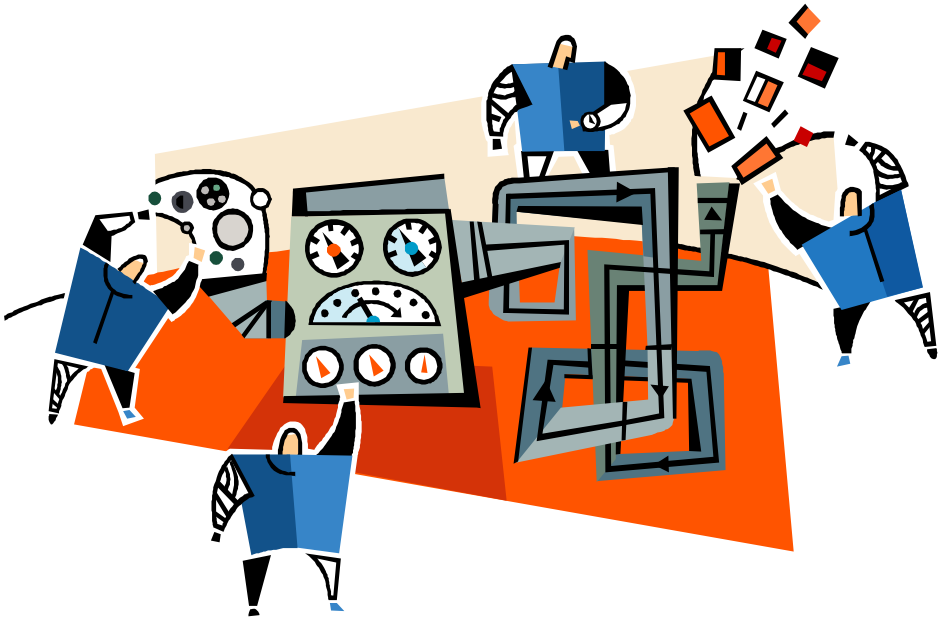
POLITECHNIKA GDAŃSKA

Automatyzacja pracy DBA w Oracle



Język PL/SQL

Kursory

Procedury składowane

Pakiety

Wyzwalacze

Oracle Scheduler

Opracowanie: Agnieszka Landowska

POLITECHNIKA GDAŃSKA

Dlaczego automatyzować?

Sprzęt jest nieokreślenie pewny.
Oprogramowanie jest określenie zawodne.
Ludzie są nieokreślenie zawodni.
Natura jest określenie pewna.
Źródło: www.linux.org

- **Niezawodność - rozwiązania automatyczne:**
  - **mogą działać w nocy, w weekend**
  - **działają niezależnie od choroby administratora**
  - **nie ma możliwości zapomnienia wykonania**
  - **nie ma możliwości popełnienia błędu przy kolejnym wykonaniu**
  - **nie stosują wyjątków (poza zaprogramowanymi)**
- **Lenistwo**
  - **„A good DBA is practically lazy”**
  - **Zasada administrowania: jeżeli coś da się zautomatyzować, należy to zautomatyzować**
  - **Zasada administrowania: automatyzacja nie powinna wpływać znacząco na obciążenie serwera i instancji (zawsze stosujemy ten mechanizm, który jest ‘najtańszy’)**
- **Typowe działania podlegające automatyzacji:**
  - **wykonywanie kopii zapasowych**
  - **monitorowanie wydajności**
  - **monitorowanie aktywności użytkowników**
  - **powiadamianie o sytuacjach nietypowych**

POLITECHNIKA GDAŃSKA

Możliwości automatyzacji Oracle

- **Procedury składowane**
  - **Język PL/SQL**
  - **Procedury i funkcje**
  - **Pakietowanie**
  - **Kursory**
  - **Składowanie procedur Java**
- **Wyzwalacze**
- **Oracle Scheduler**
  - **DBMS Jobs – wersje do 9i (prostszy)**
- **Automatyzacja oparta o poszczególne moduły Oracle**
  - **Automatic Storage Management**
  - **Flashback**
  - **Automatic Workload Repository**
  - **Automatic Undo Management**
  - **Recovery Manager**
  - **…**

POLITECHNIKA GDAŃSKA

Język PL/SQL – schemat kodu

- **Schemat kodu: PL/SQL:**

```
DECLARE
      -- blok deklaracji (opcjonalnie)
  BEGIN
      -- blok programu
  EXCEPTION
      -- obsługa wyjątków (opcjonalnie)
  END
```

- **sekcja DECLARE:**
  - **określa się typy zmiennych, stałych, kolekcji, i typów zdefiniowanych przez użytkownika**
  - **opcjonalna**
- **sekcja BEGIN i END**
  - **kod wykonywany przez procedurę, zgodny ze składnią SQL lub/i PL/SQL**
  - **obowiązkowa**
- **sekcja EXCEPTION**
  - **obsługa wyjątków**
  - **opcjonalna**

POLITECHNIKA GDAŃSKA

Procedury składowane

- **CREATE [OR REPLACE] PROCEDURE [schemat.] <procedure_name>**
  [(input/output variable declarations)]
  <IS|AS>
  [declaration block]
  BEGIN
  <PL/SQL block>
  [EXCEPTION <EXCEPTION block>]
  END;

- **OR REPLACE**
  - **Powoduje utworzenie procedury na nowo, jeśli ona już istnieje.**
- ***schemat***
  - **Schemat, w którym będzie znajdować się procedura. Jeśli argument jest pominięty, Oracle utworzy procedurę w schemacie bieżącym.**
- ***Input/output variable***
  - **deklaracja argumentów procedury. Jeżeli procedura nie posiada argumentów można pominąć nawiasy po jej nazwie.**
- **IN**
  - **Oznacza, że wartość tego argumentu jest podawana w momencie wywołania procedury**
- **OUT**
  - **Oznacza, że za pomocą tego argumentu procedura zwraca po wywołaniu wartość do środowiska, z którego została wywołana.**
- **IN OUT**
  - **Oznacza, że podanie wartości tego parametru jest wymagane do wywołania procedury, natomiast procedura zwraca za pomocą tego argumentu wartość do swojego środowiska wywołania.**
- ***typy danych w bloku deklaracji***
  - **Typ danych argumentu. Jeżeli nie poda się żadnego typu, argument może mieć dowolny typ danych dopuszczalny przez PL/SQL. Typ danych podaje się bez długości, precyzji ani skali. Poprawnym określeniem typu jest tutaj na przykład *varchar2*, niepoprawnym natomiast *varchar2*(10).**

POLITECHNIKA GDAŃSKA

Procedury składowane

- **Przykłady:**

```
CREATE OR REPLACE PROCEDURE podwyzka
        IS
        BEGIN
          UPDATE pracownicy
            SET wynagrodzenie = wynagrodzenie * 1.1;
        END;
```

```
CREATE OR REPLACE PROCEDURE podwyzka_dzial (id IN number)
       IS
       BEGIN
         UPDATE pracownicy
           SET wynagrodzenie = wynagrodzenie * 1.1
           WHERE dzial = id;
       END;
```

- **Uruchamianie:**

```
SQL> EXEC podwyzka_dzial( 3 );

podwyzka_dzial( 3 );
```

- **Usuwanie:**

```
DROP PROCEDURE procedura;
```

POLITECHNIKA GDAŃSKA

Funkcje składowane

- **Tworzenie**

```
CREATE [OR REPLACE] FUNCTION [schemat.] funkcja
       [ (argument [IN | OUT | IN OUT] typ_danych
       [, argument [IN | OUT | IN OUT] typ_danych] ... )]
       RETURN typ_danych
       IS
       {ciało_podprogramu}
```

- **Przykład**

```
CREATE OR REPLACE FUNCTION ile_pracownikow( id IN number )
       RETURN number
       IS
         ilosc number;
       BEGIN
         SELECT COUNT( * ) INTO ilosc
           FROM PRACOWNICY
           WHERE dzial = id;
         RETURN ilosc;
       END;
```

- **Wywołanie**

```
ile_pracownikow( 3 );

... WHERE ile = ile_pracownikow( 3 ) ...

SELECT ile_pracownikow( 3 ) FROM DUAL;
```

POLITECHNIKA GDAŃSKA

PL/SQL – deklaracje zmiennych i stałych

```
DECLARE
        stala    CONSTANT    number(4,2) :=17,23;
        zmienna     varchar2(20):='cos tam';
        rekord_typu       t_osoba;
        rekord_wiersz    table_name%ROWTYPE
        typ_kolumnowy  table_name.column_name%TYPE
```

- rekordy

```
TYPE t_address IS  RECORD
        (name address.name%TYPE,
         street address.street%TYPE,
        street_number address.street_number%TYPE,
         postcode address.postcode%TYPE);
```

