Pivotal

Petabyte Scale Data Warehousing Greenplum

Postgres Conf 2019

Partitioning PG versus GPDB

Marshall Presser
Craig Sylvester
Andreas Scherbaum
March 18th, 2019

Partitioning PG versus GPDB

This chapter gives you an overview of the differences between PostgreSQL and Greenplum Databases partitioning

Partitioning in Greenplum Database

- Basically available since the beginning
- 2 levels of partitioning (partition, subpartition)
- Uses „INHERIT“ functionality under the hood
- Uses „CHECK“ constraints
- Works on top of data distribution
    - Never use the same partition and distribution key
- Partition by: Date Range, Numeric Range, List, Multi-level
- Query planner can prune partitions at planning time and at run-time
- Default partitions are possible
- Subpartition templates are possible
- Splitting partitions is possible
- Exchanging partitions is possible

Partitioning in PostgreSQL

- PostgreSQL startet partition support in version 10
    - Before that, partitioning was only „inherited tables“ with handwritten code to manage the setup, no SQL support
- Supports Range and List partitions in v10, also Hash partitions in v11
- Uses „INHERIT“ functionality under the hood
- Uses „CHECK“ constraints
- Subpartitions need to be created separately
- Default partitions are possible
- Exchanging partitions are two steps (detach, attach)
- Splitting partitions is not supported
- pg_partman handles the rolling partitions
    - Can work together with pg_jobmon

Greenplum: Partition by Date Range

```
CREATE TABLE logdata (id INT, ts DATE, logtext TEXT)
DISTRIBUTED BY (id)
PARTITION BY RANGE (ts)
( START (date '2019-01-01') INCLUSIVE
   END (date '2020-01-01') EXCLUSIVE
   EVERY (INTERVAL '1 day') );
```

365/366 partitions

Greenplum: Partition by Numeric Range

```
CREATE TABLE rank (id INT, rank INT, year INT, gender CHAR(1), count INT)
DISTRIBUTED BY (id)
PARTITION BY RANGE (year)
( START (2016) END (2026) EVERY (1),
  DEFAULT PARTITION extra );
```

10 partitions

For data which does not fit into the 10 years

Greenplum: Partition by List

```
CREATE TABLE rank (id INT, rank INT, year INT, gender CHAR(1), count INT )
DISTRIBUTED BY (id)
PARTITION BY LIST (gender)
( PARTITION girls VALUES ('F'),
  PARTITION boys VALUES ('M'),
  DEFAULT PARTITION other );
```

2 partitions

For data which does not fit into F/M gender

PostgreSQL: Partition by Range

```
CREATE TABLE measurement (
    city_id         INT NOT Null,
    logdate          DATE NOT NULL,
    peaktemp          INT,
    unitsales        INT
) PARTITION BY RANGE (logdate);

CREATE TABLE measurement_y2019m02 PARTITION OF measurement
    FOR VALUES FROM ('2019-02-01') TO ('2019-03-01');

CREATE TABLE measurement_y2019m03 PARTITION OF measurement
    FOR VALUES FROM ('2019-03-01') TO ('2019-04-01');

CREATE TABLE measurement_y2019m04 PARTITION OF measurement
    FOR VALUES FROM ('2019-04-01') TO ('2019-05-01')
    PARTITION BY RANGE (peaktemp);
```

Subpartitions

PostgreSQL: pg_partman

- PostgreSQL extension
- Combination of stored procedures, some state tables, and cron jobs
- https://github.com/pgpartman/pg_partman

```
postgresql.conf:
shared_preload_libraries = 'pg_partman_bgw‘
pg_partman_bgw.interval = 3600
pg_partman_bgw.role = ‘partitions'
pg_partman_bgw.dbname = ‘partitions'
```

PostgreSQL: pg_partman

```
CREATE schema test;
CREATE TABLE test.part_test (col1 SERIAL, col2 TEXT, col3 TIMESTAMPTZ NOT NULL DEFAULT now());
```

PostgreSQL >= 11

```
SELECT partman.create_parent('test.part_test', 'col3', ’native', 'daily');
```

or

```
SELECT partman.create_parent('test.part_test', 'col1', 'partman', '100000');
```

PostgreSQL < 11

```
SELECT run_maintenance();
```

Limitations in Greenplum Database

- 32767 partitions per table
- Primary Key (or Unique Constraint) must contain all partitioned colums
- GPORCA only supports uniform partitioned tables
    - Otherwise fallback to the legacy planner (soon: Postgres Planner)
- Queries against external table partitions use the legacy planner
- External table partitions are read-only, no write operations possible
- Limited subset of ALTER PARTITION functionality if external tables are involved
- No good tool around for automatically managing rolling partitions
- Multi-level partitions create a large number of entries in pg_class and pg_attributes
- Multi-level partitions with column-based storage can lead to a large number of open file descriptors

Limitations in PostgreSQL

- No split partition support
- Multiple steps necessary for creating partitions
- During partition assignment the table is scanned for CHECK constraint violations
- The default partition is scanned for CHECK constrain violations every time partitions are changed
- Indexes are not automatically created on new partitions
- UPDATE cannot move rows between partitions
- Lower bound for MINVALUE is always inclusive, upper bound for MAXVALUE is always exclusive

Maintain Partitions

- Create a new partition:
  - PostgreSQL: CREATE TABLE <partition> PARTITION OF <parent table>
  - GPDB: ALTER TABLE <parent table> ADD PARTITION

- Drop a partition:
  - PostgreSQL: DROP TABLE <partition>
  - GPDB: ALTER TABLE <parent table> DROP PARTITION

- Detach a partition:
  - PostgreSQL: ALTER TABLE <parent table> DETACH PARTITION <partition>
  - GPDB: ALTER TABLE <parent table> EXCHANGE PARTITION <new partition>

Standalone table afterwards

Not really "DETACH"

- Truncate a partition:
  - PostgreSQL: TRUNCATE TABLE <partition>
  - GPDB: ALTER TABLE <parent table> TRUNCATE PARTITION <definition or partition name>

Need to know

- Exchange a partition:
  - PostgreSQL: ALTER TABLE <parent table> DETACH PARTITION <partition> + ATTACH PARTITION <partition>
  - GPDB: ALTER TABLE <parent table> EXCHANGE PARTITION

Pivotal®

Transforming How The World Builds Software