【サブトランザクションだけのコミット操作】

本体のトランザクション中に、分離したサブトランザクションを作成して、本体のトランザクション制御から独立して COMMIT させる方法させます。

例えば、バッチ処理の実行経過を外から見られるようにするために、『バッチ処理のステップごとの終了』をログテーブルへレコードを INSERT するような運用を考えます
バッチ処理は、異常終了のデータ復旧のためにトランザクションの最終ステップでの一括トランザクションを行うロジックが必須となります
通常であれば、バッチ処理中にこのログテーブルを SELECT しても、コミットを行っていないので、外部セッションから INSERT レコードを読み取ることはできません

外部セッションにバッチ処理の途中経過の情報を見せるために、ログテーブルへの INSERT レコードだけを独立して COMMIT させる仕組みが、**自立型トランザクション定義を使ったサブトランザクション**となります

自立型トランザクション定義を行ったプロシージャの中の COMMIT 操作は、この**プロシージャの操作だけに適用**され、本体の処理には COMMIT が適用されないような仕組みです

操作方法

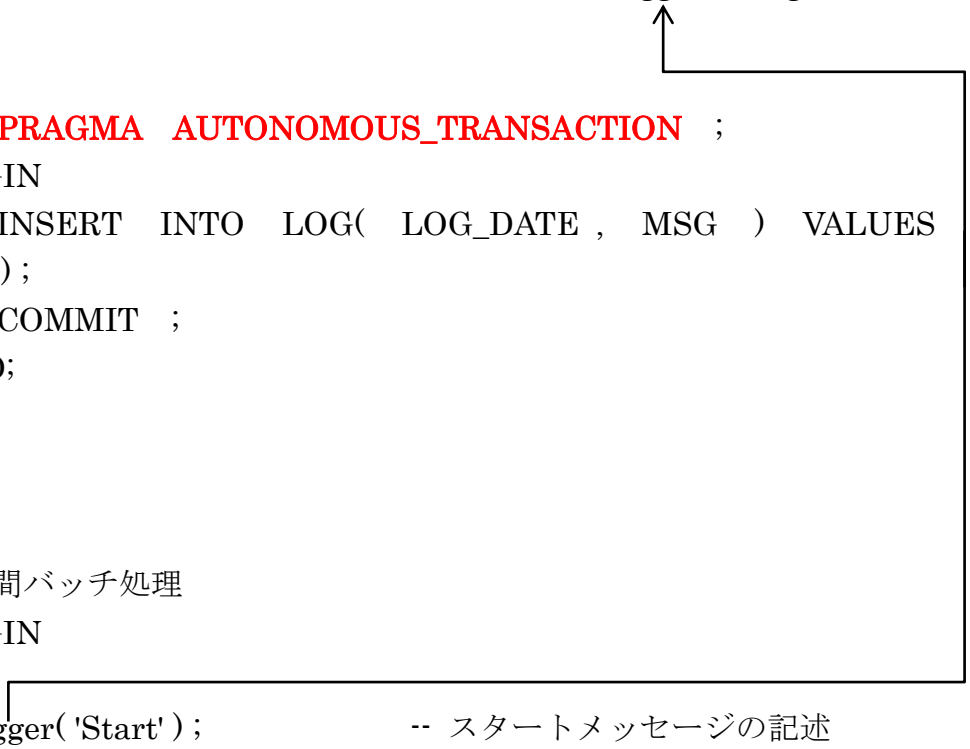
自立型トランザクションの定義

```
CREATE  OR  REPLACE  PROCEDURE  <プロシージャ名>
                                ( <変数名>  IN  ＜データ型名＞ )
IS
    -自律型トランザクション宣言
    PRAGMA  AUTONOMOUS_TRANSACTION ;
    …… -  その他のプロシージャ内の
          変数定義

BEGIN

  …… -  処理内容

  COMMIT  ;

END  ;
```

自立型トランザクションの中の処理だけに COMMIT が行われる

使用例)

```
CREATE OR REPLACE PROCEDURE logger ( i_log IN VARCHAR2 )
IS

    PRAGMA AUTONOMOUS_TRANSACTION ;
BEGIN
    INSERT INTO LOG( LOG_DATE , MSG ) VALUES ( SYSDATE , i_log ) ;
    COMMIT ;
END;
/
```



```
- 夜間バッチ処理
BEGIN
  logger( 'Start' ) ;                    -- スタートメッセージの記述
      --  ステップ 1 の処理ロジック
  logger( 'Step1  Complete' ) ;  -- ステップ 1 完了メッセージの記述
      --  ステップ 2 の処理ロジック
  logger( 'Step2  Complete' ) ;  -- ステップ 2 完了メッセージの記述
      --  ステップ 3 の処理ロジック
  logger( 'Step3  Complete' ) ;  -- ステップ 3 完了メッセージの記述
      --  ステップ 4 の処理ロジック
  logger( 'Step4  Complete' ) ;  -- ステップ 4 完了メッセージの記述
      --  ステップ 5 の処理ロジック
  logger( 'All  Step  Complete' ) ;  -- バッチ処理すべての完了メッセージの記述
  COMMIT ;
END ;
```