- tabele

- kursory

POLITECHNIKA GDAŃSKA

SQL – tworzenie typów złożonych

- Create type t_adres as object
  (miasto varchar2(10),
  kod varchar2(10),
  ulica varchar2(10),
  numer varchar2(10));
- Create type t_lista_numerow as table of varchar2(15);
- Create type t_osoba as object
  (imie varchar2(20),
  nazwisko carchar2(20),
  adres t_adres,
  telefony t_lista_numerow);
- Create type lista_osob as table of t_osoba;

POLITECHNIKA GDAŃSKA

PL/SQL – instrukcje warunkowe

IF <warunek>

THEN

<instrukcje>

[ELSEIF) THEN <instrukcje>]

[ELSE <instrukcje>]

END IF;

```
IF x = 1 THEN
   sequence_of_statements_1;
ELSIF x = 2 THEN
   sequence_of_statements_2;
ELSIF x = 3 THEN
   sequence_of_statements_3;
ELSIF x = 4 THEN
   sequence_of_statements_4;
ELSIF x = 5 THEN
   sequence_of_statements_5;
ELSE
   sequence_of_statements_N;
END IF;
```

POLITECHNIKA GDAŃSKA

PL/SQL – instrukcje CASE

```
CASE
    WHEN x = 1 THEN sequence_of_statements_1;
    WHEN x = 2 THEN sequence_of_statements_2;
    WHEN x = 3 THEN sequence_of_statements_3;
    WHEN x = 4 THEN sequence_of_statements_4;
    WHEN x = 5 THEN sequence_of_statements_5;
    ELSE sequence_of_statements_N;
END CASE;
```

```
CASE x
    WHEN 1 THEN sequence_of_statements_1;
    WHEN 2 THEN sequence_of_statements_2;
    WHEN 3 THEN sequence_of_statements_3;
    WHEN 4 THEN sequence_of_statements_4;
    WHEN 5 THEN sequence_of_statements_5;
    ELSE sequence_of_statements_N;
END CASE;
```

POLITECHNIKA GDAŃSKA

PL/SQL - pętle

- **LOOP**
  <instrukcje>
  EXIT [WHEN <warunek>]
  END LOOP;

- **WHILE <warunek>**
  LOOP
  <instrukcje>
  END LOOP;

- **FOR licznik IN kres_dolny.kres_górny**
  LOOP
  <instrukcje>
  END LOOP;

```
<<parent_loop>>
LOOP
    statements

    <<child_loop>>
    LOOP
            statements
            EXIT parent_loop WHEN <condition>;
            EXIT WHEN <condition>;
    END LOOP;

    EXIT WHEN <condition>;
END LOOP parent_loop;
```

- **wyjście z pętli:**
  - **słowo EXIT**
  - **wywołanie wyjątku**
  - **instrukcja GOTO**

POLITECHNIKA GDAŃSKA

Zapytanie SQL w PL/SQL

- **Nie wypisuje wyników na ekran**
- **Wynik musi zostać przekierowany do zmiennych**

```
SELECT expr1,expr2,...
   INTO var1,var2,...
   FROM table1,table2...
      [WHERE...]
      [GROUP BY...]
      [HAVING...]
      [FOR UPDATE OF...]
```

- **instrukcja SQL musi zawsze zwrócić jeden wiersz**
  - **wyjątki: too_many_rows lub no_data_found**
- **klauzula INTO powoduje umieszczenie wyniku zapytania na zmiennych: var1,....**
- **znaczenie FOR UPDATE OF**
- **Zmienne zapytań (dotyczy ostatnio wykonywanego zapytania):**
  - **SQL%ROWCOUNT - ile wierszy zwróciło zapytanie**
  - **SQL%FOUND=TRUE - gdy przetworzono co najmniej jeden wiersz**
  - **SQL%NOTFOUND=TRUE – brak wierszy do przetworzenia**

POLITECHNIKA GDAŃSKA

PL/SQL – kursory (1)

- **Niejawne użycie kursora w pętli**

```
FOR RecordIndex IN (SELECT person_code FROM people_table)
LOOP
  DBMS_OUTPUT.PUT_LINE(RecordIndex.person_code);
END LOOP;
```

- **Jawna deklaracja / niejawne otwarcie kursora w bloku PL/SQL**

```
DECLARE
  CURSOR cursor_person IS
    SELECT person_code FROM people_table;
BEGIN
  FOR RecordIndex IN cursor_person
  LOOP
    DBMS_OUTPUT.PUT_LINE(RecordIndex.person_code);
  END LOOP;
END;
```

POLITECHNIKA GDAŃSKA

PL/SQL – kursory (2)

- **Deklaracja:**
  - CURSOR nazwa_kursora IS instrukcja_select;
- **Korzystanie:**
  - OPEN nazwa_kursora;
  - FETCH nazwa INTO zmienna,...; -- przechwyc rekord(y)
  - EXIT WHEN nazwa%NOTFOUND; -- sprawdz czy jestesmy na koncu

  } w pętli
  - CLOSE(nazwa); -- zamknij kursor
- **Atrybuty kursora:**
  - kursor%FOUND - czy sprowadzono kolejny wiersz
  - kursor%NOTFOUND - czy koniec
  - kursor%ISOPEN - czy jest otwarty
  - kursor%ROWCOUNT - liczba sprowadzonych wierszy
- **modyfikacje na wierszach (usuwanie, zmiany) – należy założyć na nie blokady**
  - SELECT … FOR UPDATE OF column, column..., (w deklaracji kursora)
  - UPDATE / DELETE … WHERE CURRENT OF kursor (wewnątrz pętli)

POLITECHNIKA GDAŃSKA

PL/SQL – kursory (3)

- **Przykład**

```
DECLARE
  Rekosoba OsobaTbl%ROWTYPE;
  CURSOR kursor IS
    SELECT name,id FROM OsobaTbl
    FOR UPDATE OF id;
BEGIN
  OPEN kursor;
  LOOP
    FETCH kursor INTO Rekosoba;
    EXIT WHEN kursor%NOTFOUND;
    IF Rekosoba.id=0
      THEN UPDATE Osoba SET id=id+1
      WHERE CURRENT OF kuror;
    END IF;
  END LOOP;
  CLOSE kursor;
END;
```

POLITECHNIKA GDAŃSKA

PL/SQL – inne instrukcje

- **instrukcja pusta**
  - **NULL**
- **etykiety i instrukcja GOTO**

```
...
GOTO gdzies;     -- skocz do instrukcji za etykieta
...
<gdzies>         -- etykieta
...
```

- **wypisywanie na ekranie:**
  - **dbms_output.put_line(txt_expr)**
  - **np. dbms_output.put_line('abc'||zmienna_txt||to_char(1023));**
- **wcześniej w SQL*Plusie ustawienie:**
  - **SET ServerOutput ON**

POLITECHNIKA GDAŃSKA

PL/SQL - wyjątki

- **Wyjątki, błędy pojawiające się podczas wykonywania kodu, występują w jednym z dwóch typów:**
  - **wyjątki pre-definiowane (wstępnie zdefiniowane wyjątki).**
  - **wyjątki zdefiniowane przez użytkownika.**
- **Wyjątki zdefiniowane przez użytkownika można uruchomić przy pomocy komendy RAISE,**
  - **składnia: RAISE <exception_name>;**
- **Oracle zdefiniowało wstępnie podstawowe wyjątki np. NO_DATA_FOUND, TOO_MANY_ROWS, itp.**
- **Każdy wyjątek posiada numer błędu (SQL Error Number) i związaną z nim treść (SQL Error Message).**
  - **zmienne systemowe SQLCODE i SQLERRM.**
- **Wywołanie własnego błędu**
  - **raise_application_error(numer_bledu,tresc_komunikatu)**
  - **nr bledu powinien być z przedziału: <-20000..-20999>**

POLITECHNIKA GDAŃSKA

Informacje o procedurach i funkcjach

- **Informacje o funkcjach i procedurach:**
  - **SELECT * FROM user_procedures;**
  - **DESCRIBE PROCEDURE name;**
  - **DESCRIBE FUNCTION name;**
- **Jeśli zmieniają sie obiekty związane z funkcją lub procedurą, to Oracle automatycznie dokonuje ich kompilacji**
  - **SELECT status FROM user_objects**

    WHERE object_name = 'nazwa_procedury‘;
  - **jeśli status jest: INVALID to procedura wymaga kompilacji**
- **Kompilacja**
  - **ALTER PROCEDURE nazwa_procedury COMPILE;**
- **Uprawnienia do użycia procedury**
  - **GRANT EXECUTE ON nazwa_procedury TO nazwa_uzytkownika;**

POLITECHNIKA GDAŃSKA

Kiedy stosować procedury składowane?

