Laboratorium 3

Bazy Danych 2

mgr. inż. Aleksander Wojtowicz

Wyjątek to błąd, który występuje podczas wykonywania programu.

Jest sytuacją, która nie jest częścią normalnego działania programu.

Może to być błąd wynikający z czegoś tak prostego, jak dzielenie przez zero lub coś bardziej skomplikowanego, jak brak dostępu do bazy danych.

Obsługa wyjątków jest kluczowym elementem tworzenia niezawodnego i bezpiecznego kodu.

Bez odpowiedniej obsługi wyjątków, błąd może spowodować zatrzymanie działania programu lub nawet utratę danych.

Dzięki obsłudze wyjątków, programista może zdefiniować, co powinno się stać, gdy wystąpi określony błąd, co pozwala na łagodne zakończenie programu lub naprawienie błędu.

W PL/SQL mamy do czynienia z **dwoma** głównymi **typami wyjątków**: predefiniowanymi i niestandardowymi.

- **Predefiniowane wyjątki** to te, które są już zdefiniowane w PL/SQL, takie jak `NO_DATA_FOUND` czy `ZERO_DIVIDE`.
- **Niestandardowe wyjątki** to te, które definiuje programista w celu obsługi specyficznych dla aplikacji warunków błędów.

Oto przykład obsługi wyjątku w PL/SQL:

```
BEGIN
   -- blok kodu, który może spowodować błąd
EXCEPTION
   WHEN NO_DATA_FOUND THEN
   -- instrukcje do wykonania, gdy wystąpi wyjątek NO_DATA_FOUND
END;
```

W tym przykładzie, jeśli blok kodu spowoduje wyjątek NO_DATA_FOUND, zostaną wykonane instrukcje zdefiniowane w sekcji EXCEPTION.

Tabela wykorzystywana do przykładów:

```
CREATE TABLE pracownicy (
    id_pracownika  NUMBER PRIMARY KEY,
    imie           VARCHAR2(50),
    nazwisko       VARCHAR2(50),
    pensja         NUMBER
);

INSERT INTO pracownicy VALUES (1, 'Jan', 'Kowalski', 50000);
INSERT INTO pracownicy VALUES (2, 'Anna', 'Nowak', 60000);
INSERT INTO pracownicy VALUES (3, 'Piotr', 'Wiśniewski', 55000);
```

```
BEGIN
   DECLARE
      v_pensja NUMBER;
   BEGIN
      SELECT pensja INTO v_pensja
      FROM pracownicy
      WHERE id_pracownika = 999; -- Identyfikator, który nie
istnieje
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         -- Instrukcje do wykonania, gdy wystąpi wyjątek
NO_DATA_FOUND
         DBMS_OUTPUT.PUT_LINE('Nie znaleziono danych dla
podanego identyfikatora pracownika.');
   END;
END;
```

```
Nie znaleziono danych dla podanego identyfikatora pracownika.
```

1. Tworzymy zmienną **`v_pensja`**, która będzie przechowywać wynik zapytania.
2. Wykonujemy zapytanie do tabeli pracownicy, próbując pobrać pensję pracownika o nieistniejącym identyfikatorze (**999**).
3. Jeśli zapytanie nie zwróci wyników *(czyli wystąpi wyjątek **NO_DATA_FOUND**)*, wypisujemy odpowiedni komunikat.

Predefiniowane wyjątki to te, które są już zdefiniowane w PL/SQL.

Poniżej znajdują się przykłady:

1. NO_DATA_FOUND: Instrukcja SELECT INTO nie zwraca żadnych danych.
2. TOO_MANY_ROWS: Instrukcja SELECT INTO zwraca więcej niż jeden wiersz.
3. ZERO_DIVIDE: Występuje dzielenie przez zero.
4. VALUE_ERROR: Operacja arytmetyczna, konwersja lub przypisanie zwraca wynik poza zakresem.

NO_DATA_FOUND:

```
DECLARE
    pensja_pracownika NUMBER;
BEGIN
    SELECT pensja INTO pensja_pracownika FROM pracownicy WHERE
id_pracownika = 10;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Nie znaleziono pracownika o
podanym ID.');
END;
```

```
Nie znaleziono pracownika o podanym ID.
```

TOO_MANY_ROWS:

```
DECLARE
    pensja_pracownika NUMBER;
BEGIN
    SELECT pensja INTO pensja_pracownika FROM pracownicy;
EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        dbms_output.put_line('Zapytanie zwróciło więcej niż
jeden wiersz.');
END;
```

```
Zapytanie zwróciło więcej niż jeden wiersz.
```

ZERO_DIVIDE:

```
DECLARE
    wynik NUMBER;
BEGIN
    wynik := 1 / 0;
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        dbms_output.put_line('Próba dzielenia przez zero.');
END;
```

```
Próba dzielenia przez zero.
```

