日日是Oracle APEX

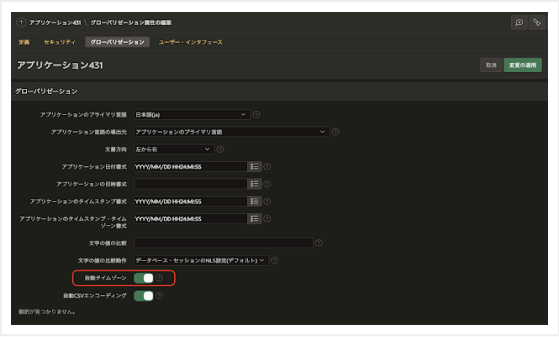
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2020年6月15日月曜日

Oracle APEXアプリケーションのグローバル化(6) - タイムゾーンの扱いと自動タイムゾーン

Oracle APEXのアプリケーションをグローバル化する方法をこれまで紹介してきました。日本国内に居住している外国籍の方を対象にしたアプリケーションであれば、各国語に翻訳を行えば対応としては十分でしょう。

そうではなく、企業などで海外の拠点からアプリケーションのアクセスがあるような場合、日時データの取り扱いも考慮する必要があります。Oracle APEXにはアプリケーションの**グローバリゼーション**属性に**自動タイムゾーン**という属性があり、これをオンにするとブラウザから取得したタイムゾーン・オフセットをOracle APEXのアプリケーションに反映させることができます。



自動タイムゾーンをオンに切り替えると、どのようなアプリケーションでもタイムゾーンを考慮する、というものではありません。その仕組みを理解して、アプリケーション自体をタイムゾーンを考慮して作る必要があります。

これから、そのタイムゾーンを考慮したアプリケーションの作成方法について紹介します。

自動タイムゾーンがOFFの場合

Oracle Databaseが持っている日時データ型には４つの種類があります。

1. DATE
2. TIMESTAMP
3. TIMESTAMP WITH LOCAL TIME ZONE
4. TIMESTAMP WITH TIME ZONE

この中で、タイムゾーンを認識するのは後ろの２つ、TIMESTAMP WITH LOCAL TIME ZONE型(以降TSLTZ型と記述します)とTIMESTAMP WITH TIME ZONE型(同TSTZ型)です。DATE型とTIMESTAMP型はタイムゾーンを認識しないため、タイムゾーンが異なる地域からの利用が想定されるアプリケーションでの使用はお勧めできません。

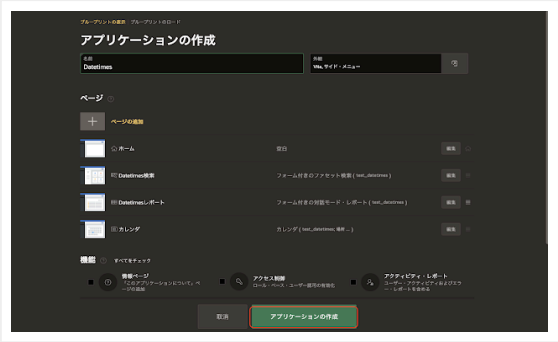
どのようなことが起こるか、実際にアプリケーションを使って確認してみます。

以下の表を作成します。

```
create table test_datetimes (
    id                  number generated by default on null as identity
                        constraint test_datetimes_id_pk primary key,
    "場所"              varchar2(80),
    "DATE型"            date,
    "TIMESTAMP型"       timestamp,
    "TSLTZ型"           timestamp with local time zone,
    "TSTZ型"            timestamp with time zone
);
```

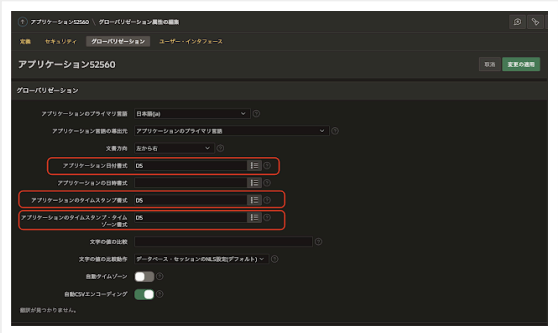
表を作成したら、SQLワークショップのオブジェクト・ブラウザから作成した表TEST_DATETIMESを選択し、**アプリケーションの作成**を実行します。

