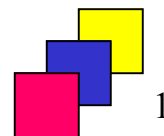


Język PL/SQL. Rozdział 6.

Procedury wyzwalane

Procedury wyzwalane, cele stosowania, typy wyzwalaczy, wyzwalacze na poleceniach DML i DDL, wyzwalacze typu INSTEAD OF, przykłady zastosowania, zarządzanie wyzwalaczami.

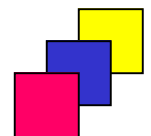


Procedury wyzwalane

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.

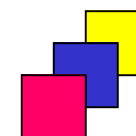
Podział:

- **Procedury wyzwalane DML:**
 - **BEFORE** - uruchamiane przed wykonaniem polecenia INSERT, UPDATE, DELETE,
 - **AFTER** - uruchamiane po wykonaniu polecenia INSERT, UPDATE, DELETE,
 - **INSTEAD OF** – uruchamiane zamiast polecenia INSERT, UPDATE, DELETE, wykonywanego na rzecz perspektywy
- **Procedury wyzwalane DDL.**
- **Procedury wyzwalane zdarzeniami systemowymi** – np. uruchomienie lub zatrzymanie bazy danych, przyłączenie się użytkownika do bazy danych, pojawianie się zdarzenia blokującego operację (brak dostępnego miejsca w przestrzeni tabel) itp.



Cele stosowania procedur wyzwalanych

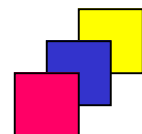
- **Zaawansowane śledzenie użytkowników**
- **Ochrona przed nieprawidłowymi transakcjami**
- **Wymuszanie więzów referencyjnych (albo więzów nie wspieranych przez deklaratywne więzy integralnościowe albo więzów między węzłami rozproszonej bazy danych)**
- **Wymuszanie złożonych reguł biznesowych**
-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



Definiowanie procedury wyzwalanej DML

```
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, tzw. wyzwalacz wierszowy



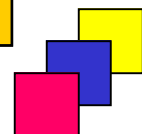
Definiowanie procedury wyzwalanej DML cd.

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*)

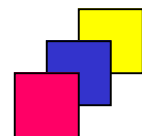
```
CREATE OR REPLACE TRIGGER test
AFTER INSERT OR UPDATE OR DELETE ON pracownicy
BEGIN
    IF INSERTING THEN ...
    ELSIF UPDATING THEN ...
    ELSIF DELETING THEN ...
    END IF;
END;
```



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;
```

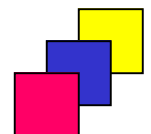
- w klauzuli **WHEN** i ciele wyzwalacza **FOR EACH ROW** można uzyskać dostęp do starej i nowej wartości atrybutu
- domyślnie stara i nowa wersja rekordu są dostępne przez pseudorekordy **OLD** i **NEW** (w ciele wyzwalacza poprzedzane dwukropkiem), ich nazwy można zmienić za pomocą klauzul **REFERENCING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



Przykład procedury wyzwalanej DML (1)

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

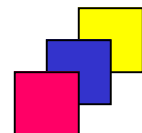
```
CREATE OR REPLACE TRIGGER trig_id_prac
BEFORE INSERT ON pracownicy
FOR EACH ROW
BEGIN
  IF (:NEW.id_prac IS NULL) THEN
    :NEW.id_prac := seq_pracownik.NEXTVAL;
  END IF;
END;
```



