【トリガーを極める】

AFTER TRIGGER :

ON句のテーブルに対する操作の後に実行されるため、他のテーブルの更新をする場合などで使用します。例1を以下に、例2を次ページに示します。

例 1

```
CREATE TRIGGER TR1 AFTER UPDATE OF 引当済数量 ON 在庫
```

トリガー名TR1の動作時期は、在庫テーブルの引当済数量が更新された後

```
REFERENCING NEW ROW AS CHKROW
```

更新後の行をCHKROWと定義。ROWは省略可

```
FOR EACH ROW
```

1行更新するたびにTRIGGER文を実行

```
WHEN (CHKROW.実在庫数量-CHKROW.引当済数量 <= CHKROW.基準在庫数量
```

WHEREではなくWHEN

更新後の行の実在庫数量と引当済数量との差が基準在庫数量以下であれば、以下の緑色部分の処理を実行

```
BEGIN ATOMIC  
  CALL PARTSORDER (CHKROW.部品番号);  
END
```

セミコロンはBEGIN~ENDの中の文末を表す

BEGIN~ENDは、複数のSQL文を1つの単位にまとめる時やストアードプロシージャを呼び出す時に使用される。ATOMICが付くことによって、単位内の一連の処理が1つでも失敗すれば、単位内の処理は全てなかったことになる。

例 2

CREATE TRIGGER TR2 AFTER UPDATE ON 商品

トリガー名TR2の動作時期は、商品テーブルが更新された後

REFERENCING OLD AS OLD2 NEW AS NEW2

更新前の行をOLD2、更新後の行をNEW2と定義

FOR EACH ROW

1行更新するたびにTRIGGER文を実行

INSERT INTO 商品履歴

VALUES (OLD2.商品コード, ..., OLD2.更新日, OLD2.適用開始日, ADD_DAYS (NEW2.適用開始日, -1))

商品履歴テーブルに追加する行の各値を設定。ADD_DAYSはユーザ定義関数であり、追加した行の適用開始日から-1日後を返すため、適用開始前の前日となる。つまり、更新前の行の適用終了日となる。

BEFORE TRIGGER

ON句のテーブルに対する操作の前に実行されるため、操作前に同テーブル内の値のチェックしたい時などに使用します。

例 3

UPDATEとSETはペア

CREATE TRIGGER TR3 BEFORE UPDATE ON 商品

トリガー名TR3の動作時期は、商品テーブルを更新する前

REFERENCING OLD AS OLD1 NEW AS NEW1

更新前の行をOLD1、更新後の行をNEW1と定義

FOR EACH ROW

1行更新するたびにTRIGGER文を実行

SET NEW1.適用開始日=COALESCE (NEW1.適用開始日, CURRENT_DATE)

更新する行の属性 {適用開始日} の値がNULLになりそうな場合には、現在の日付をセットしてから更新する。

FOR EACH ROWとFOR EACH STATEMENT

これまでの例は全て「**FOR EACH ROW**」を使用してきましたが、デフォルトは「**FOR EACH STATEMENT**」です。例えば、以下の2つのトリガー文（トリガー名TR4とTR5）のいずれかが定義されており、「**DELETE FROM TABLE1**」を実行すると、当然、TR4又はTR5が発動します。この時、TABLE1が10行で構成されている場合には、TR4は1行削除することにより実行されるため、10回実行されます。しかし、TR5はまとめて1回だけ実行されます。

現時点において、各々のメリットとデメリットは調べていません。

```
CREATE TRIGGER TR4 AFTER DELETE ON TABLE1 . . . FOR EACH ROW . . .  
CREATE TRIGGER TR5 AFTER DELETE ON TABLE1 . . . FOR EACH STATEMENT . . .
```

注意事項

- ①AFTER TRIGGERにおいて、動作時期（実行タイミング）が「今あるテーブルからの行の削除」であり、処理内容が「その削除した行を別のテーブルに挿入」とすることもできます。（これを記載した理由は、削除した行は消えてなくなるため、挿入できるのかなと思ったためである。）
- ②TRIGGERの処理内容がTRIGGERになって、別のTRIGGERを発動することができます。