ウィザードによって作成されるアプリケーションは特に変更せず、**アプリケーションの作成**を行います。ただし、確認に使用するのは**Datetimesレポート**として作成される対話モード・レポートとフォームだけです。

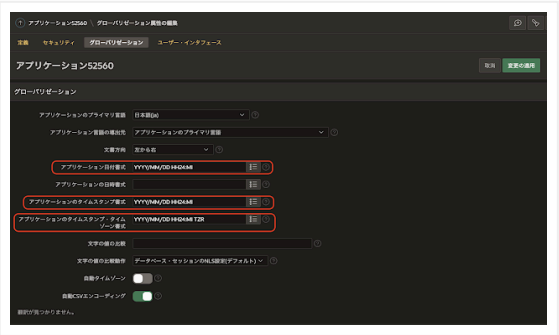


これで確認に使用するアプリケーションが出来上がりました。

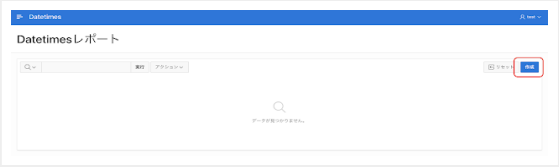
グローバリゼーションの設定を確認します。いくつかの書式のデフォルトがDSになっています。この書式では時刻が表示されません。



時刻が表示されるように**アプリケーションの日付書式**と**アプリケーションのタイムスタンプ書式**を**YYYY/MM/DD HH24:MI**へ変更します。**アプリケーションのタイムスタンプ・タイムゾーン書式**はタイムゾーンが表示されるようにTZRを追加して**YYYY/MM/DD HH24:MI TZR**に変更します。今のところは自動タイムゾーンは変更せずOFFにしておきます。

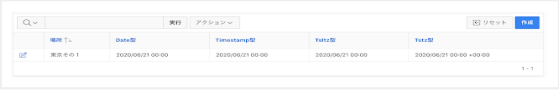


アプリケーションを実行して、**Datetimesレポート**を開きます。**作成**をクリックします。



先ほど作成した表TEST_DATETIMESにデータを登録するフォームが開きます。すべての日時型に同じデータ"**2020/06/21 00:00**"を入力し、**作成**をクリックします。

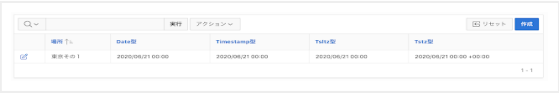
レポートに戻るので登録内容を確認します。登録したデータと同じ日時が表示されます。



TSTZ型についてはタイムゾーンの情報が+00:00と表示されています。UTCと見做せますが、アプリケーションをタイムゾーンに対応させていないので、タイムゾーン・オフセットが０になっているだけです。

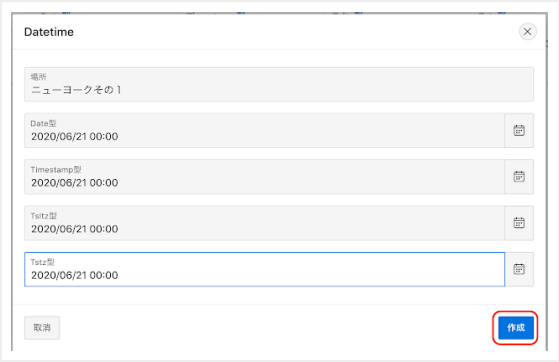
上記は東京からアクセスしています。同じアプリケーションにニューヨークからアクセスしてみます。ニューヨークに出張するわけにもいかないので、コンピュータの設定を変更します。macOSの場合、システム環境設定の日付と時刻で時間帯を切り替えることができます。同様の設定はWindowsにもLinuxにもあるでしょう。

タイムゾーンをニューヨークに切り替えたのち、再度レポートを表示させます。

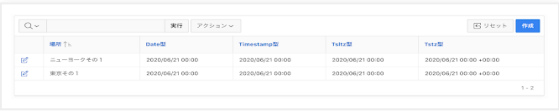


同じ情報が表示されています。アプリケーションの利用者は、表示されている時刻**2020/06/21 00:00**は東京での時刻であると理解している必要があります。

