

# 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。

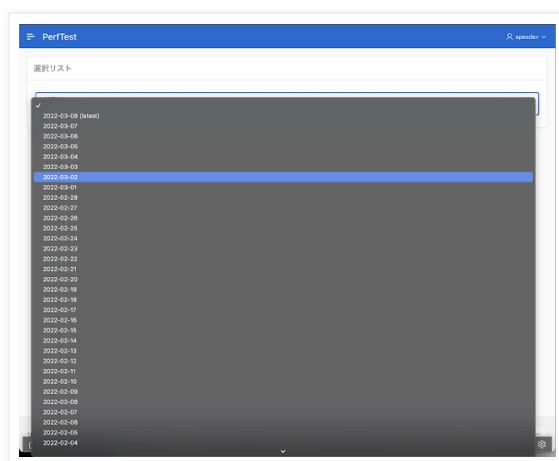
2022年3月9日水曜日

## 画面の表示に時間がかかるときのデバッグ作業

APEXで作ったアプリの画面の表示に時間がかかる、という相談を受けた時に行った作業を紹介します。

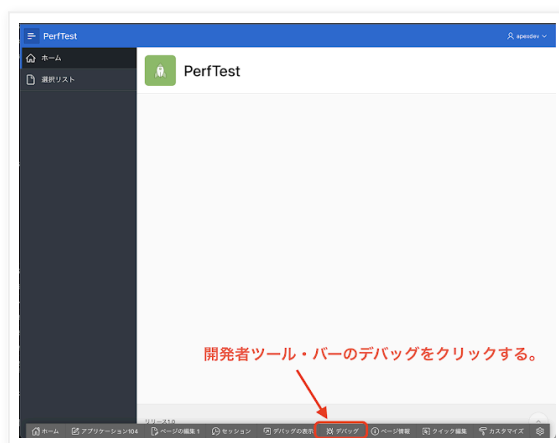
確認作業にAlways FreeのAutonomous Transaction Processingのインスタンスを使用しています。ただし、相談を受けたインスタンスはExadataではありませんでした。

元の画面はもっと複雑なようですが、結果として問題だったのはページ・アイテムの選択リストでした。以下のように選択リストのページ・アイテムがひとつだけ作成されている、簡素化したページを例に取ります。

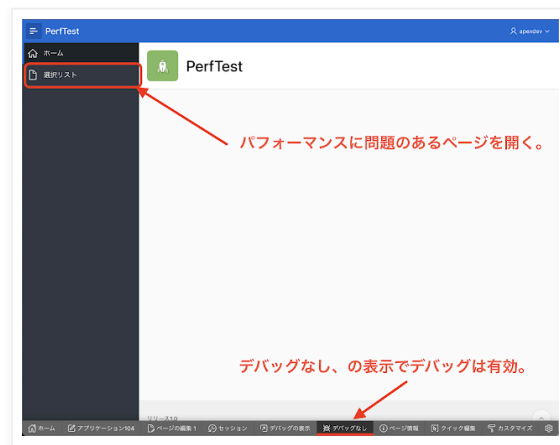


最初に**デバッグ**を有効にします。

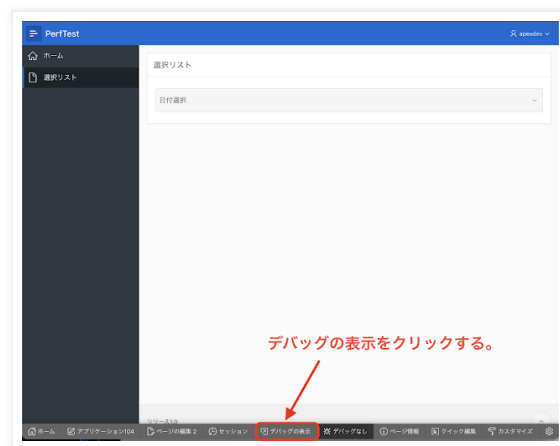
開発者ツール・バーの**デバッグ**をクリックします。



デバッグを有効にしたのち、パフォーマンスに問題のあるページを開きます。



画面が表示されたら、**開発者ツール・バーのデバッグの表示**をクリックし、デバッグ画面を開きます。



デバッグ画面が開いたら、**アプリケーション、ページやタイムスタンプ**などより、デバッグの対象にした処理に当たりをつけて、**ビュー識別子**をクリックします。

