child PARTITION OF parent
   FOR VALUES
    WITH ( MODULUS <m>, REMAINDER <r> );
```

Multi-Column Partitioning

- Multiple columns as partition key
- Supported for range and hash
- Column limit : 32

Multi-Column Partitioning

Range Partition

```
CREATE TABLE rparent (col1 int, col2 int)
  PARTITION BY RANGE (col1, col2);

CREATE TABLE rpart_1 PARTITION OF rparent
  FOR VALUES FROM (0, 0) TO (100, 50);

CREATE TABLE rpart_2 PARTITION OF rparent
  FOR VALUES FROM (100, 50)
    TO (MAXVALUE, MAXVALUE);
```

Multi-Column Partitioning

Range Partition

- Every column following `MAXVALUE / MINVALUE` must also be the same.
- The row comparison operator is used for insert
  - Elements are compared left-to-right, stopping at first unequal pair of elements.
    Consider partition `(0, 0) TO (100, 50)`
    `(0, 199), (100, 49)` fits while
    `(100, 50)` does not.

Multi-Column Partitioning

Hash Partition

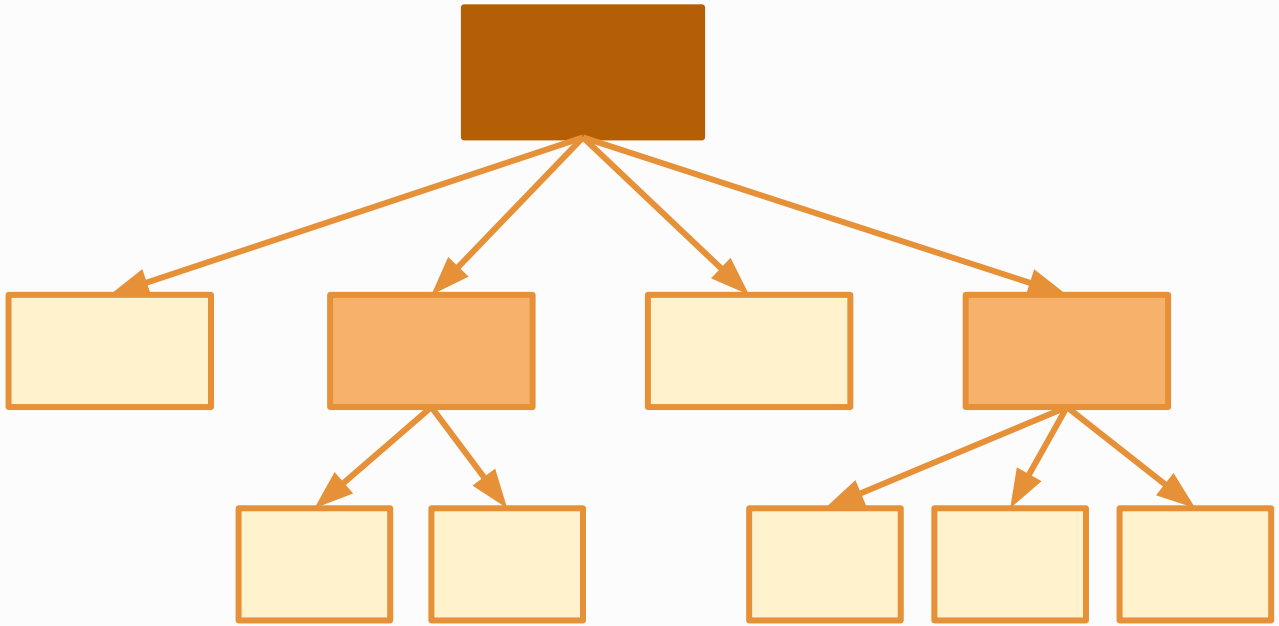
```
CREATE TABLE hparent (col1 int, col2 int)
PARTITION BY HASH (col1, col2);

CREATE TABLE hpart_2 PARTITION OF hparent FOR
VALUES WITH (MODULUS 3, REMAINDER 2);

