WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Bazy danych
Laboratorium

Zapytania DDL SQL Oracle 2

Stanislau Antanovich
nr. indeksu: 173590
gr. lab: L04

Spis treści

Spis rysunków

Spis poleceń

1 Wprowadzenie

1.1 Cel ćwiczenia

1.2 Przygotowanie

2 Realizacja

1. Tworzenie i wywołanie procedury **dodaj_etat** pozwalającej na dodanie nowego etatu

```
CREATE PROCEDURE dodaj_etat(
  p_id IN NUMBER,
  p_etat IN VARCHAR
) AS
BEGIN
  INSERT INTO ETATY (ID_ETATU, ETAT) VALUES (p_id, p_etat);
  COMMIT;
END dodaj_etat;

EXEC dodaj_etat(673, 'EMPLOYEE')
```

Polecenie 1. *Tworzenie i wywołanie procedury* ***dodaj_ etat***

	ID_ETATU	ETAT
1	667	CLERK
2	668	STAFF
3	669	ANALYST
4	670	SALESPERSON
5	671	MANAGER
6	672	PRESIDENT
7	673	EMPLOYEE

Rysunek 1: *Tworzenie i wywołanie procedury* ***dodaj_ etat***

2. Tworzenie i wywołanie procedury **usun_pracownika** pozwalającej na usunięcie pracownika na podstawie podanego id(parametr wejściowy)

```
CREATE PROCEDURE usun_pracownika(
  id_pracownika IN NUMBER;
) AS
BEGIN
  DELETE FROM PRACOWNICY WHERE ID_PRACOWNIKA = id_pracownika;
END usun_pracownika;

EXEC usun_pracownika(7369);
```

Polecenie 2. *Tworzenie i wywołanie procedury* ***usun_ pracownika***

2	7499	ALLEN	KEVIN	J	670	7698	85/02/20	1600	300	30
3	7505	DOYLE	JEAN	K	671	7839	85/04/04	2850	(null)	13
4	7506	DENNIS	LYNN	S	671	7839	85/05/15	2750	(null)	23
5	7507	BAKER	LESLIE	D	671	7839	85/06/10	2200	(null)	14
6	7521	WARD	CYNTHIA	D	670	7698	85/02/22	1250	500	30
7	7555	PETERS	DANIEL	T	670	7505	85/03/31	1250	300	13
8	7557	SHAW	KAREN	P	670	7505	85/04/02	1250	1200	13
9	7560	DUNCAN	SARAH	S	670	7506	85/05/31	1250	(null)	23
10	7564	LANGE	GREGORY	J	670	7506	85/06/01	1250	300	23
11	7566	JONES	TERRY	M	671	7839	85/04/02	2975	(null)	20
12	7569	ALBERTS	CHRIS	L	671	7839	85/04/06	3000	(null)	12
13	7600	PORTER	RAYMOND	Y	670	7505	85/04/15	1250	900	13
14	7609	LEWIS	RICHARD	M	668	7507	85/04/16	1800	(null)	24
15	7654	MARTIN	KENNETH	J	670	7698	85/09/28	1250	1400	30
16	7676	SOMMERS	DENISE	D	668	7507	85/04/19	1850	(null)	34
17	7698	BLAKE	MARION	S	671	7839	85/05/01	2850	(null)	30
18	7782	CLARK	CAROL	F	671	7839	85/06/09	2450	(null)	10
19	7788	SCOTT	DONALD	T	669	7566	86/12/09	3000	(null)	20
20	7789	WEST	LIVIA	N	670	7506	85/04/04	1500	1000	23
21	7799	FISHER	MATTHEW	G	669	7569	86/12/12	3000	(null)	12
22	7820	ROSS	PAUL	S	670	7505	85/06/01	1300	800	43
23	7839	KING	FRANCIS	A	672	(null)	85/11/17	5000	(null)	10
24	7844	TURNER	MARY	A	670	7698	85/09/08	1500	0	30
25	7876	ADAMS	DIANE	G	667	7788	87/01/12	1100	(null)	20
26	7900	JAMES	FRED	S	667	7698	85/12/03	950	(null)	30
27	7902	FORD	JENNIFER	D	669	7566	85/12/03	3000	(null)	20
28	7916	ROBERTS	GRACE	M	669	7569	87/01/04	2875	(null)	12
29	7919	DOUGLAS	MICHAEL	A	667	7799	87/01/04	800	(null)	12
30	7934	MILLER	BARBARA	M	667	7782	86/01/23	1300	(null)	10
31	7950	JENSEN	ALICE	B	667	7505	87/01/15	750	(null)	13
32	7954	MURRAY	JAMES	T	667	7506	87/01/16	750	(null)	23

