PGECons
PostgreSQL Enterprise Consortium

PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#2

組み込み関数移行調査編

製作者
担当企業名　TIS 株式会社

改訂履歴

版	改訂日	変更内容
1.0	2013/04/22	新規作成

はじめに

■本資料の目的

本資料は、異種 DBMS から PostgreSQL へ組み込み関数を移行する作業の難易度、ボリュームを事前に判断するための参考資料として利用することを想定しています。

■本資料で記載する範囲

本資料では、移行元の異種 DBMS として Oracle Database を想定し、Oracle Database から PostgreSQL へSQLを移行する際に組み込み関数の対応状況や仕様の相違により書き換えが必要となる箇所について記載します。

■本資料で扱う用語の定義

資料で記述する用語について以下に定義します。

表 1: 用語定義

No.	用語	意味
1	DBMS	データベース管理システムを指します。ここでは、PostgreSQL および異種 DBMS の総称として利用します。
2	異種 DBMS	PostgreSQL ではない、データベース管理システムを指します。 本資料では、Oracle Database が該当します。
3	Oracle	データベース管理システムの Oracle Database を指します。

■本資料で扱う DBMS およびツール

本書では以下の DBMS を前提にした調査結果を記載します。

表 2: 本書で扱う DBMS

DBMS 名称	バージョン
PostgreSQL	9.2.0
Oracle Database	11gR2 11.2.0.2.0
Orafce	3.0.3

目次

1. 組み込み関数の対応状況

1.1. OracleとPostgreSQLにおける対応状況の違い

OracleとPostgreSQLでは対応している組み込み関数に違いがあるため、移行元のOracleで使える組み込み関数が移行先のPostgreSQLに存在しないことがあります。移行元の組み込み関数が移行先に存在しない場合は、同等の機能を実現する独自のファンクションを定義する等の対応が必要となります。

OracleとPostgreSQLにおける組み込み関数の対応状況は、別紙「組み込み関数対応表（Oracle-PostgreSQL）」を参照してください。

1.2. Oracle互換関数 Orafceについて

Orafce[1]はOracleと互換性のある関数をPostgreSQLに実装することを目的としたプロジェクトです。Orafceを導入してPostgreSQLで対応する組み込み関数を追加することで、Oracleからの移行の省力化が期待できます。別紙「組み込み関数対応表（Oracle-PostgreSQL）」において、Orafceを導入することで追加される組み込み関数には「orafce列」に「○」を付与しています。

Orafceは一般的なPostgreSQLのcontribモジュールと同様にインストールすることができます。

【Orafceのインストール手順】

```
※Orafceプロジェクトからダウンロードしたソースコードを展開
# cp orafce-3.0.4.tar.gz /usr/local/src/pgsql/contrib
# cd /usr/local/src/pgsql/contrib
# tar xvf orafce-3.0.4.tar.gz
# cd orafce
※make, make installを実行し、ライブラリをインストール
# make
# make install
※データベースに関数を登録するSQLスクリプトを実行
# su - postgres
$ psql -d postgres -f ./orafunc.sql
※next_day関数の実行例
$ psql -d postgres
psql (9.1.3)
Type "help" for help.

postgres=# SELECT next_day(current_date,'saturday');
  next_day
------------
 2012-09-29
(1 row)
```

1 http://orafce.projects.pgfoundry.org/index-ja.html

2. 組み込み関数の移行方法

本章では Oracle と PostgreSQL の対応状況の違いから、そのままでは移行できない組み込み関数への対応方法を紹介します。なお、ここで紹介するのは代表的な関数のみで、全ての対応パターンは網羅していません。

2.1. 数値ファンクション

2.1.1. BITAND

Oracle の BITAND は2つの引数をビット単位で AND 計算する関数です。

【Oracle の BITAND の使用例】

```
SQL> SELECT BITAND(6,3) FROM DUAL;

BITAND(6,3)
-----------
          2     ※数値 6(バイナリ 1,1,0)と数値 3(バイナリ 0,1,1)の AND は数値 2 (0,1,0)
```

PostgreSQL には BITAND 関数が存在しませんが、&演算子で同様の演算が可能です。

【PostgreSQL での&演算子の使用例】

```
postgres=# SELECT integer '6' & integer '3';
6 & 3
-----
    2
(1 行)
```