CREATE TABLE hpart_1 PARTITION OF hparent FOR
VALUES WITH (MODULUS 3, REMAINDER 1);

CREATE TABLE hpart_0 PARTITION OF hparent FOR
VALUES WITH (MODULUS 3, REMAINDER 0);
```

Multi-Column Partitioning

Hash Partition

- Only a single modulus, remainder pair is used.
- The hash of each of partition key is calculated and then combined to get a single hash value which is divided by the modulus specified.

Multi-level Partitioning



Multi-level Partitioning

- Create a partitioned partition

```
CREATE TABLE child1 PARTITION OF parent
  FOR VALUES FROM (0) TO (100)
  PARTITION BY LIST (col3);
```

- Attach a partitioned table as a partition.
- There is no check to ensure that the sub partition bounds are a subset of the partition's own bound.

```
CREATE TABLE child1_1 PARTITION OF child1
  FOR VALUES IN (500, 2000);
```

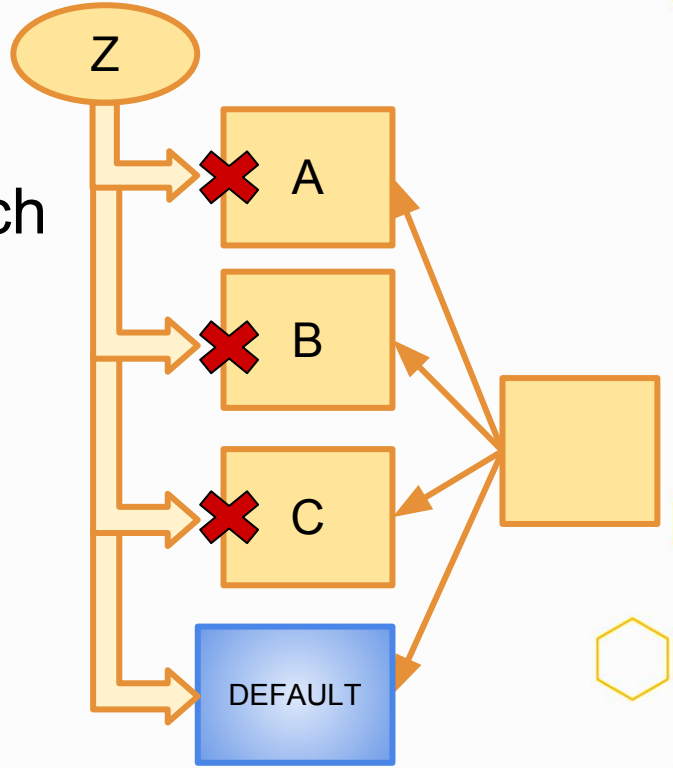
OTHER FEATURES

- Default Partition
- Handling NULL values
- Inherit Constraints
- Update tuple routing
- Foreign Table Partitions
- Partition Pruning

Default Partition

- PostgreSQL 11
- Catch tuples that do not match partition bounds of others.
- Support for: list, range
- Syntax:

```
CREATE TABLE child
   PARTITION OF parent
      DEFAULT
```

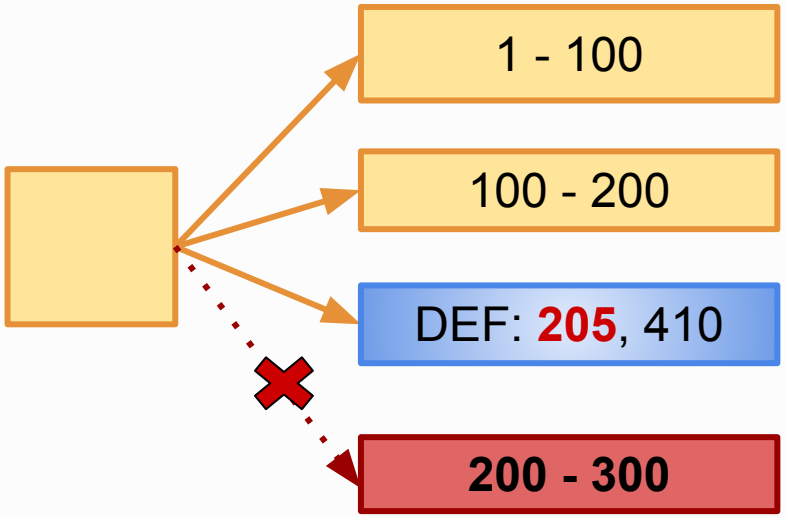


(NOT (col1 IS NOT NULL) AND
(col1 = ANY (ARRAY['A', 'B', 'C'])))

Default Partition

Add new partition

Default partition should not have rows that satisfy the new partition.




