Rozdział 13
Procedury wyzwalane

procedury wyzwalane, cel stosowania, typy wyzwalaczy, wyzwalacze na poleceniach DML i DDL, wyzwalacze typu INSTEAD OF, przykłady zastosowania, zarządzanie wyzwalaczami

Procedury wyzwalane

Procedura wyzwalana (ang. *trigger*) to program w języku PL/SQL (również Java lub C) który reaguje na zdarzenia zachodzące w bazie danych i wykonuje się po zajściu określonych warunków.

Typy procedur wyzwalanych:

- **BEFORE - uruchamiane przed wykonaniem polecenia INSERT, UPDATE, DELETE**
- **AFTER - uruchamiane po wykonaniu polecenia INSERT, UPDATE, DELETE**
- **INSTEAD OF – uruchamiane zamiast polecenia INSERT, UPDATE, DELETE**
- **systemowe – uruchamiane po zajściu określonego zdarzenia w schemacie lub bazie danych**

Cele stosowania procedur wyzwalanych

- **Zaawansowane śledzenie użytkowników**
- **Ochrona przed nieprawidłowymi transakcjami**
- **Wymuszanie więzów referencyjnych (albo więzów nie wspieranych przez deklaratywne więzy integralnościowe albo więzów między węzłami rozproszonej bazy danych)**
- **Wymuszanie złożonych reguł biznesowych**
- **Wymuszanie złożonych polityk bezpieczeństwa**
- **Zapewnianie przezroczystego zapisu wydarzeń**
- **Wypełnianie atrybutów wartościami domyślnymi**
- **Modyfikacja złożonych perspektyw**
- **Śledzenie wydarzeń systemowych**

Definiowanie procedury wyzwalanej

```
CREATE [OR REPLACE] TRIGGER nazwa
{ BEFORE | AFTER | INSTEAD OF }
{ INSERT | UPDATE | DELETE } ON { tabela | perspektywa }
[ WHEN warunek ]
[ FOR EACH ROW]
[ DECLARE /* deklaracje zmiennych i kursorów */ ]
BEGIN
        /* ciało procedury wyzwalanej */
END;
```

- **INSTEAD OF: wyzwalacz może być zdefiniowany tylko na perspektywie**
- **WHEN: wyzwalacz wykonuje się tylko dla tych krotek, dla których jest spełniony warunek**
- **FOR EACH ROW: wyzwalacz wykonuje się dla każdej modyfikowanej krotki**

Definiowanie procedury wyzwalanej cd.

Dla procedur wyzwalanych uruchamianych na skutek uaktualnienia krotek, możemy określić listę atrybutów relacji, których uaktualnienie uruchomi procedurę.

```
CREATE OR REPLACE TRIGGER test
AFTER UPDATE OF placa_pod, id_zesp ON pracownicy ...
```

Ta sama procedura może być wrażliwa na kombinację instrukcji DML (tj. INSERT, UPDATE, DELETE). (*niezgodne ze standardem*)

```
CREATE OR REPLACE TRIGGER test
AFTER INSERT OR UPDATE OR DELETE ON pracownicy
BEGIN
        IF INSERTING THEN ...
        ELSIF UPDATING THEN ...
        END IF;
END;
```


Klauzula FOR EACH ROW i WHEN

```
CREATE OR REPLACE TRIGGER test
BEFORE UPDATE ON pracownicy
FOR EACH ROW WHEN (OLD.placa_dod < 100)
BEGIN
   IF (:NEW.placa_pod <= 100) THEN ... END IF;
   IF (:NEW.etat != :OLD.etat) THEN ... END IF;
END;
```

- **w klauzuli WHEN i ciele wyzwalacza FOR EACH ROW można uzyskać dostęp do starej i nowej wartości atrybutu**
- **domyślnie stara i nowa wersja rekordu są dostępne przez nazwy OLD i NEW (w ciele wyzwalacza poprzedzane dwukropkiem), można to zmienić za pomocą klauzuli REFERENCING NEW AS n**
- **dla instrukcji INSERT stara wartość jest pusta, dla instrukcji DELETE nowa wartość jest pusta**

