Wydział Informatyki
Politechnika Białostocka

Zaawansowane bazy danych i hurtownie danych

semestr I

WYKŁAD 2:
Zaawansowane obiekty baz danych: procedury, pakiety, wyzwalacze

Agnieszka Oniśko

Podprogramy

- Procedury do przeprowadzenia akcji
- Funkcje do obliczania wartości
- Pakiety do zbierania logicznie powiązanych procedur i funkcji

Składniki podprogramu

Nagłówek podprogramu

[*deklaracje*]

```
BEGIN
  [EXCEPTION
  obsługa wyjątków]
END;
```


Podprogramy: Procedura

```
[CREATE OR REPLACE]
PROCEDURE nazwa_procedury
     [ (parametr [, parametr … ] )] IS

[deklaracje]
BEGIN

     [EXCEPTIONS
     obsługa wyjątków]

END [nazwa_procedury];
```


Podprogramy: Funkcja

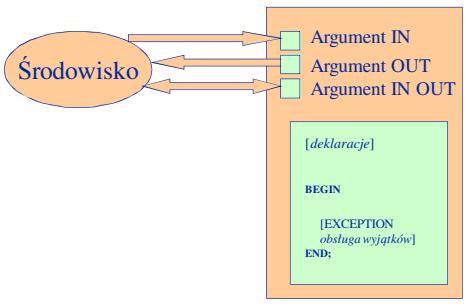
```
[CREATE OR REPLACE]
FUNCTION nazwa_funkcji
         [(parametr [, parametr … ] )]
         RETURN nazwa_typu IS

[deklaracje]
BEGIN

     [EXCEPTIONS
     obsługa wyjątków]

END [nazwa_funkcji];
```


Tryby parametrów podprogramów



Parametry w podprogramach

```
parametr
[IN | OUT [NOCOPY] | IN OUT [NOCOPY]]
     typ_danych [:= | DEFAULT wyrażenie]
```

IN	Parametry przekazywane przez wartość
OUT	Parametry przekazywane przez referencję
IN OUT	

Procedura: Przykład

```
CREATE OR REPLACE
PROCEDURE mod_cene (p_isbn books.isbn%TYPE,
                    p_wartosc NUMBER) IS

BEGIN

 UPDATE books SET retail = retail + p_wartosc
 WHERE isbn = p_isbn;

END mod_cene;
```


Funkcja: Przykład

```
CREATE OR REPLACE
FUNCTION VAT(p_isbn books.isbn%TYPE)
             RETURN NUMBER IS
v_cena books.retail%TYPE;
BEGIN
     SELECT retail INTO v_cena
     FROM books
     WHERE isbn = p_isbn;

     RETURN (v_cena*0.07);
END VAT;
```


Wywoływanie procedur/funkcji

EXECUTE nazwa_procedury[(parametry)]

EXECUTE zmienna:= nazwa_funkcji[(parametry)]

Wywoływanie procedur/funkcji: Przykład

```
EXECUTE mod_cene(1,5);
```

```
VARIABLE g_vat NUMBER
EXECUTE :g_vat := VAT(1);
PRINT g_vat
```


Użycie funkcji w instrukcji SQL

```
SELECT title, retail, VAT(1)
FROM books
WHERE isbn = 1;
```

Błędy

- Błędy bloków anonimowych pojawiają się na standardowym wyjściu
- Błędy zapamiętanych procedur są zapisywane w tabeli USER_ERRORS słownika danych

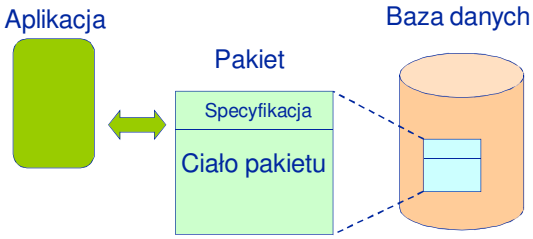
```
SHOW ERRORS
```

```
SELECT *
FROM user_errors;
```


Procedury/funkcje: Podsumowanie

- Funkcje zdefiniowane przez użytkownika mogą być stosowane w instrukcjach SQL (procedury nie!)
- Funkcja nie może zawierać instrukcji DML
- Typy danych używane w funkcji muszą być wewnętrzynymi typami serwera a nie typami danych PL/SQL

Pakiety



Specyfikacja pakietu

```
CREATE [OR REPLACE]
PACKAGE nazwa_pakietu AS
    deklaracje publiczne
    specyfikacje podprogramów
END;
```


Tworzenie ciała pakietu

```
CREATE [ OR REPLACE ]
PACKAGE BODY nazwa_pakietu AS
    deklaracje prywatne
    definicje podprogramów

[ BEGIN
instrukcje inicjalizujące pakietu]
END;
```