```
ERROR:  updated partition constraint for default
partition "part_def" would be violated by some row
```

Handling NULL values

- Hash routes based on the hash.
- Range: can be inserted in default partition.
- List: A partition can be made to accept NULL else it is routed to default partition.

```
CREATE TABLE child PARTITION OF lparent
    FOR VALUES
     IN ( NULL ) ;
```

Inherit Constraints

Constraint Types

- Check
- Not-Null
- Unique
- Primary Keys
- Foreign Keys
- Exclude

Inherit Constraints

PostgreSQL 10

- Supported: Check, Not Null
- Other constraints can be explicitly added to individual partition..
- No cascading inheritance

Inherit Constraints

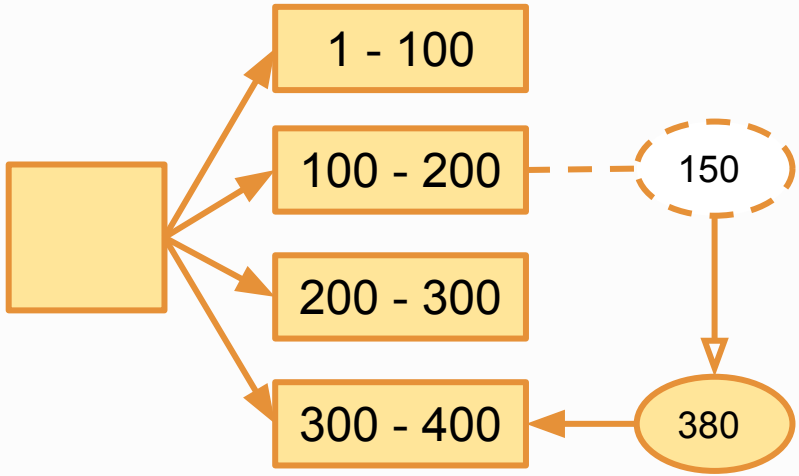
PostgreSQL 11

- Added Support: unique, primary and foreign keys
