付録.パーティションツール

1. pg_part

1.1. 環境構築

検証環境は下記で実施しました。

表 1.1: 環境

CPU	Intel(R) Xeon(R) CPU L5520 @ 2.27GHz
RAM	8GB
OS	Red Hat Enterprise Linux Server release 6.6
PostgreSQL サーバ	PostgreSQL 9.4.0

環境構築は以下の手順で実施しています。

(1)インストール

```
$cd <PostgreSQL ソース DIR>/contrib
$cp <pg_part 展開 DIR>/pg_part-master ./
$make install
$ psql -c "CREATE EXTENSION pg_part;"
CREATE EXTENSION
```

(2)確認

```
postgres=# ¥dx
                List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 pg_part | 0.1.0   | public     | Table partitioning utility


postgres-#
  select pg_namespace.nspname,pg_proc.proname
  from   pg_proc , pg_namespace
  where  pg_proc.pronamespace = pg_namespace.oid
  and    pg_namespace.nspname='pgpart';

 nspname |          proname
---------+---------------------------
 pgpart  | _get_attname_by_attnum
 pgpart  | _get_primary_key_def
 pgpart  | _get_index_column_name
 pgpart  | _get_detach_partition_def
 pgpart  | detach_partition
 pgpart  | show_partition
 pgpart  | _get_index_def
 pgpart  | _get_partition_def
 pgpart  | _get_export_query
 pgpart  | _get_import_query
 pgpart  | add_partition
 pgpart  | merge_partition
 pgpart  | _get_constraint_name
 pgpart  | _get_constraint_def
 pgpart  | _get_attach_partition_def
 pgpart  | attach_partition
(16 rows)
```

1.2. 動作検証

(1) 事前準備

・テストテーブル pttesttbl の作成

```
postgres=#CREATE TABLE pttesttbl(col1 integer, col2 text, col3 timestamp);
```

・テストデータの作成 #2016/04/01 から 10 日間(864000 秒) をデータとして生成

```
postgres=#
 insert into pttesttbl values ('1','Sampledata','2016-04-01 00:00:00.000');
 insert into pttesttbl
 values(generate_series(1,864000),'Sampledata',generate_series('2016-04-01 00:00:00.000'::timestamp,'2016-04-11 00:00:00','1 second'));
```

(2) 動作確認

・pg_part を利用した非パーティション表からパーティション表の変換

パーティションの交換は pgpart.add_partition と pgpart.merge_partition で行います。

・pgpart.add_partition 実行例

```
postgres=#SELECT pgpart.add_partition('public','pttesttbl','pttesttbl_1',' col1 >= 1 AND col1 < 86401 ', '/tmp/pttesttbl.tmp')
;
```

・pgpart.merge_partition 実行例

```
postgres=#SELECT pgpart.merge_partition('public','pttesttbl','pttesttbl_1'', null ,'/tmp/pttesttbl.tmp');
```

・pg_part を利用したパーティションの確認結果

作成されたテーブル、制約、トリガーの状況を確認する SQL

```
postgresql=#
  SELECT n.nspname as "Schema",  c.relname as "Name",
  CASE c.relkind WHEN 'r' THEN 'table' WHEN 'v' THEN 'view' WHEN 'm' THEN 'materialized view' WHEN 'i' THEN 'index' WHEN 'S'
THEN 'sequence' WHEN 's' THEN 'special' WHEN 'f' THEN 'foreign table' END as "Type",
  c.reltuples::integer,
  pg_catalog.pg_get_userbyid(c.relowner) as "Owner",
  pg_catalog.pg_size_pretty(pg_catalog.pg_table_size(c.oid)) as "Size",
  r.conname,  r.contype as "ct",
  substr(pg_catalog.pg_get_constraintdef(r.oid, true),1,30),
FROM pg_catalog.pg_class c
     LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
     LEFT JOIN pg_catalog.pg_constraint r ON r.conrelid = c.oid
WHERE c.relname like 'pttesttbl%'
ORDER BY 1,2;
```

(実行結果)

```
      Name        | Type  | reltuples |  Size   |          conname          | ct |           con_detail
------------------+-------+-----------+---------+---------------------------+----+--------------------------------
 pttesttbl        | table |    864001 | 43 MB   |                           |    |
 pttesttbl_1_86400 | table |     86400 | 4416 kB | __pttesttbl_1_86400_check | c  | CHECK (col1 >= 1 AND col1 < 86
(2 rows)
```

継承を使用した親テーブルと子テーブル確認する SQL

```
postgresql=#
select ( select relname from pg_catalog.pg_class where pg_class.oid = inhparent ) "親テーブル",count(*) "子テーブル数"
fom pg_catalog.pg_inherits
group by inhparent;
```

```
 親テーブル  | 子テーブル数
------------+--------------
 pttesttbl  |            1
(1 row)
```