- **Jeżeli operacja może być wykonana w całości na serwerze, bez wymogu pozyskania informacji od użytkownika podczas trwania operacji.**
  - **Pozwalają serwerowi wykonywać złożone operacje na bazie danych bez angażowania oprogramowania klienckiego.**
  - **Procedury zapewniają lepszą współbieżność pomiędzy klientem a serwerem. Użytkownik na maszynie klienckiej może wykonywać procedurę, która robi coś innego, podczas gdy serwer przetwarza inną procedurę**
- **Jeśli operacja wymaga przetworzenia dużej liczby wierszy, co mogłoby być nadmiernie kosztowne w kategoriach ruchu sieciowego**
  - **Zadania te zwykle wykonują się na serwerze o wiele szybciej (zwykle najsilniejsze maszyny w sieci)**
  - **Zmniejszenie ruchu sieciowego poprzez odciążenie środowisk aplikacji i przeniesienie przetwarzania na serwer**
- **Jeśli operacja musi być wykonywana okresowo lub często.**
  - **Składowane procedury są szczególnie użyteczne do wykonywania złożonych, okresowych czynności, takich jak zamykanie miesiąca lub operacja archiwizowania**
- **Jeśli operacja jest wykonywana przez wiele różnych modułów lub procesów wewnątrz aplikacji lub przez różne aplikacje**
  - **Mogą być dzielone przez wszystkich klientów mających dostęp do bazy danych. Nie trzeba oprogramowywać tej samej logiki w każdej aplikacji klienckiej; wystarczy ją zaprogramować i przetestować tylko raz na serwerze. Jeśli logika wymiana zmiany, wtedy należy ją zmienić i przetestować tylko raz, a nie wiele razy.**
- **Jeśli logika często się zmienia.**
  - **Składowane procedury umożliwiają podział złożonych zadań na mniejsze i bardziej logiczne moduły. Składowane procedury mogą być wywoływane przez siebie nawzajem, co pozwala na budowanie zestandaryzowanych procedur, które są wywoływane na różne sposoby.**
  - **Zachowane procedury oferują także podniesienie bezpieczeństwa. Mogą wykonywać operacje na tablicach, nawet jeżeli użytkownik nie posiada wystarczających uprawnień do danej operacji na tablicy. Wystarczy, żeby procedura miała odpowiedni przywilej.**

POLITECHNIKA GDAŃSKA

Wady procedur składowanych

- **Wady procedur składowanych:**
  - **komplikacja dostępu do danych przez programistów, administratorów, czasem konieczność tworzenia różnych obejść.**
  - **problem zmian procedury na potrzeby tylko jednej z korzystających aplikacji;**
  - **obciąża bazę danych lub serwer bazy danych, a to może być wąskie gardło systemu**
- **Procedura a widok**
  - **możliwe jest utworzenie składowanej procedury, która wykonuje proste wyrażenie SELECT i zwraca zbiór wynikowy (konieczność ograniczania ilości kolumn lub wierszy, do których użytkownik może mieć dostęp)**
  - **ten sam efekt jest możliwy do uzyskania przy pomocy widoku, szczególnie jeśli operacja SELECT odbywa się na widoku, a nie na tablicy**
  - **widoki są mniej kosztowne niż składowane procedury i są lepszym rozwiązaniem dla tego typu problemów**
  - **warto pamiętać o widokach zmaterializowanych!**

POLITECHNIKA GDAŃSKA

PL/SQL – pakiety

- **Umożliwiają grupowanie kursorów, zmiennych, stałych, procedur, funkcji i wyjątków w większe jednostki.**
- **Pakiety składają sie z części publicznej (specyfikacja) i prywatnej (implementacja).**
- **Deklaracja specyfikacji:**

```
CREATE [OR REPLACE]
PACKAGE name
AS
      <deklaracje obiektów publicznych>
      <specyfikacja nagłówków funkcji/procedur>
END name;
```

```
CREATE [OR REPLACE]
PACKAGE BODY name
AS
  <Definicje ciał procedur i funkcji>;
END name;
```

POLITECHNIKA GDAŃSKA

Kiedy stosować pakiety?

- **Po co tworzyć pakiety?**
  - **prywatne metody/zmienne;**
  - **współdzielenie zmiennych między procedurami,**
  - **ładowanie całego pakietu do pamięci**
- **Kiedy nie tworzyć pakietów?**
  - **gdy korzystamy tylko z 1 elementu pakietu w jednej sesji**

- **Wywołanie:**

  EXEC nazwa_pakietu.nazwa_funkcji
- **Informacje o pakietach:**

  SELECT * FROM user_packages;

POLITECHNIKA GDAŃSKA

Predefiniowane pakiety PL/SQL

- **DBMS_OUTPUT – możliwość pisania np. na ekran**
- **DBMS_JOB /DBMS_SCHEDULER – dla tworzenia zadań wyzwalanych czasowo**
- **DBMS_XPLAN – do formatowania raportów planów zapytań (Explain Plan)**
- **DBMS_SESSION – dostęp do informacji o sesji oraz operacji na sesji, np. ALTER SESSION, SET ROLE**
- **DBMS_METADATA – do wydobywania meta danych ze słownika danych Oracle Data Dictionary (np. polecenia DDL)**
- **DBMS_EPG – zarządzanie wbudowanym serwerem sieciowym (Embedded PL/SQL Gateway)**
- **UTL_FILE – do odczytu i zapisów plików z dysku**
- **UTL_HTTP – do zapytań do serwerów sieciowych z poziomu bazy danych**
- **UTL_SMTP – do wysyłania poczty z poziomu bazy danych (poprzez serwer SMTP)**
- **…**

POLITECHNIKA GDAŃSKA

Wyzwalacze

- **Czym jest wyzwalacz?**
  - **jest specjalnym rodzajem procedury składowanej, której nie można bezpośrednio wywołać, a jest ona uruchamiana w odpowiedzi na określone zdarzenia/akcje.**
- **Tworzenie wyzwalaczy**
  - **nazwa**
  - **dla jakich elementów bazy**
  - **dla jakich zdarzeń**
  - **jaki rodzaj wyzwalacza**
  - **akcja realizowana przez wyzwalacz**

```
CREATE [OR REPLACE] TRIGGER name
  {BEFORE | AFTER | INSTEAD OF} zdarzenia ON {nazwa_tabeli/nazwa_widoku/DATABASE}
      [FOR EACH ROW]
      [REFERENCING …]
      [WHEN warunek]
          BEGIN
              akcja;
          END;
```

POLITECHNIKA GDAŃSKA

Wyzwalacze - przykład

- **Przykład**

```
CREATE OR REPLACE TRIGGER limit
        BEFORE UPDATE OR INSERT ON pracownik
        FOR EACH ROW
        WHEN (:new.pensja>15000)
        BEGIN
                Raise_application_error(-20000, 'zbyt wysoka pensja');
        END;
```

- **Uwagi:**
  - **rozmiar wyzwalacza nie może przekroczyć 32k. W przypadku większych wyzwalaczy rozwiązaniem jest przeniesienie części kodu do niezależnych procedur składowanych**
  - **nazwa wyzwalacza nie może się powtarzać (także z procedurą – wspólna przestrzeń nazw), może być jednak taka sama, jak nazwa tabeli (inna przestrzeń nazw), choć nie jest to zalecane (lepiej prefix),**
  - **ograniczenia: nie można wykonać w wyzwalaczu operacji na transakcjach commit, rollback, savepoint, ponieważ wyzwalacz jest częścią instrukcji i zalicza się do jej transakcji.**
  - **w wyzwalaczach istnieją ograniczenia w posługiwaniu się typami LONG, LONG RAW, LOB**

POLITECHNIKA GDAŃSKA

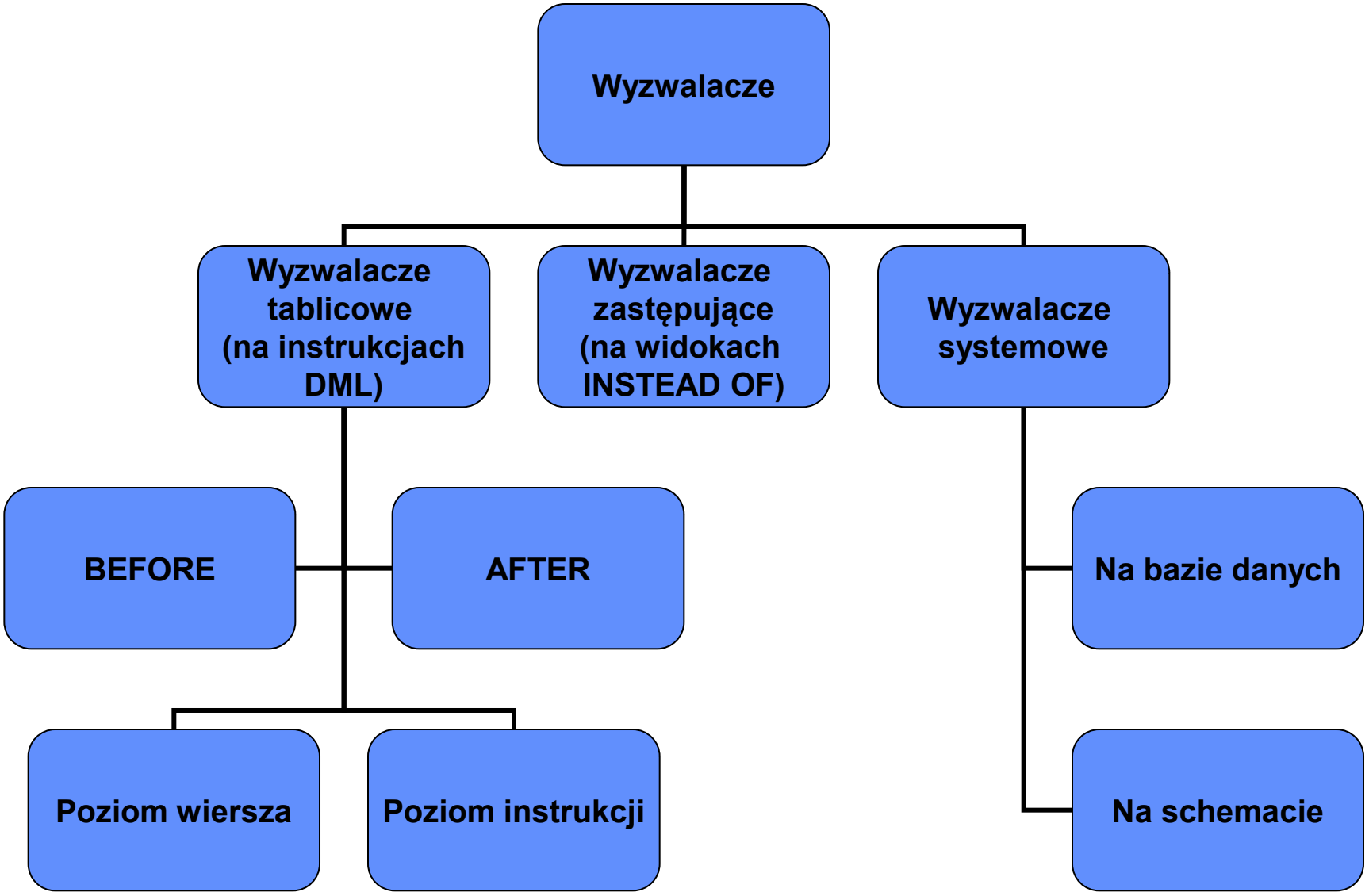
Operacje na wyzwalaczach

- **Usuwanie**
  - **DROP TRIGGER *nazwa_wyzwalacza***
- **Aktywacja/dezaktywacja**
  - **ALTER TRIGGER *nazwa_wyzwalacza* {DISABLE | ENABLE}**
  - **ALTER TABLE *nazwa_tabeli* {ENABLE | DISABLE} ALL TRIGGERS**
- **Informacje o wyzwalaczach**
  - **SELECT status FROM user_triggers;**
  - **ALL_TRIGGERS, DBA_TRIGGERS**
  - **USER_OBJECTS, ALL_OBJECTS**
  - **USER_SOURCE, ALL_SOURCE**
  - **w widokach OBJECT_TYPE jest TRIGGER**

```
select object_name, object_type, status
        from user_objects
        where object_name=nazwa;
select text
        from user_source
        where name=nazwa;
```

POLITECHNIKA GDAŃSKA

Rodzaje wyzwalaczy



POLITECHNIKA GDAŃSKA

Wyzwalacze DML – kolejność uruchamiania

- **Kolejność uruchamiania**
  - **wyzwalacz BEFORE na poziomie instrukcji**
  - **dla każdego wiersza wyzwalacz BEFORE na poziomie wiersza**
  - **dla każdego wiersza instrukcja wyzwalająca**
  - **dla każdego wiersza wyzwalacz AFTER na poziomie wiersza**
  - **wyzwalacz AFTER na poziomie instrukcji**
- **Uwagi**
  - **wyzwalacz zdefiniowany BEFORE DELETE nie wykona się, gdy będzie użyte TRUNCATE**
  - **dla tabeli można stworzyć więcej niż 12 triggerów (co to oznacza?),**
  - **usunięcie tabeli powoduje automatyczne usunięcie powiązanych z nią wyzwalaczy**

POLITECHNIKA GDAŃSKA

Wyzwalacze DML – identyfikatory korelacji i predykaty

- **Identyfikatory korelacji (tylko na poziomie wiersza):**
  - **:old – rekord wartości atrybutów danego wiersza przed wywołaniem instrukcji**
  - **:new – rekord wartości atrybutów danego wiersza po wywołaniu instrukcji**
  - **INSERT – tylko :new**
  - **DELETE – tylko :old**
  - **typ danych: tabela_wyzwalająca%ROWTYPE odwołanie kropkowe np. :old.name**
  - **w tabeli zagnieżdżonej dodatkowo :parent (wiersz tabeli nadrzędnej)**
  - **w wyzwalaczu BEFORE można zmodyfikować wartość pola w :new, zmieniona wartość zostanie wpisana do tabeli, wartości pól :old można tylko odczytywać,**
  - **REFERENCING OLD AS *nazwa_dla_old* NEW AS *nazwa_dla_new***
- **Predykaty**
  - **INSERTING/UPDATING/DELETING**
  - **CREATE TRIGGER nazwa_wyzwalacza**

```
BEFORE INSERT OR UPDATE OR DELETE
ON nazwa_tabeli
DECLARE  … BEGIN …
            IF inserting THEN …
            ELSEIF updating THEN …
END;
```

POLITECHNIKA GDAŃSKA

Wyzwalacze DML – problem tabeli mutującej i wiążącej

- **Tabela mutująca jest tabelą, która w danej chwili jest modyfikowana przez instrukcję DML, która „wywołała” wyzwalacz; przy usuwaniu kaskadowym także tabele podrzędne w chwili, gdy są z nich usuwane wartości.**
- **Tabela wiążąca jest tabelą, która może wymagać odczytu ze względu na więzy integralności referencyjnej (tabele, do których jest odwołanie w tabeli mutującej)**
- **W treści wyzwalacza na poziomie wiersza nie wolno:**
  - **Odczytywać lub modyfikować tabeli mutującej wskazanej w instrukcji wyzwalającej,**
  - **Odczytywać lub modyfikować kolumny klucza głównego, unikatowego lub obcego w tabeli wiążącej dla tabeli wyzwalającej.**
- **W treści wyzwalacza na poziomie instrukcji reguły te obowiązują tylko przy usuwaniu kaskadowym**
- **Problem: jak sprawdzić, czy po zmianie atrybutu ilość w tabeli produkty poziom zapasu nie spadł poniżej wartości krytycznej (zdefiniowanej dla kategorii)?**
  - **Nie można wykonać instrukcji select na produktach, jeżeli dla wyzwalacza jest tabelą mutującą (błąd ORA-04091)**
  - **Rozwiązanie: dwa wyzwalacze – jeden na poziomie wiersza (może zapisać wartość :new.pole do zmiennej) oraz na poziomie instrukcji (może wykonać zapytanie na tabeli mutującej) plus zmienne zdefiniowane w pakiecie (każda sesja tworzy własny egzemplarz zmiennych pakietowych – nie ma problemu nadpisywania wartości przy konfliktach);**

POLITECHNIKA GDAŃSKA

Wyzwalacze zastępujące INSTEAD OF

- wykonywane na widokach
- widok modyfikowalny – taki, na którym można wykonać instrukcję DML, widok jest modyfikowalny o ile nie zawiera:
  - Operatorów zbioru (union/minus)
  - Funkcji agregujących (sum, avg),
  - Klauzul group by
  - Operatora distinct,
  - Złączeń,
- mimo niemodyfikowalności perspektywy czasem istnieje konieczność jej modyfikacji – wtedy wykorzystuje się wyzwalacze zastępujące
- CREATE OR REPLACE VIEW zmiany_pensji

  AS SELECT nazwisko, oldpensja, newpensja

  FROM pracownik, podwyzki where pracownik.id=komu;

  INSERT INTO zmiany_pensji VALUES () → błąd
- CREATE TRIGGER insert_zmiana

  INSTEAD OF INSERT ON zmiany_pensji

  DECLARE …

  BEGIN

  …

  END;

POLITECHNIKA GDAŃSKA

Wyzwalacze systemowe

- nie są związane z konkretną tabelą/widokiem
- poziom bazy lub poziom schematu
- zdarzenia:
  - startup (after), shutdown (before) – tylko poziom bazy
- zdarzenia:
  - servererror (after), logon (after), logoff (before), create, drop, alter (before, after) – poziom bazy lub schematu
- dostępne zmienne dla wszystkich zdarzeń:
  - sysevent, instance_num, database_name, login_user
- dostępne zmienne dla servererror:
  - servererror
- dostępne zmienne dla CREATE, ALTER i DROP:
  - dictionary_obj_type, dictionary_obj_name, dictionary_obj_owner

POLITECHNIKA GDAŃSKA

Wyzwalacze systemowe - przykład

```
CREATE OR REPLACE TRIGGER nazwa_wyzwalacza
    AFTER CREATE ON DATABASE
        BEGIN
            ...
        END;

CREATE OR REPLACE TRIGGER nazwa_wyzwalacza
    AFTER ALTER ON ztbd30.schema
        BEGIN
            ...
        END;
```

POLITECHNIKA GDAŃSKA

Wyzwalacze – kiedy stosować? (1)

- **Wymuszanie reguł poprawności danych**
  - automatyczne generowanie wartości
  - wypełnienie kolumn wartościami domyślnymi (wyliczalnymi),
  - realizacja złożonej logiki biznesowej (np. do wykonywania obliczeń, które trzeba zrealizować przed wstawieniem lub aktualizacją nowego wiersza, używamy wyzwalaczy typu BEFORE)
  - np. wartości unikalne, ale dopuszczenie NULL
  - Atrybut chcemy porównać z wartością zmiennej systemowej (np. sysdate)
- **Wymuszanie zachowania więzów integralności, których nie można uzyskać w sposób deklaratywny**
  - zapewnienie niestandardowych więzów integralności
  - do sprawdzania nietypowych więzów spójności używamy wyzwalaczy typu BEFORE. W przypadku wystąpienia błędu, wykonanie odpowiedniej operacji może zostać anulowane.
  - np. wartość jakiegoś atrybutu nie może być wyższa/niższa/taka sama jak wartość w innej tabeli
- **Wymuszanie spójności przy celowej nadmiarowości bazy**
  - realizacja synchronicznej replikacji tabel,
  - gdy zmiany w jednej tabeli muszą dotknąć innych tabel (kolumn, wierszy) powiązanych
  - np. wygenerowanie dokumentu sprzedaży mogłoby automatycznie zmniejszać liczbę produktów na stanie,
  - np. gdy zmieni się ilość produktów na stanie, sprawdzany jest poziom zapasu (przewidziany dla kategorii produktów), w razie przekroczenia – dodawany jest wpis do tabeli produktów do zamówienia

POLITECHNIKA GDAŃSKA

Wyzwalacze – kiedy stosować? (2)

- **Audyt operacji DDL i DML**
  - **zapobieganie błędnym transakcjom,**
  - **prowadzenie statystyk**
- **Wprowadzanie dodatkowych mechanizmów bezpieczeństwa**
  - **wprowadzenie skomplikowanych funkcji autoryzujących**
  - **monitorowanie działań użytkowników**
  - **zapewnienie dodatkowych reguł bezpieczeństwa poprzez warstwę widoków**
- **Automatyzacja zadań administracyjnych**
  - **przypominacze,**
  - **nietypowa obsługa błędów (np. po błędnym logowaniu do bazy)**
  - **archiwizacja,**
  - **monitorowanie wydajności/niezawodności/bezpieczeństwa**

POLITECHNIKA GDAŃSKA

Wyzwalacze – kiedy NIE stosować?

- Jeżeli ten sam mechanizm (integralności, spójności, bezpieczeństwa) da się wykonać innymi metodami wbudowanymi w SQL lub/i moduły serwera Oracle
- Zasada administrowania: jeśli coś da się zapewnić za pomocą innego mechanizmu niż wyzwalacz, należy użyć tego mechanizmu np.:
  - Widoki, widoki zmaterializowane
  - Ograniczenia typu NOT NULL, CHECK,
  - Polecenie AUDIT
- Wady:
  - komplikacja dostępu do danych
  - obciążenie wydajnościowe (dla wyzwalaczy wierszowych),
  - problem dostępności (dla wyzwalaczy na całej bazie np. na logon)!

POLITECHNIKA GDAŃSKA

Wyzwalacze – problem wyboru zdarzenia

- **Przed/po każdym insert/update/delete? Dla każdego wiersza? Dla instrukcji? Na poziomie systemowym?**
- **Problem: wartość jakiegoś atrybutu nie może być wyższa/niższa/taka sama jak wartość w innej tabeli**
- **Problem: gdy zmieni się ilość produktów na stanie, sprawdzany jest poziom zapasu (przewidziany dla kategorii produktów), w razie przekroczenia – dodawany jest wpis do tabeli produktów do zamówienia**
- **Problem: mamy tabelę z kwotami sprzedaży dla poszczególnych kategorii produktów, kiedy ją uaktualniać?**
- **Problem: chcemy usuwać przeterminowane rezerwacje na filmy w bazie wypożyczalni video – kiedy?**

POLITECHNIKA GDAŃSKA

Oracle Scheduler

- **Harmonogramowanie i automatyczne uruchamianie zadań**
  - **administracyjnych**
  - **wynikających z logiki baz danych**
- **Ustawienie wykonania zadania:**
  - **na podstawie czasu – Oracle Scheduler**
  - **na podstawie zdarzenia – wyzwalacze**
- **Obsługa:**
  - **Element Oracle Enterprise Managera (uruchamianego w przeglądarce)**
  - **Pakiet procedur DBMS_Scheduler**
- **Trzy podstawowe komponenty:**
  - **Schedule – ustawienie kiedy i jak często zadanie powinno być wykonywane (harmonogram)**
  - **Program – ustawienie zadania do wykonywania (procedury, kodu Java, itp.)**
  - **Job – uruchomienie zadania (Program) z wybranym harmonogramem (Schedule)**

POLITECHNIKA GDAŃSKA

Harmonogram (Schedule)

- **Kiedy i jak często zadanie ma być wykonywane**
  - **Jednorazowo lub okresowo**
  - **Częstość powtarzania**
  - **Zakres dat obowiązywania**
- **Jeden harmonogram może być wykorzystywany do wielu zadań**

```
DBMS_SCHEDULER.CREATE_SCHEDULE (
   schedule_name           IN VARCHAR2,
   start_date              IN TIMESTAMP WITH TIMEZONE,
   repeat_interval         IN VARCHAR2,
   end_date                IN TIMESTAMP WITH TIMEZONE,
   comments                IN VARCHAR2);
```

- **Atrybuty:**
  - **schedule_name** – unikalny identyfikator dla danego Schedule
  - **start_date** – oznacza datę wystąpienia zadania (w przypadku zadań powtarzających się, oznacza datę pierwszego wystąpienia zadania)
  - **repeat_interval** – określa, jak często zadanie jest powtarzane
  - **end_date** – data, po której zadanie nie będzie wywoływane
  - **comments** – komentarz

Create Schedule

Show SQL | Cancel | OK

* Name: Myschedule

* Owner: HR

Description:

Schedule

Time Zone: GMT -7:00

Start

- (•) Immediately
- ( ) Later

Date: Aug-02-2003 (example: Dec-12-2002)

Time: 3 : 20 : 00 ( ) AM (•) PM

Repeat

Frequency: 1 Days

- Minutes
- Hours
- **Days**
- Weeks
- Months
- Years

Repeat Until

- (•) Indefinite
- ( ) Custom

Date: Aug-02-2003 (example: Dec-12-2002)

Time: 3 : 30 : 00 ( ) AM (•) PM

(Ignored except when repeating by minutes or hours.)

POLITECHNIKA GDAŃSKA

Zadanie (Program)

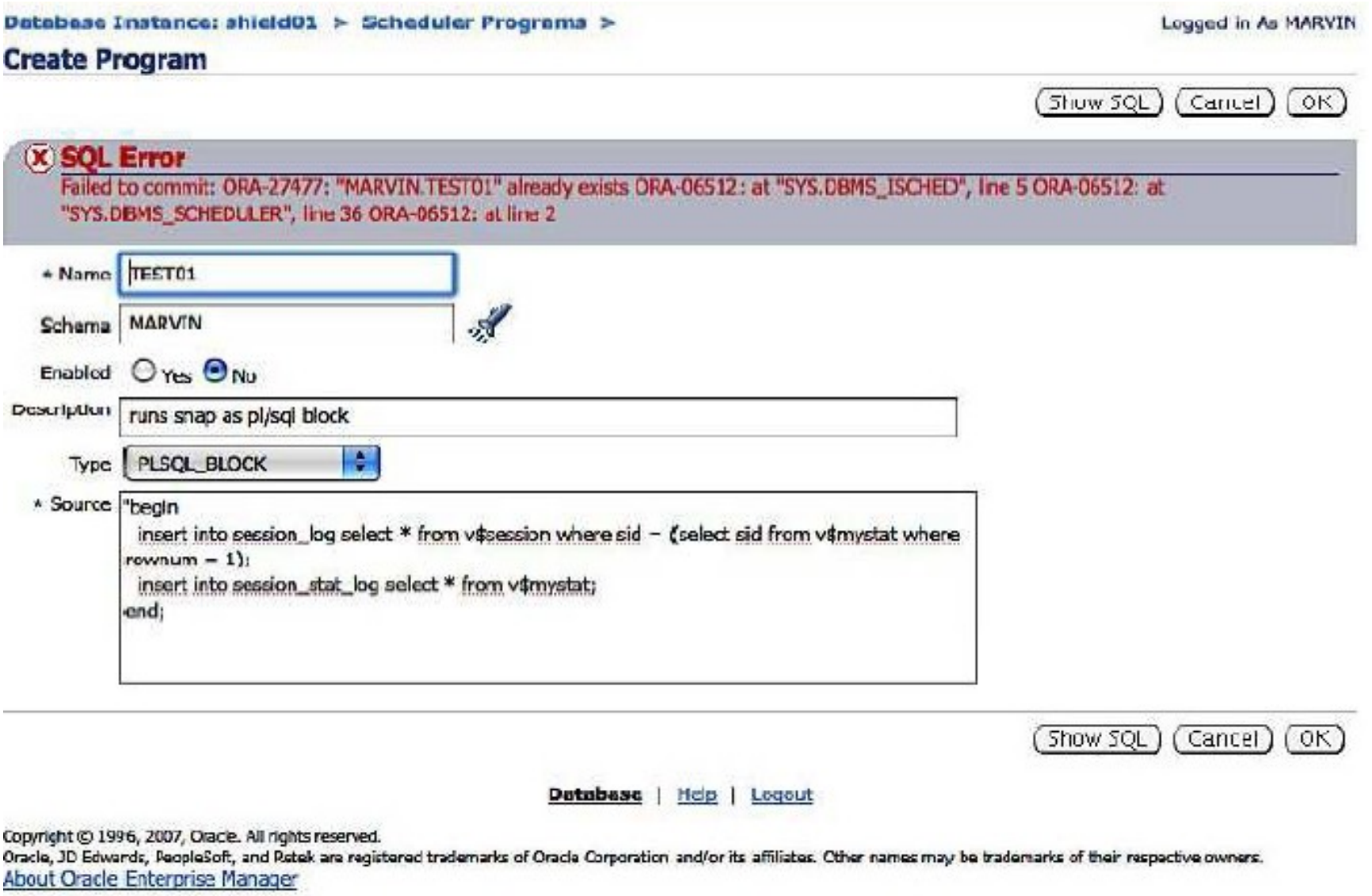
- **co (jaki skrypt) będzie przedmiotem zadania**
- **nazwy Schedule, Program i Job – nie mogą się powtarzać (wspólna przestrzeń nazw)**

```
DBMS_SCHEDULER.CREATE_PROGRAM (
   program_name              IN VARCHAR2,
   program_type              IN VARCHAR2,
   program_action            IN VARCHAR2,
   number_of_arguments       IN PLS_INTEGER,
   enabled                   IN BOOLEAN,
   comments                  IN VARCHAR2);
```

- **Atrybuty:**
  - **program_name** – unikatowy identyfikator programu
  - **program_type –** typ programu
  - **program_action –** określa nazwę procedury, nazwę pliku do wykonania lub kod w postaci bloku PL/SQL
  - **number_of_arguments –** liczba przekazywanych argumentów (domyślnie 0)
  - **Enabled** – określa, czy program ma być stworzony w stanie ENABLED czy nie (domyślnie FALSE)
  - **comments** – komentarz

POLITECHNIKA GDAŃSKA

Tworzenie Programu



POLITECHNIKA GDAŃSKA

Zadanie (Job) (1)

- określa, co musi być wykonane i kiedy
- typ zadania (atrybut job_type): program w C, funkcja w PL/SQL, program w Java lub skrypt shellowy.
- Job = Program i Schedule
- tworzenie zadania za pomocą wcześniej utworzonych obiektów Schedule i Program (jeśli te obiekty nie były utworzone, to należy zamienić atrybuty program_name i Schedule_name na atrybuty z których te obiekty się tworzy):

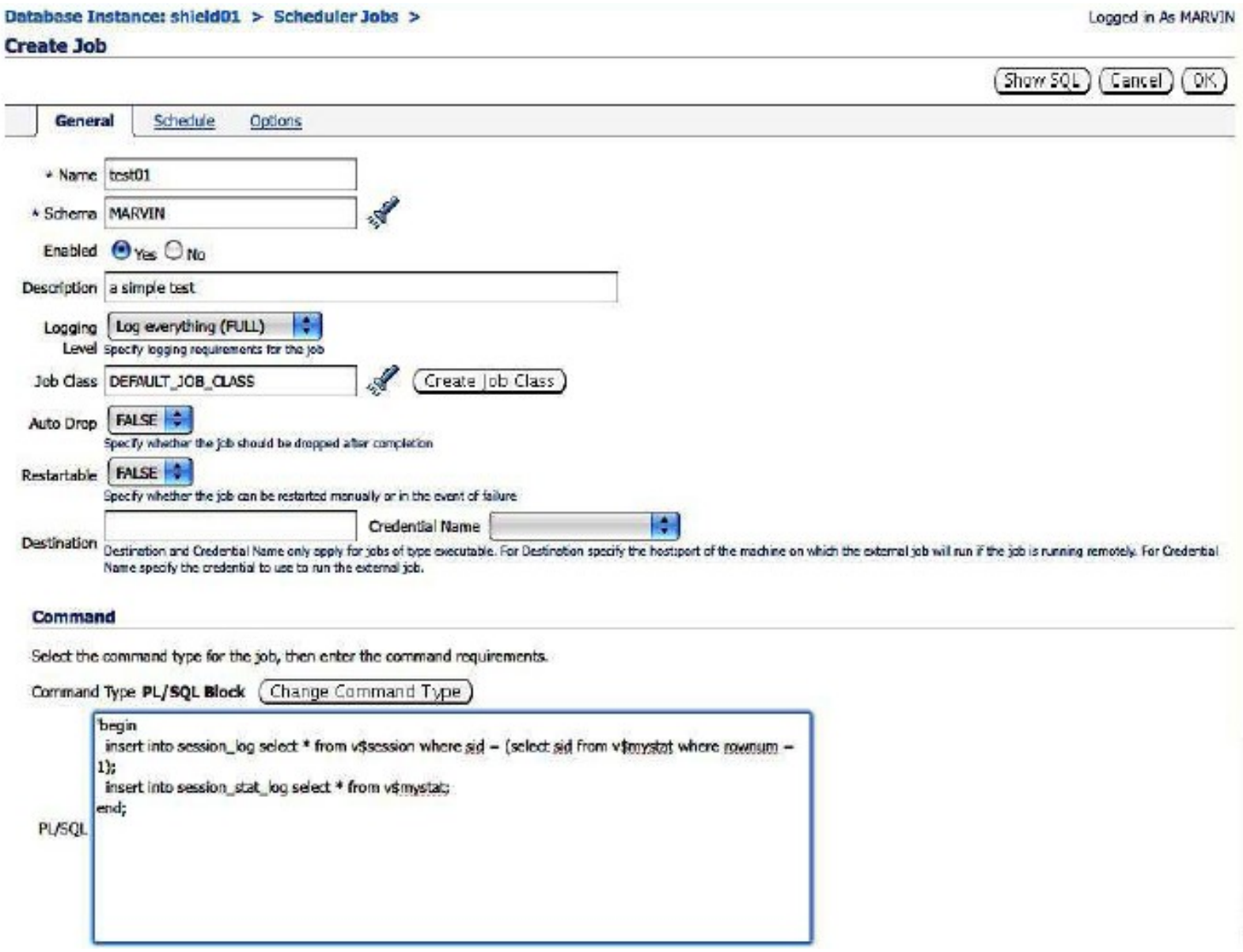
```
DBMS_SCHEDULER.CREATE_JOB (
   job_name                IN VARCHAR2,
   program_name            IN VARCHAR2,
   schedule_name           IN VARCHAR2,
   job_class               IN VARCHAR2,
   enabled                 IN BOOLEAN,
   auto_drop               IN BOOLEAN,
   comments                IN VARCHAR2);
```

POLITECHNIKA GDAŃSKA

Zadanie (Job) (2)

- Atrybuty:
  - **job_name** – unikatowy identyfikator zadania
  - **program_name** – nazwa obiektu Program, z którego składa się zadanie
  - **schedule_name** – nazwa obiektu Schedule, z którego składa się zadanie
  - **job_class** – klasa zadań (domyślnie 'DEFAULT_JOB_CLASS')
  - **enabled** – określa, czy po stworzeniu zadania jest ono w stanie ENABLED (domyślnie FALSE)
  - **auto_drop** – jeśli znacznik ustawiony na TRUE, oznacza automatyczne usunięcie zadania po jego zakończeniu (domyślnie TRUE)
  - **comments** – komentarz
- tworzenie zadania jeśli nie było Schedule i Program:

```
DBMS_SCHEDULER.CREATE_JOB (
   job_name                IN VARCHAR2,
   job_type                IN VARCHAR2,
   job_action              IN VARCHAR2,
   number_of_arguments     IN PLS_INTEGER,
   start_date              IN TIMESTAMP WITH TIME ZONE,
   repeat_interval         IN VARCHAR2,
   end_date                IN TIMESTAMP WITH TIME ZONE,
   job_class               IN VARCHAR2,
   enabled                 IN BOOLEAN,
   auto_drop               IN BOOLEAN,
   comments                IN VARCHAR2);
```


Database Instance: shield01 > Scheduler Jobs >
Logged in As MARVIN
Create Job
Show SQL
Cancel
OK
General
Schedule
Options
* Name test01
* Schema MARVIN
Enabled Yes No
Description a simple test
Logging Level Log everything (FULL)
Specify logging requirements for the job
Job Class DEFAULT_JOB_CLASS
Create Job Class
Auto Drop FALSE
Specify whether the job should be dropped after completion
Restartable FALSE
Specify whether the job can be restarted manually or in the event of failure
Destination
Credential Name
Destination and Credential Name only apply for jobs of type executable. For Destination specify the host:port of the machine on which the external job will run if the job is running remotely. For Credential Name specify the credential to use to run the external job.
Command
Select the command type for the job, then enter the command requirements.
Command Type PL/SQL Block
Change Command Type
PL/SQL
begin
insert into session_log select * from v$session where sid = (select sid from v$mystat where rownum = 1);
insert into session_stat_log select * from v$mystat;
end;

POLITECHNIKA GDAŃSKA

Schedule i Program - przykład

```
begin
  DBMS_SCHEDULER.CREATE_SCHEDULE(
  schedule_name => 'INTERVAL_DAILY_2200',
  start_date=> trunc(sysdate)+18/24,
       -- start today 18:00 (06:00 p.m.)
  repeat_interval=> 'FREQ=DAILY; BYDAY=MON,TUE,WED,THU,FRI,SAT,SUN; BYHOUR=22;',
       -- daily from Monday to Sunday at 22:00 (10:00 p.m.)
  comments=>'Runtime: Every day (Mon-Sun) at 22:00 o'clock');
end;
```

```
begin
  DBMS_SCHEDULER.CREATE_PROGRAM(
    program_name=> 'PROG_COLLECT_SESS_DATA',
    program_type=> 'STORED_PROCEDURE',
    program_action=> 'pkg_collect_data.prc_session_data',
      -- Call a procedure of a database package
    enabled=>true,
    comments=>'Procedure to collect session information');
end;
```

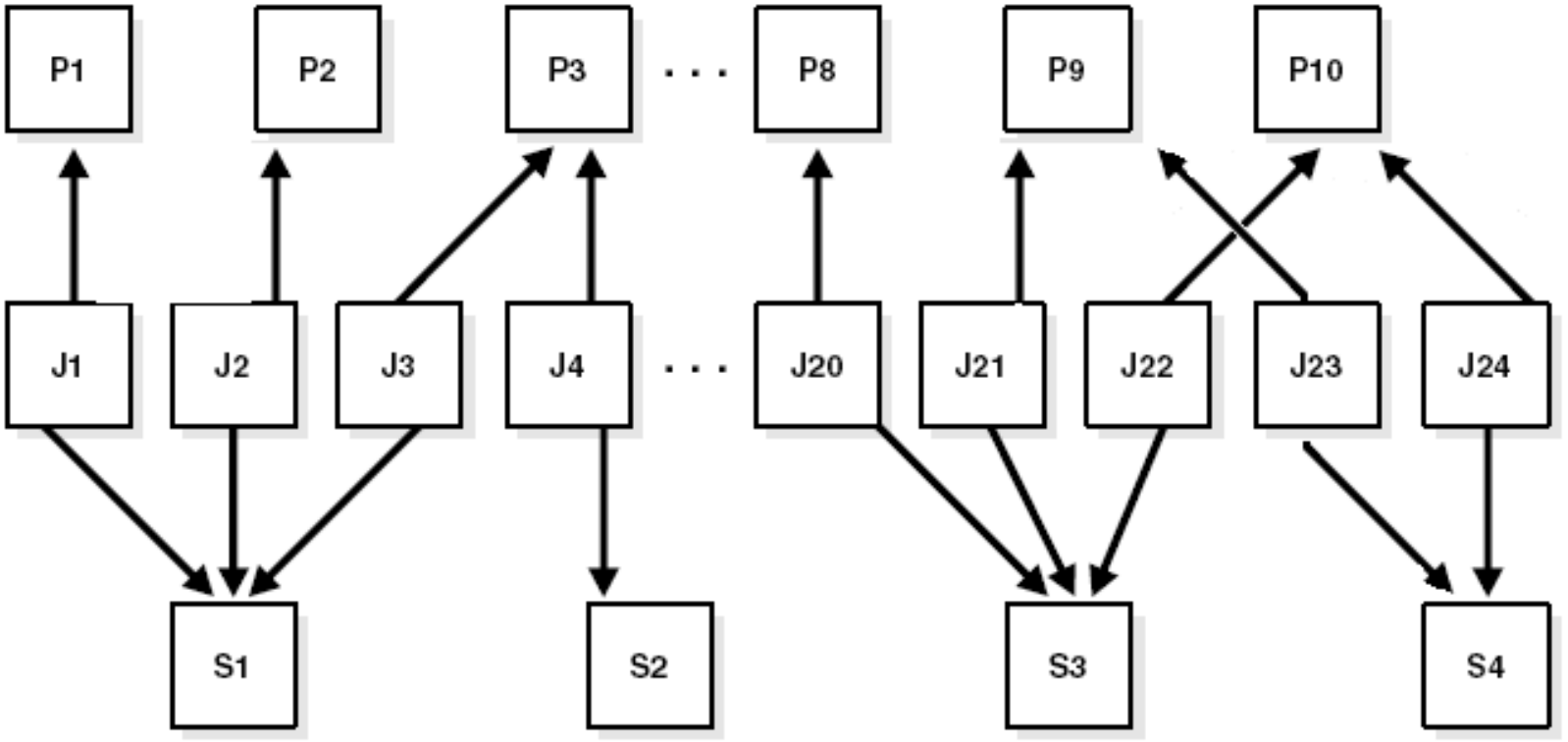
POLITECHNIKA GDAŃSKA

Job - przykład

```
begin
  -- Connect both dbms scheduler parts by creating the final job
  DBMS_SCHEDULER.CREATE_JOB(
    job_name => 'JOB_COLLECT_SESS_DATA',
    program_name=> 'PROG_COLLECT_SESS_DATA',
    schedule_name=>'INTERVAL_EVERY_5_MINUTES',
    enabled=>true,
    auto_drop=>false,
    comments=>'Job to collect data about session values every 5 minutes');
end;
```

POLITECHNIKA GDAŃSKA

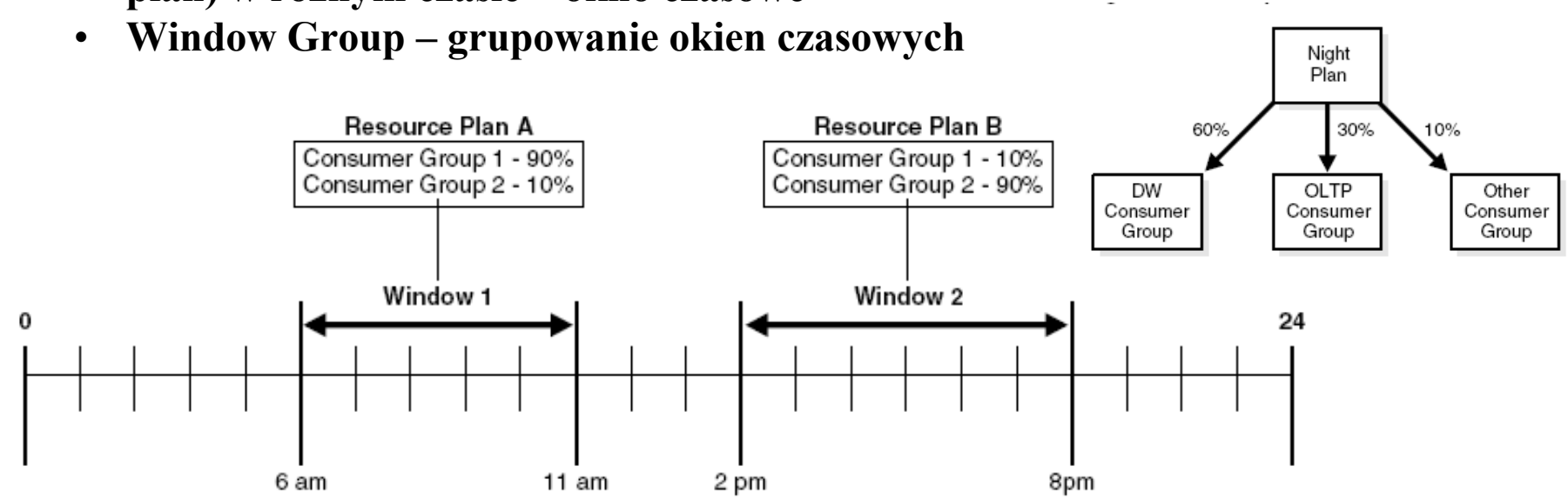
Zależności – Program, Job, Schedule



- **Przykładowo:**
  - **S1 – codziennie przed rozpoczęciem dnia pracy np. o 5:00**
  - **S2 – codziennie po zakończeniu dnia pracy np. o 21:00**
  - **S3 – co tydzień w piątek o 22:00**
  - **S4 – co miesiąc, ostatniego dnia miesiąca o 22:00**

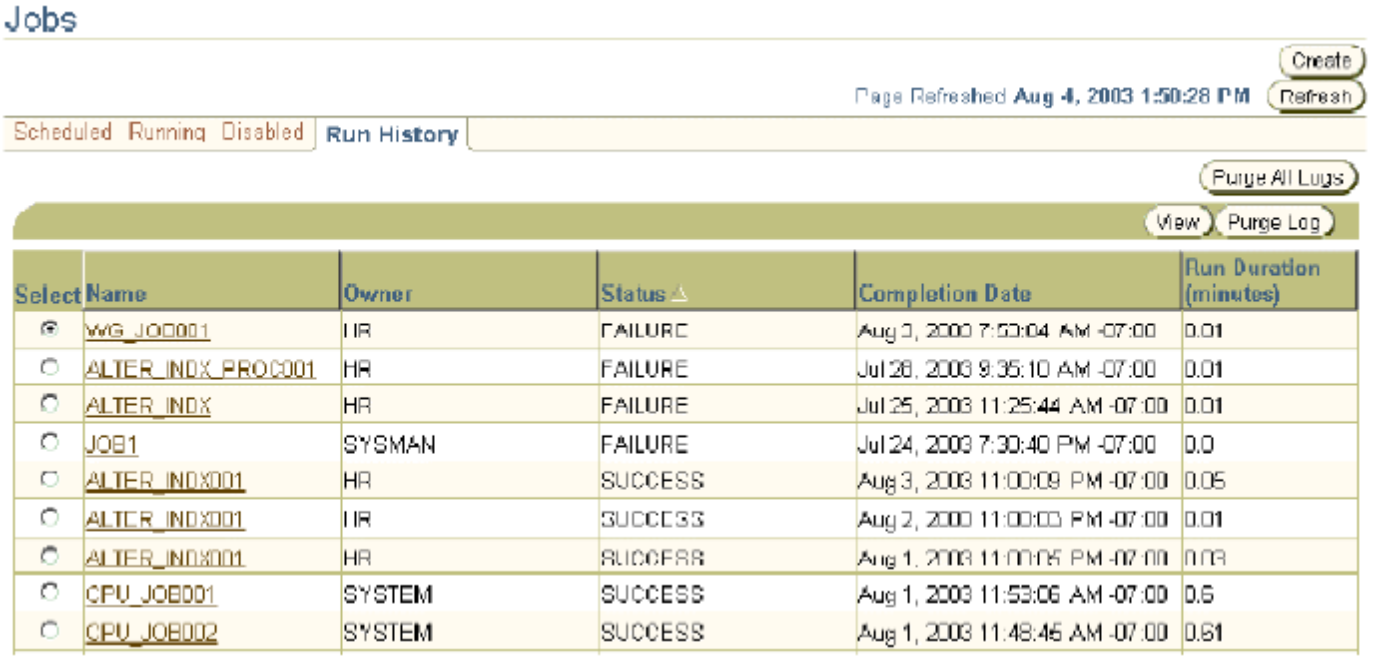
Inne możliwości Oracle Scheduler

- Uprawnienia – Schedule, Program i Job to zwykłe elementy bazodanowe
- Chains – zbiory zadań do wykonania w pewnej sekwencji, warunkowo lub alternatywnie
- Job classes – grupowanie zadań ze względu na podobne właściwości (priorytety, uprawnienia)
  - DEFAULT_JOB_CLASS – uprawnienia EXECUTE to PUBLIC
  - Logging_level
  - Log_history
  - Przydzielanie odpowiednich zasobów do każdej z klas (za pomocą modułu Resource Manager) – krytyczne funkcje mogą mieć priorytety w dostępie do zasobów
- Windows – automatyczne aktywowanie różnych planów zasobów (Resource plan) w różnym czasie – okno czasowe
- Window Group – grupowanie okien czasowych



Monitorowanie wykonania zadań

- SELECT JOB_NAME, STATUS, ERROR#
  FROM DBA_SCHEDULER_JOB_RUN_DETAILS
  WHERE JOB_NAME = 'MY_JOB1';
- Przydatne widoki:
  - *_SCHEDULER_JOBS, *_SCHEDULER_SCHEDULES, *_SCHEDULER_PROGRAMS, *_SCHEDULER_RUNNING_JOBS, *_SCHEDULER_JOB_LOG, *_SCHEDULER_JOB_RUN_DETAILS



Select	Name	Owner	Status	Completion Date	Run Duration (minutes)
◉	WG_JOB001	HR	FAILURE	Aug 3, 2003 7:53:04 AM -07:00	0.01
○	ALTER_INDX_PROC001	HR	FAILURE	Jul 28, 2003 9:35:10 AM -07:00	0.01
○	ALTER_INDX	HR	FAILURE	Jul 25, 2003 11:25:44 AM -07:00	0.01
○	JOB1	SYSMAN	FAILURE	Jul 24, 2003 7:30:40 PM -07:00	0.0
○	ALTER_INDX001	HR	SUCCESS	Aug 3, 2003 11:00:09 PM -07:00	0.05
○	ALTER_INDX001	HR	SUCCESS	Aug 2, 2003 11:00:03 PM -07:00	0.01
○	ALTER_INDX001	HR	SUCCESS	Aug 1, 2003 11:00:05 PM -07:00	0.03
○	CPU_JOB001	SYSTEM	SUCCESS	Aug 1, 2003 11:53:06 AM -07:00	0.6
○	CPU_JOB002	SYSTEM	SUCCESS	Aug 1, 2003 11:48:46 AM -07:00	0.61

POLITECHNIKA GDAŃSKA

Job logs

```
SELECT JOB_NAME, OPERATION, OWNER FROM DBA_SCHEDULER_JOB_LOG;

JOB_NAME        OPERATION     OWNER
--------        ---------     -----
MY_JOB13        CREATE        SYS
MY_JOB14        CREATE        OE
MY_NEW_JOB3     ENABLE        SYS
MY_EMP_JOB1     UPDATE        SYS
MY_JOB1         CREATE        SCOTT
MY_EMP_JOB1     UPDATE        SYS
MY_EMP_JOB      CREATE        SYS
MY_JOB14        RUN           OE
MY_JOB14        RETRY_RUN     OE
MY_JOB14        RETRY_RUN     OE
MY_JOB14        RETRY_RUN     OE
MY_JOB14        RETRY_RUN     OE
MY_JOB14        BROKEN        OE
MY_JOB14        DROP          OE
```

- Parametr

  logging_level = LOGGING_FULL
- additional_info column zawiera wartości zmodyfikowanego atrybutu (przed i po), wartości wszystkich atrybutów (dla drop)

- Więcej o Oracle Scheduler:
  - Oracle Administrator Guide,
  - rozdziały 26 i 27