Pakiety: Przykład

```
CREATE OR REPLACE
PACKAGE p_zamowienia AS
PROCEDURE usun_zamowienie
(v_id orders.order#%TYPE);

PROCEDURE dodaj_pozycje
    (v_id orderitems.order#%TYPE,
    v_isbn orderitems.isbn%TYPE,
    v_ilosc orderitems.quantity%TYPE);
END;
```

Pakiety: Przykład

```
CREATE PACKAGE BODY p_zamowienia AS
PROCEDURE usun_zamowienie
(v_id orders.order#%TYPE)
IS
BEGIN
     DELETE FROM orderitems
     WHERE order# = v_id;

     DELETE FROM orders
     WHERE order# = v_id;
END;
```


Pakiety: Przykład

```
PROCEDURE dodaj_pozycje
     (v_id orderitems.order#%TYPE,
     v_isbn orderitems.isbn%TYPE,
     v_ilosc orderitems.quantity%TYPE);
     IS
     BEGIN
        INSERT INTO orderitmes
        VALUES(v_id, v_isbn, SYSDATE,
                              v_ilosc);
     END;

END; -- koniec pakietu
```


Pakiety: Wywołanie

```
EXECUTE p_zamowienia.usun_zamownienie(5);
```

```
EXECUTE p_zamowienia.dodaj_pozycje(,,,);
```


Pakiety: Zalety korzystania

- Poprawa zarządzania procedurami i funkcjami
- Poprawa bezpieczeństwa (uprawnienia do pakietu!)
- Poprawa efektywności

Pakiety wbudowane

DBMS_ALERT	DBMS_SQL
DBMS_OUTPUT	DBMS_DEBUG
DBMS_PIPE	DBMS_DDL
UTL_FILE	DBMS _RANDOM
UTL_HTTP	DBMS_SESSION

```
SET SERVEROUTPUT ON
```

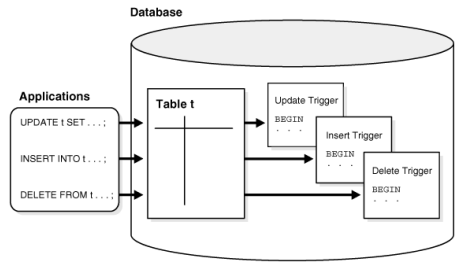

Wyzwalacze

Procedury składowane w bazie danych, automatycznie uruchamiane przez system przy wystąpieniu określonego zdarzenia w bazie danych

Wyzwalacze (liczba wykonanych akcji)

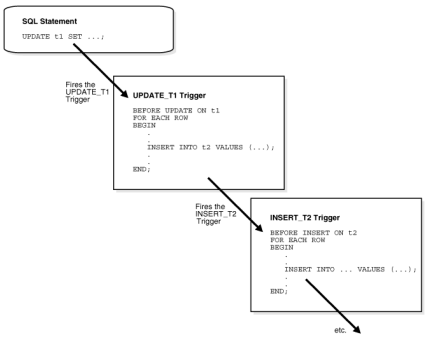
- Wyzwalacze wierszowe: akcja wykonywana jest tyle razy, ile jest wierszy, których dotyczy polecenie aktywujące wyzwalacz
- Wyzwalacze poleceniowe: akcja wykonywana jest tylko jeden raz, niezależnie od ilości wierszy, których dotyczy polecenie aktywujące wyzwalacz

Tworzenie wyzwalaczy



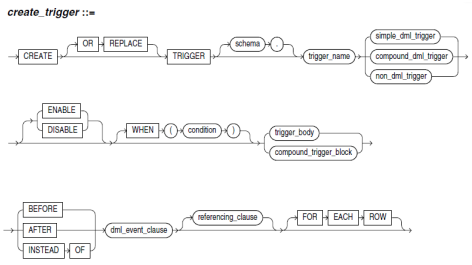
http://docs.oracle.com/cd/B19306_01/server.102/b14220/triggers.htm

Wyzwalacze wywoływane kaskadowo



http://docs.oracle.com/cd/B19306_01/server.102/b14220/triggers.htm

Tworzenie wyzwalaczy



Tworzenie wyzwalaczy

```
CREATE OR REPLACE
TRIGGER nazwa_wyzwalacza
{ BEFORE | AFTER | INSTEAD OF }
{ DELETE | INSERT | UPDATE
[ OF kolumna, [, kolumna ] … ]  }
[ OR { DELETE | INSERT | UPDATE
[ OF kolumna, [, kolumna ] … ]  } ]
ON tabela
[ FOR EACH ROW [ WHEN (warunek) ] ]

blok pl/sql
```


Rodzaje wyzwalaczy (czas wykonania)

- Wyzwalacze przeprowadzające określoną akcję przed wykonaniem polecenia, które aktywizuje wyzwalacz
- Wyzwalacze przeprowadzające akcję po wykonaniu polecenia, które aktywizuje wyzwalacz

Stare i nowe wartości wierszy

INSERT	new
UPDATE	new, old
DELETE	old

```
:identyfikator.kolumna
```


Modyfikowanie wartości

BEFORE	Można zmieniać wartości “new”
AFTER	Nie można zmieniać wartości “new”