アイテム	ページ	操作	実行	PL/SQL	デバッグ	セッション	エラー
ビュー識別子	アプリケーション = 1004	ページ = 2	実行				
ビュー識別子	アプリケーション = 1004	ページ = 2	実行				
ビュー識別子	アプリケーション = 1004	ページ = 2	実行				
ビュー識別子	アプリケーション = 1004	ページ = 2	実行				
ビュー識別子	アプリケーション = 1004	ページ = 2	実行				

デバッグ・メッセージの一覧が表示されます。グラフを見ると、突出して実行時間が長い処理があることが分かります。

経過	実行	メッセージ	レベル	グラフ
0.00095	0.00000	Reset NLS settings	4	
0.00701	0.00020	alter session set NLS_LANGUAGE='AMERICAN' NLS_TERRITORY='AMERICA' NLS_SORT='BINARY' NLS_CYP='BINARY' NLS_CAI='BINARY' NLS_CAD='BINARY'	4	
0.00736	0.00040	...NLS_DATE_FORMAT='MM-DD HH:MM:SS.FF' AM timestamp_format='MM-DD HH:MM:SS.FF' AM timestamp_tz_format='MM-DD HH:MM:SS.FF' AM TZ_INTERVAL='00:00:00' group separator=','	4	
0.00782	0.00000	...Setting session time_zone to +09:00	4	
0.00786	0.00054	RESET show	5	
0.01040	0.00040	Language derived from: PLSQL_PRIMARY_LANGUAGE, current browser language: ja	4	
0.01082	0.00002	alter session set nls_language='JAPANESE' nls_territory='JAPAN'	4	
0.01084	0.00004	...NLS_CAI='BINARY' NLS_CAD='BINARY'	4	
0.01088	0.00020	Setting NLS_DATE_FORMAT='MM-DD HH:MM:SS.FF' AM timestamp_format='MM-DD HH:MM:SS.FF' AM timestamp_tz_format='MM-DD HH:MM:SS.FF' AM TZ_INTERVAL='00:00:00' group separator=','	4	
...	...	...NLS_DATE_FORMAT='MM-DD HH:MM:SS.FF' AM timestamp_format='MM-DD HH:MM:SS.FF' AM timestamp_tz_format='MM-DD HH:MM:SS.FF' AM TZ_INTERVAL='00:00:00' group separator=','	4	

また、デバッグ・メッセージの一覧は対話モード・レポートなので、例えば実行時間が10秒以上かかっている処理といった条件でも検索できます。



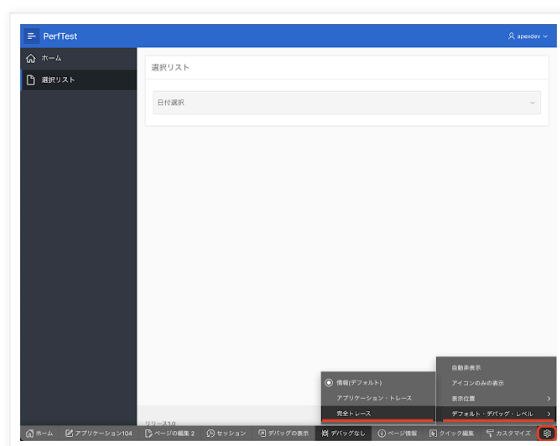
デバッグ・メッセージの一覧より、実行時間の長い部分の近辺を確認します。

ページ・アイテムP2\_DATEの処理に時間がかかっている、ということ以外は分かりません。

経過	実行	メッセージ	レベル	グラフ
0.02054	0.00002	← APEX_APPLICATION_PAGE_ITEM P2_DATE	3	
0.02056	0.00018	← APEX_APPLICATION_PAGE_REGION 選択リスト	3	
0.02054	0.00019	← APEX_APPLICATION_PAGE_REGION 選択リスト	3	
0.02053	0.00019	← APEX_APPLICATION_PAGE_ITEM P2_DATE	3	
0.03350	0.00002	← APEX_APPLICATION_PAGE_ITEM P2_DATE	3	
0.03352	0.00004	← APEX_APPLICATION_PAGE_REGION 選択リスト	3	
0.03356	0.00005	Evaluate which regions should be rendered for display point BODY_1	4	
0.03359	0.00007	Evaluate which regions should be rendered for display point BODY_2	4	