Przykład procedury wyzwalanej DML (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 = 'ASYSTENT';
  IF :NEW.placa_pod NOT BETWEEN v_min AND v_max THEN
    RAISE_APPLICATION_ERROR(-20001,'Za wysoka placa');
  END IF;
END;
```

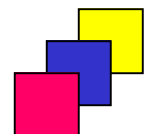


Ograniczenia wierszowych procedur wyzwalanych

- Wyzwalacz wierszowy nie może wykonywać zapytań i modyfikować relacji, na której został założony – zapobiega to odczytowi przez wyzwalacz niespójnych danych (ograniczenie to nie dotyczy wyzwalaczy INSTEAD OF)

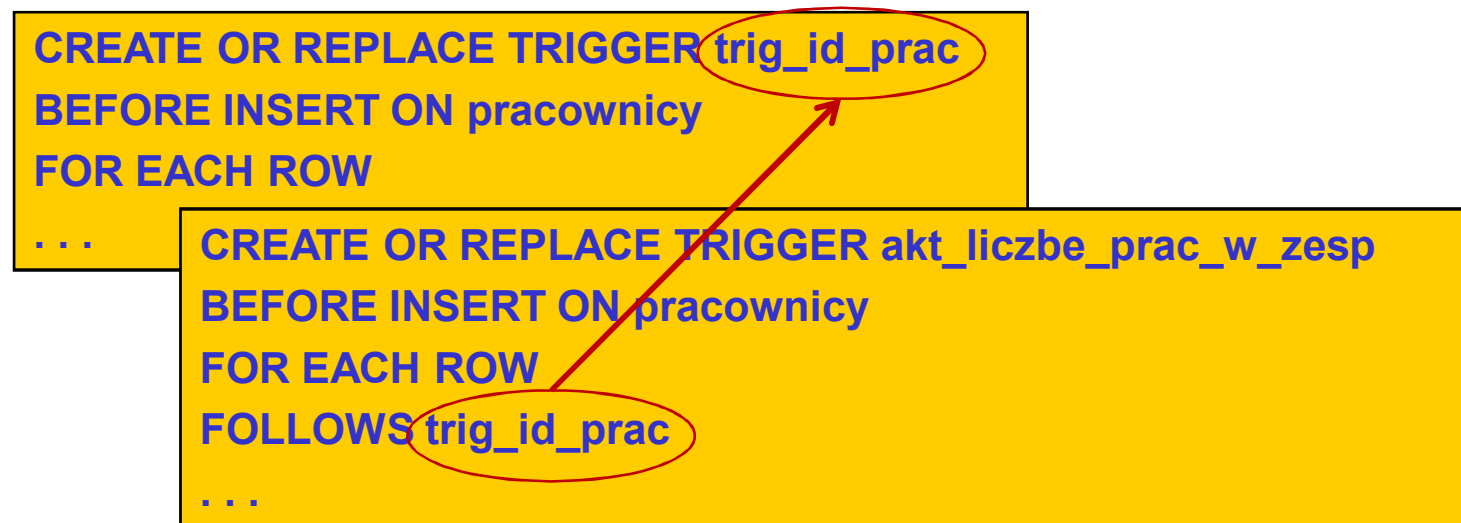
```
CREATE TRIGGER PoliczPracownikow
  AFTER DELETE ON pracownicy
  FOR EACH ROW
  DECLARE
    v_ilu NUMBER(5);
  BEGIN
    SELECT COUNT(*) INTO v_ilu FROM pracownicy;
    dbms_output.put_line('Liczba pracowników: '||v_ilu);
  END;

SQL> DELETE pracownicy WHERE nazwisko = 'HAPKE';
ORA-04091: tabela SCOTT.PRACOWNICY ulega mutacji,
wyzwalacz/funkcja może tego nie widzieć
```



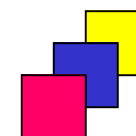
Określanie porządku uruchamiania wyzwalaczy przypisanych do tego samego zdarzenia

- W przypadku istnienia wielu wyzwalaczy opartych na tym samym zdarzeniu mogą one być uruchamiane w nieokreślonej kolejności (teoretycznie za każdym razem różnej)
- Do określenia kolejności uruchamiania wyzwalaczy służy klauzula **FOLLOWS** wskazująca wyzwalacz poprzedzający



Wyzwalacze złożone

- Zazwyczaj wyzwalacz posiada określony poziom (wierszowy lub polecenia) a także jest określonego typu (BEFORE lub AFTER).
- Ze względu na różne swoje możliwości często wiele wyzwalaczy "współpracuje" ze sobą w celu uzyskania odpowiedniego rezultatu.
- Taka "współpraca" nie zawsze jest oczywista i wymaga analizy wykonywanych przez wyzwalacze operacji.
- Zamiast tworzenia wielu wyzwalaczy współpracujących istnieje możliwość skorzystania z tzw. wyzwalaczy złożonych.
- W wyzwalaczu złożonym osobne sekcje:
 - **BEFORE STATEMENT** – odpowiednik wyzwalacza BEFORE polecenia,
 - **AFTER STATEMENT** – odpowiednik wyzwalacza AFTER polecenia,
 - **BEFORE EACH ROW** – odpowiednik wyzwalacza wierszowego BEFORE,
 - **AFTER EACH ROW** – odpowiednik wyzwalacza wierszowego AFTER.
- Nie jest konieczne użycie wszystkich sekcji.



Wyzwalacze złożone – przykłady (1)

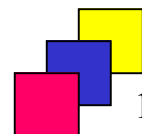
```
CREATE OR REPLACE TRIGGER obsluga_wstawiania
FOR INSERT ON pracownicy
COMPOUND TRIGGER
v_licznik NUMBER(10);