ニューヨークからデータを入力します。東京と同様に2020/06/21 00:00を設定します。



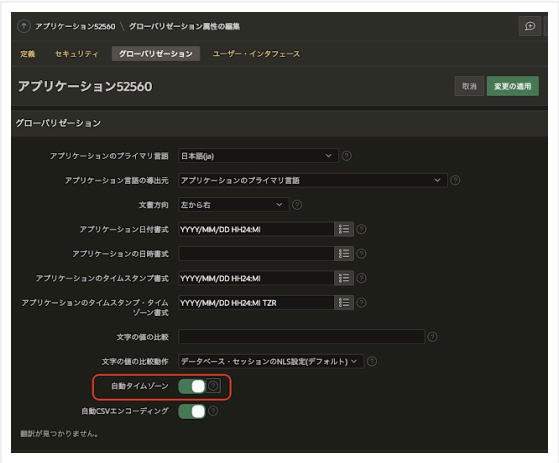
登録されたデータがレポートに表示されます。東京で登録したデータとニューヨークで登録したデータは、レポートから見て、全く同一になります。



アプリケーションが保持する日時データはすべて東京のタイムゾーンである、と決めている場合はどの日時型を使用しても、あまり問題はありません。しかし、このままでは東京の利用者が東京の現地時刻でアプリケーションを使用し、ニューヨークの利用者はニューヨークの現地時刻でアプリケーションを使用する、といった状況に対応できません。ニューヨークの利用者は時差計算を自分で行う必要があり、アプリケーションとしての使い勝手は低くなります。

自動タイムゾーンがONの場合

自動タイムゾーンをONへ変更します。

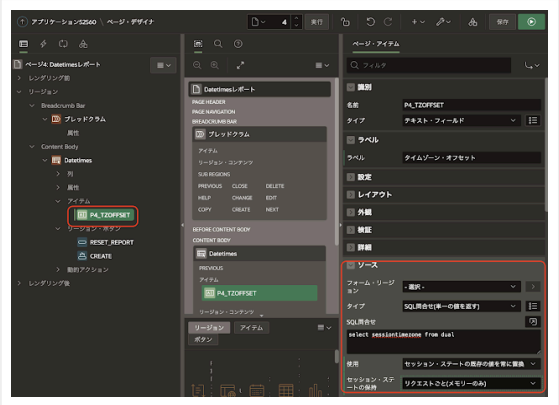


また、アプリケーションに少し変更を加えます。Datetimesレポートに認識されているタイムゾーン・オフセットを表示するページ・アイテムを追加します。

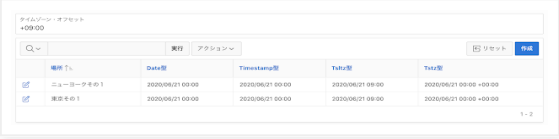
名前を**P4_TZOFFSET**、**ラベル**を**タイムゾーン・オフセット**、**ソース**の**タイプ**を**SQL問合わせ(単一の値を返す)**として、以下のSQL文を**SQL問合わせ**に指定します。

select sessiontimezone from dual

使用は**セッション・ステートの既存の値を常に置換**、**セッション・ステートの保持**は**リクエストごと(メモリーのみ)**としておきます。



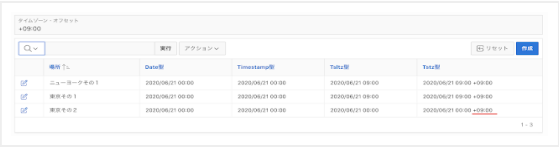
アプリケーションにページ・アイテムを追加した後、レポートを表示します。



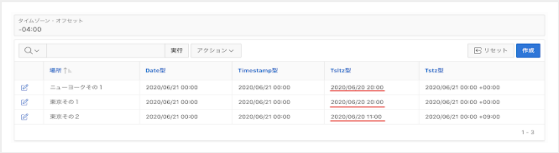
タイムゾーン・オフセットとして +09:00 が設定されていることが確認できます。レポートを見ると、DATE型、TIMEZONE型、TSTZ型はタイムゾーン・オフセットの影響を受けていないことがわかります。TSLTZ型つまりTIMESTAMP WITH LOCAL TIME ZONE型は、保存されているデータにタイムゾーン・オフセットが適用され、**2020/06/21 09:00**として時刻が表示されています。

