Postgres Plus Advanced Server におけるパーティション・テーブル

EnterpriseDB Corporation

2014年9月

PPASのパーティショニングの利点

PostgreSQLのパーティション・テーブル構成に比べ、PPASの場合は、下記のような特徴、利点があります。

- Oracle 互換性の PARTITION BY および SUBPARTITION BY 構文が、エラーを最少限に留める
- より多くの Oracle 互換性構文が、容易かつ精巧なデータ管理技術を提供
  - ADD / DROP によるパーティションの追加・削除
  - SPLIT によるパーティション分割
  - EXCHANGE による新しいパーティションのスワップイン
  - パーティション・データのTRUNCATE
  - MOVE による、パーティションを異なるテーブルスペースへ移動
- ファースト・プルーニングと制約排除サポートを伴う、PPAS によるパフォーマンスの改善
- パーティション向けシステムカタログビュー
  - ALL_PART_TABLES
  - ALL_TAB_PARTITIONS, ALL_TAB_SUBPARTITIONS
  - ALL_PART_KEY_COLUMNS, ALL_SUBPART_KEY_COLUMNS

作成時、PPAS 構文がエラーと複雑製を最少化

【PPAS】

```
CREATE TABLE sales (
  dept_no number,
  part_no varchar2,
  country varchar2(20),
  date date,
  amount number )
PARTITION BY RANGE(date)
(
PARTITION q1_2014 VALUES LESS THAN('2014-Apr-01'),
PARTITION q2_2014 VALUES LESS THAN('2014-Jul-01'),
PARTITION q3_2014 VALUES LESS THAN('2014-Oct-01'),
PARTITION q4_2014 VALUES LESS THAN('2015-Jan-01')
);
CREATE INDEX sales_date on sales(date);
```