Przykład procedury wyzwalanej (1)

Poniższa procedura wyzwalana uruchamia się przed wstawieniem nowego pracownika i nadaje mu kolejny identyfikator pobierany z licznika (sekwencji)

```
CREATE OR REPLACE TRIGGER trig_id_prac
BEFORE INSERT ON pracownicy
FOR EACH ROW
BEGIN
        IF (:NEW.id_prac IS NULL) THEN
                SELECT seq_pracownik.NEXTVAL
                INTO :NEW.id_prac FROM DUAL;
        END IF;
END;
```


Przykład procedury wyzwalanej (2)

Poniższa procedura sprawdza, czy płaca przyznana asystentowi nie przekracza widełek płacowych dla asystenta.

```
CREATE OR REPLACE TRIGGER trig_placa_asystenta
BEFORE UPDATE OF placa_pod ON pracownicy
FOR EACH ROW
WHEN ( NEW.ETAT = 'ASYSTENT' )
DECLARE
 v_max NUMBER; v_min NUMBER;
BEGIN
 SELECT placa_min, placa_max INTO v_min, v_max
 FROM etaty WHERE nazwa = :NEW.etat;
 IF :NEW.placa_pod NOT BETWEEN v_min AND v_max THEN
   RAISE_APPLICATION_ERROR(-20001,'Za wysoka placa');
 END IF;
END;
```

Przykład procedury wyzwalanej (3)

Procedura wpisuje do tabeli HISTORY datę utworzenia, typ i nazwę każdego obiektu tworzonego wewnątrz bieżącego schematu.

```
CREATE TABLE HISTORY (
  CR_DATE DATE,
  CR_OBJECT VARCHAR2(50),
  CR_NAME VARCHAR2(50) );

CREATE OR REPLACE TRIGGER TR_SCHEMA
AFTER CREATE ON SCHEMA
BEGIN
  INSERT INTO HISTORY(CR_DATE,CR_OBJECT,CR_NAME)
  VALUES (SYSDATE,ORA_DICT_OBJ_TYPE,ORA_DICT_OBJ_NAME);
END;
```


Procedura wyzwalana INSTEAD OF

Pozwala na zapewnianie modyfikowalności złożonych perspektyw.

```
CREATE OR REPLACE VIEW zesp_count AS
SELECT z.nazwa, count(*) AS pracownicy
FROM pracownicy p, zespoly z WHERE z.id_zesp = p.id_zesp
GROUP BY z.nazwa;
```

```
CREATE OR REPLACE TRIGGER trig_instead
INSTEAD OF INSERT ON zesp_count
FOR EACH ROW
BEGIN
  INSERT INTO zespoly(id_zesp,nazwa,adres)
  VALUES(80,:NEW.nazwa,NULL);
END;
```


Zarządzanie procedurami wyzwalanymi

- **Wszystkie procedury wyzwalane związane z daną relacją można zablokować (odblokować) pojedynczym poleceniem:**

```
ALTER TABLE nazwa_relacji
DISABLE [ENABLE] ALL TRIGGERS;
```

- **Każda procedura wyzwalana może być w jednym z dwóch stanów: odblokowania lub zablokowania. Do zablokowania (odblokowania) pojedynczej procedury wyzwalanej służy polecenie:**

```
ALTER TRIGGER nazwa DISABLE [ENABLE];
```

- **Do usunięcie wyzwalacza służy polecenie**

```
DROP TRIGGER nazwa;
```


Słownik bazy danych

Informacje o procedurach wyzwalanych użytkownika mieszczą się w perspektywie systemowej USER_TRIGGERS

Informacje o zależnościach można podejrzeć w perspektywie słownika bazy danych USER_DEPENDENCIES

```
SELECT TRIGGER_NAME, TRIGGER_TYPE, TRIGGERING_EVENT,
TABLE_NAME, TRIGGER_BODY
FROM USER_TRIGGERS;
```

```
SELECT NAME, TYPE, REFERENCED_TYPE
FROM USER_DEPENDENCIES
WHERE REFERENCED_NAME = 'PRACOWNICY';
```