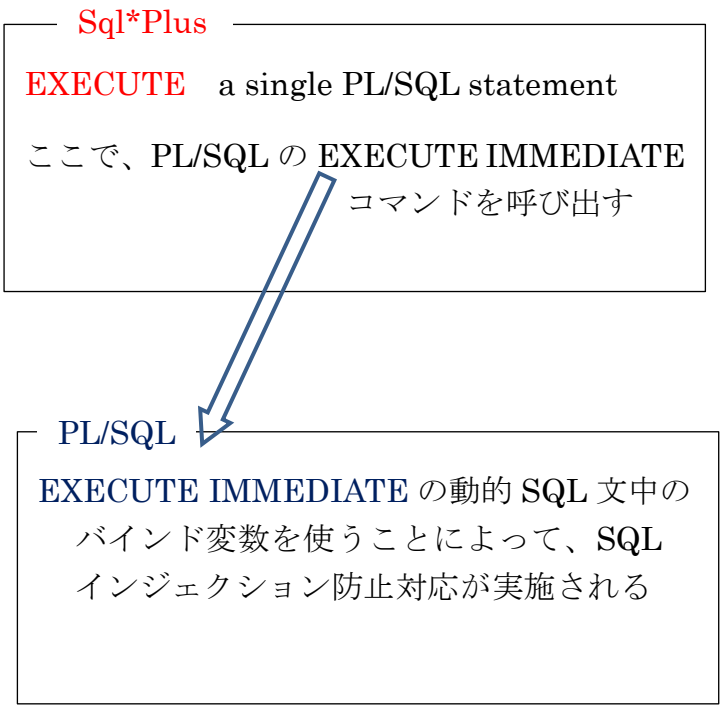
単独コマンドで、動的 SQL 文内のバインド変数を使う方法

単独コマンドでの、SQL インジェクション対応

SQL の純粋コマンド単体では、SQL インジェクション防止のための対応は実装されていない

対応方法としては、PL/SQL の EXECUTE IMMEDIATE コマンドの動的 SQL 文中のバインド変数が対応しているので、これを１つのコマンドで呼び出すようにする



他のアプリケーションでの『SQL インジェクション防止対応』

前ページの方法を使って、他のアプリケーションから SQL インジェクション防止対応を実施しようとした場合には、そのアプリケーションが PL/SQL を呼び出すためのインターフェイス・コマンドを実装していることが必須になる

これは、純粋 SQL コマンドで行っているのではなく、PL/SQL コマンドの EXECUTE IMMEDIATE コマンドの中で行っているためである

別の方法での SQL インジェクション防止対応としては、プロシージャの中で、SQL インジェクション防止対応を行うようにして、このプロシージャを呼び出せば、防止対応を実施したことになる

Sql*Plus の単独コマンドでの実行例）

```
variable   SQL_Statement        VARCHAR2 ( 300 ) ;
variable   Hikaku_Parameter     NUMBER ;
variable   Member_name           VARCHAR2 ( 50 ) ;
variable   Deptno_Value         NUMBER ;

execute   :SQL_Statement   :=   'SELECT   member ,   deptno   FROM   emp
                                          Where   empno   =   :Parameter1'   ;
execute   :hikaku_parameter   :=   1 ;
```

```
--    Sql*Plus の EXECUTE コマンドから、PL/SQL の EXECUTE   IMMEDIATE
--    コマンドの呼び出し

EXECUTE   EXECUTE   IMMEDIATE   :SQL_Statement   -
                                 INTO   :Member_name ,   :Deptno_Value      -
                                 USING   :Hikaku_Parameter ;

--    USING 句を使って動的 SQL 文内のバインド変数への値セットを行うことに
--    よって、自動的に SQL/PLUS によって SQL インジェクション防止のための
--    フィルターが実施される
```

```
print    Member_name   Deptno_Value
```

実行結果

```
MEMBER_NAME
--------------------------
  愛川こずえ

DEPTNO_VALUE
--------------------------
                    1
```