

日々是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

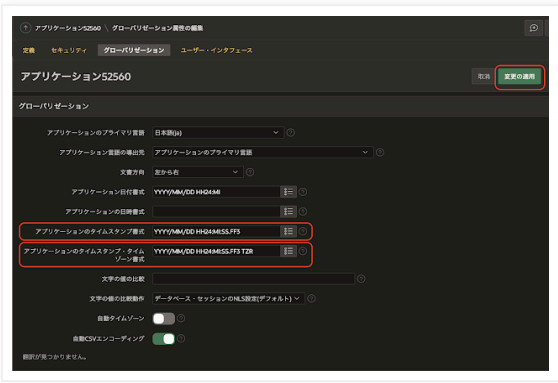
2020年6月18日 木曜日

Oracle APEXアプリケーションのグローバル化(8) - 日時データ型と現在時刻

Oracle APEXのアプリケーションへタイムゾーンを適用する方法について説明してきました。この記事では、Oracle APEXのアプリケーションを作成する上で必要な、Oracle Databaseの日時データ型と現在時刻の扱いについて説明してみます。

日時データ型の表示フォーマットについて

日時データ型がOracle APEXのアプリケーションでどのように認識されているかを目視で確認するために、今まで使用してきたサンプル・アプリケーションのグローバル化の設定を変更します。



日時データ型に関する書式設定を以下のように指定します。アプリケーション日付書式はそのまま変更しません。

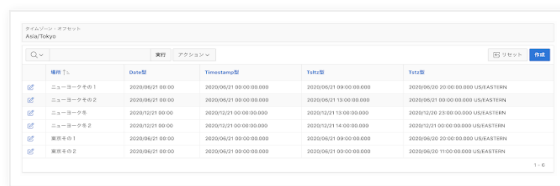
対象	書式	セッション設定
アプリケーション日付書式	YYYY/MM/DD HH24:MI	NLS_DATE_FORMAT
アプリケーションのタイムスタンプ書式	YYYY/MM/DD HH24:MI:SS.FF3	NLS_TIMESTAMP_FORMAT
アプリケーションのタイムスタンプ・タイムゾーン書式	YYYY/MM/DD HH24:MI:SS.FF3 TZR	NLS_TIMESTAMP_TZ_FORMAT

ここで指定された書式はページ処理の開始時点で、以下のSQL文と同等の処理によりセッションに設定されます。

```
alter session set NLS_DATE_FORMAT = 'YYYY/MM/DD HH24:MI';
alter session set NLS_TIMESTAMP_FORMAT = 'YYYY/MM/DD HH24:MI:SS.FF3';
alter session set NLS_TIMESTAMP_TZ_FORMAT = 'YYYY/MM/DD HH24:MI:SS.FF3 TZR';
```

結果としてOracle APEXのアプリケーションでも扱う日時データ型にしたがって(ページ・アイテムやレポートの列に書式マスクが設定されていない限り)、セッションに設定された書式が適用されます。

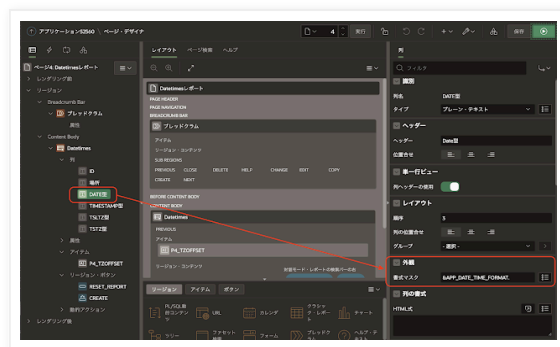
上記の書式が適用されたレポートを確認します。



	Date	Timestamp	Time	Time
1	2020/06/27 00:00	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000 LOCALTIME
2	2020/06/27 00:00	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000 LOCALTIME
3	2020/06/27 00:00	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000 LOCALTIME
4	2020/06/27 00:00	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000 LOCALTIME
5	2020/06/27 00:00	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000 LOCALTIME
6	2020/06/27 00:00	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000	2020/06/27 00:00:00.000 LOCALTIME

DATE型には**アプリケーション日付書式(NLS_DATE_FORMAT)**、**TIMESTAMP型**および**TSLTZ型(TIMESTAMP WITH LOCAL TIME ZONE型)**には**アプリケーションのタイムスタンプ書式(NLS_TIMESTAMP_FORMAT)**、**TSTZ型(TIMESTAMP WITH TIME ZONE型)**には**アプリケーションのタイムスタンプ・タイムゾーン書式(NLS_TIMESTAMP_TZ_FORMAT)**が適用されていることが確認できます。

グローバル化の設定で**アプリケーションの日時書式**のみ空白にしています。これはセッションの設定には使用されず、APP_DATE_TIME_FORMATという置換文字列としてアプリケーション内で利用することができます。



書式マスクを個別に指定する場所に、置換文字列として**&APP_DATE_TIME_FORMAT**を指定することにより、書式の変更を一箇所で出来るようになります。