なお、Orafce には BITAND 関数が実装されていますので、Orafce を導入した PostgreSQL では関数を変更せずに移行することができます。

2.1.2. COSH

Oracle の COSH は引数の双曲線コサインを計算する関数です。

【Oracle の COSH の使用例】

```
SQL> SELECT COSH(0) "Hyperbolic cosine of 0" FROM DUAL;

Hyperbolic cosine of 0
----------------------
                     1
```

PostgreSQL には COSH 関数が存在しませんが、EXP 関数を使って、「(exp(n) + exp(-n)) / 2」という式に置き換えることで同様の演算が可能です。

なお、Orafce には COSH 関数が実装されていますので、Orafce を導入した PostgreSQL では関数を変更せずに移行することができます。

2.1.3. SINH

Oracle の SINH は引数の双曲線サインを計算する関数です。PostgreSQL にはSINH関数が存在しませんが、2.1.2 と同様に EXP 関数を使って、「(exp(n)-exp(-n))/2」という式に置き換えることで同様の演算が可能です。

また、Orafce には SINH 関数が実装されていますので、Orafce を導入した PostgreSQL では関数を変更せずに移行することができます。

2.1.4. TANH

Oracle の TANH は引数の双曲線タンジェントを計算する関数です。PostgreSQL には TANH 関数が存在しませんが、2.1.2 と同様に EXP 関数を使って、「(exp(n)-exp(-n))/(exp(n)+exp(-n))」という式に置き換えることで同様の演算が可能です。

また、Orafce には TANH 関数が実装されていますので、Orafce を導入した PostgreSQL では関数を変更せずに移行することができます。

2.1.5. NANVL

Oracle の NANVL は引数として n1,n2 をとり、n2 が非数値の場合は代替値 n1 を、数値の場合は n2 を戻す関数です。一方、PostgreSQL には NANVL 関数が存在しませんが、Orafce に NANVL 関数が実装されていますので、Orafce を導入した PostgreSQL では関数を変更せずに移行することができます。

2.1.6. REMAINDER

Oracle の REMAINDER は引数として n1,n2 をとり、n1 を n2 で割った余りを求める関数です。

PostgreSQL、Orafce ともに REMAINDER 関数が存在しません。また、余りを求める点では MOD と類似した関数ですが、内部の計算方法が異なる[2]ため、REMAINDER を MOD で置換することはできません。同様の演算を実現するには「n1 - n2 * ROUND (n1 / n2) 」という式に置き換える必要があります。

2.2. 文字値を戻す文字ファンクション

2.2.1. SUBSTR

Oracle の SUBSTR は引数として char, position, substring_length を取り、char の position 番目から substring_length 文字分の文字列を抜き出して戻す関数です。PostgreSQL にも SUBSTR 関数が存在するため、特に変更するなく利用できます。ただし、第 2 引数の position が負値の場合の挙動が異なる点に注意が必要です。

【Oracle の SUBSTR の使用例 】

```
SQL> SELECT SUBSTR('ABCDEFG',3,4) "Substring" FROM DUAL;

Substring
---------
CDEF

※ Oracle では第 2 引数が負値の場合、 文字列の後ろから開始位置をカウントする。
SQL> SELECT SUBSTR('ABCDEFG',-5,4) "Substring" FROM DUAL;

Substring
---------
CDEF
```

【PostgreSQL の SUBSTR の使用例 】

```
postgres=# select substr('ABCDEFG',3,4) "Substring";
 Substring
-----------
 CDEF
(1 行)

※PostgreSQL では第 2 引数が負値の場合の挙動が Oracle と異なる。
postgres=# select substr('ABCDEFG',-5,4) "Substring";
 Substring
-----------

(1 行)
```

なお、Orafce では上記の点について互換性を向上させた SUBSTR 関数が実装されていますので、Orafce を導入した PostgreSQL ではそのまま移行することができます。

2 Oracle の MOD 関数の計算式: n1 - n2 * TRUNC (n1 / n2)、
Oracle の REMAINDER 関数の計算式: n1 - n2 * ROUND (n1 / n2)

2.3. 数値を戻す文字ファンクション

2.3.1. INSTR

Oracle の INSTR は引数 string の substring を検索する関数で、見つかった substring の位置を返します。

【Oracle の INSTR の使用例】

```
SQL> SELECT INSTR('abcdefg','b') FROM DUAL;

INSTR('abcdefg','b')
--------------------
                   2
```