シンプルで宣言的な PARTITION BY

シングルインデックス コマンド

【PostgreSQL】

```
CREATE TABLE sales (
  dept_no number,
  part_no varchar2,
  country varchar2(20),
  date date,
  amount number
);
CREATE TABLE q1_2014 ( CHECK ( date >= DATE '2014-Jan-01' AND date
< DATE '2014-Apr-01' )
) INHERITS (sales);
CREATE TABLE q2_2004 ( CHECK ( date >= DATE '2014-Apr-01' AND date
< DATE '2014-Jul-01' )
) INHERITS (sales);
...
CREATE INDEX q1_2014_date ON q1_2014(date);
CREATE INDEX q2_2014_date ON q2_2014(date);
...
CREATE OR REPLACE FUNCTION sales_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF ( NEW.date >= DATE '2014-Jan-01' AND
         NEW.date < DATE '2014-Apr-01' ) THEN
        INSERT INTO q1_2014 VALUES (NEW.*);
    ELSIF ( NEW.date >= DATE '2014-Apr-01' AND
            NEW. date < DATE '2014-Jul-01' ) THEN
        INSERT INTO q2_2014 VALUES (NEW.*);
    ...
    ELSE
        RAISE EXCEPTION 'Date out of range.  Fix the
sales_insert_trigger() function!';
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

「マスター」テーブル

CHECK と INHERITS を使用した「子」テーブル

各テーブルにおけるインデックス

INSERT、UPDATE DELETE を管理する複雑なトリガ関数

EDB ENTERPRISEDB

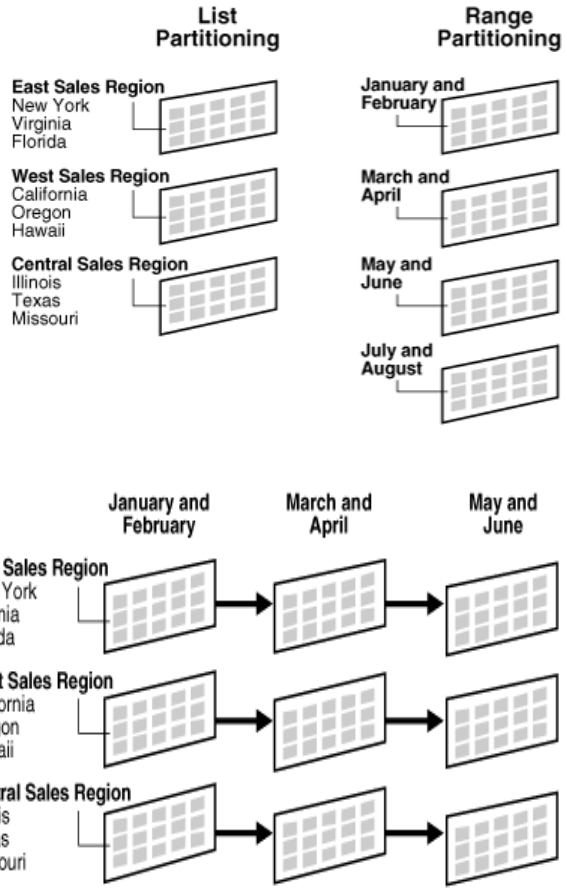
パーティション・タイプ

- サポートされるパーティション・タイプ
  - ①リスト・パーティション

    シングル・パーティショニング・キー列。正確な値に基づく
  - ②レンジ・パーティション

    一つ以上のパーティショニング・キー列。二つの極値間の値に基づく
  - ③ハッシュ・パーティション

    指定されたカラム名のハッシュ値に基いて、データを均等に分割。PPAS9.4よりサポート
- サブ・パーティション
  - パーティション化されたテーブルより、より小さな、サブセットへ分割可能
  - 親と異なる、パーティション・タイプを使用可能
  - 全てのデータは、サブ・パーティションに保管される



Images from http://docs.oracle.com/cd/E18283_01/server.112/e16541/partition.htm

PPAS は、便利なパーティション管理を提供

- パーティションの追加／削除
  - `ALTER TABLE sales ADD PARTITION q1_2015 VALUES LESS THAN('01-APR-2015');`
- パーティションの分割
  - `ALTER TABLE sales SPLIT PARTITION q1_2014 AT ('01-FEB-2014') INTO (PARTITION m1_2014, PARTITION m23_2014);`
    `ALTER TABLE sales SPLIT PARTITION m23_2014 AT ('01-MAR-2014') INTO (PARTITION m2_2014, PARTITION m3_2014);`
- 既存テーブルのパーティション化
  - `ALTER TABLE sales EXCHANGE PARTITION q1_2014 WITH TABLE new_q1_2014_data;`
- パーティション・データのtruncate
  - `ALTER TABLE sales TRUNCATE PARTITION final_archive;`
- パーティションの異なるテーブルスペースへの移動
  - `ALTER TABLE sales MOVE PARTITION q1_2010 TABLESPACE 4yr_archived_tables;`

PPAS による、パーティショニング・パフォーマンスの改善

Advanced Server のクエリプランナーが二つの最適化技術を使用して、効率的なプランを算出:

- 制約排除 (constraint_exclusion = partition or on)
  - PostgreSQL が提供
  - SELECT w/ WHERE:クエリプランナーは、クエリフラグメントをどのパーティションに送るかを決定する前に、各パーティション向けに定義された CHECK 制約を検証しなくてはならない
- ファースト・プルーニング (edb_partition_pruning = on)
  - Query Plan プロセスの初期に生じる。Oracle スタイルのパーティション化テーブルにおける、パーティション間の関係性を把握。
  - SELECT w/ WHERE: クエリプランナーは、各パーティション向けに定義された制約を検証せずに、一部のパーティションのみが値を有すると推論できる
  - リストあるいは単一値レンジ・パーティションにおいて使用可能 (サブ・パーティション化テーブル、あるいは多値レンジ・パーティション化テーブルには使用不可)
  - WHERE 句における >、>=、=、<=、<、AND、BETWEEN 演算子と使用可能

まとめ

- パーティショニングは、沢山のデータを有する際に便利
- Postgres が基盤を提供
  - テーブル・インヘリタンス機能、CHECK 制約、マルチ・インデックスおよび複雑なトリガを介したサポート
  - `constraint_exclusion` の最適化
- 改善された PPAS！
  - Oracle 互換性の宣言的 PARTITION BY 構文
  - 包括的なパーティション管理機能
  - ファースト・パーティション・プルーニングによる、パフォーマンスの改善

The End.