現在時刻について

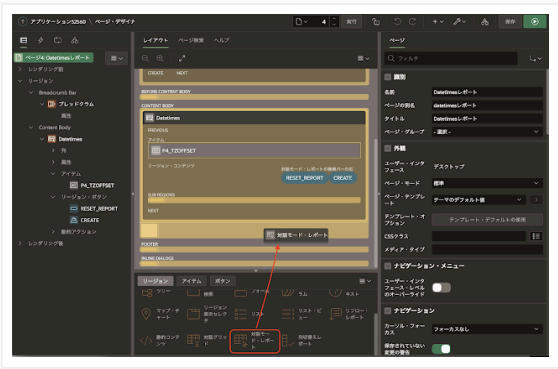
Oracle Databaseでは、現在の時刻を取得するために使用できるリテラルにいくつか種類があります。

- SYSDATE
- SYSTIMESTAMP
- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP

Oracle APEXのアプリケーションがこれらの現在時刻を表す日時リテラルをどのように認識するか確認するために、レポートとして表示させてみます。

Datetimesレポートのページに、対話モード・レポートのリージョンを追加します。ページ・デザインにてDatetimesレポートのページを開き、**コンポーネント・ギャラリー**より**対話モード・レポー**

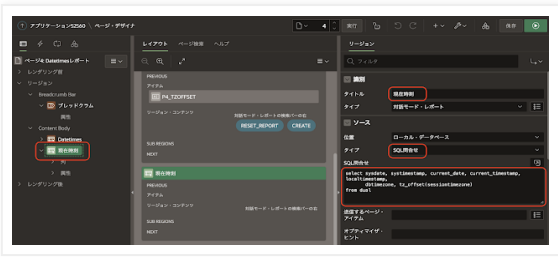
トをDatetimesレポートの直下に配置されるよう、ドラッグ&ドロップします。



レポートとして表示するSQLとして以下を与えます。

```
select sysdate, systimestamp, current_date, current_timestamp, localtimestamp,
       dbtimezone, tz_offset(sessiontimezone)
from dual
```

対話モード・レポートの設定は以下のようになります。タイトルは現在時刻としています。ソースの位置はローカル・データベース、タイプはSQL問い合わせとして、上記のSQLを設定します。



作成したレポートを、ページを実行して確認してみます。タイムゾーン・オフセットはAsia/Tokyoに設定されています。

Sysdate	Systimestamp	Current Date	Current Timestamp	Localtimestamp	dbtimezone	TZ_OFFSET(sessiontimezone)
2020/06/18 05:47	2020/06/18 05:47:38.111 +09:00	2020/06/18 14:47	2020/06/18 14:47:38.111 ASIA/TOKYO	2020/06/18 14:47:38.111	-05:00	+09:00

見やすく記述します。日本時間の6月18日14時47分には実行しています。

SYSDATE	2020/06/18 05:47
SYSTIMESTAMP	2020/06/18 05:47:38.111 +00:00
CURRENT_DATE	2020/06/18 14:47
CURRENT_TIMESTAMP	2020/06/18 14:47:38.111 ASIA/TOKYO
LOCALTIMESTAMP	2020/06/18 14:47:38.111
DBTIMEZONE	-05:00
TZ_OFFSET(SESSIONTIMEZONE)	+09:00

タイムゾーン・オフセットをUS/Easternに切り替えて表示してみます。

Sysdate	Systimestamp	Current Date	Current Timestamp	Localtimestamp	dbtimezone	TZ_OFFSET(sessiontimezone)
2020/06/18 05:52	2020/06/18 05:52:47.888 +09:00	2020/06/18 05:52	2020/06/18 05:52:47.888 US/EASTERN	2020/06/18 05:52:47.888	-08:00	+09:00

見やすく記述します。日本時間の6月18日14時52分に実行しています。

SYSDATE	2020/06/18 05:52
SYSTIMESTAMP	2020/06/18 05:52:41.988 +00:00
CURRENT_DATE	2020/06/18 01:52
CURRENT_TIMESTAMP	2020/06/18 01:52:41.988 US/EASTERN
LOCALTIMESTAMP	2020/06/18 01:52:41.988
DBTIMEZONE	-05:00
TZ_OFFSET(SESSIONTIMEZONE)	-04:00

1. SYSDATE

データベースが稼働しているオペレーティング・システムから得られる時刻をDATE型で返します。タイムゾーンを含まないので、Oracle APEXのアプリケーション利用者はサーバーと同じタイムゾーンからアクセスしていることが前提でなければ、現在時刻として扱うことはできません。

2. SYSTIMESTAMP

データベースが稼働しているオペレーティング・システムから得られる時刻をTSTZ型(TIMESTAMP WITH TIME ZONE型)で返します。時刻のみではなくタイムゾーンも含みますが、オペレーティング・システムから得られるタイムゾーンになります。

ユーザーのタイムゾーンでの時刻で表示するにはSYSTIMESTAMP AT LOCALとして、AT LOCAL演算子を適用する必要があります。SYSTIMESTAMP AT LOCALの結果はCURRENT_TIMESTAMPになるため、そのような場合はCURRENT_TIMESTAMPを使用すべきでしょう。