一方、PostgreSQL には INSTR 関数が存在しませんが、strpos 関数に置き換えることで同等の機能を実現できます。また、Orafce には INSTR 関数が実装されていますので、Orafce を導入した PostgreSQL では関数を変更せずに移行することができます。

【PostgreSQL の strpos 関数で置き換えた例】

```
postgres=# select strpos('abcdefg','b');
 strpos
--------
      2
(1 row)
```

2.4. 日時ファンクション

2.4.1. ADD_MONTHS

Oracle の ADD_MONTHS は月を演算する関数で、引数の日付に月数を加えて戻します。

【Oracle の ADD_MONTHS の使用例 】

```
SQL> SELECT ADD_MONTHS('2013/3/22',1) FROM DUAL;

ADD_MONT
--------
13-04-22
```

一方、PostgreSQL には ADD_MONTHS 関数が存在しないため、算術演算子を使った書き換えが必要です。

【PostgreSQL で 2013/3/22 の 1 か月後の日付を求める例】

```
postgres=# SELECT date '2013-03-22' + interval '1 months';
  ?column?
---------------------
 2013-04-22 00:00:00
(1 行)
```

なお、Orafce には ADD_MONTHS 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

【Orafce 導入済の PostgreSQL での ADD_MONTHS の使用例】

```
postgres=# SELECT add_months(date '2013-03-22',1);
add_months
------------
 2013-04-22
(1 行)
```

2.4.2. CURRENT_DATE

Oracle の CURRENT_DATE は 、データベースサーバー の OS のシステムコールで取得した日付をセッションタ

イムゾーンに変換した現在日付と時刻（年～秒）を DATE 型で戻します。

【Oracle の CURRENT_DATE の使用例 】

```
SQL> select CURRENT_DATE from dual;

CURRENT_DATE
-------------------
2013-04-22 16:28:54
```

一方、PostgreSQL にも現在日付を戻す同名の関数が存在します。ただし、PostgreSQL の CURRENT_DATE で戻される DATE 型は日付までのデータしか保有しない点に注意が必要です。時分秒までのデータが必要な場合は、CURRENT_TIMESTAMP に置き換えてください。

【PostgreSQL の CURRENT_DATE の使用例 】

```
postgres=# SELECT CURRENT_DATE;
CURRENT_DATE
------------
 2013-04-22
(1 行)
```

2.4.3. CURRENT_TIMESTAMP

Oracle の CURRENT_TIMESTAMP は 、データベースサーバー の OS のシステムコールで取得した日付をセッションタイムゾーンに変換した現在日付と時刻（年～秒）を TIMESTAMP WITH TIME ZONE データ型の値で戻します。

【Oracle の CURRENT_TIMESTAMP の使用例 】

```
SQL> select CURRENT_TIMESTAMP from dual;

CURRENT_TIMESTAMP
---------------------------------
13-04-22 16:37:04.281000 +09:00
```

PostgreSQL の CURRENT_TIMESTAMP も Oracle と同じ TIMESTAMP WITH TIMEZONE 型を戻すので、そのまま移行することができます。

【PostgreSQL の CURRENT_TIMESTAMP の使用例 】

```
postgres=# SELECT CURRENT_TIMESTAMP;
CURRENT_TIMESTAMP
------------
2013-04-22 14:39:53.662522+09
(1 行)
```

ただし、Oracle と PostgreSQL でデフォルトの出力フォーマットは異なるため、to_char() 関数等を使って出力フォーマットは揃える必要があります。

2.4.4. SYSDATE

Oracle の SYSDATE は、データベースサーバーが稼動する OS のシステムコールから取得した日付と時刻（年～秒）を元に現在日付を DATE 型で戻します。

【Oracle の SYSDATE の使用例 】

```
SQL> select SYSDATE from dual;

SYSDATE
-------------------
2013-04-22 16:07:15
```

一方、PostgreSQL には SYSDATE 関数が存在しないため、前述の CURRENT_DATE、CURRENT_TIMESTAMP として置き換えることを検討します。

なお、PostgreSQL の CURRENT_DATE、CURRENT_TIMESTAMP は「現在のトランザクションの開始時刻に基づいた値」を返す仕様のため、実際の現在時刻を取得したい場合は CLOCK_TIMESTAMP を利用します。

2.4.5. SYSTIMESTAMP

Oracle の SYSTIMESTAMP は、データベースサーバーが稼動する OS のシステムコールから取得した秒の小数部（ミリ秒～ナノ秒）と「タイムゾーンを含む」日付と時間を戻します。