デバッグ・レベルを完全トレースまで上げて、ログを取得します。

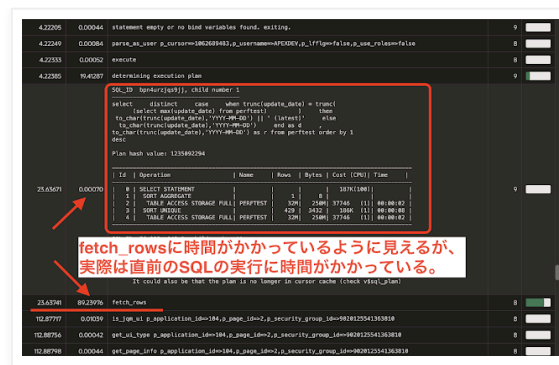
開発者ツール・バー・オプションを開き、デフォルト・デバッグ・レベルを完全トレースに変更します。



デバッグ・レベルを上げたのち、パフォーマンスが問題となっている操作を再度行います。

デバッグ・メッセージの中から、実行時間の長い処理を確認します。

メッセージとしてfetch\_rowsの実行時間が長い場合があります。これは直前のSELECT文の実行時間です。SELECT文の実行時間は、実際の実行時間ではなく実行計画の表示にかかった時間とされます。



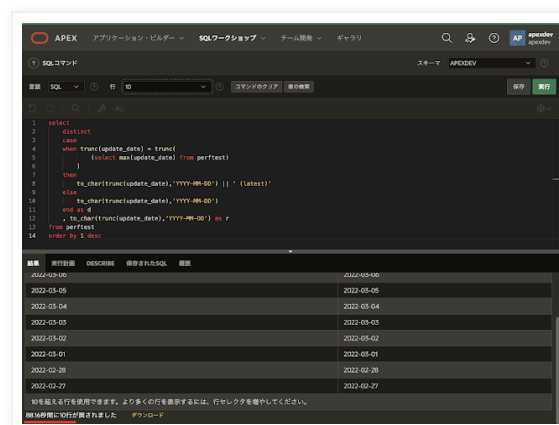
fetch\_rowsの直前に表示されているSELECT文に時間がかかっていることが分かるので、このSELECT文をSQLコマンドで実行します。

選択リストのLOVのソースとして、最初は以下のSELECT文が使われていました。

検索対象の表PERFTESTには列UPDATE\_DATEがあります。131,072,000行のデータが投入されています。

```
select
  distinct
  case
    when trunc(update_date) = trunc(
      (select max(update_date) from perfctest)
    )
  then
    to_char(trunc(update_date), 'YYYY-MM-DD') || ' (latest)'
  else
    to_char(trunc(update_date), 'YYYY-MM-DD')
  end as d
, to_char(trunc(update_date), 'YYYY-MM-DD') as r
from perfctest
order by 1 desc
```

SELECT文の実行に88.16秒かったことが分かります。



実行計画を確認することもできます。

```

1 select
2   distinct
3   case
4     when trunc(update_date) = trunc(
5       (select max(update_date) from perfest)
6     )
7   then
8     to_char(trunc(update_date), 'YYYY-MM-DD') || ' (latest)'
9   else
10    to_char(trunc(update_date), 'YYYY-MM-DD')
11  end as d,
12  rownum as r
13  from perfest
14  order by d desc

```

操作	実行計画	DESCRIPTION	実行されたSQL	行数
SELECT STATEMENT				429
Sort	AGGREGATE			1
TABLE ACCESS	STORAGE FULL	PERFESET		61072,000
Sort	UNIQUE			429
TABLE ACCESS	STORAGE FULL	PERFESET		61072,000

SELECT文が意図しているのは以下です。

1. 列UPDATE\_DATEの内容を、日付を単位として選択リストに表示する。
2. 降順で表示する。
3. 最も最近の日付に、ラベル(latest)を付加する。