確認結果より pg_part で提供された関数を使用することで、簡単に継承を利用した子テーブルと制約が作成されることが確認できます。

2. pg_partman

2.1. 環境構築

検証は下記環境で実施しました。

表 2.1: 環境

CPU	Intel(R) Xeon(R) CPU L5520 @ 2.27GHz
RAM	8GB
OS	Red Hat Enterprise Linux Server release 6.6
PostgreSQL サーバ	PostgreSQL 9.4.0

環境構築は以下の手順で実施。

(1)事前準備

```
・dblink の インストール
  $cd <PostgreSQL ソース DIR>/contrib/dblink
  $make
  $make install
  $psql -c "CREATE EXTENSION dblink;"

・pg_jobman の インストール
  $cd <PostgreSQL ソース DIR>/contrib
  $cp <pg_part 展開 DIR>/pg_part-master ./
  $make
  $make install
  $ psql -c "CREATE SCHEMA jobmon;"
  $ psql -c "CREATE EXTENSION pg_jobmon SCHEMA jobmon;"
```

(2)インストール

```
・pg_partman の インストール
  $cd <PostgreSQL ソース DIR>/contrib
  $cp <pg_partman 展開 DIR>/pg_partman-master ./
  $make
  $make install
  $ psql -c "CREATE SCHEMA partman;"
  $ psql -c "CREATE EXTENSION pg_partman SCHEMA partman;"
```

(3)確認

```
 $  psql -c "¥dx"
                                 List of installed extensions
   Name    | Version |   Schema   |                          Description
-----------+---------+------------+------------------------------------------------------------
dblink     | 1.1     | public     | connect to other PostgreSQL databases from within a database
pg_jobmon  | 1.3.2   | jobmon     | Extension for logging and monitoring functions in PostgreSQL
pg_partman | 2.3.1   | partman    | Extension to manage partitioned tables by time or ID
```

2.2. 動作検証

(1) 事前準備

```
・テストテーブル pttesttbl の作成
 postgres=#CREATE TABLE pttesttbl(col1 integer, col2 text, col3 timestamp);

・テストデータの作成 #2016/04/01 から 10 日間(864000 秒) をデータとして生成
 postgres=#
  insert into pttesttbl values ('1','Sampledata','2016-04-01 00:00:00.000');
  insert into pttesttbl
  values(generate_series(1,864000),'Sampledata',generate_series('2016-04-01 00:00:00.000'::timestamp,'2016-04-11 00:00:00','1 second'));
```

(2) 動作確認

・pg_partman を利用した非パーティション表からパーティション表の変換

パーティションの交換は partman.create_parent と partman.undo_partition_id/ partman.undo_partition_time で行います。

・partman.create_parent Id base パーティション実行例

```
postgres=#
        alter table pttesttbl alter column col1 set not null;
        SELECT partman.create_parent('public.pttesttbl', 'col1', 'id', '86400');
```

・partman.create_parent time base パーティション実行例

```
postgres=#
        alter table pttesttbl alter column col1 set not null;
        SELECT partman.create_parent('public.pttesttbl', 'col3', 'time', 'daily');
```

・partman.undo_partition_id 実行例

```
postgres=#SELECT partman.undo_partition_id('public.pttesttbl',20,p_keep_table := false);
```

・partman.undo_partition_time 実行例

```
postgres=#SELECT partman.undo_partition_time('public.pttesttbl',20,p_keep_table := false);
```

pg_partman を利用したパーティションの確認

作成されたテーブル、制約、トリガーの状況を確認する SQL

```
postgresql=#
SELECT c.relname as "Name",
  CASE c.relkind WHEN 'r' THEN 'table' WHEN 'v' THEN 'view' WHEN 'm' THEN 'materialized view' WHEN 'i' THEN 'index' WHEN 'S'
THEN 'sequence' WHEN 's' THEN 'special' WHEN 'f' THEN 'foreign table' END as "Type", c.reltuples::integer,
  pg_catalog.pg_size_pretty(pg_catalog.pg_table_size(c.oid)) as "Size",  r.conname,  r.contype as "ct",
  substr(pg_catalog.pg_get_constraintdef(r.oid, true),1,30) con_detail,
case c.relhastriggers  when true  then ( select tgname from pg_trigger where tgrelid = c.oid LIMIT 1 ) when false then
'false'
       else 'other' end as "tr"
FROM pg_catalog.pg_class c
     LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
     LEFT JOIN pg_catalog.pg_constraint r ON r.conrelid = c.oid
WHERE c.relname like 'pttesttbl%'
ORDER BY 1,2;

            Name              | Type  | reltuples |    Size    |                conname                | ct |
con_detail           |         tr
------------------------------+-------+-----------+------------+---------------------------------------+----
+------------------------------+--------------------
pttesttbl                     | table |         0 | 48 kB      |                                       |    |
               | pttesttbl_part_trig
pttesttbl_p2016_04_04         | table |     59262 | 3056 kB    | pttesttbl_p2016_04_04_partition_check | c  | CHECK (col3 >=
'2016-04-04 00: | false
pttesttbl_p2016_04_04_col3_idx | index |     59262 | 1312 kB    |                                       |    |
               | false
pttesttbl_p2016_04_05         | table |     86400 | 4440 kB    | pttesttbl_p2016_04_05_partition_check | c  | CHECK (col3 >=
'2016-04-05 00: | false
pttesttbl_p2016_04_05_col3_idx | index |     86400 | 1904 kB    |                                       |    |
               | false
```

