SQLで不足レコードを補うクエリの発行

2017年11月16日
自分用メモ。

処理対象データ

PostgreSQLに以下のデータがある。

```
$ psql -h 192.168.1.30 kamonohashi kamo
psql (9.6.5)
"help" でヘルプを表示します.

kamonohashi=> SELECT * FROM salelog;
         DAY          | item  | qty
----------------------+-------+-----
 2017-10-01 00:00:00 | ITEM1 |  30
 2017-10-02 00:00:00 | ITEM2 |  15
 2017-10-02 00:00:00 | ITEM3 |  20
 2017-10-11 00:00:00 | ITEM1 |  25
 2017-10-13 00:00:00 | ITEM1 |  15
 2017-10-13 00:00:00 | ITEM2 |   5
(6行)

kamonohashi=>
```

これにレコードを補い、以下のように各日で必ずITEM1～ITEM3のレコードが現れるようにしたい。

```
         DAY          | item  | qty
----------------------+-------+-----
 2017-10-01 00:00:00 | ITEM1 |  30
 2017-10-01 00:00:00 | ITEM2 |   0
 2017-10-01 00:00:00 | ITEM3 |   0
 2017-10-02 00:00:00 | ITEM1 |   0
 2017-10-02 00:00:00 | ITEM2 |  15
 2017-10-02 00:00:00 | ITEM3 |  20
 2017-10-11 00:00:00 | ITEM1 |  25
 2017-10-11 00:00:00 | ITEM2 |   0
 2017-10-11 00:00:00 | ITEM3 |   0
 2017-10-13 00:00:00 | ITEM1 |  15
 2017-10-13 00:00:00 | ITEM2 |   5
 2017-10-13 00:00:00 | ITEM3 |   0
```

方法1：3回のクエリ発行

単純に、ITEM1～ITEM3のいずれかの記録がされた日付に、記録がないアイテムのレコードを追加する。

例えばITEM1の不足レコードを追加するには以下のクエリとなる。

```
WITH t AS (
  SELECT DISTINCT DAY FROM salelog
)
INSERT INTO salelog
  SELECT t.day, 'ITEM1', 0 FROM t
   WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day AND
m.item='ITEM1')
;
```

実行してみる。

```
kamonohashi=> WITH t AS (
kamonohashi(>   SELECT DISTINCT DAY FROM salelog
kamonohashi(> )
kamonohashi-> INSERT INTO salelog
kamonohashi->   SELECT t.day, 'ITEM1', 0 FROM t
kamonohashi->    WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day
AND m.item='ITEM1')
kamonohashi-> ;
INSERT 0 1
kamonohashi=> SELECT * FROM salelog;
         DAY         | item  | qty
---------------------+-------+-----
 2017-10-01 00:00:00 | ITEM1 |  30
 2017-10-02 00:00:00 | ITEM2 |  15
 2017-10-02 00:00:00 | ITEM3 |  20
 2017-10-11 00:00:00 | ITEM1 |  25
 2017-10-13 00:00:00 | ITEM1 |  15
 2017-10-13 00:00:00 | ITEM2 |   5
 2017-10-02 00:00:00 | ITEM1 |   0
(7行)

kamonohashi=>
```

2017-10-02 に ITEM1 のレコードが追加された。

続けて ITEM2,ITEM3 のクエリも実行してみる。

```
kamonohashi=> WITH t AS (
kamonohashi(>   SELECT DISTINCT DAY FROM salelog
kamonohashi(> )
kamonohashi-> INSERT INTO salelog
kamonohashi->   SELECT t.day, 'ITEM2', 0 FROM t
kamonohashi->    WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day
AND m.item='ITEM2')
kamonohashi-> ;
INSERT 0 2
kamonohashi=> WITH t AS (
kamonohashi(>   SELECT DISTINCT DAY FROM salelog
```

```
kamonohashi(> )
kamonohashi-> INSERT INTO salelog
kamonohashi->   SELECT t.day, 'ITEM3', 0 FROM t
kamonohashi->    WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day
AND m.item='ITEM3')
kamonohashi-> ;
INSERT 0 3
kamonohashi=> SELECT * FROM salelog;
         DAY          | item  | qty
----------------------+-------+-----
 2017-10-01 00:00:00 | ITEM1 |  30
 2017-10-02 00:00:00 | ITEM2 |  15
 2017-10-02 00:00:00 | ITEM3 |  20
 2017-10-11 00:00:00 | ITEM1 |  25
 2017-10-13 00:00:00 | ITEM1 |  15
 2017-10-13 00:00:00 | ITEM2 |   5
 2017-10-01 00:00:00 | ITEM2 |   0
 2017-10-11 00:00:00 | ITEM2 |   0
 2017-10-01 00:00:00 | ITEM3 |   0
 2017-10-11 00:00:00 | ITEM3 |   0
 2017-10-13 00:00:00 | ITEM3 |   0
 2017-10-02 00:00:00 | ITEM1 |   0
(12行)
```

