リソース資源の使用制限　使用制限対象：oracle ユーザー毎

Oracle ユーザーに対し、リソース使用制限

１つの SQL 文の実行時間が指定 CPU 秒を超える処理は、SQL 処理を強制終了させられる

ここで制限されるのは、CPU 使用時間であり、待機イベントでの待ち時間は、含まれない

使用制限が掛る対象範囲には、ユーザー名で指定を行う

CPU 実行時間を超えた場合には、以下のようなエラー・メッセージが出力され、処理は、中止（ロールバック）される

ORA-00040: アクティブな時間制限を超えました - コールは異常終了しました

```
begin
  for  i  in  35..300000  loop
      insert into  "SYSTEM"."EMP"  values( i , 'dummy', mod( i , 5 ) , 'いろ' ) ;
  end  loop  ;
end  ;
/
```

```
行 1 でエラーが発生しました。:
ORA-00040: アクティブな時間制限を超えました
                    - コールは異常終了しました
```

設定例）

指定 CPU 時間：　　5CPU 秒
制限対象範囲：　　ユーザー名 KOZUE

順序番号は、設定概要で記述した番号に合わせて表記変更し、コマンド実行の記述順序は、書籍にあった例題の表示どおりに従う

```
BEGIN
-- 1.ペンディングエリア作成
    DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;
    DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;

-- 2.コンシューマ・グループ作成
    DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(
        consumer_group => 'ONLINE_GRP' ,        -- コンシューマ・グループ
        comment => 'online user'
    ) ;

-- 3.コンシューマ・グループの割り当て
--          コンシューマ・グループに『リソース使用制限の対象範囲』を設定
    DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP(
        user => 'kozue' ,
        consumer_group => 'ONLINE_GRP'
    ) ;
```

この設定の操作は、手順９）の後で行っても構わない

```
-- 4.リソースプラン作成
    DBMS_RESOURCE_MANAGER.CREATE_PLAN(
        plan => 'SQL_TIMEOUT' ,                 --リソースプラン名
        comment => 'sql timeout plan'
    ) ;

-- 5.リソースディレクティブ作成（リソース使用制限の対象範囲の指定）
    DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
        PLAN => 'SQL_TIMEOUT' ,
        GROUP_OR_SUBPLAN => 'OTHER_GROUPS' ,
```

```
        Comment => 'other group'
    ) ;
```

※ リソースの使用制限をまったく行わないリソースディレクティブを作成し、『リソースプラン』に紐付ける
これにより、別のリソースディレクティブで対象範囲外となった者へ、この使用制限無しリソースディレクティブを適用する
このために、記述されている

```
    DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
        plan => 'SQL_TIMEOUT' ,
        group_or_subplan => 'ONLINE_GRP' ,
        switch_group => 'CANCEL_SQL' ,              -- SQL を取り消し
        switch_time => 5 ,                          -- 5 秒でタイムアウト
        switch_estimate => FALSE ,
        Comment => 'online group'
    ) ;
```

ここで制限されるのは、CPU 使用時間である
待機イベントでの待ち時間は、含まれない

```
-- 6.リソースプラン検証（設定の有効性の確認）
    DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA( );


-- 7. リソースプラン有効化（ペンディングエリアのクローズ）
    DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA( );


-- 8.コンシューマ・グループへのリソース使用制限切替えの権限付与
    DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER
    _GROUP(
        grantee_name => 'kozue' ,
        consumer_group => 'ONLINE_GRP' ,
        grant_option => FALSE
    ) ;

END;
/
```

作成した＜リソースプラン名＞の名前を指定する

```
-- 9. 完成したリソースプランの運用のための適用
```

ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'SQL_TIMEOUT' SCOPE = ＜MEMORY or SPFILE or BOTH＞;

```
-- 8.コンシューマ・グループへのリソース使用制限切替えの権限付与
BEGIN
    DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER
    _GROUP(
        grantee_name => 'ikura' ,
        consumer_group => 'ONLINE_GRP' ,
        grant_option => FALSE
        ) ;
END;
/
```

```
-- 3.  （コンシューマ・グループ割当）
--          コンシューマ・グループに『リソース使用制限の対象範囲』を設定
    EXECUTE
            DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_
            GROUP( user => 'ikura' , consumer_group => 'ONLINE_GRP' ) ;
```

【参考情報】

リソースプランを削除する手順

```
ALTER SYSTEM RESET RESOURCE_MANAGER_PLAN
                    SCOPE=MEMORY;

execute DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;

execute DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;

execute DBMS_RESOURCE_MANAGER.DELETE_PLAN_CASCADE
                    ( plan => 'SQL_TIMEOUT' );

execute DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA( );
```