Włączanie i wyłączanie wyzwalaczy

```
ALTER TRIGGER nazwa_wyzwalacza
{ ENABLE | DISABLE}
```

```
ALTER TRIGGER nazwa_wyzw ENABLE;
ALTER TRIGGER nazwa_wyzw DISABLE;
```


Wyzwalacze systemowe

BEFORE LOGON	AFTER LOGON
BEFORE CREATE	AFTER CREATE
BEFORE ALTER	AFTER ALTER
BEFORE DROP	AFTER DROP
BEFORE AUDIT	AFTER AUDIT
BEFORE NOAUDIT	AFTER NOAUDIT
BEFORE DDL	AFTER DDL
BEFORE GRANT	AFTER GRANT
BEFORE RENAME	AFTER RENAME
BEFORE REVOKE	AFTER REVOKE

Wyzwalacze systemowe

```
CREATE OR REPLACE
TRIGGER trig_logowanie
AFTER LOGON
ON SCHEMA
BEGIN
    INSERT INTO komunikat(str1)
    VALUES(‘zalogowano sie’ || SYSDATE);
END;
```


Wykrywanie rodzaju instrukcji DML

Polecenie-1 DML

Polecenie-2 DML

⇒ Wyzwalacz

INSERTING
UPDATING
DELETING

Wykrywanie rodzaju instrukcji DML

```
CREATE OR REPLACE
TRIGGER trig_ceny
BEFORE INSERT OR UPDATE OF
retail ON books
FOR EACH ROW
WHEN (new.retail < 100)
DECLARE
    v_akcja VARCHAR2(20);
BEGIN
IF :new.retail > :old.retail*1.1 THEN
    :new.retail := :old.retail*1.1;
END IF;
```


Przykład - cd.

```
IF INSERTING THEN
    v_akcja := 'wstawiono wiersz';
ELSE
    v_akcja := 'zaktualizowano wiersz';
END IF;
INSERT INTO komunikat
    VALUES('ksiazka', SYSDATE, v_akcja);
END;
```


Rodzaje wyzwalaczy

- Wyzwalacze reagujące
  - na instrukcje DML: `BEFORE`, `AFTER` (operują na tabelach)
  - `INSTEAD OF` (operują na perspektywach)

Sekwencje

- Obiekty bazy danych używane do generowania unikatowych wartości (zwykle na potrzeby kluczy głównych)
- Dostęp do kolejnych wartości umożliwiają pseudokolumny
  - `NEXTVAL` (zwraca wartość zwiększoną w kolejnym kroku)
  - `CURRVAL` (zwraca bieżącą wartość)

Sekwencje

```
CREATE SEQUENCE nazwa_sekwencji

[   INCREMENT BY wartość
  | START WITH   wartość
  | {MAXVALUE  wartość | NOMAXVALUE}
  | {MINVALUE   wartość  | NOMINVALUE }
  | {CYCLE | NOCYCLE }  ]
```


Sekwencje: Przykład

```
CREATE SEQUENCE sekw_2
INCREMENT BY 2
START WITH 100
MAXVALUE 120
CYCLE;
```

Sekwencje: Przykład

```
BEGIN
    FOR i IN 1..5 LOOP
        INSERT INTO komunikat(nr)
        VALUES(sekw_2.NEXTVAL);
    END LOOP;
END;
```


Sekwencje

```
ALTER SEQUENCE nazwa_sekwencji

[ INCREMENT BY wartość
| {MAXVALUE wartość | NOMAXVALUE}
| {MINVALUE wartość | NOMINVALUE}
| {CYCLE | NOCYCLE } ]
```


Sekwencje

- Mogą być używane:
  - w klauzuli `SET` polecenia `UPDATE`
  - najbardziej zewnętrznym podzapytaniu
  - w listach wartości polecenia `INSERT`
- Nie mogą być używane:
  - w podzapytaniach
  - w poleceniu `SELECT` z klauzulami `ORDER BY`, `GROUP BY`, `CONNECT BY`, `HAVING`
  - z kwantyfikatorem `DISTINCT`

Synonimy: Zastosowanie

- Synonimy umożliwiają nadawanie alternatywnych nazw dla niektórych obiektów bazy danych
- Łatwiejsze wprowadzanie poleceń poprzez skrócenie nazwy obiektów znajdujących się w innych schematach tej samej bazy danych lub innych bazach
- Odizolowanie aplikacji i użytkowników od zmian zachodzących w konstrukcji bazy danych

Synonimy

```
CREATE [PUBLIC] SYNONYM
nazwa_synonimu FOR obiekt;
```

Synonimy prywatne: tworzone dla określonej grupy użytkowników

Synonimy publiczne: tworzone dla wszystkich użytkowników

Synonimy: Przykład

```
CREATE PUBLIC SYNONYM
s_ksiazka FOR books;
```

```
RENAME stara_nazwa TO nowa_nazwa;
```