3つのクエリを実行し、6レコードが追加された。 見易くソートをかけると、期待通りの結果であることがわかる。

```
kamonohashi=> SELECT * FROM salelog ORDER BY DAY, item;
         DAY          | item  | qty
----------------------+-------+-----
 2017-10-01 00:00:00 | ITEM1 |  30
 2017-10-01 00:00:00 | ITEM2 |   0
 2017-10-01 00:00:00 | ITEM3 |   0
 2017-10-02 00:00:00 | ITEM1 |   0
 2017-10-02 00:00:00 | ITEM2 |  15
 2017-10-02 00:00:00 | ITEM3 |  20
 2017-10-11 00:00:00 | ITEM1 |  25
 2017-10-11 00:00:00 | ITEM2 |   0
 2017-10-11 00:00:00 | ITEM3 |   0
 2017-10-13 00:00:00 | ITEM1 |  15
 2017-10-13 00:00:00 | ITEM2 |   5
 2017-10-13 00:00:00 | ITEM3 |   0
(12行)

kamonohashi=>
```

方法2：1回のクエリ発行

事前に以下のテーブルを用意する。

```
kamonohashi=> SELECT * FROM itemlist;
 item
-------
 ITEM1
 ITEM2
 ITEM3
(3行)

kamonohashi=>
```

このテーブル itemlist を組み合わせるクエリは以下となる。

```
WITH t AS (
  SELECT d.day, m.item
    FROM ( SELECT DISTINCT DAY FROM salelog) d
      CROSS JOIN itemlist m
)
INSERT INTO salelog
  SELECT t.day, t.item, 0 FROM t
   WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day AND
m.item=t.item)
;
```

実行結果は期待通りになっている。

```
kamonohashi=> WITH t AS (
kamonohashi(>   SELECT d.day, m.item
kamonohashi(>     FROM ( SELECT DISTINCT DAY FROM salelog) d
kamonohashi(>       CROSS JOIN itemlist m
kamonohashi(> )
kamonohashi-> INSERT INTO salelog
kamonohashi->   SELECT t.day, t.item, 0 FROM t
kamonohashi->    WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day
AND m.item=t.item)
kamonohashi-> ;
INSERT 0 6
kamonohashi=> SELECT * FROM salelog ORDER BY DAY, item;
         DAY          | item  | qty
----------------------+-------+-----
 2017-10-01 00:00:00 | ITEM1 |  30
 2017-10-01 00:00:00 | ITEM2 |   0
 2017-10-01 00:00:00 | ITEM3 |   0
 2017-10-02 00:00:00 | ITEM1 |   0
 2017-10-02 00:00:00 | ITEM2 |  15
 2017-10-02 00:00:00 | ITEM3 |  20
 2017-10-11 00:00:00 | ITEM1 |  25
 2017-10-11 00:00:00 | ITEM2 |   0
 2017-10-11 00:00:00 | ITEM3 |   0
 2017-10-13 00:00:00 | ITEM1 |  15
 2017-10-13 00:00:00 | ITEM2 |   5
 2017-10-13 00:00:00 | ITEM3 |   0
```

```
(12行)

kamonohashi=>
```

説明

キー項目でINSERT先の既存レコードの存在を確認し、レコードが存在しないならそのキー項目を持つレコードを追加している。 この例では、項目day□項目item□が該当する。

レコードの存在有無は EXISTS()関数で確認できる。 レコードがあれば EXISTS(...)の戻り値はTRUEとなる。追加先のテーブル salelog 内にキーで示すレコードが存在しないならEXISTS(...)の結果はFALSEになるので□NOT EXISTS(...)で条件が成立する。
※EXISTS(...)=TRUE時はすでにそのキーのレコードがあるので追加してはいけない

EXISTS()関数内のクエリで使うキー項目を持つレコードセットが t になる。

方法２ではレコードセット t を作るためにテーブル itemlist を使ったが、このテーブルも置き換えできる。例えば

```
WITH t AS (
  SELECT d.day, m.item
    FROM ( SELECT DISTINCT DAY FROM salelog) d
      CROSS JOIN ( SELECT DISTINCT item FROM salelog) m
)
INSERT INTO salelog
  SELECT t.day, t.item, 0 FROM t
   WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day AND
m.item=t.item)
;
```

でもいい。

また、レコードセット t は、WITHを使わずサブクエリにしても構わない。

```
INSERT INTO salelog
  SELECT t.day, t.item, 0
    FROM (
      SELECT d.day, m.item
        FROM ( SELECT DISTINCT DAY FROM salelog) d
          CROSS JOIN ( SELECT DISTINCT item FROM salelog) m
    ) t
   WHERE NOT EXISTS(SELECT * FROM salelog m WHERE m.day=t.day AND
m.item=t.item)
;
```

database, SQL, 不足レコード補填, PostgreSQL, 技術資料