VALUE_ERROR:

```
DECLARE
    liczba NUMBER;
BEGIN
    liczba := 'abc';
EXCEPTION
    WHEN VALUE_ERROR THEN
        dbms_output.put_line('Nieprawidłowa wartość liczby.');
END;
```

```
Nieprawidłowa wartość liczby.
```

Niestandardowe wyjątki to te, które definiuje programista w celu obsługi specyficznych dla aplikacji warunków błędów.

```
DECLARE
   -- Definicja niestandardowego wyjątku
   moj_wyjatek EXCEPTION;
BEGIN
   -- blok kodu, który może spowodować błąd
   RAISE moj_wyjatek;  -- zgłoszenie niestandardowego wyjątku
EXCEPTION
   WHEN moj_wyjatek THEN
      -- do wykonania, gdy wystąpi niestandardowy wyjątek
      DBMS_OUTPUT.PUT_LINE('Wystąpił niestandardowy wyjątek');
END;
```

W tym kodzie, `moj_wyjatek` *to niestandardowy wyjątek, który jest zgłaszany w bloku* `BEGIN`*. Gdy ten wyjątek jest zgłaszany, blok* `EXCEPTION` *go przechwytuje i wyświetla komunikat 'Wystąpił niestandardowy wyjątek'.*

Niestandardowe wyjątki są użyteczne, gdy chcemy obsłużyć specyficzne dla naszej aplikacji warunki błędów, które nie są obsługiwane przez predefiniowane wyjątki. Na przykład, możemy chcieć zgłosić niestandardowy wyjątek, gdy dane wejściowe nie spełniają określonych kryteriów.

Dobre praktyki obsługi wyjątków

1. **Gdy zgłaszamy wyjątek, dostarcz jak najwięcej informacji o błędzie.**

2. **Jeżeli nie ma pewności, jak obsłużyć wyjątek, lepiej go nie tłumić.**

   Nieobsłużony wyjątek spowoduje zatrzymanie programu, co jest lepsze niż kontynuowanie działania z błędem.

3. **Wyjątki są kosztowne pod względem wydajności, dlatego nie powinny być używane do sterowania przepływem programu.**

   Na przykład, zamiast zgłaszać wyjątek, gdy dane wejściowe nie spełniają określonych kryteriów, lepiej użyć instrukcji IF do sprawdzenia tych kryteriów.

4. **Podczas pisania kodu upewnij się, że przetestowano wszystkie możliwe wyjątki i że są one odpowiednio obsługiwane.**

   Można to zrobić, pisząc testy jednostkowe dla kodu.

Podsumowanie laboratorium

1. **Wyjątki** to błędy, które występują podczas wykonywania programu.
2. **Obsługa wyjątków** pozwala na łagodne zakończenie programu lub naprawienie błędu.
3. **Predefiniowane wyjątki** są już zdefiniowane w PL/SQL, takie jak `NO_DATA_FOUND` czy `ZERO_DIVIDE`.
4. **Niestandardowe wyjątki** są definiowane przez programistę w celu obsługi specyficznych dla aplikacji warunków błędów.
5. **Dobre praktyki** obsługi wyjątków obejmują dokładne zgłaszanie błędów, nietłumienie wyjątków, nieużywanie wyjątków do sterowania przepływem programu i testowanie obsługi wyjątków.

Zadania:

1. Napisz procedurę PL/SQL, która spróbuje przypisać nazwisko pracownika do zmiennej, ale dane są za długie na zmienną VARCHAR2 i obsługuje wyjątek wypisując komunikat.

2. Napisz procedurę PL/SQL, która spróbuje podzielić pensję pracownika przez zero i obsługuje wyjątek wypisując komunikat.

3. Napisz procedurę PL/SQL, która spróbuje przypisać imię i nazwisko pracownika o identyfikatorze, który nie istnieje, do zmiennych. Obsłuż wątek i wypisz komunikat.

4. Napisz procedurę PL/SQL, która spróbuje przypisać do zmiennej wynik operacji porównania dwóch ciągów znaków i liczby. Obsłuż wątek i wypisz komunikat.

5. Napisz procedurę PL/SQL, która spróbuje przypisać pensję pracownika do zmiennej, ale pensja jest mniejsza niż zero. Obsłuż wątek i wypisz komunikat.

6. Napisz procedurę PL/SQL, która spróbuje podnieść pensję pracownika o wartość niebędącą liczbą. Obsłuż wątek i wypisz komunikat.

7. Napisz procedurę PL/SQL, która spróbuje przypisać do zmiennej wynik dzielenia dwóch ciągów znaków. Obsłuż wątek i wypisz komunikat.