自動タイムゾーンがONの状態で、再度、同じ時刻を登録します。

レポートに表示される結果を確認します。TSTZ型に+09:00というタイムゾーンが加わっています。またTSLTZ型は登録した**2020/06/21 00:00**が表示されされています。データの登録時に東京での時間で**2020/06/21 00:00**と認識されているので、同じ表示になります。DATE型、TIMESTAMP型はタイムゾーンの影響を受けないため、自動タイムゾーンがOFFのときに登録したデータと変わりがありません。



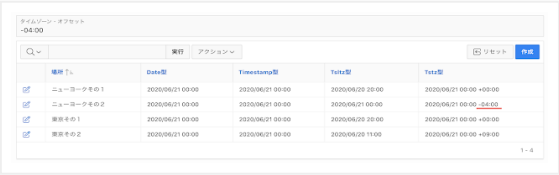
次にニューヨークから自動タイムゾーンをONにしたアプリケーションにアクセスします。



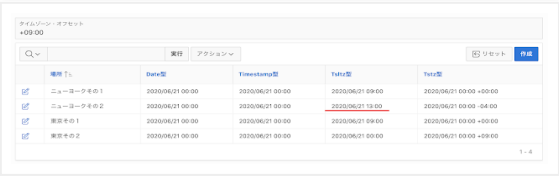
タイムゾーン・オフセットとして -04:00 が設定されていることが確認できます。TSLTZ型には、設定されたタイムゾーン・オフセットが適用されています。つまり、ここで表示されている時刻はニューヨークの現地時間です。13時間(4+9)進めると日本時間になります。

ニューヨークからも同様に2020/06/21 00:00の時刻を指定してデータを登録します。

レポートを確認すると、TSTZ型で -04:00 が追加されていますが、それ以外は登録したままの情報になっています。



再度、東京からレポートを確認します。

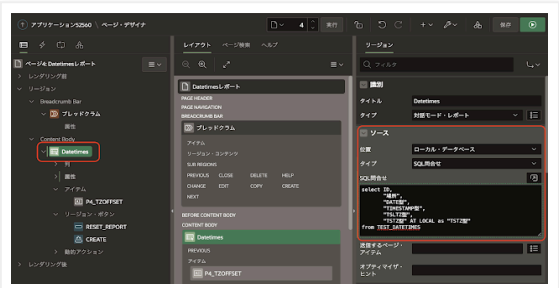


TSLTZ型のみ、タイムゾーン・オフセットが反映された時刻が表示されています。先ほどニューヨークで入力した**2020/06/21 00:00**はニューヨークの現地時間であり、東京からみたレポートでは13時間進んだ(4+9=13)、**2020/06/21 13:00**として表示されています。

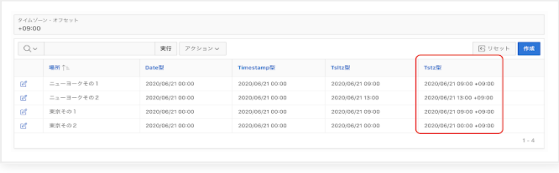
以上のように、**グローバリゼーション**属性の**自動タイムゾーン**をONにすることで、TSLTZ型(TIMESTAMP WITH LOCAL TIME ZONE型)は常にブラウザが認識した(タイムゾーン・オフセットを適用した)現地時刻で表示されるようになります。TSTZ型(TIMESTAMP WITH TIME ZONE型)も同様にタイムゾーンを認識しますが、データそのものにタイムゾーンが含まれているため、現地時刻を表示しません。現地時刻として表示するには、**AT LOCAL**演算子を適用します。

先ほどのDatetimesレポートのTSTZ型を現地時刻で表示させるには、レポートのソースを以下のSQLに変更します。

```
select ID,
        "場所",
        "DATE型",
        "TIMESTAMP型",
        "TSLTZ型",
        "TSTZ型" AT LOCAL as "TSTZ型"
from TEST_DATETIMES
```



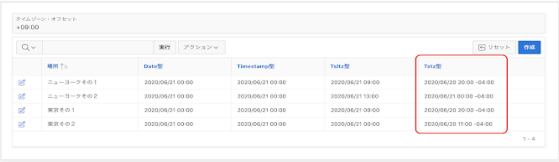
この変更の結果、TSTZ型の表示は現行のタイムゾーン・オフセットでの時刻表示に変わります。TSLTZ型はタイムゾーン・オフセットが適用された時刻ですので、TSTZ型にAT LOCAL演算子を適用すると、時刻表示は同じになります。