PostgreSQL には SYSTIMESTAMP 関数が存在しないため、SYSDATE と同様に CURRENT_TIMESTAMP もしくは CLOCK_TIMESTAMP を利用します。

2.4.6. LAST_DAY

Oracle の LAST_DAY は、引数の日付の月末を求める関数です。

【Oracle の LAST_DAY の使用例 】

```
SQL> SELECT LAST_DAY('2013/4/22') FROM DUAL;

LAST_DAY
--------
13-04-30
```

一方、PostgreSQL には LAST_DAY 関数が存在しません。LAST_DAY 関数を呼び出す SQL を書き換えずに移行するためには、同様の計算を行うファンクションを作成する方法があります。

なお、Orafce には LAST_DAY 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

【Orafce 導入済の PostgreSQL での LAST_DAY の使用例】

```
postgres=# SELECT last_day(date '2013-04-22');
last_day
------------
 2013-04-30
(1 行)
```

2.4.7. NEXT_DAY

Oracle の NEXT_DAY は、指定した曜日で引数の日付より後の最初の日付を求める関数です。

【Oracle の NEXT_DAY の使用例 】

```
SQL> SELECT NEXT_DAY('2013/4/22','SUNDAY') FROM DUAL;

NEXT DAY
----------
2013-04-28
```

一方、PostgreSQL には NEXT_DAY 関数が存在しません。NEXT_DAY 関数を呼び出す SQL を書き換えずに移行するためには、同様の計算を行うファンクションを作成する方法があります。

なお、Orafce には NEXT_DAY 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

【Orafce 導入済の PostgreSQL での NEXT_DAY の使用例】

```
postgres=# SELECT next_day(date '2013-04-22', 'sunday');
next_day
------------
 2013-04-28
(1 行)
```

2.4.8. MONTHS_BETWEEN

Oracle の MONTHS_BETWEEN は、2 番目の引数から 1 番目の引数までの月数を求める関数です。

【Oracle の MONTHS_BETWEEN の使用例 】

```
SQL> SELECT MONTHS_BETWEEN('2013/03/15','2012/02/20') FROM DUAL;

MONTHS_BETWEEN('2013/03/15','2012/02/20')
-----------------------------------------
                                12.8387097
```

一方、PostgreSQL には MONTHS_BETWEEN 関数が存在しません。MONTHS_BETWEEN 関数を呼び出す SQL を書き換えずに移行するためには、同様の計算を行うファンクションを作成する方法があります。

なお、Orafce には MONTHS_BETWEEN 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

【Orafce 導入済の PostgreSQL での MONTHS_BETWEEN の使用例】

```
postgres=# SELECT months_between(date '2013-03-15', '2012-02-20');
months_between
------------
 12.8387097
(1 行)
```

2.4.9. ROUND

Oracle の ROUND は、引数の日付を書式モデルで指定した単位に丸めた結果を求める関数です、

【Oracle の ROUND の使用例 】

```
SQL> SELECT ROUND (TO_DATE ('27-OCT-12'),'YEAR') "New Year" FROM DUAL;

New Year
---------
01-JAN-13
```

一方、PostgreSQL には日付を引数に取る ROUND 関数は存在しません[3]。ROUND 関数を呼び出す SQL を書き換えずに移行するためには、同様の計算を行うファンクションを作成する方法があります。

なお、Orafce には ROUND 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

【Orafce 導入済の PostgreSQL での ROUND の使用例】

```
postgres=# SELECT round(date '2012-07-12', 'yyyy');
round
----------
2013-01-01
(1 行)
```

2.4.10. TRUNC

Oracle の TRUNC は、引数の日付を書式モデルで指定した単位まで切り捨てた結果を求める関数です。

【Oracle のTRUNCの使用例 】

```
SQL> SELECT TRUNC(TO_DATE('27-OCT-12','DD-MON-YY'), 'YEAR') "New Year" FROM DUAL;

New Year
---------
01-JAN-12
```

3　引数に数値を取り、指定した小数点位置で丸める ROUND 関数は存在します。

一方、PostgreSQL には日付を引数に取る TRUNC 関数は存在しません[4]。TRUNC 関数を呼び出す SQL を書き換えずに移行するためには、同様の計算を行うファンクションを作成する方法があります。