若干、SELECT文が複雑すぎるようです。ファンクションto\_charやtruncといった処理が呼び出される回数が減るようにSELECT文を書き直します。

```

select
  case when rownum = 1 then
    update_date || ' (latest)'
  else
    update_date
  end as d,
  update_date r
from
(
  select update_date from
  (
    select
      to_char(update_date, 'YYYY-MM-DD') update_date
    from perfest
  )
  group by update_date order by update_date desc
)

```

実行時間は**43.44秒**まで改善されました。

```

1 select
2   case when rownum = 1 then
3     update_date || ' (latest)'
4   else
5     update_date
6   end as d,
7   update_date r
8  from
9  (
10   select update_date from
11   (
12     select
13       to_char(update_date, 'YYYY-MM-DD') update_date
14     from perfest
15   )
16   group by update_date order by update_date desc
17  )

```

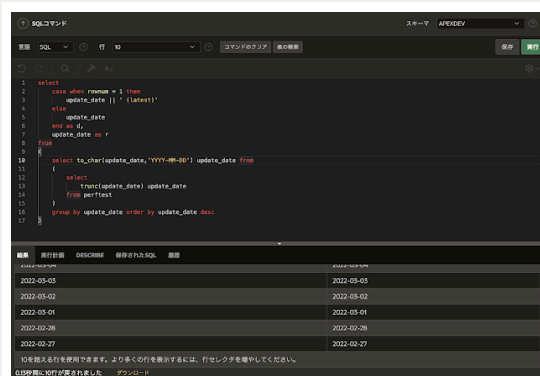
操作	実行計画	DESCRIPTION	実行されたSQL	行数
SELECT STATEMENT				429
Sort	AGGREGATE			1
TABLE ACCESS	STORAGE FULL	PERFESET		61072,000

(確認した結果) to\_charよりtruncの方がCPU負荷が低かったなので、表PERFESETの全行に対して実行するファンクションをto\_charからtruncに変え、集計した結果にto\_charを適用するように

SELECT文を変更しました。

```
select
  case when rownum = 1 then
    update_date || ' (latest)'
  else
    update_date
  end as d,
  update_date as r
from
(
  select to_char(update_date,'YYYY-MM-DD') update_date from
  (
    select
      trunc(update_date) update_date
    from perfctest
  )
  group by update_date order by update_date desc
)
```

実行結果が**0.13**秒まで改善しました。

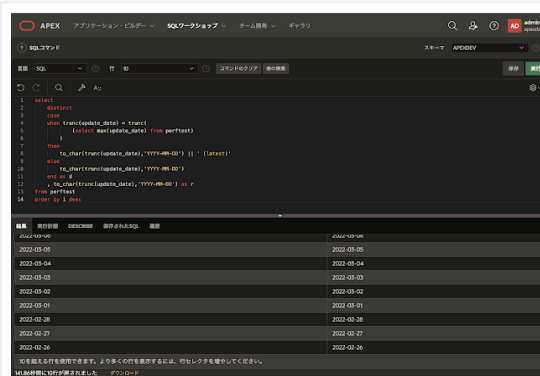


update_date	update_date
2022-05-03	2022-05-03
2022-05-02	2022-05-02
2022-05-01	2022-05-01
2022-02-28	2022-02-28
2022-02-27	2022-02-27

これで解決かと思ったのですが、Autonomous DatabaseだとCPUを使う処理を減らすと（それ以外が高速なので）結果は良くなるのですが、Exadataでない場合はそこまで改善しないようです。

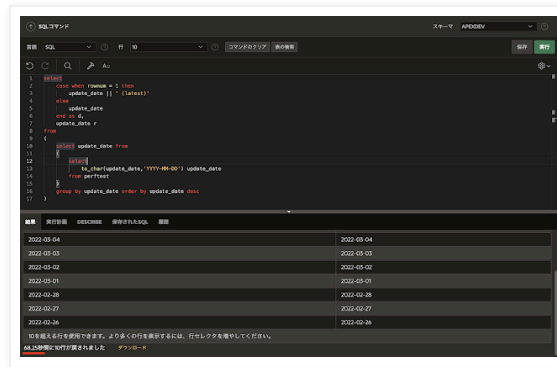
比較のため、手元のVirtualBoxで構成したOracle Database 21c Express EditionのAPEXで実行してみました。