ユーザーのタイムゾーンで扱う必要のない場合、例えばプロシージャやトリガー内で使用できません。

3. CURRENT_DATE

セッションのタイムゾーン・オフセットを適用した現在時刻をDATE型で返します。Oracle APEXアプリケーションを使用しているユーザーのタイムゾーンでの現在時刻になります。

4. CURRENT_TIMESTAMP

セッションのタイムゾーン・オフセットを適用した現在時刻をTSTZ型(TIMESTAMP WITH TIME ZONE型)で返します。時刻部分がユーザーのタイムゾーンでの時刻表示になり、かつ、タイムゾーンの情報も含まれます。

5. LOCALTIMESTAMP

セッションのタイムゾーン・オフセットを適用した現在時刻をTIMESTAMP型で返します。時刻部分がユーザーのタイムゾーンでの時刻表示になります。

アプリケーションを作成する上では、DATE型が必要であればCURRENT_DATE、TIMESTAMP型が必要であればLOCALTIMESTAMP、タイムゾーンの情報が必要であればCURRENT_TIMESTAMP、またはSYSTIMESTAMPを使うことになります。

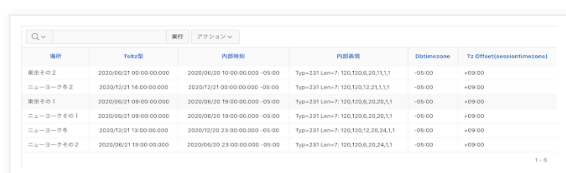
TSLTZ型とDBTIMEZONE

DBTIMEZONEというのはデータベースを作成するときに指定するタイムゾーンです。TSLTZ型はこのDBTIMEZONEで指定されたタイムゾーンの時刻として保存されます。タイムゾーン自体の情報は含みません。

内部表現を確認してみます。先ほどと同様な手順で対話モード・レポートを作成します。ソースとなるSQLとして以下を設定します。

```
select "場所",
       "TSLTZ型",
       "TSLTZ型" at time zone dbtimezone as "内部時刻",
       dump("TSLTZ型") as "内部表現",
       dbtimezone,
       tz_offset(sessiontimezone)
from test_datetimes
```

レポートの結果は以下になります。



場所	TSLTZ型	内部時刻	内部表現	Dbtimezone	To Offset(sessiontimezone)
東京その1	2020/06/21 09:00:00.000	2020/06/21 09:00:00.000 -05:00	Typ=231 Len=7: 120,120,6,20,20,1,1	-05:00	+09:00
ニューヨークその2	2020/06/21 19:00:00.000	2020/06/21 09:00:00.000 -05:00	Typ=231 Len=7: 120,120,6,20,20,1,1	-05:00	+09:00
東京その1	2020/06/21 19:00:00.000	2020/06/21 19:00:00.000 -05:00	Typ=231 Len=7: 120,120,6,20,20,1,1	-05:00	+09:00
ニューヨークその1	2020/06/21 09:00:00.000	2020/06/21 19:00:00.000 -05:00	Typ=231 Len=7: 120,120,6,20,20,1,1	-05:00	+09:00
ニューヨークその2	2020/06/21 19:00:00.000	2020/06/21 19:00:00.000 -05:00	Typ=231 Len=7: 120,120,6,20,20,1,1	-05:00	+09:00
ニューヨークその2	2020/06/21 19:00:00.000	2020/06/21 19:00:00.000 -05:00	Typ=231 Len=7: 120,120,6,20,20,1,1	-05:00	+09:00

東京その1のデータは以下になっています。

場所	東京その1
TSLTZ型	2020/06/21 09:00:00.000
内部時刻	2020/06/20 19:00:00.000 -05:00
内部表現	Typ=231 Len=7: 120,120,6,20,20,1,1
DBTIMEZONE	-05:00
TZ_OFFSET(SESSIONTIMEZONE)	+09:00

TSLTZ型は内部的には**2020/06/20 19:00:00.000(-05:00はデータとしては含まれない)**として保存されています。実際には**Typ=231 Len=7: 120,120,6,20,20,1,1**となっています。データが保存されるとき、取り出されるときに、DBTIMEZONEとSESSIONTIMEZONEの差分をオフセットとして適用します。

気を付けなければならないことは、TSLTZ型を使うとタイムゾーンが保存されないことです。一旦、TSLTZ型に保存された後はデータの入力がどのタイムゾーンから行われたのか確認する術が無くなります。であればTSTZ型を使うと良いのですが、TSTZ型ではほとんどの操作でタイムゾーンの変換が必要になり、パフォーマンス上好ましくはありません。

DBTIMEZONEで時刻を保存しても、時刻にタイムゾーンの情報を含んでも、結局は同じ時刻を表していることに違いはありません。そのため、アプリケーション上の要件としてはTIMESTAMP WITH LOCAL TIME ZONE型で不足はないでしょう。

時刻を表示する際、つねにタイムゾーンも同時に表示させる、または、同一のユーザでもタイムゾーンを切り替えてデータの入力を行うといったアプリケーションでは、TSTZ型を使った方がよいかもしれません。

続く

[ウェブ バージョンを表示](#)

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

[詳細プロフィールを表示](#)

Powered by Blogger.