なお、Orafce には TRUNC 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

【Orafce 導入済の PostgreSQL での TRUNC の使用例】

```
postgres=# SELECT trunc(date '2012-07-12', 'yyyy');
round
----------
2012-01-01
(1 行)
```

2.5. 変換ファンクション

2.5.1. CONVERT

Oracle の CONVERT は、引数として char,dest_char_set,source_char_set を取り、変換対象文字列 char を変換前のキャラクタセット source_char_set から、変換後のキャラクタセット dest_char_set に変換した文字列を戻す関数です。

【Oracle の CONVERT の使用例 】

```
※Latin-1(WE8ISO8859P1)文字列を ASCII(US7ASCII)に変換する
SQL> SELECT CONVERT('A E I O O A B C D E ', 'US7ASCII', 'WE8ISO8859P1') FROM DUAL;
```

一方、PostgreSQL にも CONVERT 関数が存在しますが、変換前の符号化方式と変換後の符号化方式を引数で与える順序が逆になる点に注意が必要です。

【PostgreSQL の CONVERT の使用例】

```
※UTF-8 で登録されているデータ EUC-JP に変換して表示する
postgres=# SELECT convert(kanjiitem,'UTF-8','EUC-JP') FROM abctbl;
```

また、Oracle で指定できるキャラクタセットと PostgreSQL で指定できる符号化方式は名称が異なるため、引数の書き換えが必要となります。詳細は『Oracle Database グローバリゼーション・サポート・ガイド[5]』および『PostgreSQL 9.2.0 文書[6]』を参照してください。

2.6. エンコーディング・ファンクションおよびデコーディング・ファンクション

2.6.1. DECODE

Oracle の DECODE は、引数として expr,search,result を取り、expr が search と等しい場合に対応する result を返す関数です。

【Oracle の DECODE の使用例 】

```
※ warehouse_id が 1 の場合「Southlake」、2 の場合「San Francisco」、3 の場合「New Jersey」、
   4 の場合「Seattle」、1、2、3、4 のいずれでもない場合、「Non domestic」を戻す。
SQL> SELECT product_id,DECODE (warehouse_id, 1, 'Southlake',
                               2, 'San Francisco',
                               3, 'New Jersey',
                               4, 'Seattle',
                                  'Non domestic') "Location"
  FROM inventories  WHERE product_id < 1775  ORDER BY product_id;
```

PostgreSQL には DECODE 関数が存在しませんが、CASE 式に置き換えることで同等の機能を実現できます。

【Oracle の DECODE 使用例を CASE 式で置換えた例】

4 引数に数値を取り、指定した小数点位置で切り捨てる TRUNC 関数は存在します。

5 http://docs.oracle.com/cd/E16338_01/server.112/b56307/applocaledata.htm#i635016

6 http://www.postgresql.jp/document/9.2/html/functions-string.html#CONVERSION-NAMES

```
postgres=# SELECT product_id,
  CASE warehouse_id
    WHEN 1 THEN 'Southlake'
    WHEN 2 THEN 'San Francisco'
    WHEN 3 THEN 'New Jersey'
    WHEN 4 THEN 'Seattle'
    ELSE 'Non domestic'
  FROM inventories  WHERE product_id < 1775  ORDER BY product_id;
```

なお、Orafce には DECODE 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

2.7. NULL 関連ファンクション

2.7.1. NVL

Oracle の NVL は、引数として expr1,expr2 を取り、expr1 が NULL の場合 expr2 を、NULL でない場合 expr1 を返す関数です。

【Oracle の NVL の使用例 】

```
SQL> SELECT last_name, NVL(TO_CHAR(commission_pct), 'Not Applicable') commission FROM employees;
```

PostgreSQL には NVL 関数が存在しませんが、COALESCE 関数に置き換えることで同等の機能を実現できます。

【Oracle の NVL 使用例を COALESCE 関数で置換えた例】

```
postgres=# SELECT last_name, COALESCE(TO_CHAR(commission_pct), 'Not Applicable') commission
  FROM employees;
```

なお、Orafce には NVL 関数が実装されていますので、Orafce を導入した PostgreSQL では SQL を書き換えずに移行することができます。

3. 別紙一覧

- 別紙: 組み込み関数対応表(Oracle-PostgreSQL)

著者

版	所属企業・団体名	部署名	氏名
組み込み関数移行調査編 1.0 （2012 年度 WG2）	TIS 株式会社	戦略技術センター	中西 剛紀