Rysunek 2: *Tworzenie i wywołanie procedury* ***usun_pracownika***

3. Tworzenie i wywołanie proceduty **edytuj_pracownika** pozwalającej na edytowanie kolumn pracownika na podstawie podanych parametrów

```
CREATE PROCEDURE edytuj_pracownika(
  id_pracownika IN NUMBER;
  nazwisko IN VARCHAR;
  imie IN VARCHAR;
  pensja IN NUMBER;
  prowizja IN NUMBER;
) AS
BEGIN
  UPDATE PRACOWNICY SET NAZWISKO = nazwisko, IMIE = imie, PENSJA = pensja, PROWIZJA =
      prowizja WHERE ID_PRACOWNIKA = id_pracownika;
END edytuj_pracownika

EXEC edytuj_pracownika(7499, 'ANTANOVICH', 'STANISLAU', 5000, 0);
```

Polecenie 3. *Tworzenie i wywołanie procedury* ***edytuj_pracownika***

ID_PRACOWNIKA	NAZWISKO	IMIE	DRUGIE_IMIE	ID_ETATU	ID_SZEFA	DATA_ZATRUDNIENIA	PENSJA	PROWIZJA	ID_WYDZIALU
7499	ANTANOVICH	STANISLAU	J	670	7698	85/02/20	5000	0	30

Rysunek 3: *Tworzenie i wywołanie procedury* ***edytuj_pracownika***

4. Modyfikacja procedury **dodaj_etat** w taki sposób, aby uniknąć wystąpienia duplikującego się rekordu.

```
CREATE PROCEDURE dodaj_etat(
  id_etatu IN NUMBER,
      etat IN VARCHAR
) AS etat_count NUMBER;
BEGIN
```

```
  SELECT COUNT(*) INTO etat_count FROM ETATY WHERE ID_ETATU = id_etatu;
  IF etat_count = 0 THEN
    INSERT INTO ETATY (ID_ETATU, ETAT)
    VALUES (id_etatu, etat);
    COMMIT;
  END IF;
END dodaj_etat;
/

EXEC dodaj_etat(666, 'NOWY ETAT');
```

Polecenie 4. *Modyfikacja procedury* ***dodaj_etat***

	ID_ETATU	ETAT
1	667	CLERK
2	668	STAFF
3	669	ANALYST
4	670	SALESPERSON
5	671	MANAGER
6	672	PRESIDENT
7	673	EMPLOYEE
8	666	NOWY_ETAT

Rysunek 4: *Modyfikacja procedury* ***dodaj_etat***

5. Zaproponowana funkcja **srednia_pensja**, która zwraca średnią pensję wszystkich pracowników

```
CREATE FUNCTION srednia_pensja RETURN NUMBER
IS
  avg_pensja NUMBER;
BEGIN
  SELECT AVG(PENSJA) INTO avg_pensja FROM PRACOWNICY;
  RETURN avg_pensja;
END srednia_pensja;
/

SELECT srednia_pensja() FROM DUAL;
```

Polecenie 5. *Zaproponowana funckja* ***srednia_pensja***

```
FUNCTION srednia_pensja compiled
SREDNIA_PENSJA()
----------------
        2003,125
```

Rysunek 5: *Zaproponowana funkcja* ***srednia_pensja***

6. Zaproponowana funkcja **zmien_prowizje**, która umożliwi na podstawie id pracownika zmienić jego prowizję. W przypadku wprowadzonej wartości poniżej zera funkcja ma zgłosić odpowiedni wyjątek

```
CREATE FUNCTION zmien_prowizje(
  id_pracownika IN NUMBER,
  prowizja IN NUMBER
) RETURN VARCHAR
IS
BEGIN
  IF prowizja < 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Prowizja nie moze byc ponizej zera')
  ELSE
    UPDATE PRACOWNICY SET PROWIZJA = prowizja WHERE ID_PRACOWNIKA = id_pracownika;
    COMMIT;
    RETURN 'Prowizja zostala zmieniona'
  END IF;
```

```
END zmien_prowizje;

DBMS_OUTPUT.PUT_LINE(zmien_prowizje(7499, 2000));
```

Polecenie 6. *Zaproponowana funkcja* ***zmien_prowizje***

ID_PRACOWNIKA	NAZWISKO	IMIE	DRUGIE_IMIE	ID_ETATU	ID_SZEFA	DATA_ZATRUDNIENIA	PENSJA	PROWIZJA	ID_WYDZIALU
7499	ANTANOVICH	STANISLAU	J	670	7698	85/02/20	5000	2000	30