それ以外に、TSLTZ、 TSTZ型はともに**AT TIME ZONE 'オフセットまたはタイムゾーン名'**演算子を適用することにより、指定したタイムゾーンの時刻に変換することができます。例えば、タイムゾーン・オフセットが東京(+09:00)と認識されている状態でニューヨークの時間で表示するにはAT TIME ZONE '-04:00'を指定します。

```
select ID,
        "場所",
        "DATE型",
        "TIMESTAMP型",
        "TSLTZ型",
        "TSTZ型" AT TIME ZONE '-04:00' as "TSTZ型"
from TEST_DATETIMES
```

このようにレポートのソースを変更した結果が以下になります。

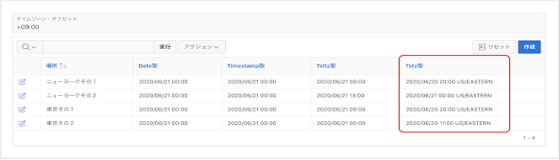


認識されているタイムゾーン・オフセットにかかわらず、ニューヨークの時間で表示されます。

AT TIME ZONEでの指定ではタイムゾーン・オフセットという数値だけではなく、名前による指定も可能です。利用可能な名前は以下のSQLにて確認できます。

```
SELECT TZNAME, TZABBREV
FROM V$TIMEZONE_NAMES
ORDER BY TZNAME, TZABBREV;
```

先ほどの指定はAT TIME ZONE 'US/Eastern'と指定することもできます。変更した結果は以下になります。

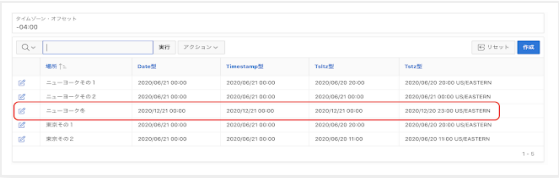


-04:00とUS/Easternの両方で同じ時刻の表示になっています。タイムゾーン・オフセットの数値と名前の指定の違いは夏時間の扱いです。名前による指定であれば、夏時間を考慮しますが、数値でタイムゾーン・オフセットを指定する場合はその数値がそのまま適用されます。ですので、夏時間を考慮した時刻表示を行う場合は、AT TIME ZONE演算子はタイムゾーン名による指定が必要になります。

自動タイムゾーンでは、夏時間のときはブラウザがタイムゾーン・オフセットとして-04:00を返し、そうでない場合は-05:00を返してくると想定しています。夏時間のない日本では問題になりませんが、夏時刻がある地域ではタイムゾーン・オフセットの扱いに問題が発生します。

利用者がニューヨークにいて夏時間であれば、タイムゾーン・オフセットは-04:00です。その状況で先ほどのアプリケーションにて、**2020/12/21 00:00**の時刻を設定します。

レポートは以下のように表示されます。



TSLTZ型、TSTZ型ともに保存時はブラウザで認識されているタイムゾーン・オフセット-04:00が適用されます。TSLTZ型は表示時も認識されているタイムゾーン・オフセットが適用され、結果として時刻の表示は変わらず2020/12/21 00:00ですが、TSTZ型はAT TIME ZONE 'US/Eastern'として夏時間を考慮させているため、表示ではタイムゾーン・オフセットとして-05:00が適用され、**2020/12/20 23:00**となります。

夏時間の対応が必要な場合は、Oracle APEXの自動タイムゾーンは少々力不足です。

夏時間まで考慮するとタイムゾーンは名前で設定する必要があります。これは自動タイムゾーンではできません。Oracle APEXが提供するAPEX_UTILパッケージに

SET_SESSION_TIME_ZONEプロシージャ
GET_SESSION_TIME_ZONEファンクション

に含まれており、こちらを利用することでセッション単位で名前による指定を含む任意のタイムゾーンを設定することが可能です。記事が長くなりましたので、これらの解説はまた項を改めて行います。

続く

Yuji N. 時刻: 19:18

共有

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.