- Not supported: exclude constraints
- Cascading: New constraints to parent propagates to partitions.
- Other constraints can be explicitly added to individual partition.

Update Tuple Routing

PostgreSQL 11

- When an updated tuple no longer fits the partition, route it to the correct one.



Foreign Table Partitions

- Can add foreign table as a partition.

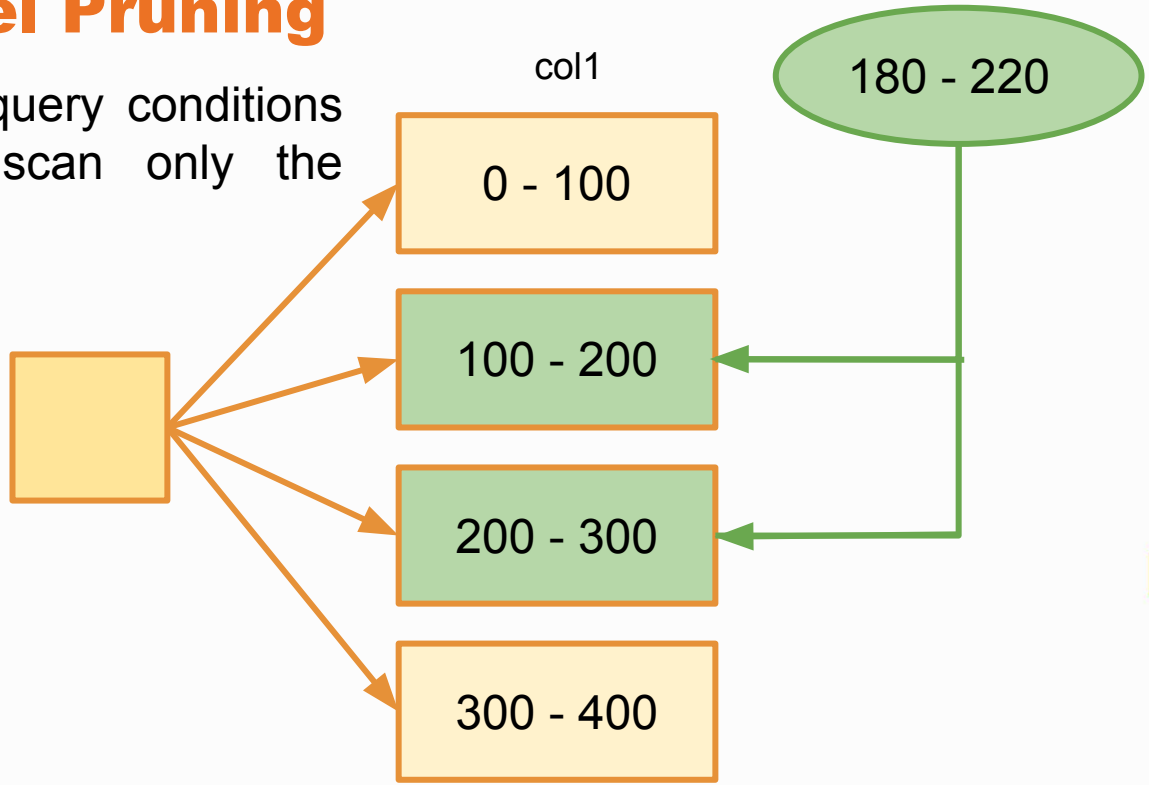
```
CREATE FOREIGN TABLE lpart_f1 PARTITION OF
lparent FOR VALUES IN ('C' , 'E') SERVER s0
```

- Tuples can be inserted from v11.
- Cannot be partitioned.
- Updated tuples can me moved to foreign partitions but not from them.

Partition Pruning

Planner Level Pruning

Depending on the query conditions on partition key, scan only the relevant partitions.



Partition Pruning

Planner Level Pruning