BEFORE STATEMENT IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Przed wstawieniem - raz dla polecenia');
  v_licznik := 0;
END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Przed wstawieniem - dla rekordu z id_prac = '||:new.id_prac);
END BEFORE EACH ROW;

AFTER EACH ROW IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Po wstawieniu - dla rekordu z id_prac = '||:new.id_prac);
  v_licznik := v_licznik + 1;
END AFTER EACH ROW;

AFTER STATEMENT IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Po wstawieniu - raz dla polecenia');
  DBMS_OUTPUT.PUT_LINE('Liczba wstawionych rekordów: '||v_licznik);
END AFTER STATEMENT;
END obsluga_wstawiania;
```

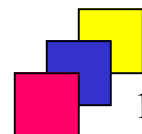


Wyzwalacze złożone – przykłady (2)

```
ALTER TABLE pracownicy ADD liczba_podwl NUMBER(2);  
UPDATE pracownicy s SET liczba_podwl = (SELECT COUNT(*) FROM pracownicy WHERE id_szeffa = s.id_prac);
```

```
CREATE OR REPLACE TRIGGER akt_liczba_podwl  
FOR INSERT OR DELETE OR UPDATE OF ID_SZEFA  
ON pracownicy COMPOUND TRIGGER  
TYPE tab_num IS TABLE OF NUMBER(2) INDEX BY PLS_INTEGER;  
v_do_akt tab_num;  
v_id_szeffa NUMBER(3);  
AFTER EACH ROW IS  
BEGIN  
  IF (DELETING or UPDATING) and :OLD.ID_SZEFA is not null THEN  
    IF v_do_akt.EXISTS(:OLD.ID_SZEFA) THEN  
      v_do_akt(:OLD.ID_SZEFA) := v_do_akt(:OLD.ID_SZEFA) - 1;  
    ELSE  
      v_do_akt(:OLD.ID_SZEFA) := -1;  
    END IF;  
  END IF;  
  IF (INSERTING or UPDATING) and :NEW.ID_SZEFA is not null THEN  
    IF v_do_akt.EXISTS(:NEW.ID_SZEFA) THEN  
      v_do_akt(:NEW.ID_SZEFA) := v_do_akt(:NEW.ID_SZEFA) + 1;  
    ELSE  
      v_do_akt(:NEW.ID_SZEFA) := 1;  
    END IF;  
  END IF;  
END AFTER EACH ROW;  
...
```

```
...  
AFTER STATEMENT IS  
BEGIN  
  v_id_szeffa := v_do_akt.FIRST;  
  WHILE (v_id_szeffa IS NOT NULL) LOOP  
    DBMS_OUTPUT.PUT_LINE(v_id_szeffa||':'||  
      v_do_akt(v_id_szeffa));  
    UPDATE pracownicy  
    SET   liczba_podwl = liczba_podwl +  
          v_do_akt(v_id_szeffa)  
    WHERE id_prac = v_id_szeffa;  
    v_id_szeffa := v_do_akt.NEXT(v_id_szeffa);  
  END LOOP;  
END AFTER STATEMENT;  
END akt_liczba_podwl;
```



Wyzwalacze złożone – przykłady (3)

```
SELECT * FROM pracownicy  
WHERE liczba_podwl > 0;
```

NAZWISKO	LICZBA_PODWL
WEGLARZ	4
BLAZEWICZ	1
SLOWINSKI	1
BRZEZINSKI	5
MORZY	2

```
update pracownicy  
SET id_szefa = 100;
```

```
14 rows updated.  
100:10  
110:-1  
120:-1  
130:-5  
140:-2
```

```
SELECT * FROM pracownicy  
WHERE liczba_podwl > 0;
```

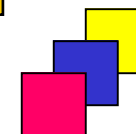
NAZWISKO	LICZBA_PODWL
WEGLARZ	14

Procedura wyzwalana INSTEAD OF

Pozwala na zapewnianie modyfikowalności złożonych perspektyw.

```
CREATE OR REPLACE VIEW zesp_count AS
SELECT nazwa, count(id_prac) AS pracownicy
FROM pracownicy RIGHT JOIN zespoly USING (id_zesp)
GROUP BY nazwa;
```

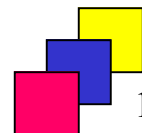
```
CREATE OR REPLACE TRIGGER trig_instead
INSTEAD OF INSERT ON zesp_count
FOR EACH ROW
BEGIN
  INSERT INTO zespoly(id_zesp,nazwa,adres)
  VALUES(seq_zesp.NEXTVAL,:NEW.nazwa,NULL);
END;
```



Przykład procedury wyzwalanej DDL

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

```
CREATE TABLE HISTORY (  
  CR_DATE DATE,  
  CR_OBJECT VARCHAR2(50),  
  CR_NAME VARCHAR2(50));  
  
CREATE OR REPLACE TRIGGER TR_CREATE  
AFTER CREATE ON scott.SCHEMA  
BEGIN  
  INSERT INTO HISTORY(CR_DATE,CR_OBJECT,CR_NAME)  
  VALUES (SYSDATE, ORA_DICT_OBJ_TYPE, ORA_DICT_OBJ_NAME);  
END;
```



Procedury wyzwalane DDL

Przykłady zdarzeń DDL

- ALTER
- ANALYZE
- AUDIT
- CREATE
- DDL
- DROP
- GRANT
- REVOKE
- RENAME

Przykłady dostępnych atrybutów

- ORA_CLIENT_IP_ADDRESS
- ORA_DES_ENCRYPTED_PASSWORD
- ORA_DICT_OBJ_NAME
- ORA_DICT_OBJ_OWNER
- ORA_DICT_OBJ_TYPE
- ORA GRANTEE
- ORA_IS_ALTER_COLUMN
- ORA_IS_DROP_COLUMN
- ORA_LOGIN_USER
- ORA_WITH_GRANT_OPTION

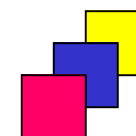
Przykład procedury wyzwalanej zd. systemowym

Procedura zabrania przyłączać się do bazy danych w weekend.

```
CREATE OR REPLACE TRIGGER TR_WEEKEND
AFTER LOGON ON DATABASE
BEGIN
  IF to_char(sysdate, 'DAY') in ('SOBOTA', 'NIEDZIELA') THEN
    raise_application_error(-20010, 'Logowanie w weekendy zabronione!');
  END IF;
END;
```

Procedury wyzwalane zd. systemowymi

- Zdarzenia systemowe mogą być definiowane na poziomie schematu (ON SCHEMA) lub bazy danych (ON DATABASE)
- Dostępne są następujące zdarzenia systemowe
 - **STARTUP (AFTER)**
 - **SHUTDOWN (BEFORE)**
 - **LOGON (AFTER)**
 - **LOGOFF (BEFORE)**
 - **DB_ROLE_CHANGE**
 - **SERVERERROR (AFTER)**
- W przypadku wyzwalacza uruchamianego po wystąpieniu (większości) błędów systemowych przydatne mogą być następujące atrybuty
 - **ORA_SERVER_ERROR** – numer błędu
 - **ORA_SERVER_ERROR_MSG** – komunikat o błędzie
 - **ORA_SERVER_ERROR_NUM_PARAMS** – liczba parametrów błędu
 - **ORA_SERVER_ERROR_PARAM** – numer błędu dla wskazanego parametru



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 usunięcia 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';
```