Rysunek 6: *Zaproponowana funkcja* ***zmien_prowizje***

7. Zaproponowana funkcja **ilosc_pracownicy_z_pensja** umożliwiająca zwrócenie ilości pracowników, których pensja mieści się w podanym przedziale <min;max>

```
CREATE FUNCTION ilosc_pracownicy_z_pensja(
  min_pensja IN NUMBER,
  max_pensja IN NUMBER
) RETURN NUMBER
IS
  e_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO e_count FROM PRACOWNICY WHERE PENSJA BETWEEN min_pensja AND
      max_pensja;
  RETURN e_count;
END ilosc_pracownicy_z_pensja;
/

SELECT ilosc_pracownicy_z_pensja(1000,2500) FROM DUAL;
```

Polecenie 7. *Zaproponowana funkcja* ***ilosc_pracownicy_z_pensja***

ILOSC_PRACOWNICY_Z_PENSJA(1000,2500)
16

Rysunek 7: *Zaproponowana funkcja* ***ilosc_pracownicy_z_pensja***

8. Zaproponowana funkcja **zwieksz_pensje**, która na podstawie id pracownika zmienia wartość kolumny pensja zgodnie z zasadami:
   - jeżeli pracownik nie ma zwierzchnika jego pensja nie ulega zmianie
   - jeżeli pracownik ma zwierzchnika, ale ma prowizje, pensja zwiększa się o wartość 100
   - jeżeli pracownik ma zwierzchnika, ale nie ma prowizji, pensja zwiększa się o 10%
   - funkcja zwraca nową pensję jako wartość

```
CREATE FUNCTION zwieksz_pensje(
  id_pracownika IN NUMBER
) RETURN NUMBER
IS
  pensja NUMBER;
  z_pensja NUMBER;
  prowizja NUMBER;
BEGIN
  SELECT PENSJA, PROWIZJA INTO pensja, prowizja FROM PRACOWNICY WHERE ID_PRACOWNIKA =
      id_pracownika;
  IF prowizja IS NULL THEN
  SELECT PENSJA INTO z_pensja FROM PRACOWNICY WHERE ID_PRACOWNIKA = (SELECT ID_SZEFA
      FROM PRACOWNICY WHERE ID_PRACOWNIKA = id_pracownika) FETCH FIRST ROW ONLY;
```

```
  pensja := pensja * 1.1;
    IF z_pensja IS NOT NULL THEN
      pensja := pensja + 100;
    END IF;
  ELSE
    pensja := pensja + 100;
  END IF;

  UPDATE PRACOWNICY SET PENSJA = pensja WHERE ID_PRACOWNIKA = id_pracownika;
  COMMIT;
  RETURN pensja;
END zwieksz_pensje;
/

SELECT zwieksz_pensje(7499) FROM DUAL;
```

Polecenie 8. *Zaproponowana funkcja* ***zwieksz_ pensje***

ID_PRACOWNIKA	NAZWISKO	IMIE	DRUGIE_IMIE	ID_ETATU	ID_SZEFA	DATA_ZATRUDNIENIA	PENSJA	PROWIZJA	ID_WYDZIALU
7499	ANTANOVICH	STANISLAU	J	670	7698	85/02/20	5100	2000	30

Rysunek 8: *Zaproponowana funkcja* ***zwieksz_ pensje***

9. Wywołanie komendy usunięcia dowolnej procedury i funkcji

```
DROP PROCEDURE dodaj_etat;
```

Polecenie 9. *Wywołanie komendy usunięcia dowolnej procedury i funkcji*

```
procedure DODAJ_ETAT dropped.
```

Rysunek 9: *Wywołanie komendy usunięcia dowolnej procedury i funkcji*

10. Zaproponowany typ pozwalający przechowywać dane obiektu **ETAT** oraz tabelę etatów

```
CREATE TYPE typ_etat AS OBJECT (
  id_etatu NUMBER,
  etat VARCHAR(255)
);
/

CREATE TYPE tabela_etatow AS TABLE OF typ_etat;
/
```

Polecenie 10. *Zaproponowany typ pozwalający przechowywać dane obiektu* ***ETAT*** *oraz tabelę etatów*

```
TYPE typ_etat compiled
TYPE tabela_etatow compiled
```

Rysunek 10: *Zaproponowany typ pozwalający przechowywać dane obiektu* ***ETAT*** *oraz tabelę etatów*

3 Wnioski

W wyniku przeprowadzonych działań udało mi się efektywnie wykorzystać narzędzie SQL Developer do zarządzania bazą danych. W ramach zadania, stworzyłem i wywołałem szereg procedur oraz funkcji, które umożliwiły mi dodawanie, usuwanie i edycję danych pracowników oraz etatów zgodnie z określonymi regułami.

Zaimplementowałem procedury takie jak dodaj_etat, usuń_pracownika oraz edytuj_pracownika, które pozwoliły na manipulację danymi w sposób kontrolowany i bezpieczny. