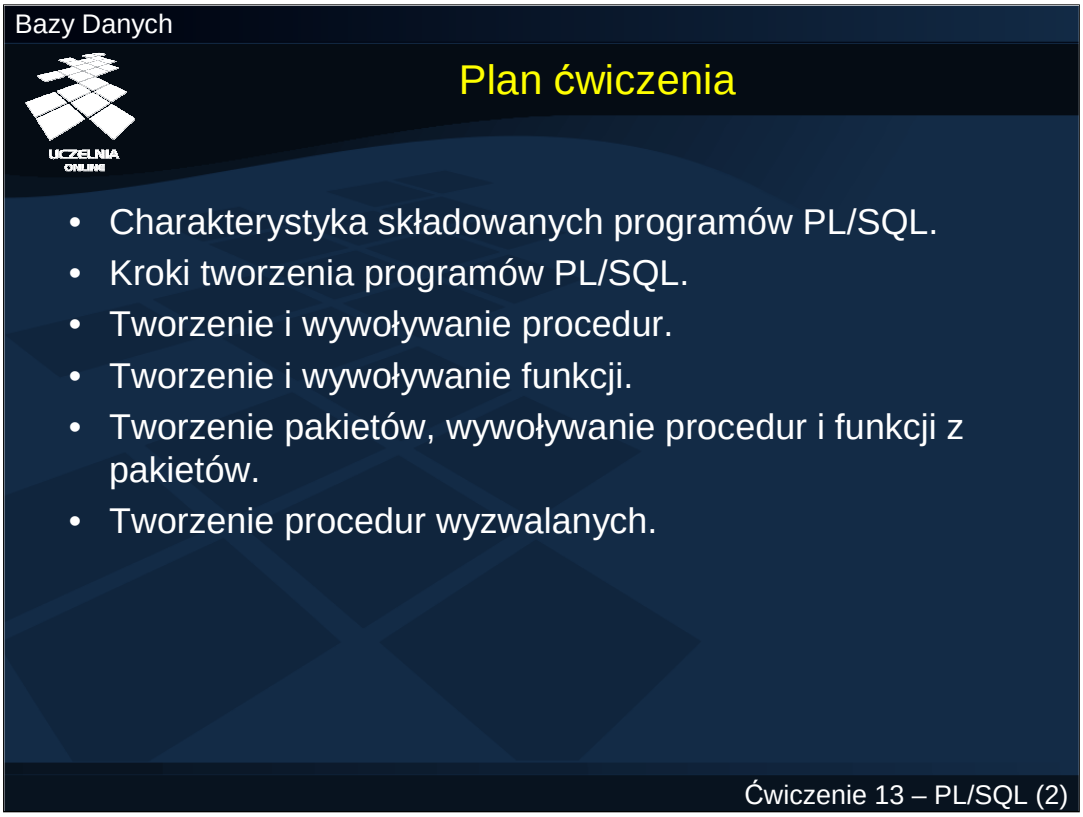
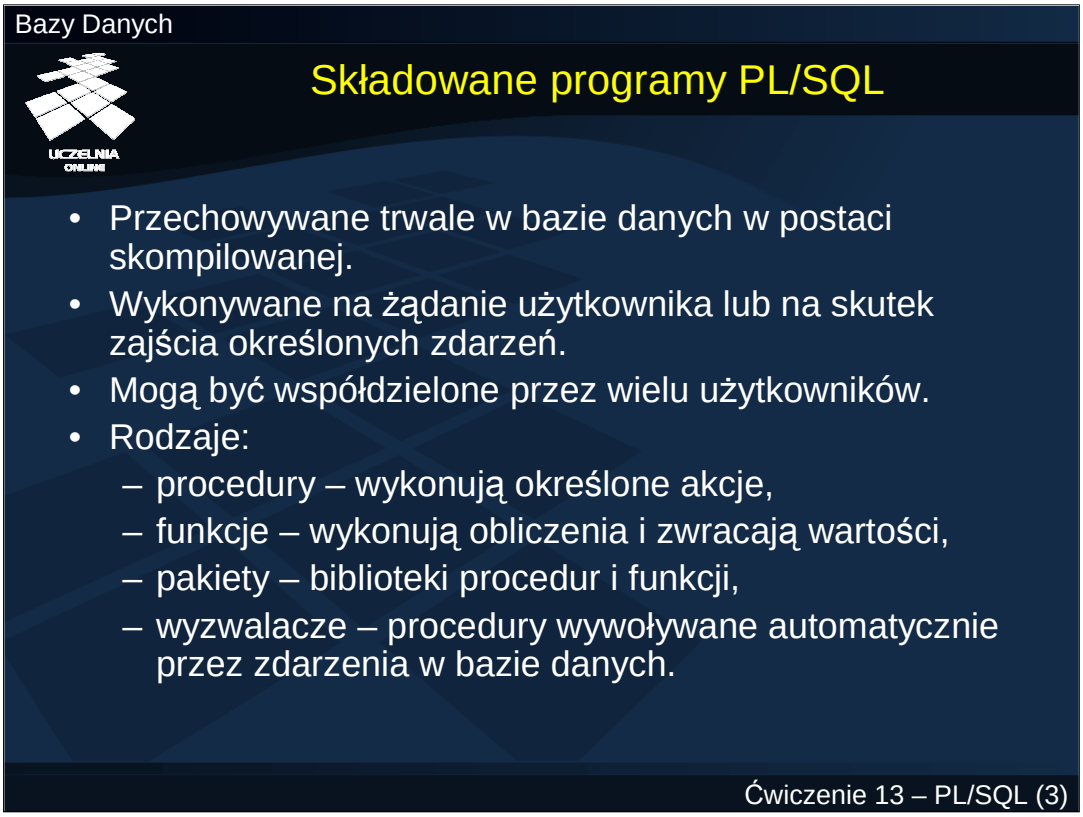
Niniejsze ćwiczenie zaprezentuje składowane w bazie danych programy PL/SQL: procedury, funkcje, pakiety oraz procedury wyzwalane.

Wymagania:

Umiejętność tworzenia zapytań w języku SQL, znajomość operacji z grup DML i DDL, umiejętność wykorzystania zaawansowanych konstrukcji programowych PL/SQL.



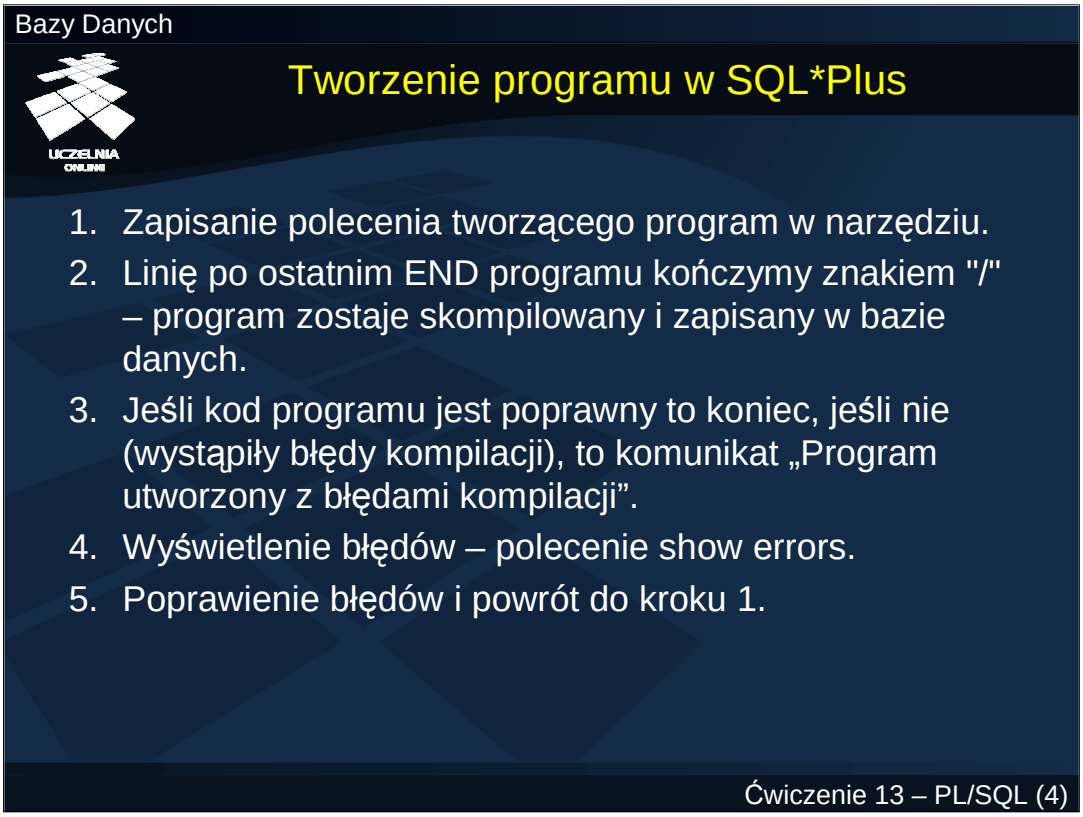
Na wstępie ćwiczenia scharakteryzujemy składowane programy PL/SQL. Dalej omówione zostaną poszczególne kroki procesu tworzenia programów PL/SQL, wspólne dla wszystkich rodzajów programów. Następne slajdy przedstawią polecenia tworzące procedury i funkcje oraz metody wywoływania procedur i funkcji z innych programów. Kolejne zagadnienie to tworzenie pakietów procedur i funkcji. Ćwiczenie zakończy omówienie procedur wyzwalanych.



Anonimowy blok PL/SQL, którym zajmowaliśmy się w poprzednich ćwiczeniach, jest wykonywany zaraz po jego utworzeniu. Blok taki nie jest przechowywany trwale w bazie danych – jeśli chcemy ponownie wykonać blok, musimy dokonać jego ponownej definicji.

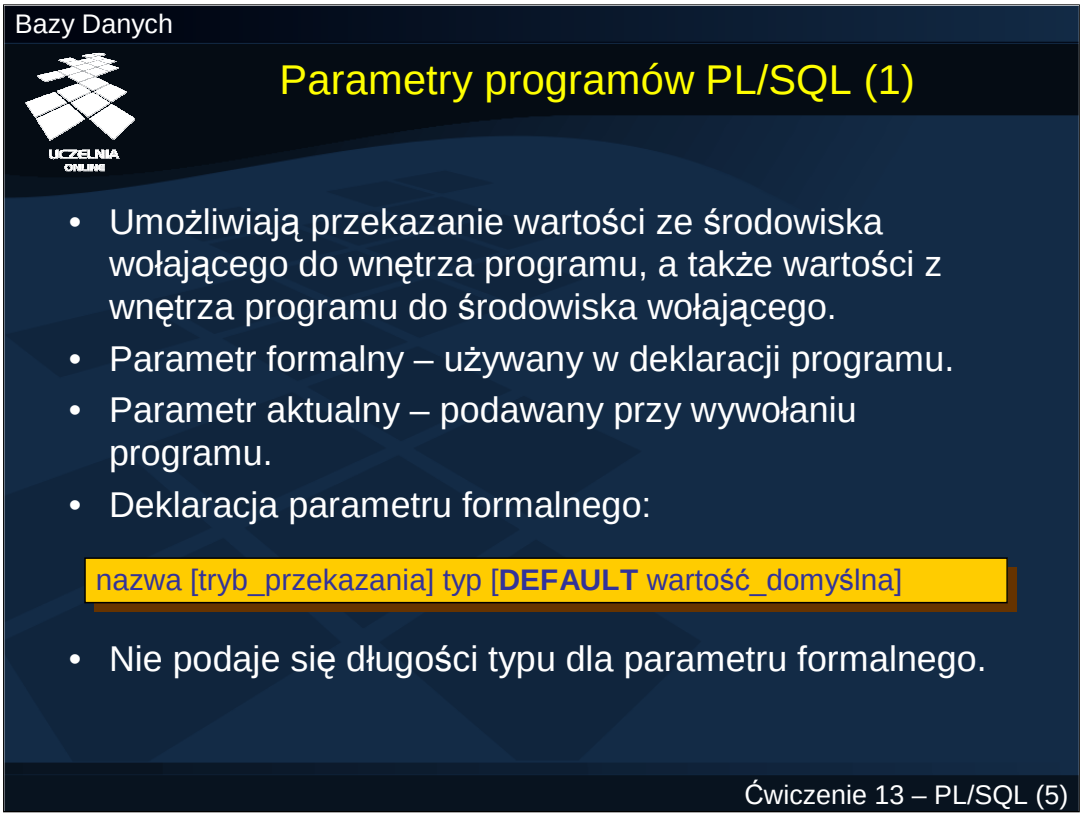
SZBD Oracle umożliwia tworzenie trwale przechowywanych w bazie danych programów, tzw. programów składowanych. Programy te przechowywane są w postaci skompilowanej, przez co ich wykonanie jest szybsze niż wykonanie anonimowego bloku PL/SQL. Program składowany zostaje uruchomiony albo na żądanie użytkownika albo na skutek zajścia w bazie danych określonych zdarzeń.

SZBD Oracle umożliwia tworzenie następujących rodzajów programów składowanych: procedur, wykonujących określone akcje, funkcji, wykonujących obliczenia i zwracających wartości, pakietów, będących bibliotekami procedur i funkcji, oraz wyzwalaczy, będących procedurami wywoływanymi automatycznie (bez ingerencji użytkownika) na skutek zajścia w bazie danych określonych zdarzeń.



Omówimy teraz proces tworzenia programu składowanego przy pomocy narzędzia SQL*Plus, wchodzącego w skład dystrybucji SZBD Oracle.

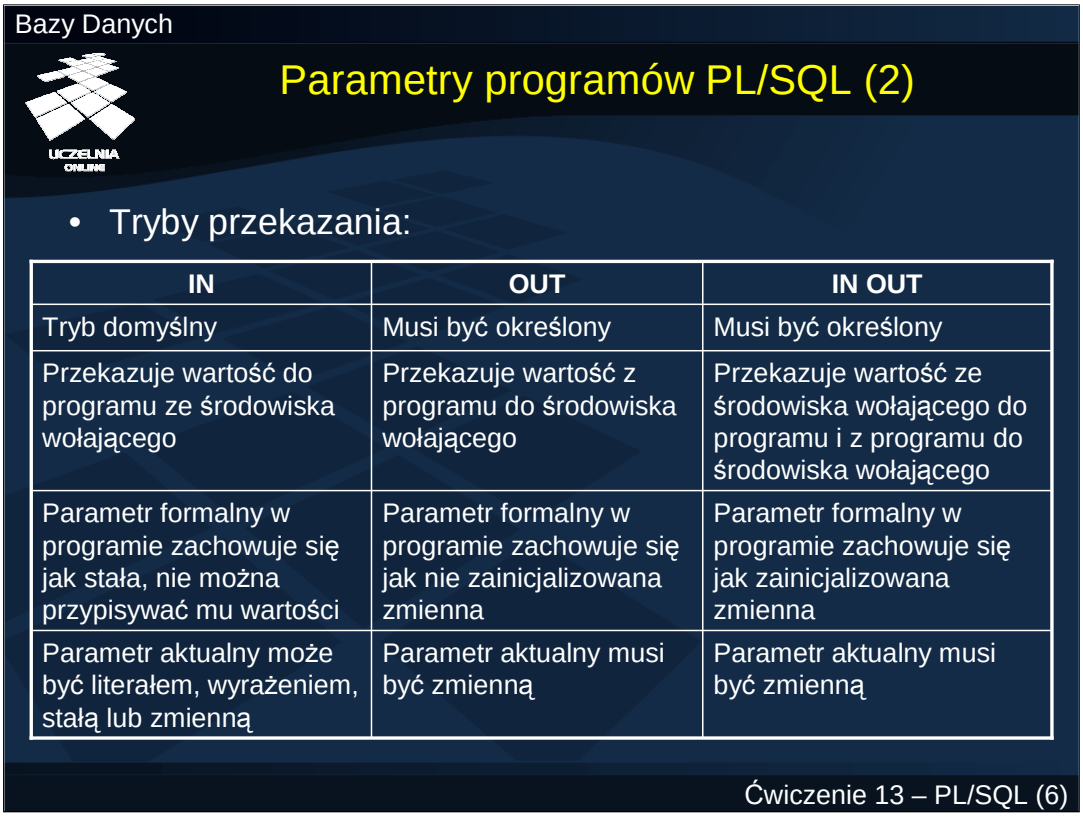
Pierwszy krok jest analogiczny do procesu tworzenia anonimowego bloku PL/SQL i polega na zapisaniu w narzędziu polecenia tworzącego program. W linii następującej po słowie kluczowym END, kończącym program, umieszczamy symbol „/” (ukośnik), oznaczający koniec definicji programu. Narzędzie przesyła kod programu do SZBD Oracle, tam kod ten jest sprawdzany pod kątem poprawności syntaktycznej i semantycznej, skompilowany i trwale zapisany w bazie danych. Jeśli kod programu jest poprawny, na konsoli zostaje wyświetlony komunikat „Program został utworzony” i program jest gotowy do wykonania. Jeśli jednak podczas sprawdzania poprawności kodu i kompilacji programu wystąpiły błędy, otrzymujemy na konsoli komunikat „Program został utworzony z błędami kompilacji”. Program, mimo że został trwale zachowany w bazie danych, nie jest poprawny i nie może być wywoływany. Używając polecenie show errors wyświetlamy błędy, jakie zostały znalezione w programie. Błędy należy poprawić, utworzyć ponownie program (krok 1.). Sekwencję tą powtarzamy tak długo, aż nie uzyskamy poprawnego programu.



Program PL/SQL może zostać wyposażony w listę parametrów, które umożliwiają przekazanie do wnętrza programu wartości ze środowiska wołającego (najczęściej) lub przekazanie wartości z programu do środowiska wołającego (rzadziej). Omawiając parametry programów musimy wprowadzić rozróżnienie pomiędzy parametrami formalnymi, używanymi w trakcie tworzenia programu, a parametrami aktualnymi, podawanymi w miejsce parametrów formalnych w momencie wywoływania programu.

Deklarując parametr formalny w definicji programu musimy podać jego nazwę – będzie ona używana wewnątrz programu. Dobrym zwyczajem jest tworzenie nazw dla parametrów formalnych z przedrostkiem „p_” (np. p_id_zesp), dzięki temu programy stają się bardziej czytelne. Dalej podajemy tryb przekazania parametru, następny slajd zostanie poświęcony w całości temu elementowi deklaracji parametru.

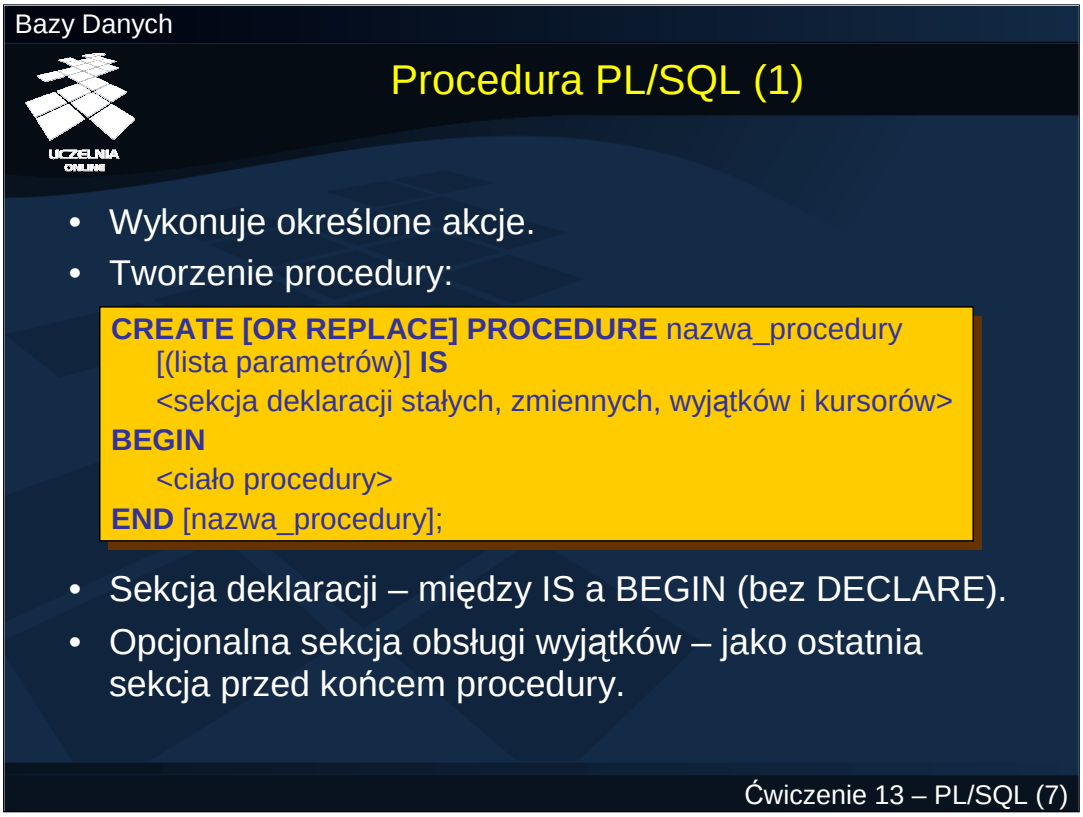
Po określeniu trybu przekazania podajemy typ danych parametru, pamiętajmy jednak, że dla parametrów nie podajemy nigdy długości typu. Wreszcie, jeśli parametr ma mieć wartość domyślną, co oznacza, że dla takiego parametru nie jest konieczne podanie parametru aktualnego przy wywołaniu programu, wartość tą podaje się po słowie DEFAULT.



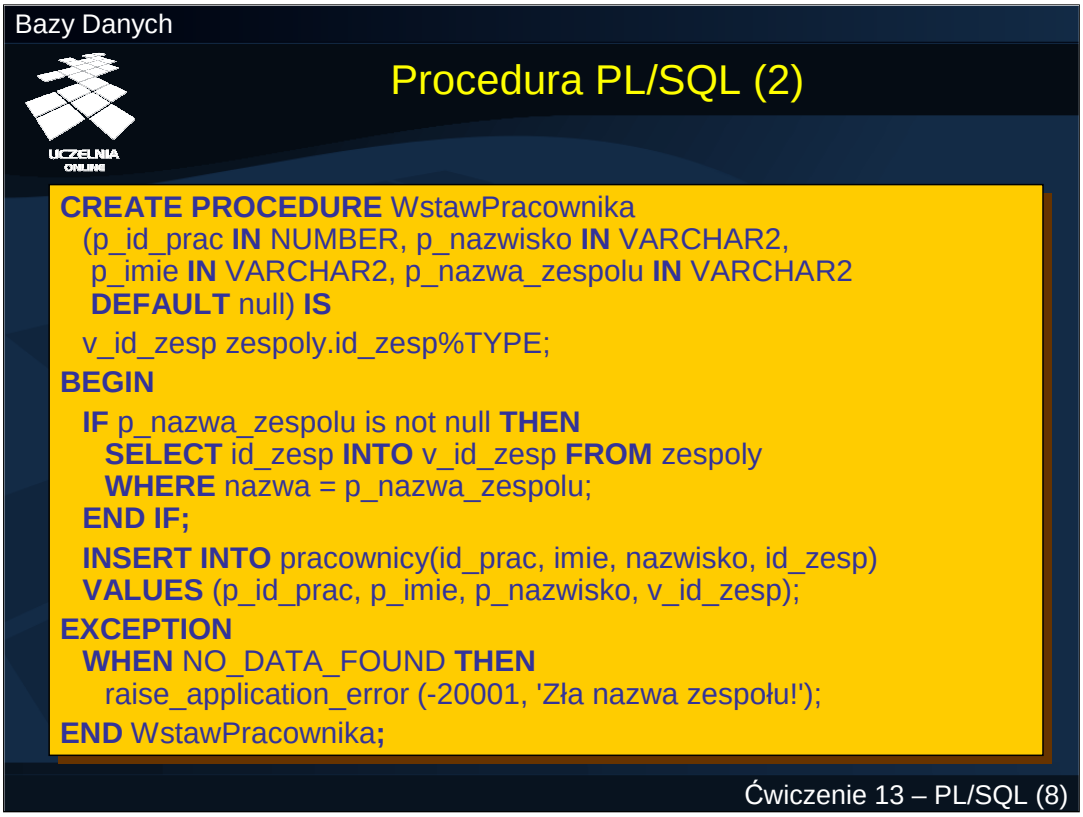
Przyjrzymy się teraz dokładnie trybowi przekazania parametrów programy. Mamy tutaj trzy możliwości. Pierwsza to tryb oznaczony przez słowo IN, będący trybem domyślnym (jeśli pominiemy określenie trybu, zostaje przyjęty tryb IN). Parametr formalny z trybem IN przekazuje wartość ze środowiska wołającego do programu. Parametr taki w ciele programu zachowuje się jak stała – może być tylko odczytywany. Parametr aktualny dla parametru formalnego przekazywanego w trybie IN może być literałem, wyrażeniem, stałą lub zmienną. Ten tryb przekazania jest trybem najczęściej stosowanym.

Z kolei parametr przekazywany w trybie OUT służy do przekazania wartości z wnętrza programu do środowiska wołającego. W programie taki parametr zachowuje się jak niezainicjalizowana zmienna (przechowuje wartość pustą), można (wręcz należy) do takiego parametru przypisać w programie jakąś wartość. Parametr aktualny dla parametru formalnego, przekazywanego w trybie OUT, musi być zmienną.

Ostatni tryb przekazywania parametrów, tryb IN OUT jest kombinacją dwóch wcześniej wymienionych trybów, parametr w tym trybie służy do przekazywania zarówno wartości ze środowiska wołającego do programu jak i w drugim kierunku, z programu do środowiska wołającego. W programie parametr taki zachowuje się jak zainicjalizowana zmienna (jej wartością jest wartość parametru aktualnego), natomiast parametr aktualny musi w tym przypadku być zmienną.

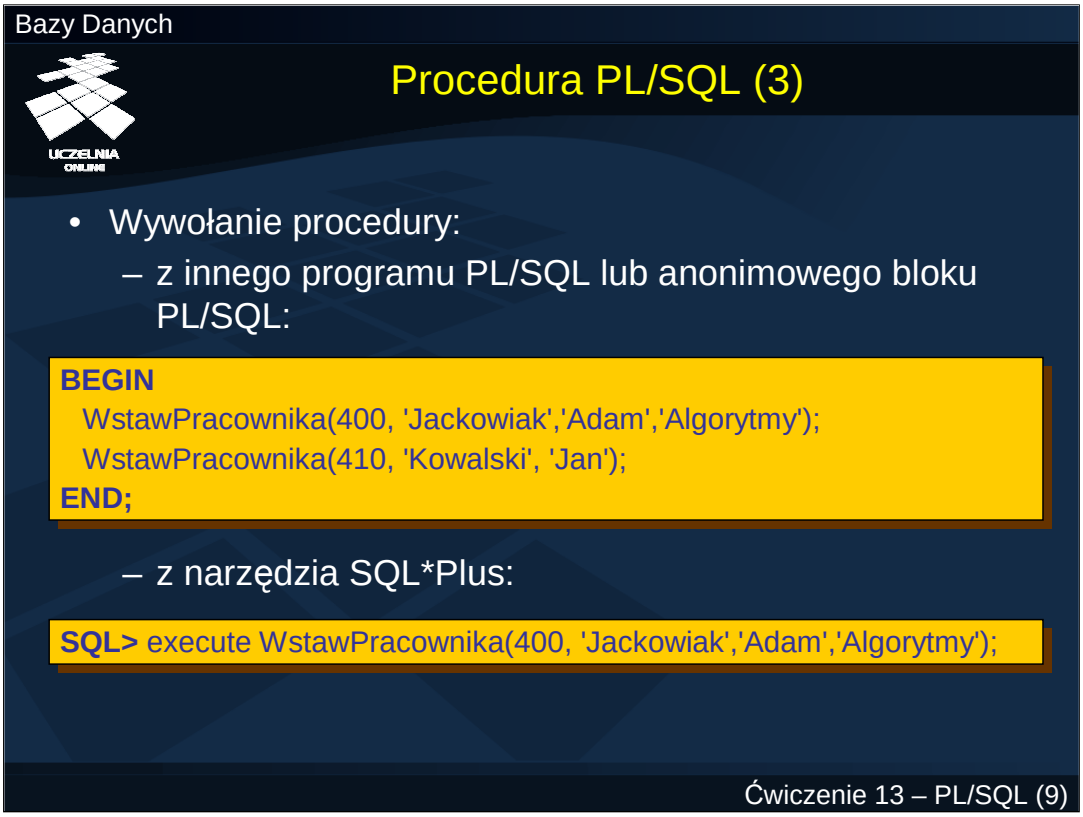


Rozpoczniemy teraz omawianie poleceń tworzących poszczególne rodzaje programów składowanych. Rozpoczniemy od polecenia tworzącego procedurę. Przypomnijmy: procedura jest programem, wykonującym określone akcje. Polecenie rozpoczyna się od słów CREATE PROCEDURE, po których podajemy nazwę tworzonej procedury. Jeśli procedurę redefiniujemy (poprawiamy istniejącą już w bazie danych procedurę), musimy dodać słowa OR REPLACE, jeśli je pominiemy, SZBD nie pozwoli na utworzenie nowej procedury o nazwie takiej samej jak procedura już istniejąca w bazie danych. Po nazwie procedury umieszczamy listę parametrów formalnych. Lista otoczona jest nawiasami, poszczególne parametry oddzielone są od siebie przecinkami. Jeśli procedura nie posiada w ogóle parametrów, pomijamy również nawiasy. Następnie po słowie IS rozpoczyna się sekcja deklaracji stałych, zmiennych, wyjątków i kursorów, jakie zostaną użyte w ciele procedury. Jest to odpowiednik sekcji deklaracji bloku anonimowego, rozpoczynającej się od słowa DECLARE. Słowo BEGIN rozpoczyna ciało procedury, w którym umieszczamy instrukcje PL/SQL (również możemy zamieszczać zagnieżdżone bloki PL/SQL). Opcjonalna sekcja obsługi błędów, podobnie jak w przypadku bloku anonimowego, musi być ostatnią sekcją ciała procedury. Ciało kończymy słowem kluczowym END, po którym można powtórzyć nazwę procedury (nie jest to jednak wymagane).



Bieżący slajd przedstawia polecenie tworzące procedurę o nazwie WstawPracownika. Procedura ma cztery parametry formalne: p_id_prac, p_nazwisko, p_imie oraz p_nazwa_zespolu, wszystkie parametry przekazywane są w trybie IN. Parametr p_nazwa_zespolu ma zdefiniowaną wartość domyślną (wartość pustą NULL). W sekcji deklaracji procedury zadeklarowano jedną zmienną numeryczną o nazwie v_id_zesp.

Omówimy teraz ciało procedury. Jeśli wartość parametru p_nazwa_zespolu nie jest pusta, wówczas wykonywane jest zapytanie odczytujące identyfikator zespołu, którego nazwa została przekazana do procedury przez parametr p_nazwa_zespolu. Wartość odczytanego identyfikatora zostaje zapamiętana w zmiennej v_id_zesp. Jeśli polecenie nie znajdzie zespołu (oznacza to, że w parametrze p_nazwa_zespolu przekazano do procedury nazwę nieistniejącego zespołu), generuje ono wyjątek NO_DATA_FOUND, który jest przechwytywany przez klauzulę obsługi błędów procedury. W przypadku wystąpienia tego wyjątku procedura jest przerywana przez wykonanie polecenia raise_application_error z zasygnalizowaniem błędu o numerze –20001 i wyświetleniem komunikatu „Zła nazwa zespołu”. Jeśli odczyt identyfikatora zespołu się powiedzie lub parametr p_id_zespolu przechowuje wartość pustą, wykonywane jest drugie polecenie w ciele procedury, polecenie INSERT, które wstawia do relacji PRACOWNICY dane nowego pracownika. Zauważmy, że wartości dla atrybutów ID_PRAC, NAZWISKO i IMIE pobierane są z parametrów formalnych procedury, wartość dla ID_ZESP jest pobierana ze zmiennej v_id_zesp (może być pusta).



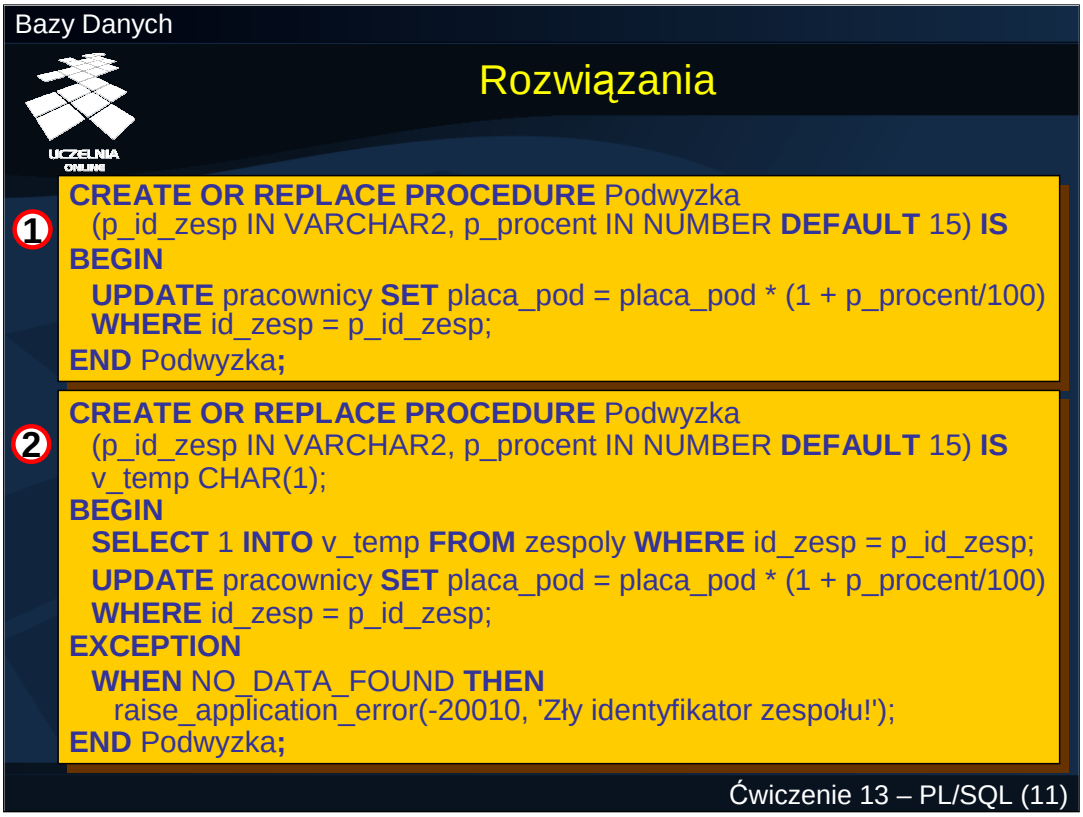
Na bieżącym slajdzie omówiono dwa sposoby wywołania poprawnie zdefiniowanej procedury. Pierwszy sposób to wywołanie procedury z innego programu PL/SQL lub anonimowego bloku PL/SQL. W zaprezentowanym przykładzie procedurę WstawPracownika wywołano dwukrotnie. Pierwszy raz z parametrami aktualnymi 400, Jackowiak, Adam i Algorytmy dla parametrów formalnych, odpowiednio: p_id_zesp, p_nazwisko, p_imie i p_nazwa_zespolu. W drugim wywołaniu pominięto parametr aktualny dla parametru formalnego p_nazwa_zespolu (ten parametr ma zdefiniowaną wartość domyślną).

Drugi przykład pokazuje, jak wywołać procedurę z narzędzia PL/SQL. Tutaj używamy specjalnego polecenia EXECUTE, po którym podajemy nazwę procedury wraz z listą parametrów aktualnych.

Zadania

1. Napisz procedurę Podwyzka, która wszystkim pracownikom zespołu o danym numerze (parametr) podniesie płacę podstawową o podany procent (parametr). Domyślnie podwyżka powinna wynosić 15%.
2. Dodaj do powyższej procedury obsługę błędów – jeśli podano numer nieistniejącego zespołu, procedura powinna wygenerować błąd o numerze –20010 i odpowiednim komunikacie.

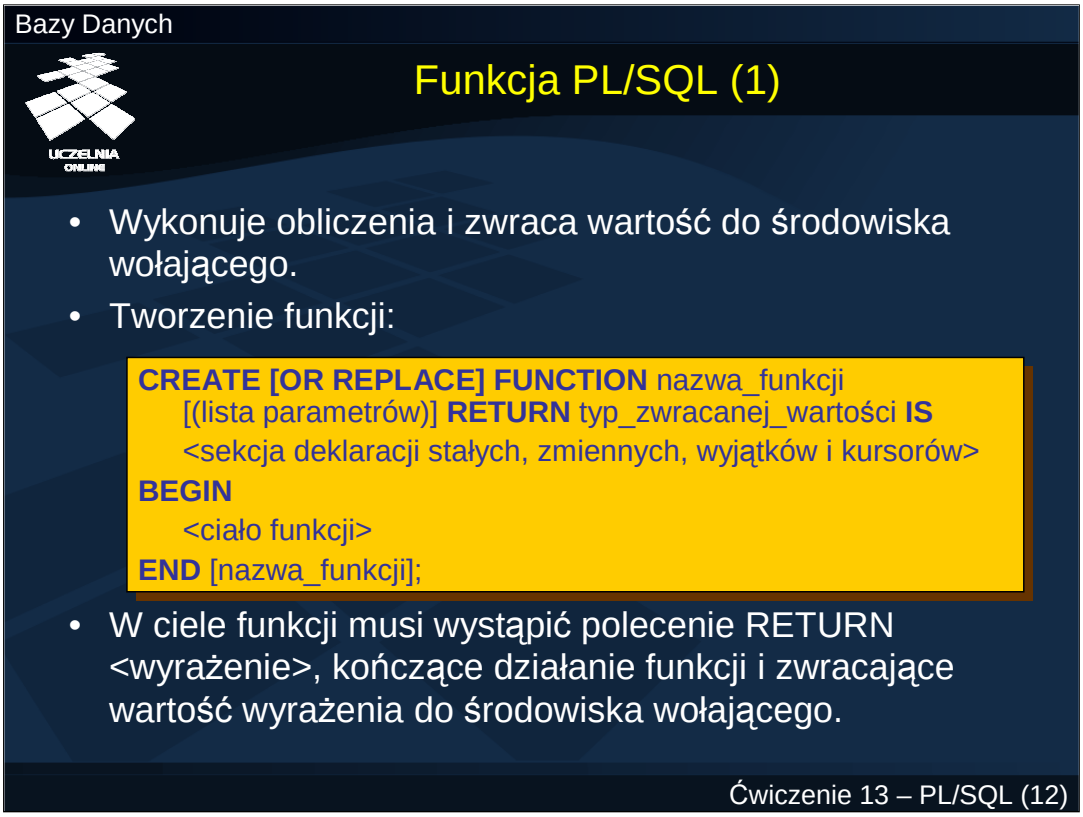
Bieżący slajd rozpoczyna zbiór zadań, których celem jest utrwalenie wiadomości o procedurach składowanych PL/SQL.



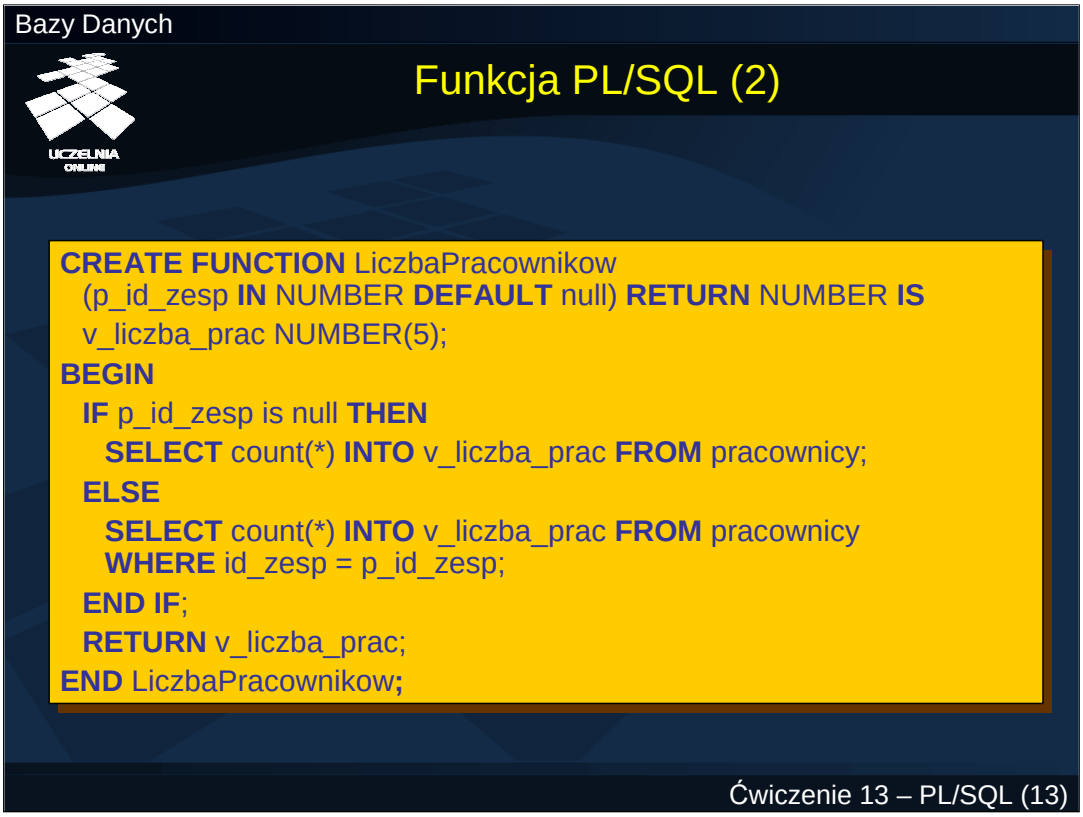
Bieżący slajd przedstawia rozwiązania zadań (1) i (2), których treść zacytowano poniżej.

(1) Napisz procedurę Podwyzka, która wszystkim pracownikom zespołu o danym numerze (parametr) podniesie płacę podstawową o podany procent (parametr). Domyślnie podwyżka powinna wynosić 15%.

(2) Dodaj do powyższej procedury obsługę błędów – jeśli podano numer nieistniejącego zespołu, procedura powinna wygenerować błąd o numerze –20010 i odpowiednim komunikacie.



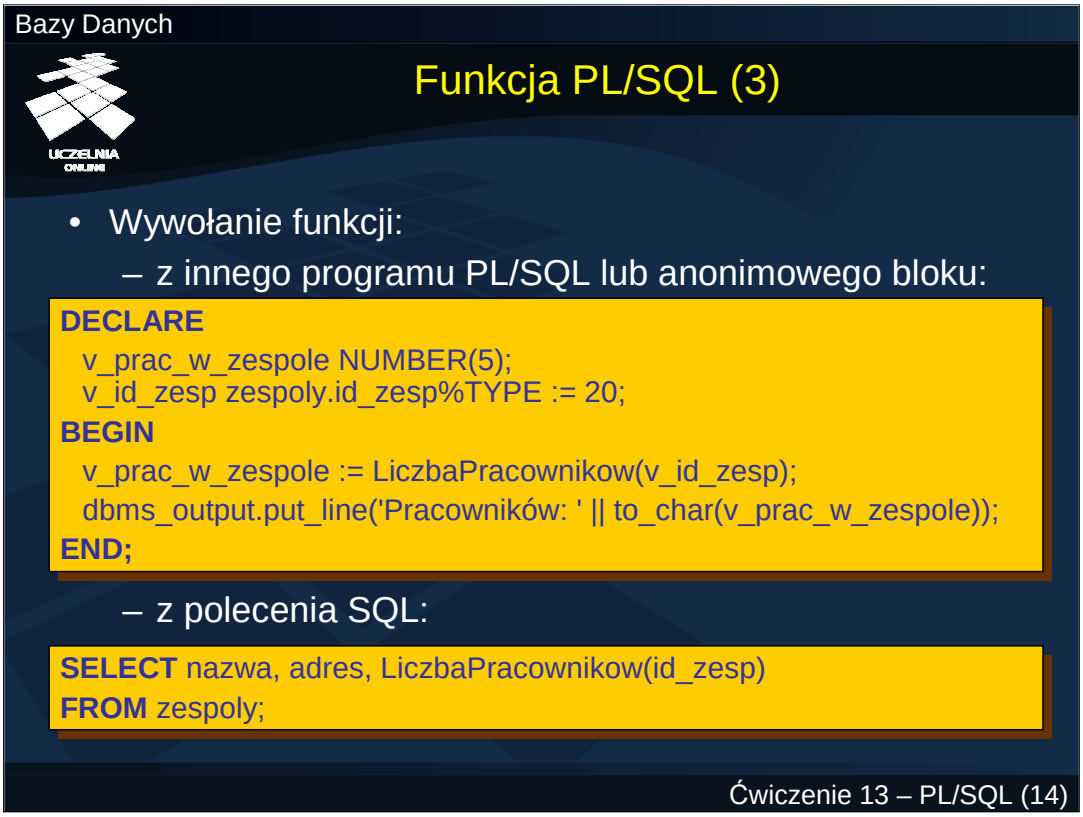
Bieżący slajd omawia polecenie tworzące składowaną funkcję PL/SQL. Funkcje wykonują obliczenia i zwracają wartość do środowiska wołającego. Elementy polecenia CREATE FUNCTION, tworzącego funkcję, są analogiczne do elementów tworzących procedurę. Dodatkowa klauzula RETURN po opcjonalnej liście parametrów funkcji, określa typ wartości, jaką zwróci funkcja po wykonaniu. W ciele funkcji musi wystąpić polecenie RETURN, po którym umieszcza się wyrażenie. Polecenie RETURN kończy działanie funkcji, zwracając do środowiska wołającego wartość wyrażenia. Należy tak konstruować funkcję, aby każde jej wykonanie zakończyło się realizacją polecenia RETURN. W przeciwnym wypadku przy wywołaniu funkcji zostanie wygenerowany komunikat o błędzie wykonania.



Na bieżącym slajdzie zaprezentowano polecenie tworzące funkcję o nazwie LiczbaPracownikow. Funkcja ma jeden parametr formalny o nazwie p_id_zesp z wartością domyślną równą NULL. W sekcji deklaracji zadeklarowano zmienną liczbową v_liczba_prac. Omówimy teraz ciało funkcji.

Jeśli wartość parametru p_id_zesp jest pusta, wykonane zostaje polecenie znajdujące liczbę wszystkich pracowników, liczba ta zostaje zapisana w zmiennej v_liczba_prac. W przeciwnym wypadku znaleziona zostaje liczba pracowników, którzy należą do zespołu o identyfikatorze równym wartości parametru p_id_zesp. Liczba ta zostaje zapisana w zmiennej v_liczba_prac. Polecenie RETURN kończy działanie funkcji, przekazując do środowiska wołającego wartość zmiennej v_liczba_prac.

Reasumując: jeśli w parametrze p_id_zesp przekazano identyfikator zespołu, funkcja wylicza liczbę pracowników tego zespołu, jeśli w wywołaniu funkcji parametr p_id_zesp został pominięty, funkcja wylicza liczbę wszystkich pracowników.



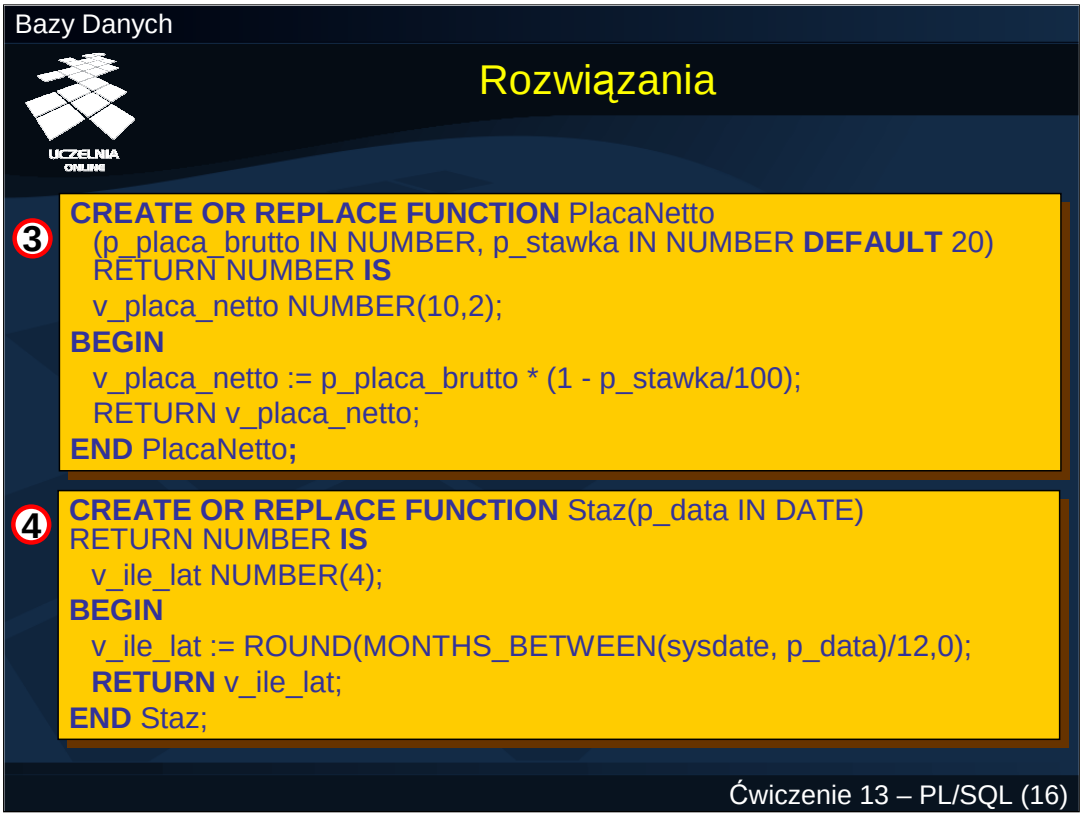
Bieżący slajd przedstawia dwie metody wywołania funkcji. Pierwsza metoda to wywołanie z programu lub bloku PL/SQL. W zaprezentowanym przykładzie zadeklarowano w bloku dwie zmienne: v_prac_w_zespole oraz v_id_zesp, inicjalizowanej wartością 20. W części wykonywalnej bloku wywołujemy funkcję LiczbaPracownikow w podobny sposób jak procedurę, ale uwaga – trzeba umożliwić funkcji przekazanie obliczanej przez nią wartości. W tym przykładzie wartość od funkcji odbierze zmienna v_prac_w_zespole. Zauważmy, że parametrem aktualnym w tym przykładzie jest zmienna v_id_zesp. Wartość, przekazana przez funkcję do zmiennej v_prac_w_zespole, zostaje wypisana na konsoli.

Drugi sposób wywołania funkcji, niedostępny dla procedur, to wywołanie w poleceniu SQL. W zaprezentowanym przykładzie funkcja LiczbaPracownikow zostaje użyta w zapytaniu do relacji ZESPOLY. Parametrem aktualnym funkcji jest wartość atrybutu ID_ZESP, funkcja zostanie wykonana jednokrotnie dla każdego rekordu, odczytanego przez zapytanie z relacji ZESPOLY.

Zadania

3. Napisz funkcję PlacaNetto, która dla podanej płacy brutto (parametr) i podanej stawki podatku (parametr o wartości domyślnej 20%) wyliczy płacę netto.
4. Napisz funkcję Staz, która dla daty zatrudnienia pracownika (parametr) wyliczy staż pracy w latach.
5. Napisz funkcję Silnia, która dla danego n obliczy n! = 1 * 2 * ... * n (zastosuj iterację).

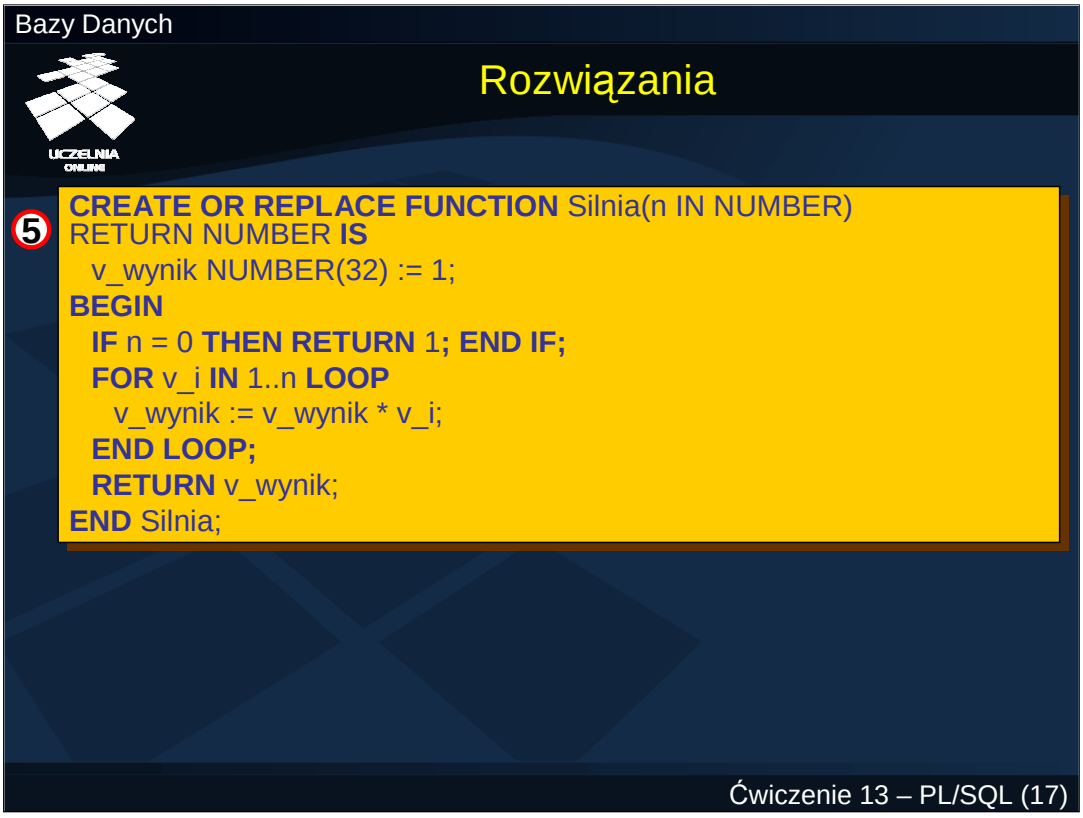
Bieżący slajd prezentuje zbiór zadań, których celem jest utrwalenie wiadomości o składowanych funkcjach PL/SQL.



Bieżący slajd przedstawia rozwiązania zadań (3) i (4), których treść zacytowano poniżej.

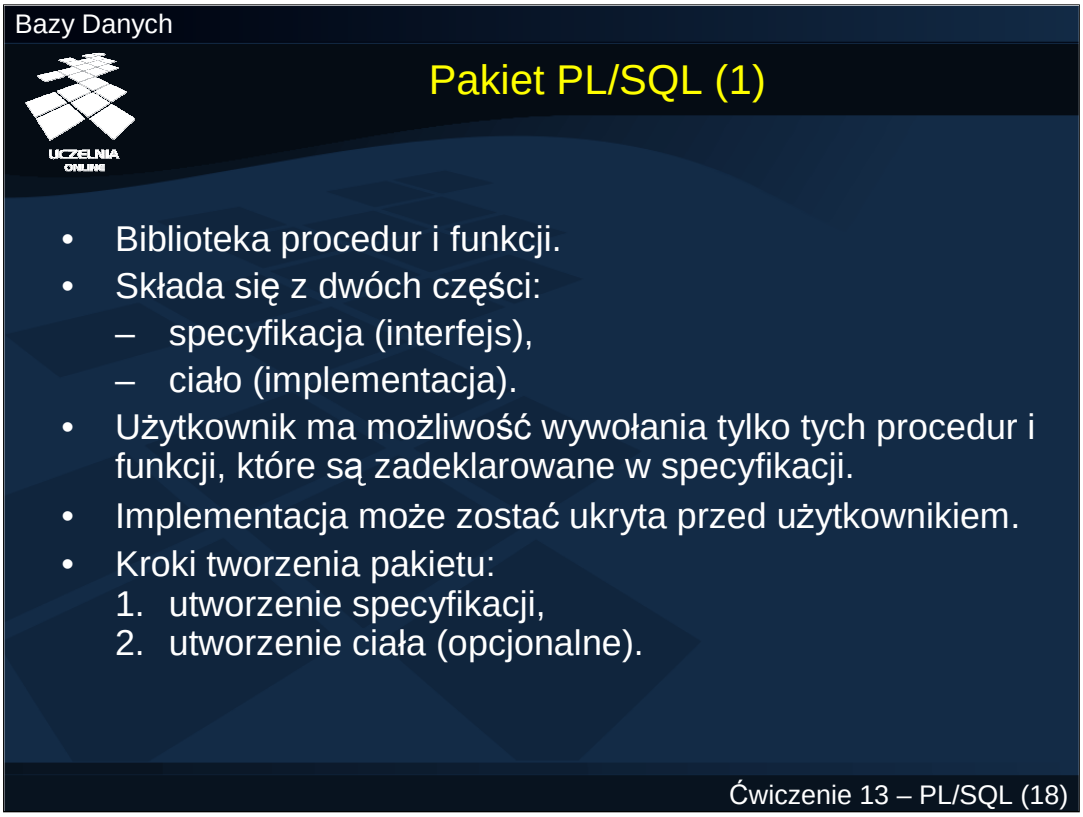
(3) Napisz funkcję PlacaNetto, która dla podanej płacy brutto (parametr) i podanej stawki podatku (parametr o wartości domyślnej 20%) wyliczy płacę netto.

(4) Napisz funkcję Staz, która dla daty zatrudnienia pracownika (parametr) wyliczy staż pracy w latach.



Bieżący slajd przedstawia rozwiązanie zadania (5), którego treść zacytowano poniżej.

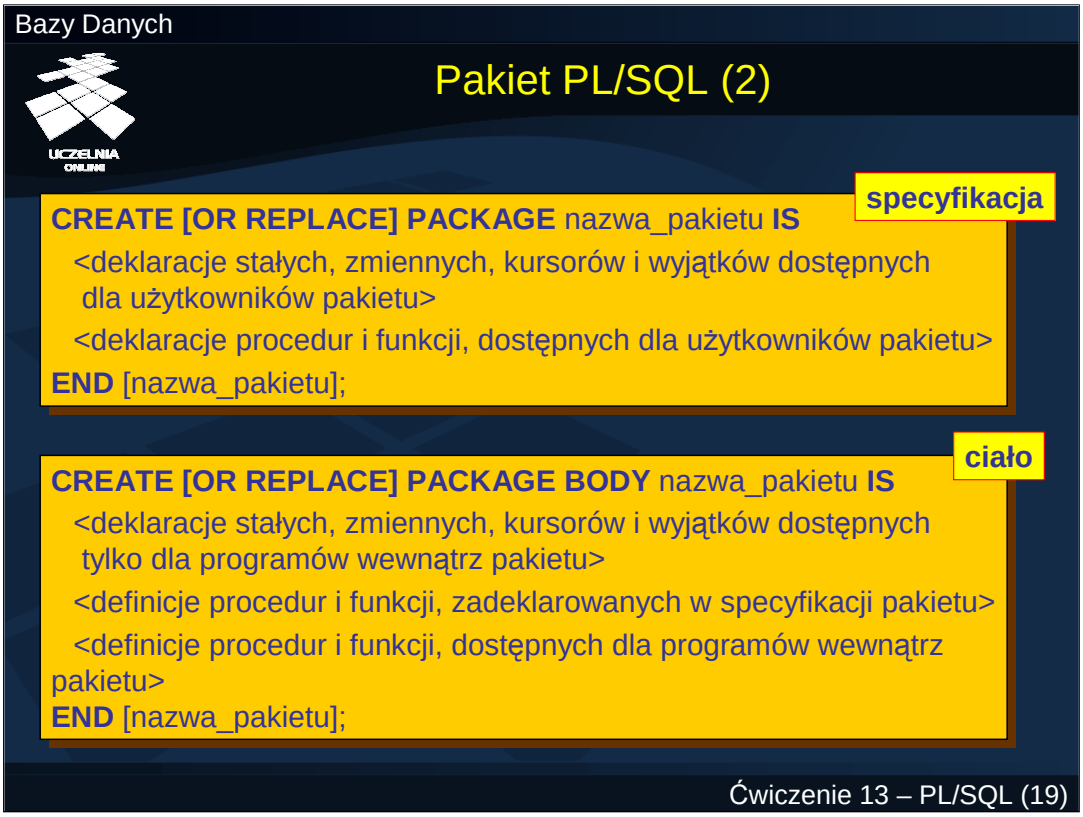
(5) Napisz funkcję Silnia, która dla danego n obliczy n! = 1 * 2 * ... * n (zastosuj iterację).



Bieżący slajd rozpoczyna omawianie pakietów PL/SQL. Pakiet to biblioteka, zawierająca procedury i funkcje. Takie połączenie procedur i funkcji w pakiet jest korzystne – znacznie ułatwia czynności administracyjne w SZBD, dodatkowo programista uzyskuje pewne mechanizmy, niedostępne w zwykłych procedurach i funkcjach.

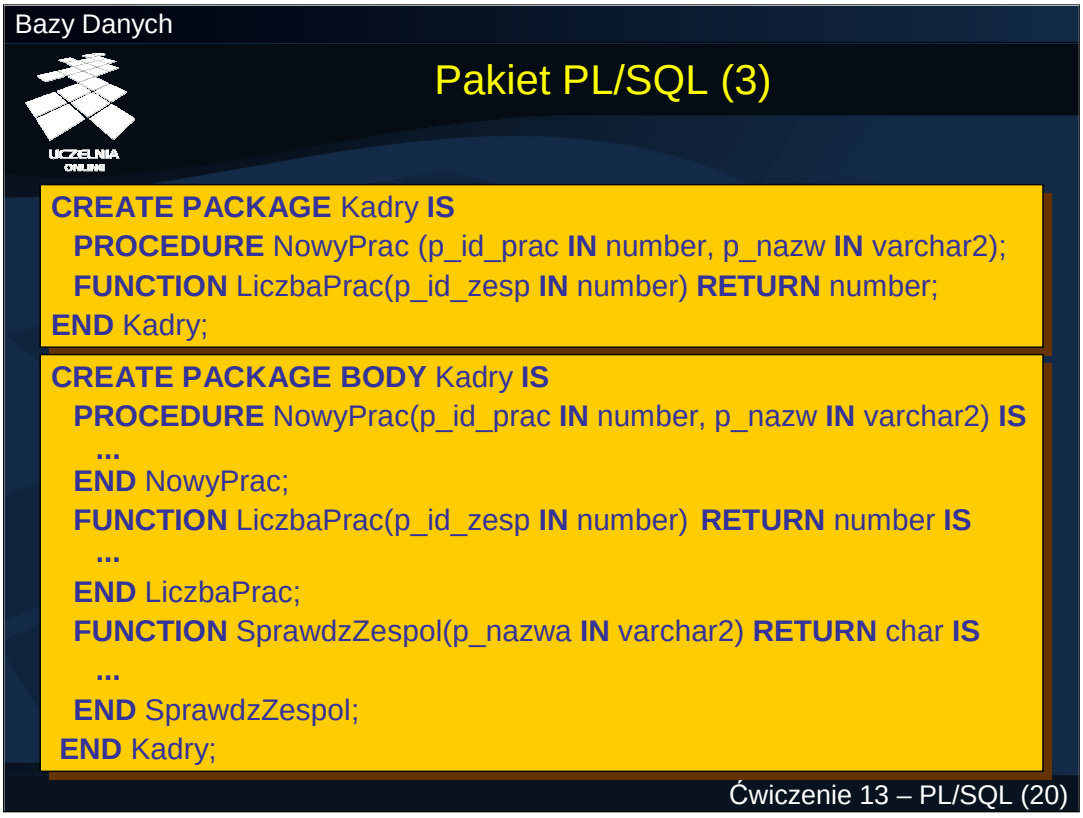
Pakiet składa się z dwóch części: specyfikacji, nazywanej również interfejsem, oraz ciała, inaczej nazywanego implementacją. W specyfikacji umieszcza się deklaracje procedur i funkcji pakietu, które mają być dostępne dla użytkowników pakietu. Dodatkowo można tutaj umieścić deklaracje zmiennych, stałych, wyjątków oraz kursorów, z których mogą korzystać użytkownicy pakietu. Z kolei w ciele pakietu umieszcza się definicje elementów zadeklarowanych w specyfikacji oraz definicje pozostałych elementów, a więc procedur, funkcji, stałych, zmiennych, wyjątków i kursorów, które jednak nie są dostępne dla użytkowników pakietu, a mogą z nich korzystać tylko inne procedury lub funkcje pakietu. Podział pakietu na specyfikację i ciało umożliwia ukrycie przed użytkownikiem końcowym implementacji pakietu – użytkownik może korzystać tylko z tych elementów pakietu, które określi w specyfikacji twórca pakietu.

Tworzenie pakietu składa się z dwóch kroków: 1. utworzenia specyfikacji oraz 2. utworzenia ciała. Są to osobne polecenia, które zostaną przedstawione na następnym slajdzie.



Specyfikację pakietu tworzy się wykonując polecenie CREATE PACKAGE lub CREATE OR REPLACE PACKAGE w przypadku redefinicji istniejącego pakietu, za którym umieszcza się nazwę definiowanego pakietu. Następnie po słowie IS umieszcza się deklaracje elementów, które mają być dostępne dla użytkowników pakietu, a więc stałych, zmiennych, kursorów, wyjątków, oraz oczywiście procedur i funkcji. Składnia deklaracji jest identyczna jak w przypadku omówionych już we wcześniejszych ćwiczeniach elementów. Jednak w przypadku deklaracji procedur i funkcji pomijamy słowa CREATE OR REPLACE, deklarację w przypadku procedury kończymy po liście parametrów, a w przypadku funkcji po nazwie typu zwracanej przez funkcję wartości. Specyfikację pakietu kończy słowo END, po którym można umieścić nazwę pakietu.

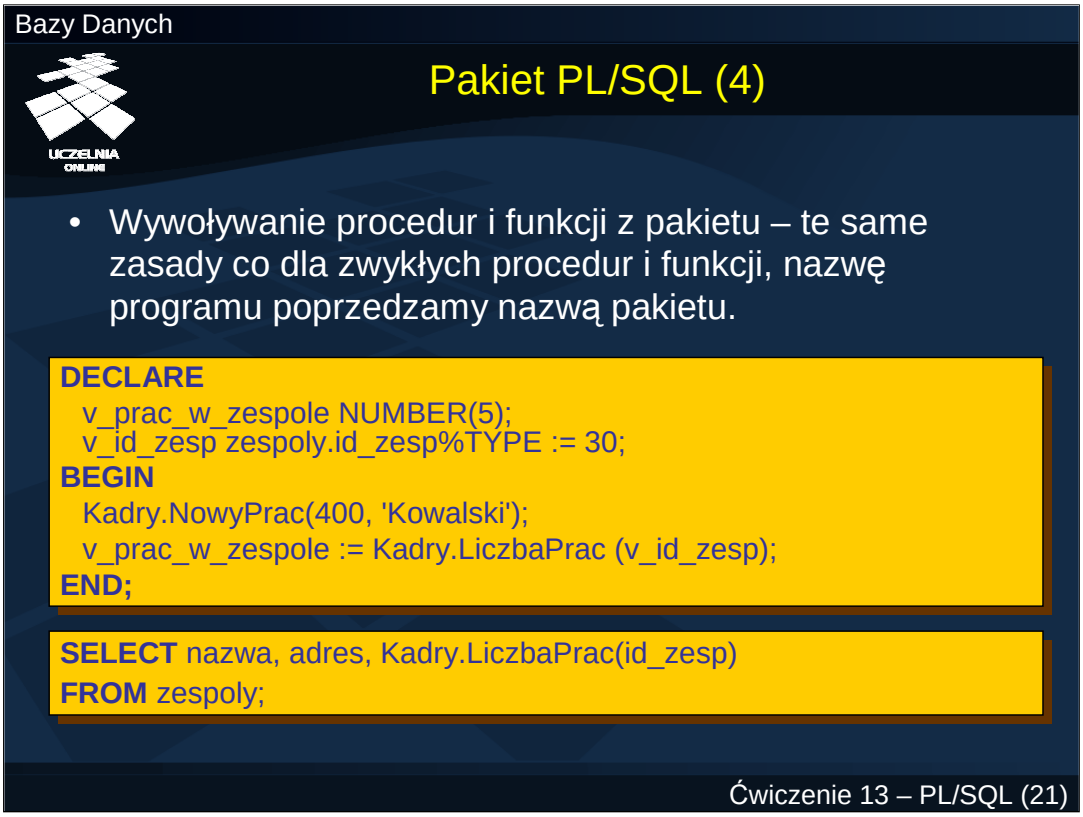
Polecenie tworzące ciało pakietu rozpoczyna się od słów CREATE PACKAGE BODY lub CREATE OR REPLACE PACKAGE BODY w przypadku redefinicji ciała istniejącego już pakietu. Następnie podaje się nazwę pakietu (jest to ta sama nazwa, jaką podano przy tworzeniu specyfikacji pakietu), po słowie IS umieszczamy definicję elementów pakietu. Definicję ciała pakietu kończy słowo END, po którym można dodatkowo umieścić nazwę pakietu.



Bieżący slajd tworzy pakiet o nazwie Kadry. Pierwsze polecenie definiuje specyfikację pakietu, w niej zadeklarowano procedurę NowyPrac i funkcję LiczbaPrac. Te dwa programy będą dostępne dla użytkowników pakietu.

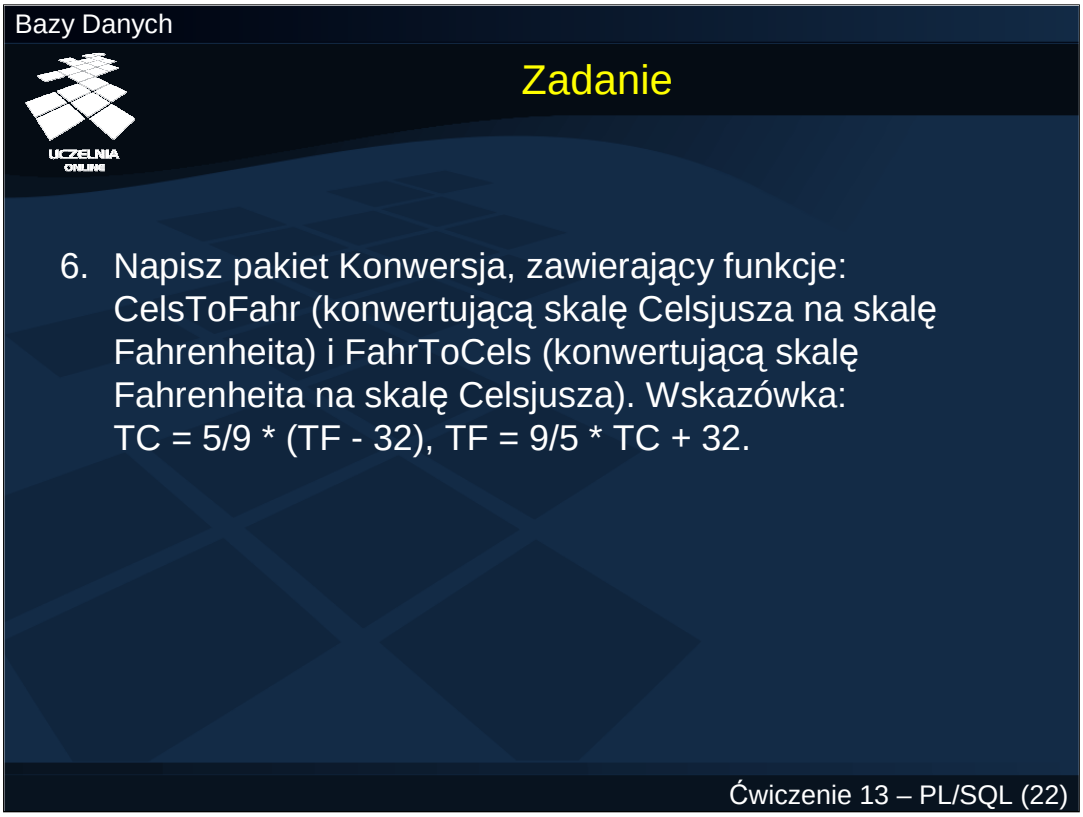
Drugie polecenie tworzy ciało pakietu. W ciele umieszczono definicję programów, zadeklarowanych w specyfikacji. Trzeci program, funkcja SprawdzZespol, nie została zadeklarowana w specyfikacji pakietu, nie jest więc dostępna dla użytkowników pakietu, a może być wywołana jedynie z procedury NowyPrac lub funkcji LiczbaPrac.

Z racji braku miejsca w ciele przykładowego pakietu pominięto ciała programów.

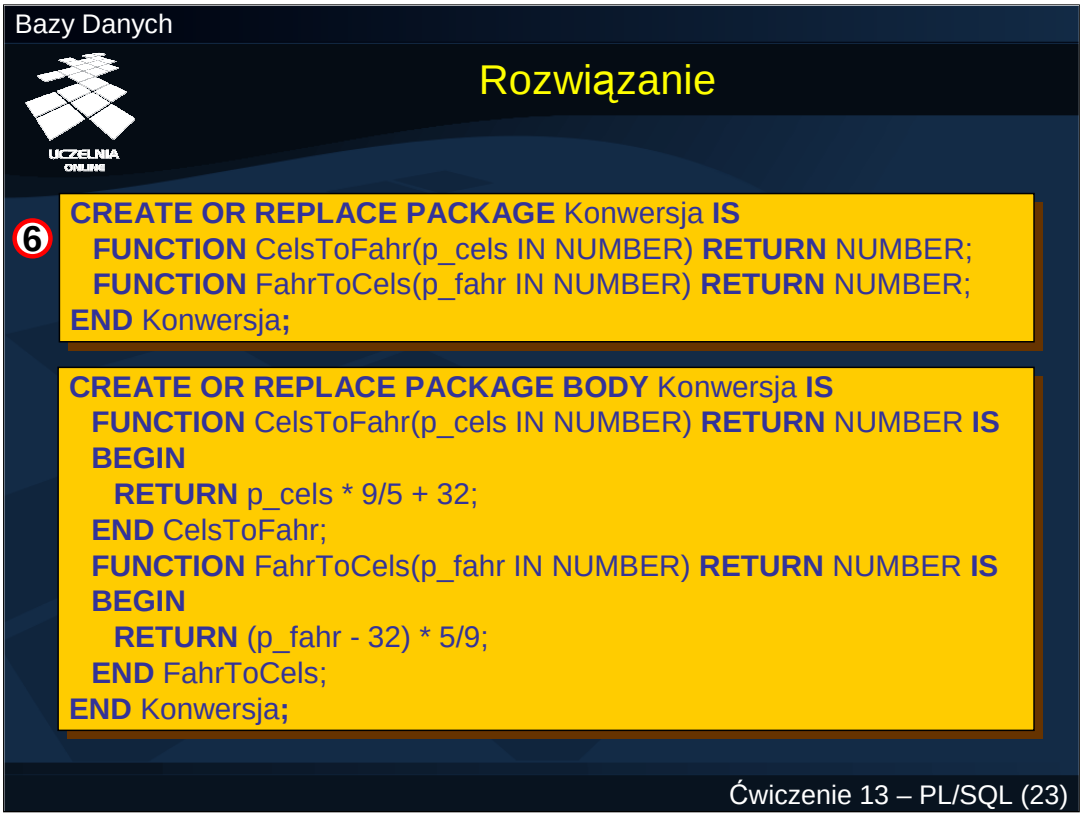


Wywołanie procedur i funkcji, zadeklarowanych w specyfikacji pakietu, jest niemalże identyczne jak wywołanie zwykłych procedur i funkcji. Jedyna różnica to konieczność umieszczenia nazwy pakietu przed nazwą procedury lub funkcji, nazwa pakietu oddzielona jest od nazwy programu kropką.

Na bieżącym slajdzie w pierwszym przykładzie zaprezentowano wywołanie procedury NowyPrac i funkcji LiczbaPrac pakietu Kadry z anonimowego bloku PL/SQL. Drugi przykład pokazuje wywołanie funkcji LiczbaPrac pakietu Kadry z zapytania SQL.

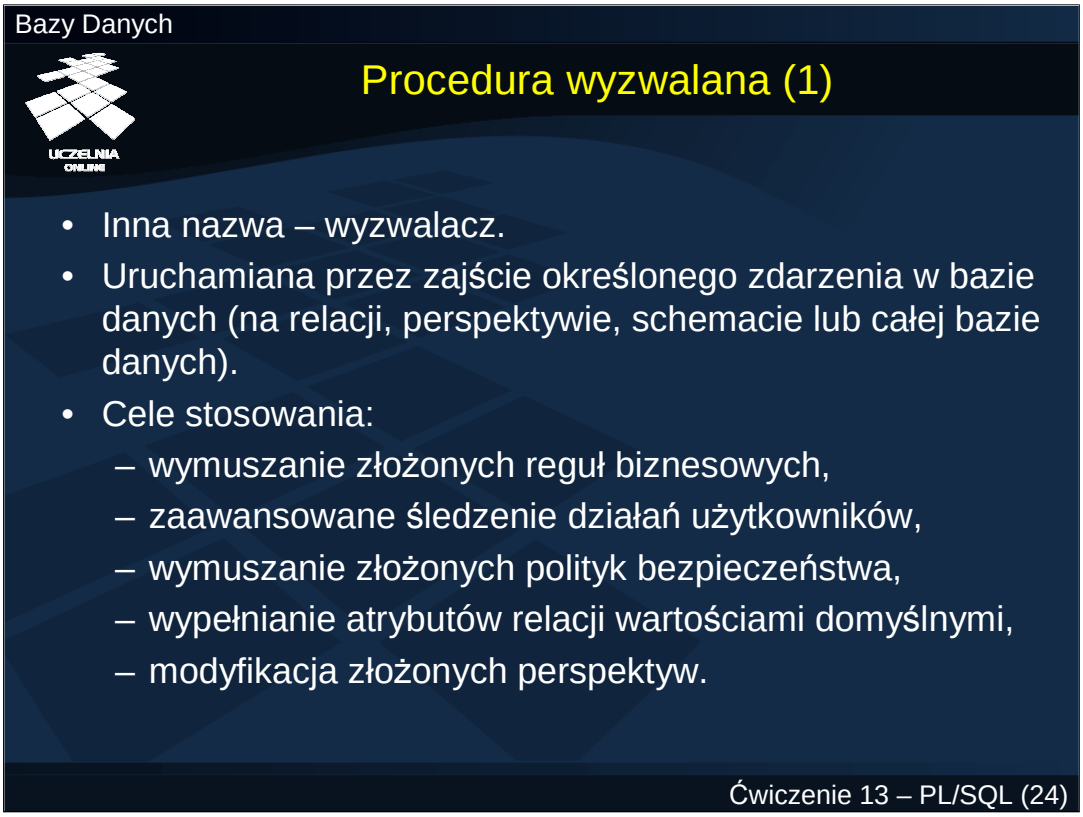


Bieżący slajd prezentuje zadanie, które ma utrwalić wiadomości o pakietach PL/SQL.



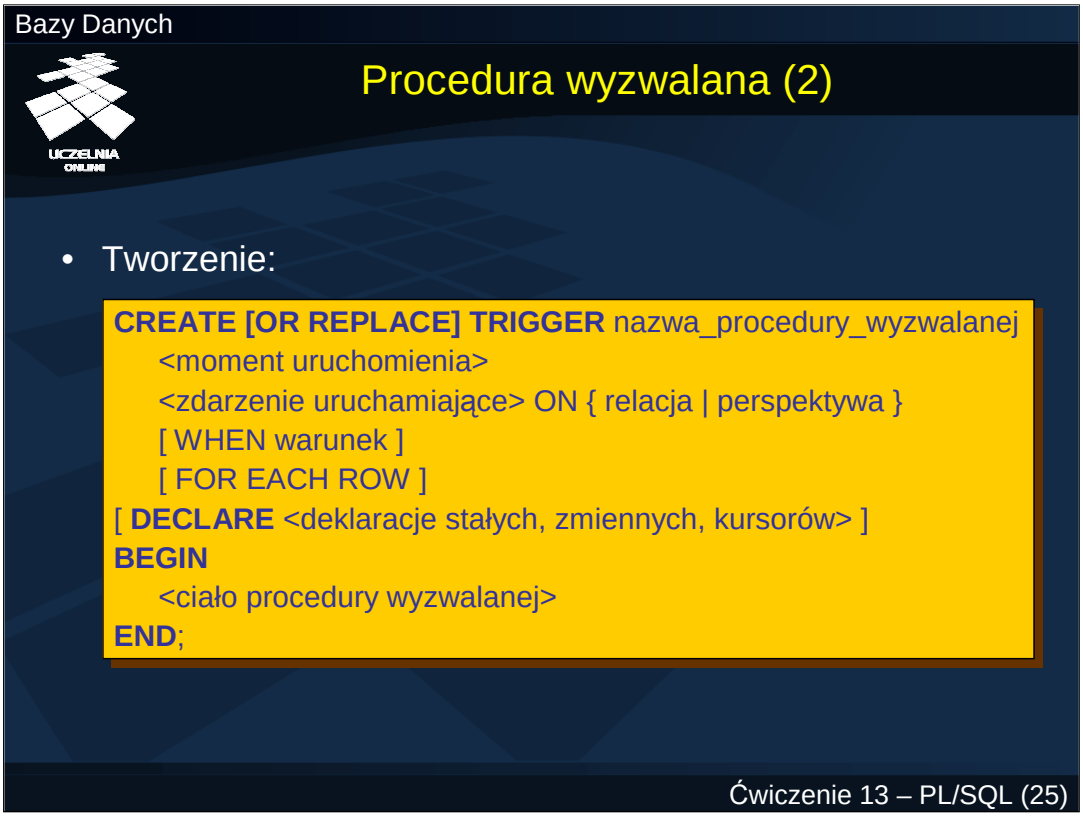
Bieżący slajd przedstawia rozwiązanie zadania (6), którego treść zacytowano poniżej.

(6) Napisz pakiet Konwersja, zawierający funkcje: CelsToFahr (konwertującą skalę Celsjusza na skalę Fahrenheita) i FahrToCels (konwertującą skalę Fahrenheita na skalę Celsjusza). Wskazówka:
TC = 5/9 * (TF - 32), TF = 9/5 * TC + 32.

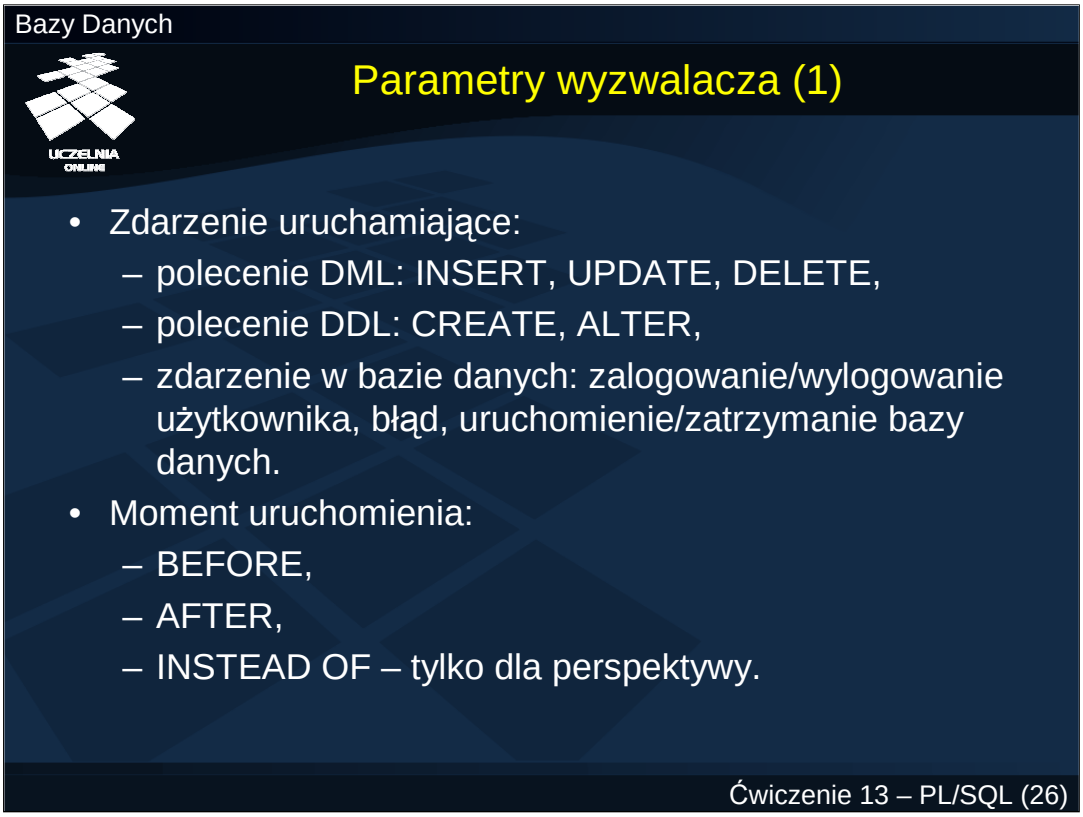


Ostatni z omawianych w bieżącym ćwiczeniu rodzajów programów PL/SQL to procedury wyzwalane, nazywane inaczej wyzwalaczami. W przeciwieństwie do pozostałych rodzajów podprogramów, wyzwalacze nie są uruchamiane na żądanie użytkownika, ale automatycznie na skutek zajścia określonych zdarzeń w bazie danych. Zdarzenia te mogą być zdefiniowane dla relacji lub perspektywy (np. wstawienie, usunięcie lub modyfikacja rekordu), określonego schematu (np. utworzenie nowej relacji w schemacie) lub całej bazy danych (np. przyłączenie użytkownika do bazy danych).

Przedstawimy teraz cele stosowania wyzwalaczy. Jednym z nich jest wymuszanie złożonych reguł biznesowych, np. zależności rekordów jednej relacji od rekordów innej relacji. Kolejne zastosowania to zaawansowane śledzenie operacji, realizowanych przez użytkowników bazy danych, wymuszanie złożonych polityk bezpieczeństwa (np. uniemożliwianie pracy użytkownikom w odpowiednich porach), wypełnianie atrybutów relacji wartościami domyślnymi przy wstawianiu nowych rekordów. Bardzo ważnym zastosowaniem jest umożliwianie modyfikacji złożonych perspektyw relacyjnych (problem ten zostanie omówiony w dalszej części ćwiczenia).

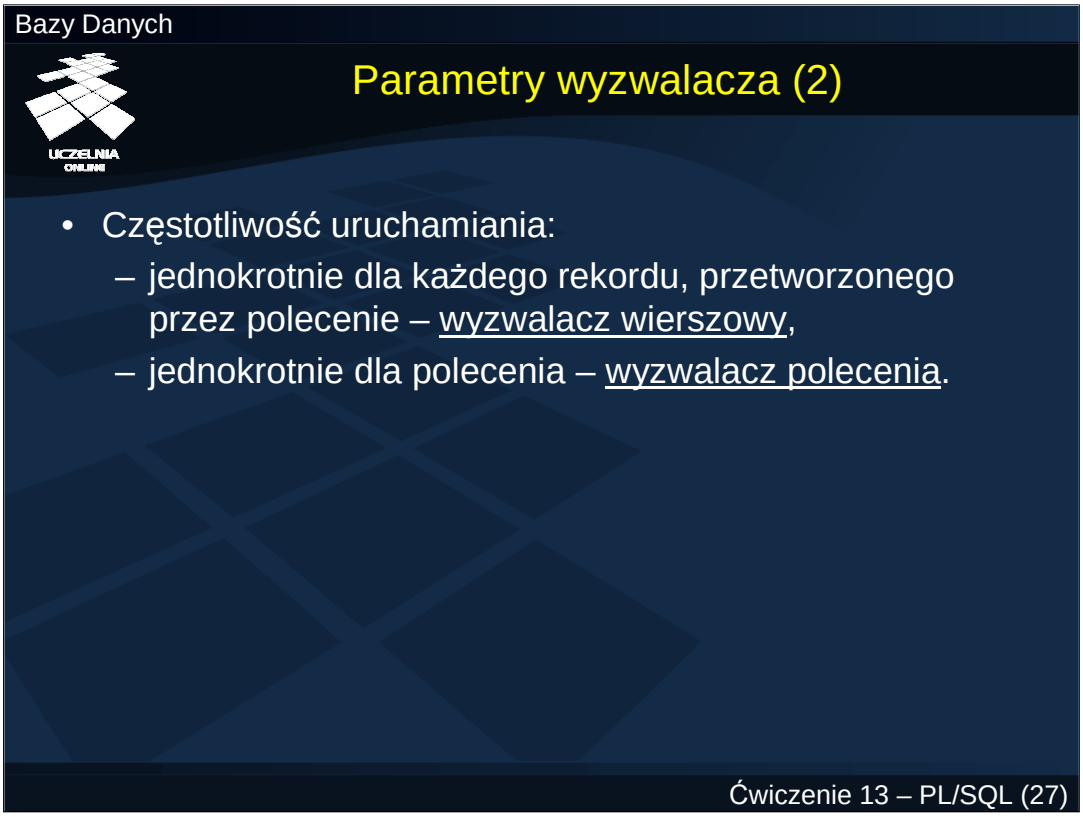


Bieżący slajd przedstawia polecenie tworzące wyzwalacz. Polecenie rozpoczyna się od słów kluczowych CREATE TRIGGER lub CREATE OR REPLACE TRIGGER w przypadku redefinicji istniejącego już wyzwalacza, po których podaje się nazwę wyzwalacza. Następnie definiuje się parametry wyzwalacza: moment uruchomienia i zdarzenie uruchamiające. Dalej możliwe jest podanie dodatkowego warunku uruchomienia wyzwalacza. Następnie określa się częstotliwość uruchomienia wyzwalacza (klauzula FOR EACH ROW). Dalej umieszcza się blok PL/SQL z opcjonalnymi sekcjami deklaracji oraz obsługi wyjątków, zawierający polecenia, tworzące ciało procedury wyzwalanej.

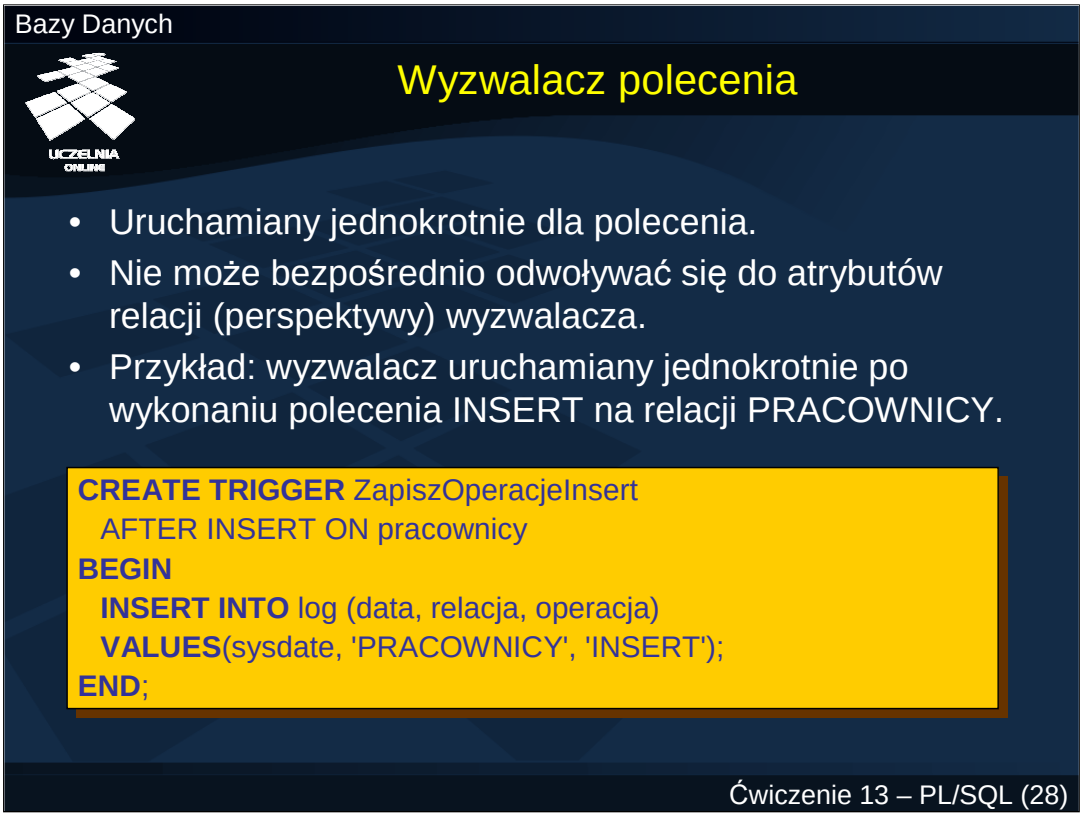


Omówimy teraz parametry wyzwalacza. Pierwszy z nich określa zdarzenie, którego wystąpienie ma uruchomić wyzwalacz. Może to być polecenie DML: INSERT, UPDATE lub DELETE, w tym wypadku po słowie ON podaje się nazwę relacji lub perspektywy, do której będzie kierowane polecenie. Dodatkowo w przypadku polecenia UPDATE można wprost podać listę atrybutów relacji lub perspektywy, których uaktualnienie ma spowodować uruchomienie wyzwalacza. Realizuje się to używając konstrukcji „UPDATE OF lista_atrybutów”. Inny rodzaj poleceń, które mogą uruchomić wyzwalacz, to dwa polecenia z grupy DDL: CREATE lub ALTER. W takim przypadku wyzwalacz może zostać uruchomiony przez utworzenie nowego obiektu w danym schemacie bądź zmianę definicji obiektu istniejącego. Wreszcie wyzwalacz może być uruchomiony przez zdarzenie w bazie danych, takie jak zalogowanie lub wylogowanie użytkownika, wystąpienie błędu, uruchomienie czy też zatrzymanie bazy danych.

Drugi parametr określa moment wykonania wyzwalacza w stosunku do zdarzenia, które spowodowało uruchomienie wyzwalacza. BEFORE oznacza, że wyzwalacz ma zostać wykonany przed realizacją polecenia odpowiadającego za zdarzenie, na którym zdefiniowano wyzwalacz. Np. wyzwalacz BEFORE INSERT zostaje uruchomiony wydaniem przez użytkownika polecenia INSERT, wyzwalacz jest wykonywany przed wykonaniem polecenia INSERT w bazie danych. AFTER powoduje, że wyzwalacz zostaje wykonany po realizacji polecenia odpowiadającego za zdarzenie, na którym zdefiniowano wyzwalacz. Z kolei INSTEAD OF powoduje wykonanie wyzwalacza zamiast polecenia, odpowiadającego za uruchomienie wyzwalacza. INSTEAD OF może być jednak definiowany jedynie dla perspektyw.

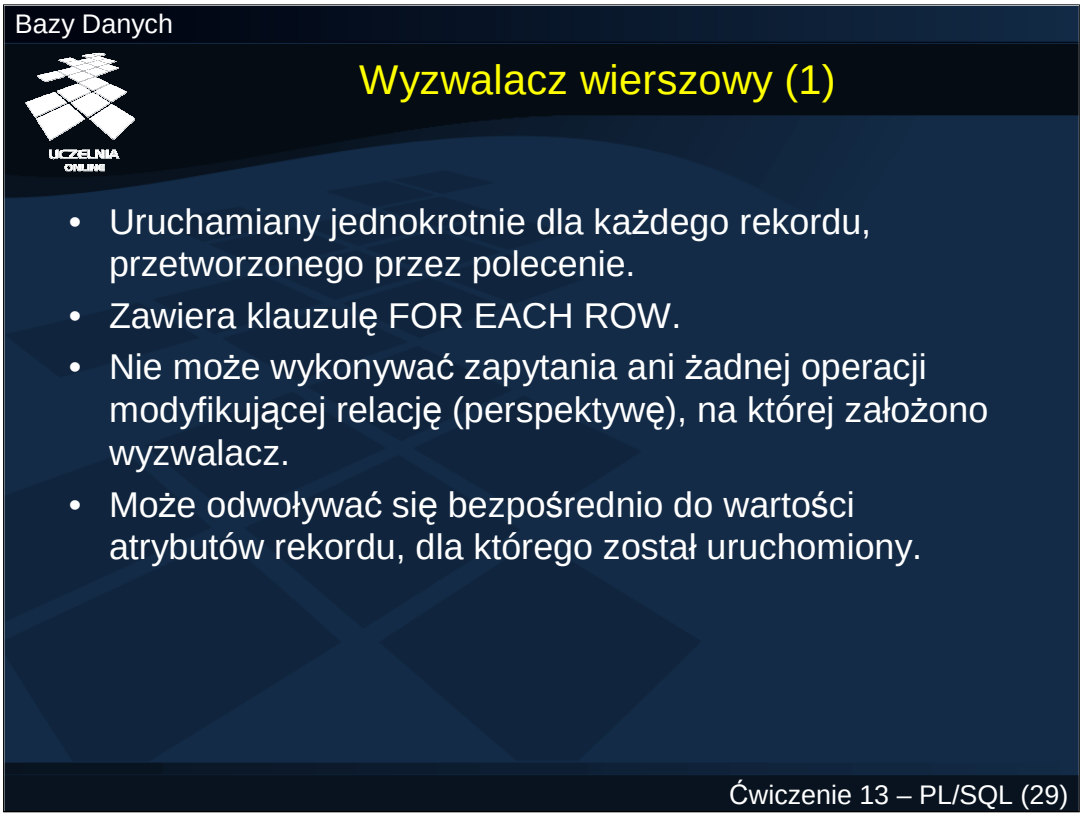


Trzeci parametr określa częstotliwość uruchamiania wyzwalacza. Mamy tu dwie możliwości. Wyzwalacz polecenia wykonywany jest zawsze jednokrotnie, niezależnie od liczby rekordów, jakie przetworzyło polecenie. Z kolei wyzwalacz wierszowy jest wykonywany tyle razy, ile rekordów przetwarza polecenie, które odpowiada za zdarzenie, na którym zdefiniowano wyzwalacz.

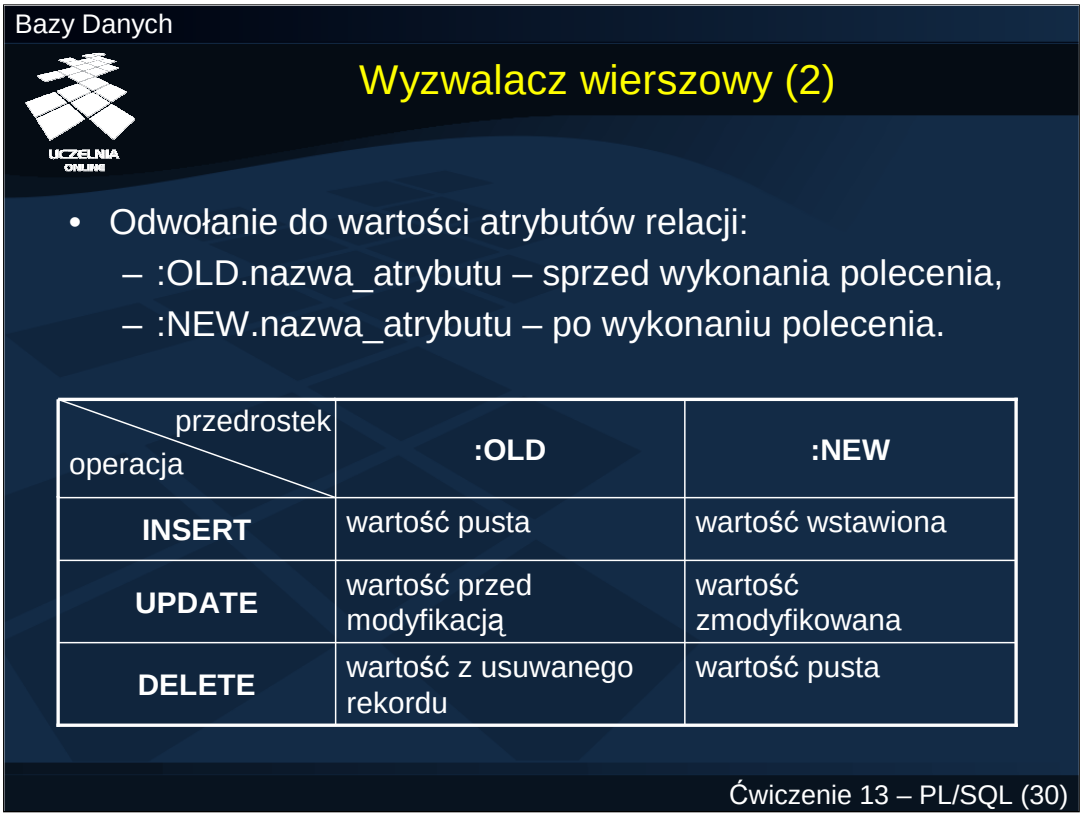


Bieżący slajd dokładnie wyjaśnia wyzwalacze polecenia. Wyzwalacz taki wykonywany jest zawsze jednokrotnie, niezależnie od liczby rekordów, jakie przetworzyło polecenie. Ograniczeniem takiego wyzwalacza jest niemożność bezpośredniego odwołania w ciele wyzwalacza do danych relacji lub perspektywy, na której założono wyzwalacz (takie odwołanie jest możliwe w w przypadku wyzwalaczy wierszowych).

Zanalizujmy wyzwalacz polecenia o nazwie ZapiszOperacjeInsert. Wyzwalacz został założony na relacji PRACOWNICY i zostaje wykonany po realizacji operacji INSERT. Jedyne polecenie w ciele wyzwalacza wstawia do relacji LOG rekord, opisujący datę realizacji operacji wstawienia rekordów do relacji PRACOWNICY. Reasumując, omawiany wyzwalacz śledzi operacje wstawienia rekordów do relacji PRACOWNICY.

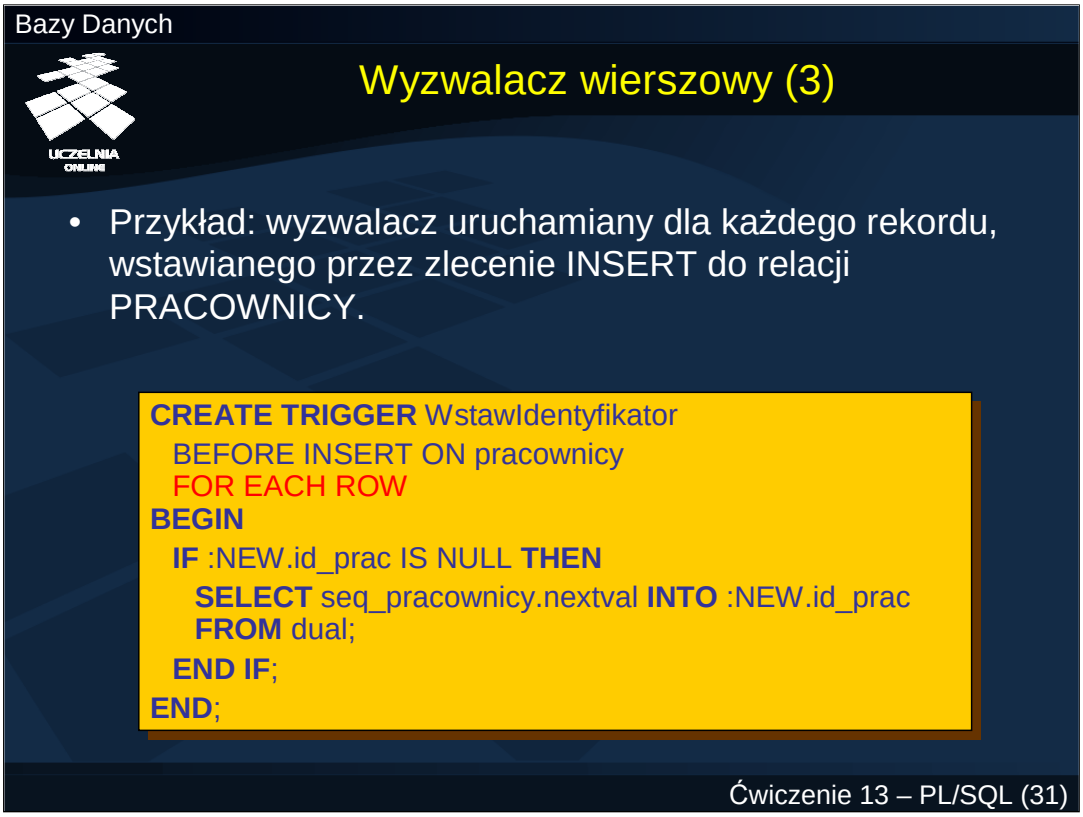


Wyzwalacz wierszowy jest wykonywany jednokrotnie dla każdego rekordu, przetworzonego przez polecenie uruchamiające wyzwalacz. Aby wyzwalacz był wyzwalaczem wierszowym, należy w jego definicji podać klauzulę FOR EACH ROW. Wyzwalacz wierszowy ma jedno poważne ograniczenie – w jego ciele nie może zostać wykonana żadna operacja odczytu lub modyfikacji danych relacji lub perspektywy, dla której zdefiniowano wyzwalacz. Gdyby realizacja takich operacji była dopuszczalna w wyzwalaczu wierszowym, SZBD nie mógłby zagwarantować spójności operacji. Za to w wyzwalaczu wierszowym można bezpośrednio odwołać się do wartości atrybutów rekordu relacji lub perspektywy, dla którego wyzwalacz został uruchomiony. Zostanie to omówione na następnym slajdzie.

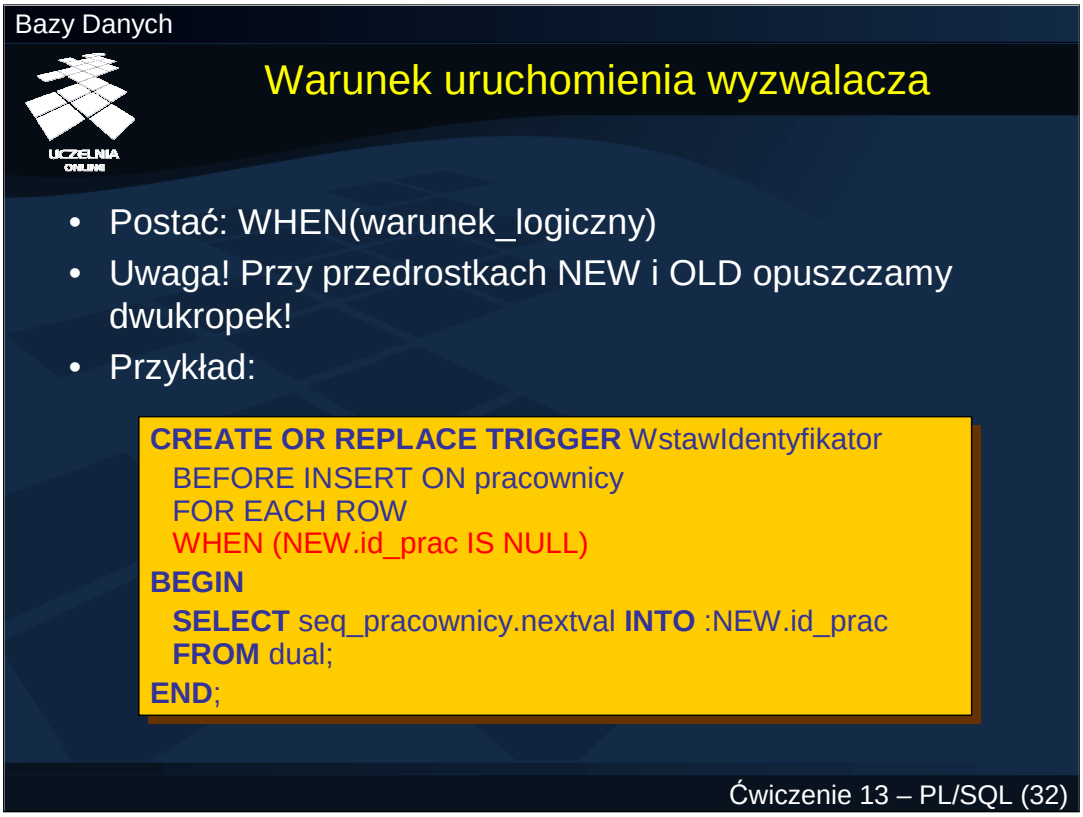


W wyzwalaczu wierszowym mamy możliwość bezpośredniego odwołania do wartości atrybutów rekordu relacji lub perspektywy, dla którego wyzwalacz został uruchomiony. Co więcej, mamy dostęp do wartości atrybutu zarówno sprzed wykonania polecenia, używamy wówczas przedrostka „:OLD”, jak i po wykonaniu polecenia, wówczas stosujemy przedrostek „:NEW”.

Należy pamiętać, że nie we wszystkich sytuacjach odczyt obu wartości atrybutów rekordu ma sens. W przypadku wyzwalacza wierszowego dla polecenia INSERT wartości atrybutów sprzed wykonania polecenia (przedrostek :OLD) są puste – rekord jeszcze nie istnieje. Z kolei w przypadku polecenia DELETE wartości atrybutów po wykonaniu polecenia (przedrostek :NEW) są puste – rekord już nie istnieje. Jedynie w wyzwalaczu dla polecenia UPDATE sensowne jest odczytywanie wartości zarówno sprzed wykonania polecenia (rekord jeszcze nie zmodyfikowany) jak i po jego wykonaniu (rekord po modyfikacji).



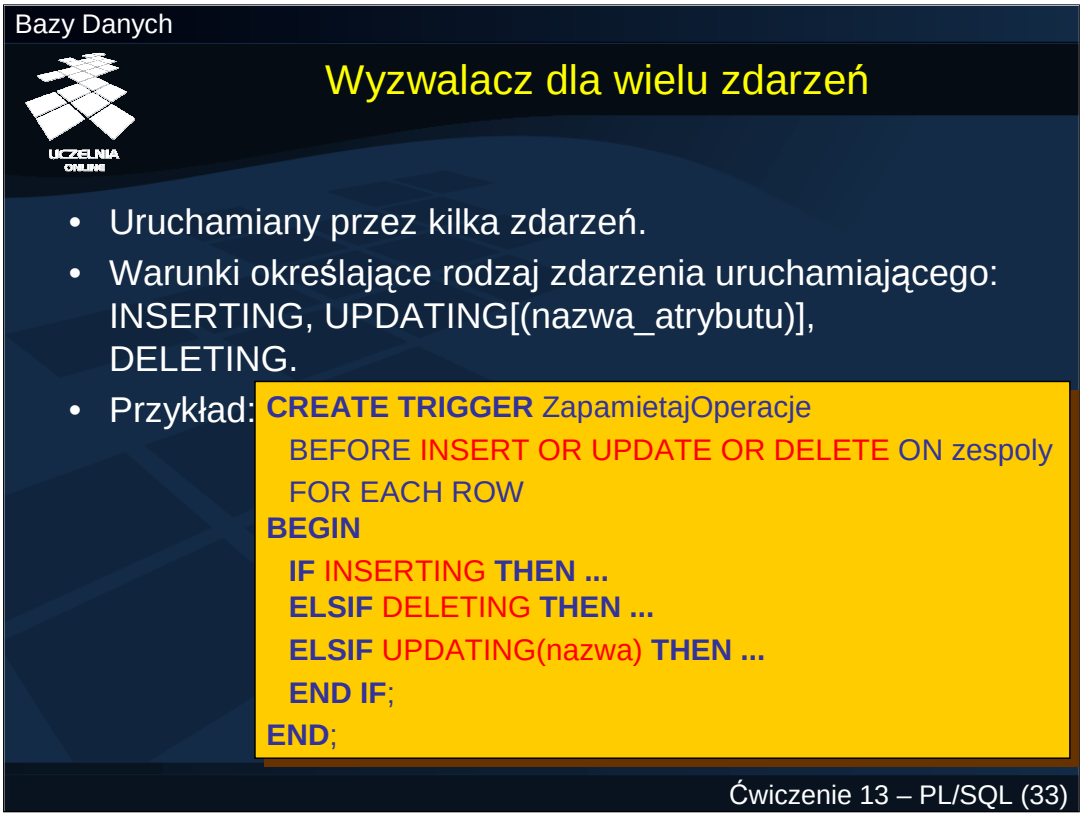
Bieżący slajd przykład wyzwalacza wierszowego o nazwie WstawIdentyfikator, wykonywanego przed poleceniem INSERT na relacji PRACOWNICY. W ciele wyzwalacza zostaje sprawdzona wartość atrybutu ID_PRAC, jaka zostanie umieszczona w rekordzie po operacji INSERT. Jeśli jest ona pusta (oznacza to, że użytkownik nie podał wartość dla ID_PRAC w poleceniu INSERT), wyzwalacz pobiera wartość z sekwencji o nazwie SEQ_PRACOWNICY i wstawia ją do atrybutu ID_PRAC. Dzięki temu operacja INSERT będzie mogła być zrealizowana – atrybut ID_PRAC jest atrybutem obowiązkowym.



Opcjonalnym elementem definicji wyzwalacza jest warunek, sprawdzany przy uruchomieniu wyzwalacza. Warunek ten umieszcza się w nawiasach po klauzuli WHEN. Jeśli warunek został zdefiniowany, po wystąpieniu zdarzenia uruchamiającego wyzwalacz warunek jest sprawdzany, jeśli jest on prawdziwy, wyzwalacz zostaje uruchomiony, w przeciwnym wypadku uruchomienie wyzwalacza zostaje zaniechane.

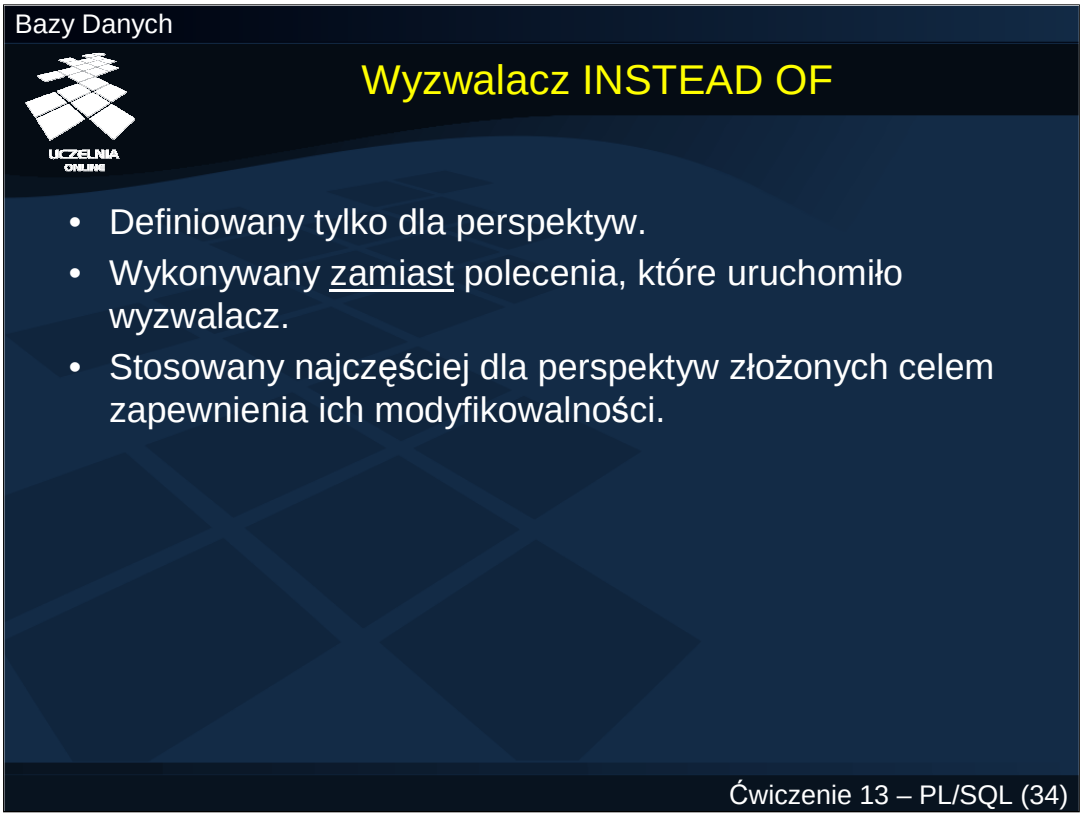
Uwaga! Stosując w warunku w klauzuli WHEN przedrostki :NEW i :OLD pomijamy dwukropek!

Zaprezentowany na slajdzie przykład jest modyfikacją wyzwalacza z poprzedniego slajdu. Warunek w instrukcji warunkowej IF zastąpiono warunkiem w klauzuli WHEN. Wyzwalacz wykona się tylko wtedy, gdy wartość atrybutu ID_PRAC w rekordzie wstawianym do relacji PRACOWNICY jest pusta.

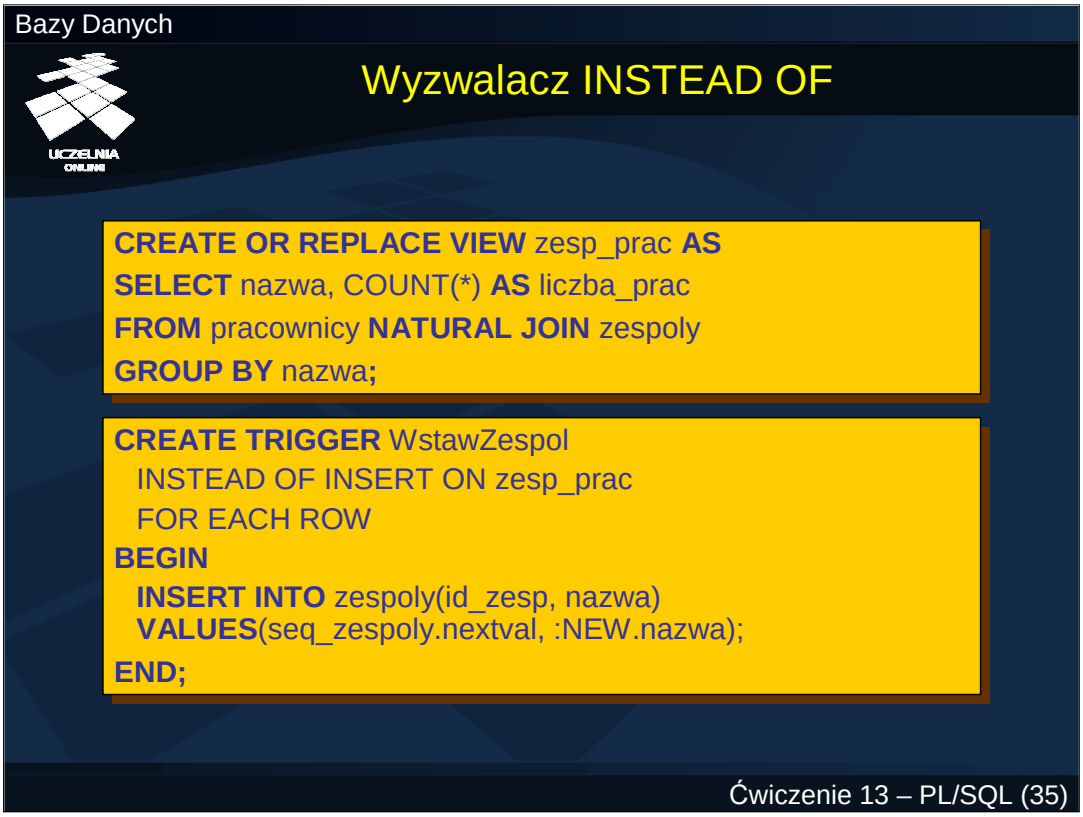


Istnieje możliwość zdefiniowania wyzwalacza, uruchamianego przez kilka różnych zdarzeń na tej samej relacji lub perspektywie. Realizuje się to łącząc poszczególne zdarzenia spójnikiem logicznym OR. W ciele takiego wyzwalacza można selekcjonować kod, który ma być wykonany w przypadku wystąpienia określonego zdarzenia. Używa się w tym celu zmiennych logiczny INSERTING, DELETING i UPDATING, które przyjmują wartość prawdy jeśli zdarzeniem uruchamiającym wyzwalacz jest odpowiednio zdarzenie INSERT, DELETE bądź UPDATE. Dodatkowo można zastosować konstrukcję „UPDATING(nazwa_atrybutu)”, która jest prawdą jeśli wyzwalacz uruchomiło uaktualnienie wartości atrybutu o podanej nazwie.

Zaprezentowany na bieżącym slajdzie wyzwalacz o nazwie ZapamietajOperacje jest wywoływany dla operacji INSERT, UPDATE lub DELETE na relacji ZESPOLY.



Omówimy teraz ostatni rodzaj wyzwalacza, który może być definiowany wyłącznie dla perspektyw. Jest to wyzwalacz INSTEAD OF i jest on wykonywany zamiast operacji, która odpowiada za zdarzenie uruchamiające wyzwalacz. Wyzwalacz INSTEAD OF stosuje się najczęściej w celu umożliwienia modyfikacji danych perspektyw złożonych, które normalnie nie akceptują poleceń DML. W takim przypadku operacja DML zostaje zastąpiona wykonaniem wyzwalacza, który modyfikuje dane relacji bazowych perspektywy zamiast danych udostępnianych bezpośrednio przez perspektywę. Następny slajd prezentuje przykład takiego zastosowania.



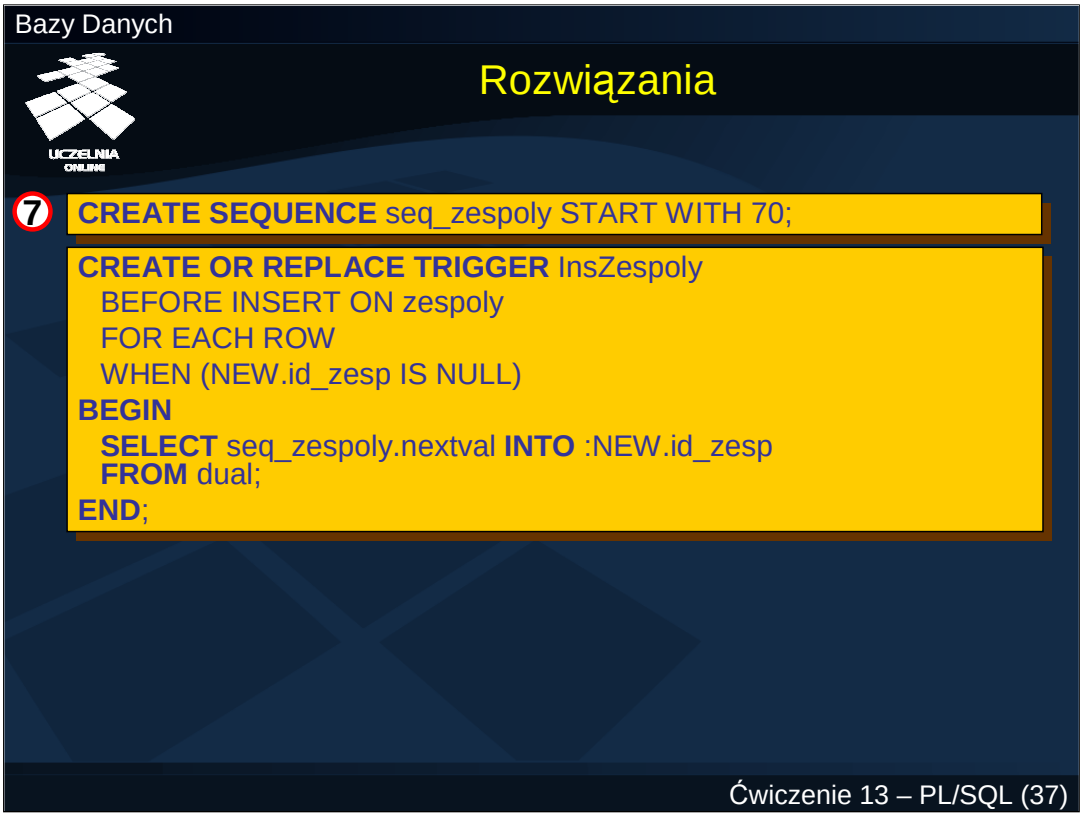
Pierwsze polecenie tworzy perspektywę o nazwie ZESP_PRAC, która wykonuje połączenie naturalne danych relacji PRACOWNICY i ZESPOLY, a następnie grupuje wynik połączenia ze względu na wartość atrybutu NAZWA. Jak widzimy, jest to perspektywa złożona i nie akceptuje ona żadnych poleceń modyfikacji danych.

Dla perspektywy zdefiniowano wyzwalacz wierszowy o nazwie WstawZespol, który będzie wykonywany zamiast operacji INSERT, kierowanych do perspektywy. Ciało wyzwalacza wstawia do relacji ZESPOLY rekord.

Zadania

7. Napisz wyzwalacz, który będzie automatycznie przyznawał kolejne identyfikatory nowym zespołom (utwórz i wykorzystaj sekwencję). Przetestuj działanie wyzwalacza.
8. Stwórz relację HISTORIA o schemacie (*id_prac, placa_pod, etat, zespol*, data). Napisz wyzwalacz, który po każdej modyfikacji płacy podstawowej, etatu lub zespołu w relacji PRACOWNICY będzie wpisywał wartości historyczne do relacji HISTORIA. Atrybut *data* będzie przechowywała datę modyfikacji.

Na bieżącym slajdzie zamieszczono dwa zadania, utrwalające zaprezentowane wcześniej informacje dotyczące procedur wyzwalanych.

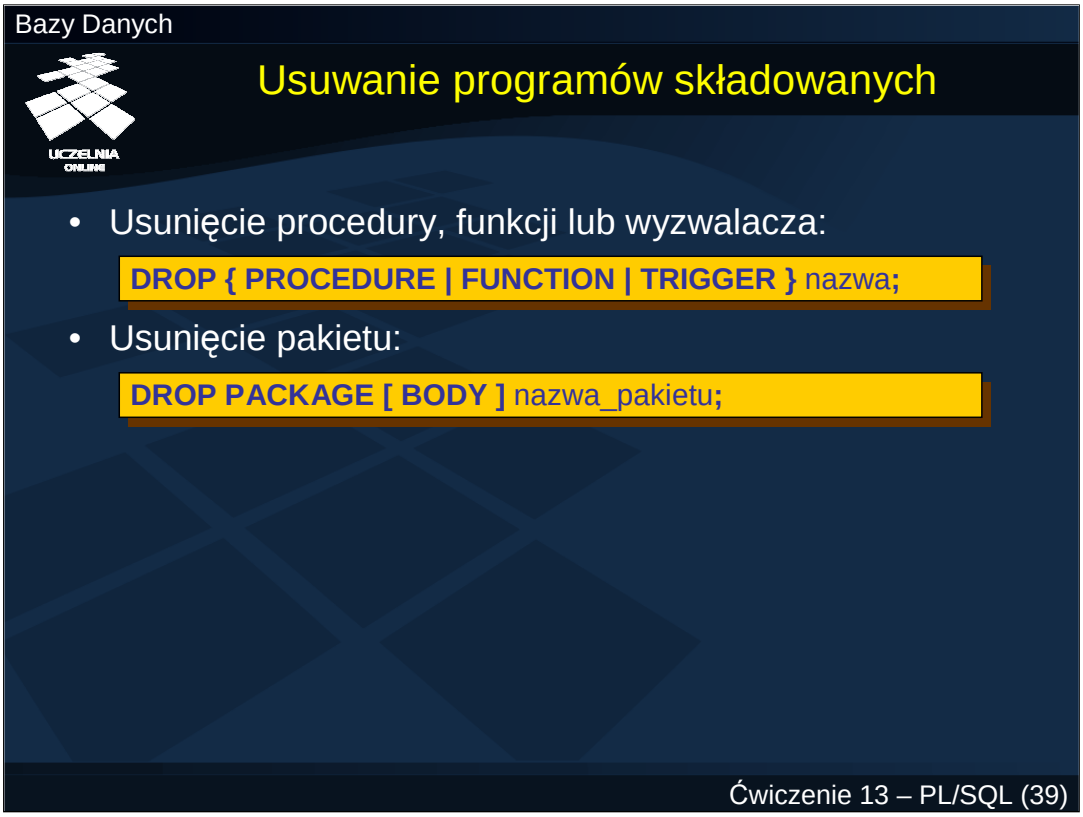


Bieżący slajd przedstawia rozwiązanie zadania (7), którego treść zacytowano poniżej.

(7) Napisz wyzwalacz, który będzie automatycznie przyznawał kolejne identyfikatory nowym zespołom (utwórz i wykorzystaj sekwencję). Przetestuj działanie wyzwalacza.

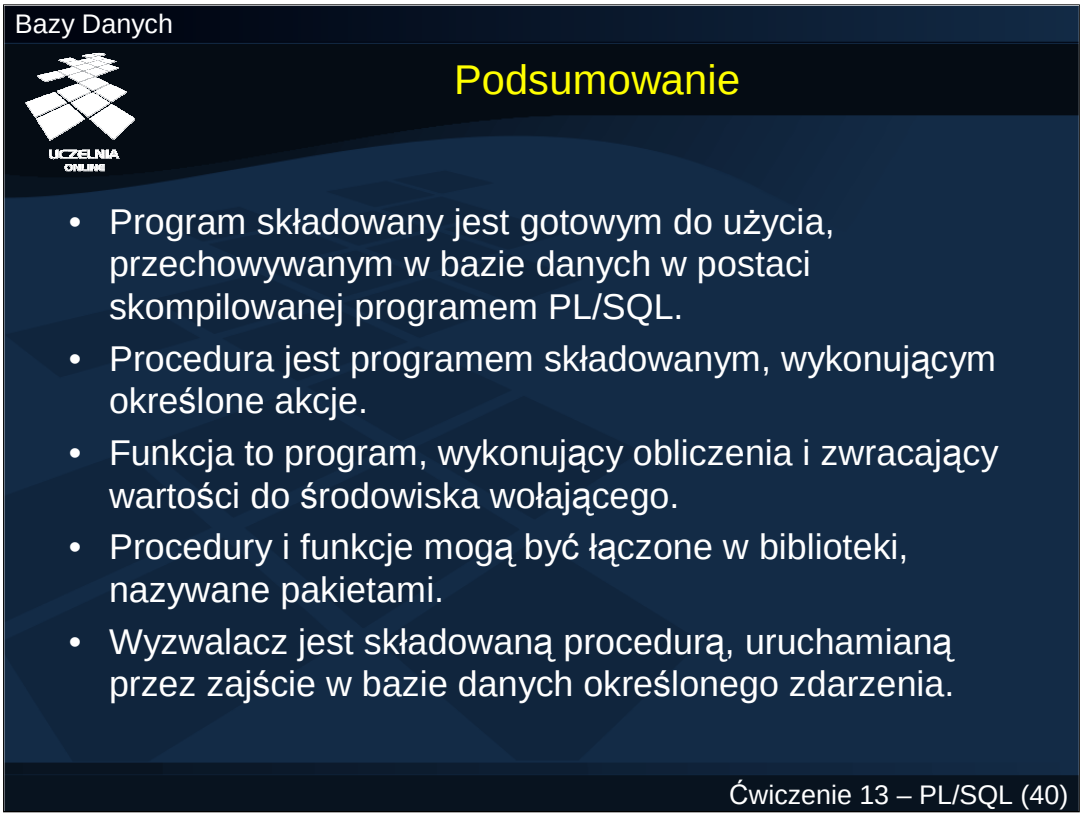
Bieżący slajd przedstawia rozwiązanie zadania (8), którego treść zacytowano poniżej.

(8) Stwórz relację HISTORIA o schemacie (ID_PRAC, PLACA_POD, ETAT, ZESPOL, DATA). Napisz wyzwalacz, który po każdej modyfikacji płacy podstawowej, etatu lub zespołu w relacji PRACOWNICY będzie wpisywał wartości historyczne do relacji HISTORIA. Atrybut DATA będzie przechowywała datę modyfikacji.



Na bieżącym slajdzie zaprezentowano polecenia usuwające z bazy danych składowane programy PL/SQL.

Pierwsze polecenie usuwa procedurę, funkcję bądź wyzwalacz z bazy danych. Drugie polecenie służy do usuwania całego pakietu (DROP PACKAGE) bądź tylko jego ciała (DROP PACKAGE BODY).



W zakończonym ćwiczeniu zaprezentowano koncepcję programów składowanych. Są to procedury, funkcje i pakiety, trwale zapisane w bazie danych w postaci skompilowanej. Procedura wykonuje w bazie danych określone akcje. Funkcja wykonuje obliczenia i zwraca wartość do środowiska wołającego. Pakiet jest biblioteką procedur i funkcji.

Procedura wyzwalana, nazywana również wyzwalaczem, jest procedurą składowaną, uruchamianą jednak nie przez użytkownika, ale przez zajście w bazie danych określonego zdarzenia, np. wstawienia rekordu do określonej relacji, utworzenia nowej perspektywy we wskazanym schemacie, przyłączenia się użytkownika do bazy danych.

Każde z omówionych w ćwiczeniu zagadnień zostało utrwalone przez serię zadań.