```
=# EXPLAIN (COSTS OFF, TIMING OFF, ANALYZE) SELECT *
FROM rparent WHERE col1 BETWEEN 180 AND 220;

 Append (actual rows=41 loops=1)
   ->  Seq Scan on rpart_2 (actual rows=20 loops=1)
         Filter: ((col1 >= 180) AND (col1 <= 220))
         Rows Removed by Filter: 80
   ->  Seq Scan on rpart_3 (actual rows=21 loops=1)
         Filter: ((col1 >= 180) AND (col1 <= 220))
         Rows Removed by Filter: 79
```

Partition Pruning

Runtime Partition Pruning

- PostgreSQL 11
- Performed at two levels
  - Executor Initialization - prepared query
  - Actual Execution - nested loop join

Partition Pruning

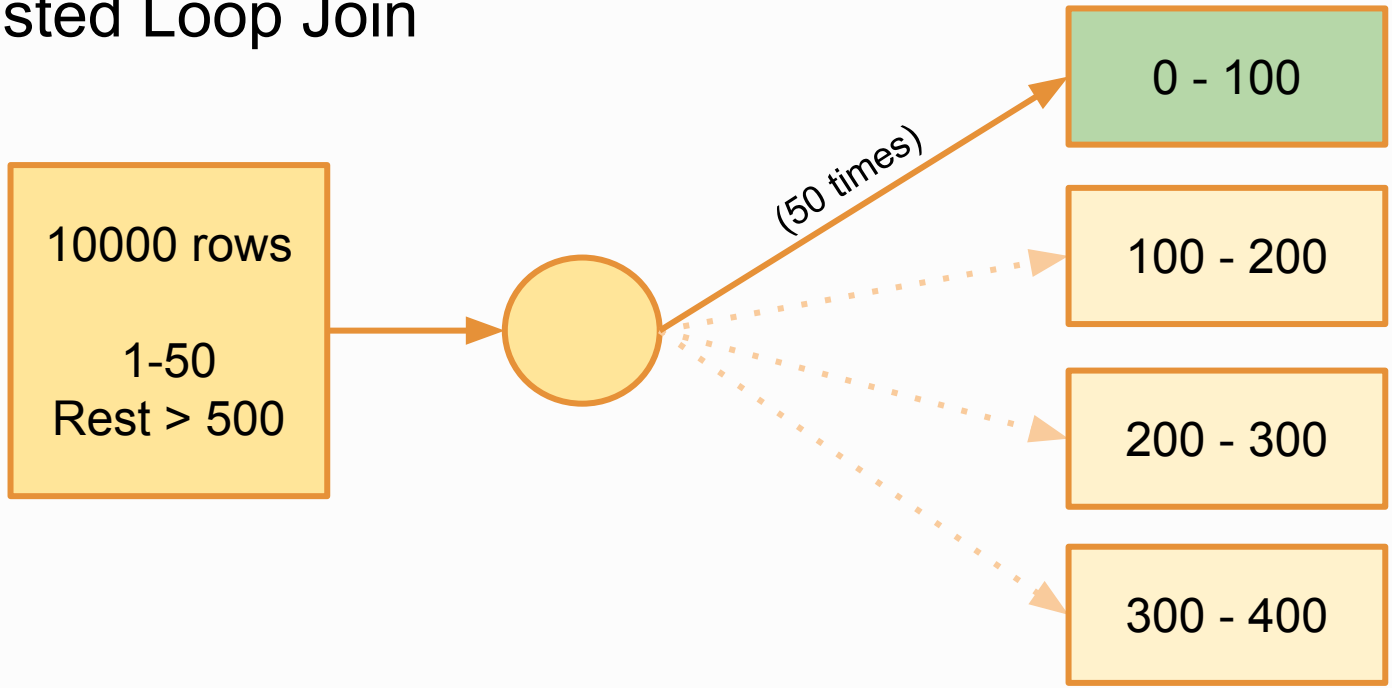
Runtime Pruning - Executor Initialization

```
=# PREPARE r_q1 (int) as SELECT * from rparent where
col1 < $1;
=# EXPLAIN execute r_q1(150);

 Append  (cost=0.00..168.06 rows=3012 width=8)
   Subplans Removed: 2
   ->  Seq Scan on rpart_1
         Filter: (col1 < $1)
   ->  Seq Scan on rpart_2
         Filter: (col1 < $1)
(6 rows)
```

Partition Pruning

Runtime Pruning - Actual execution

Nested Loop Join



Partition Pruning

Runtime Pruning - Actual execution

```
=#EXPLAIN SELECT * FROM tbl JOIN rparent ON id= col1;
 Nested Loop (actual rows=50 loops=1)
   ->  Seq Scan on tbl (actual rows=10000 loops=1)
   ->  Append (actual rows=0 loops=10000)
       ->  Index Scan using rpart_1_pkey on rpart_1 ( actual rows=1
loops=50)
             Index Cond: (col1 = tbl.id)
       ->  Index Scan using rpart_2_pkey on rpart_2  (never executed)
             Index Cond: (col1 = tbl.id)
       ->  Index Scan using rpart_3_pkey on rpart_3  (never executed)
             Index Cond: (col1 = tbl.id)
       ->  Index Scan using rpart_4_pkey on rpart_4  (never executed)
             Index Cond: (col1 = tbl.id)
```

We are Hiring!

- Java Full Stack Developer
- Java Architect
- Python Full Stack Developer
- Technical Sales Engineer
- Technical Support
- Sales
- Database Consultant
- UI Developers

THANK YOU!

EDB POSTGRES