DBMS_PIPE パッケージ

機能

データベースのサーバー・プロセス（ユーザー・プログラム）間で、パイプを使用してメッセージ通信を行う方法

【通信開始タイミングは、パイプ書出し実行即時】

（レコード内容の変更などの操作が起こった場合に、使う）

※ 受信側プロセスは、メッセージ受取りステートメントで待機状態になる

DBMS_ALTER との違いは、COMMIT を行う前に（ 即時に ）メッセージが送られる

使い方概要

【パイプ処理の流れ】

［受信側］　　［送信側］

① メッセージ受取り待機処理

＜ステータス変数＞　:=

DBMS_PIPE.RECEIVE_MESSAGE('パイプ名' , タイムアウト秒数) ;

ステータス：０　受信成功

１　タイムアウト

② パイプで使うバッファの初期化

DBMS_PIPE.RESET_BUFFER ;

③ バッファへメッセージをセット

DBMS_PIPE.PACK_MESSAGE(

'メッセージ') ;

④ パイプにメッセージを送信

＜ステータス変数＞　:=

DBMS_PIPE.SEND_MESSAGE(

'パイプ名' , タイムアウト秒数,

最大メッセージバイト数) ;

⑤ 受取りメッセージをローカル変数にセット

DBMS_PIPE.UNPACK_MESSAGE(＜メッセージ受取り変数＞) ;

サンプル・コード

```
CREATE  OR  REPLACE  PROCEDURE  pipe_receive
IS
  msg  VARCHAR2 ( 4000 ) ;
  stat  INTEGER  :=  0 ;
  tim  number  :=  900 ;
  pipe_end  EXCEPTION ;  ←― 例外発生対応用変数の定義
  PRAGMA  EXCEPTION_INIT( pipe_end, -06556 ) ; ←―空バッファからのメッセ
                                                  ージ取出しエラー発生時
                                                  のエラーコード
BEGIN
  LOOP
    stat  :=  DBMS_PIPE.RECEIVE_MESSAGE( 'mod_pipe' ,  tim ) ; ←― ①
    DBMS_PIPE.UNPACK_MESSAGE( msg ) ;  ←― ⑤
    DBMS_OUTPUT.PUT_LINE( msg ) ;
    tim  :=  1 ;
  END  LOOP ;

EXCEPTION  ←― 例外処理ルーチン
  WHEN  pipe_end  THEN  NULL ;
END ;

/          ←―プロシージャの登録のための／（スラッシュ）


CREATE  OR  REPLACE  TRIGGER  pipe_sent
    AFTER  DELETE  ON  emp_table
DECLARE
  stat  INTEGER ;
BEGIN
  DBMS_PIPE.RESET_BUFFER ; ←― ②
  DBMS_PIPE.PACK_MESSAGE( 'emp_table 表の行が削除されました。' ) ;  ←― ③
  stat  :=  DBMS_PIPE.SEND_MESSAGE( 'mod_pipe' , 30) ; ←― ④
END;


/          ←―トリガーの登録のための／（スラッシュ）
```

【パイプ処理の**実行結果**】

［受信側］　　［送信側］

SQL> SET　SERVEROUTPUT　ON　　SQL> SET　SERVEROUTPUT ON

（メッセージ出力を、SQL*Plus 画面に表示させる）

SQL> EXECUTE　pipe_receive

　　|

　　|　　SQL> DELETE　FROM　dept

メッセージの待機中　　WHERE　deptno　=　55 ;

　　|

　　↓

「 emp_table 表の行を削除しました。 」

と、表示される←― コッミト操作前

SQL> COMMIT ;