最初のSELECT文の実行は**141.86**秒でした。

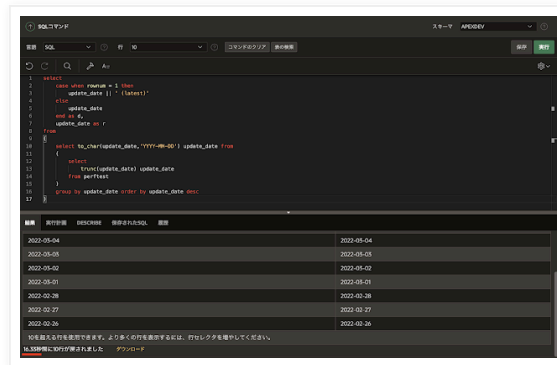


update_date	update_date
2022-05-05	2022-05-05
2022-05-04	2022-05-04
2022-05-03	2022-05-03
2022-05-02	2022-05-02
2022-05-01	2022-05-01
2022-02-28	2022-02-28
2022-02-27	2022-02-27
2022-02-26	2022-02-26

2つめのSELECT文の実行は**68.25**秒でした。



最後のSELECT文の実行は**16.33秒**で、実用に耐えられる実行時間ではありません。



これ以上の改善は難しいだろうということで、選択リストが使うマテリアライズド・ビューを作り、定期的にリフレッシュすることになります。

```
create materialized view perftest_days
build immediate
refresh complete on demand
as
select to_char(update_date,'YYYY-MM-DD') update_date from
(
select
    trunc(update_date) update_date
from perftest
)
group by update_date;
```

以下のSQLでマテリアライズド・ビューのリフレッシュを行います。

```
begin
    dbms_mview.refresh('PERFTEST_DAYS','C');
end;
```

マテリアライズド・ビューのリフレッシュには時間がかかりますが、1日に1回実行すればことが足ります。

選択リストのSQL問合せは、マテリアライズド・ビューを使う前提だと以下になります。

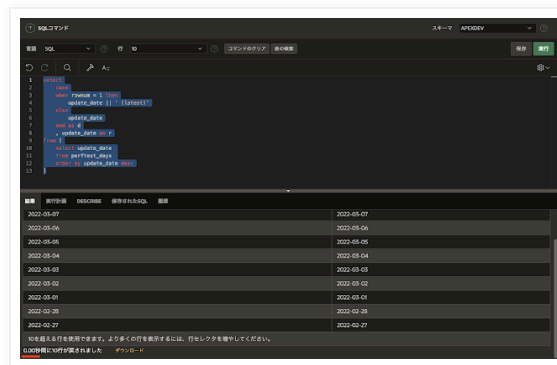
```
select
    case
        when rownum = 1 then
            update_date || ' (latest)'
        else
            update_date
        end as d
    , update_date as r
```

```

from (
  select update_date
  from perfctest_days
  order by update_date desc
)

```

実行時間は**0.00**秒となり、ほとんど無視できるようになりました。

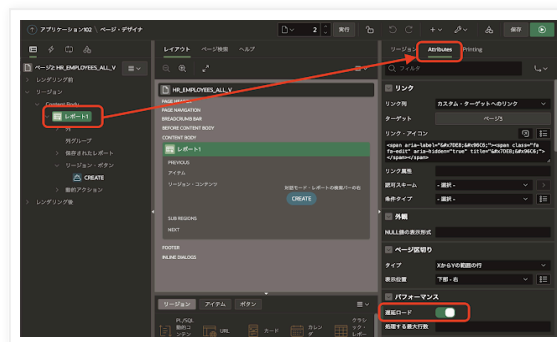


デバッグ作業の紹介は以上になります。

Oracle APEXのアプリケーション作成の参考になれば幸いです。

## 追記

対話グリッドや対話モード・レポートなど、リージョンで**遅延ロード**というプロパティを持つものがあります。



遅延ロードが有効なリージョンのソースとなるSQLは、ページのロードとは別に実行されます。デバッグ・メッセージの一覧を見ると、パス情報が**ajax plugin**になっているものが、遅延ロードがONのリージョンのソースとなるSQLの実行です。



デバッグをするときは、**遅延ロードをOFFにするとデバッグが容易になる**でしょう。

完



[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.

---