継承を使用した親テーブルと子テーブル確認する SQL

```
postgresql=#
select ( select relname from pg_catalog.pg_class where pg_class.oid = inhparent ) "親テーブル",count(*) "子テーブル数"
from pg_catalog.pg_inherits
group by inhparent;

 親テーブル  | 子テーブル数
------------+--------------
 pttesttbl  |           20
(1 row)
```

3. pg_shard

3.1. 環境構築手順

本検証では、検証環境として Microsoft Azure を使っているため、その環境構築手順を示します。なお、サーバ環境とクライアント環境は同一サーバ上で構築を行っています。

3.1.1. 構築環境

ハードウェア環境とソフトウェア環境は、以下の環境になります。

表 3.1: ハードウェア環境

仮想環境	Microsoft Azure
インスタンス	Standard D3
CPU	Intel(R) Xeon(R) CPU E5-2660 2.20GHz @ 4core
RAM	14GB
DISK	SSD 200GB

表 3.2: ソフトウェア環境

OS	CentOS 7.2.1511
PostgreSQL サーバ	PostgreSQL 9.4.5 (PGDG)
pg_shard	1.2.3
gcc	4.8.5

3.1.2. 構築手順

構築手順は以下の通りです。
※SELinux と firewall は予め停止しています。

設定するにあたっての前提条件になります。
DB ユーザ:postgres（パスワード:postgres）
DB 名:pgbench
DB サーバ、クライアントの IP アドレス:10.0.0.1

(1) PostgreSQL のインストール

```
$ sudo rpm -i pgdg-centos94-9.4-2.noarch.rpm
$ sudo yum install postgresql94 postgresql94-server postgresql94-contrib postgresql94-devel postgresql94-libs
```

(2) pg_shard のインストール

```
$ export PATH=/usr/pgsql-9.4/bin:$PATH
$ tar zxf pg_shard-1.2.3.tar.gz
$ cd pg_shard-1.2.3
$ make
$ sudo make install
```

(3) .pgpass の設定

```
*:*:pgbench:postgres:postgres
```

(4) データベースクラスタのインストールと設定

```
$ initdb --encoding=UTF-8 --no-locale -A md5 -W
```

(5) pg_hba.conf の設定

```
host    all             all             10.0.0.1/32             md5
```

(6) postgresql.conf の設定

```
listen_addresses = '*'
shared_buffers = 3500MB
work_mem = 16MB
maintainance_work_mem = 128MB
shared_preload_libraries = 'pg_shard'
synchronous_commit = off
wal_buffers = 16MB
checkpoint_segments = 64
checkpoint_timeout = 30min
checkpoint_completion_target = 0.8
checkpoint_warning = 30min
hot_standby = on
random_page_cost = 2.0
effective_cache_size = 7GB
logging_collector = on
log_filename = 'postgresql-%Y-%m-%d.log'
log_line_prefix = '[%t %d] '
```

(7) pg_worker_list.conf の作成

パーティショニング振り分けの対象とする PostgreSQL サーバのアドレスとポート番号を記載します。

```
10.0.0.1        5432
```

ファイルのパーミッションは、0600 に設定します。

(8) データベースを起動

```
$ pg_ctl start
```

(9) データベース作成

```
$ createdb pgbench
```

(10) pg_shard 拡張を追加

```
$psql pgbench
pgbench=# create extension pg_shard;
```

(11) テーブル作成

今回は、pgbench を動作させるため、4 つのテーブルを作成する

```
pgbench=# create table pgbench_accounts
(
  aid bigint not null,
  bid int,
  abalance int,
  filler char(84)
) with (fillfactor=100);
```

(12) pg_shard 構築

master_create_woker_shards 関数の第 2 引数でパーティションの分割数を指定します。

```
pgbench=# select master_create_distributed_table('pgbench_accounts','aid');
pgbench=# select master_create_worker_shards('pgbench_accounts',1, 1);
```

(13) vacuum と primary key の設定

データロード後に、vacuum と primary key の設定を行います。

```
pgbench=# vacuum analyze pgbench_accounts;
pgbench=# vacuum analyze pgbench_accounts_xxxxx;（全パーティションテーブル）
pgbench=# alter table pgbench_accounts add primary key (aid);
pgbench=# alter table pgbench_accounts_xxxxx add primary key (aid);（全パーティションテーブル）
```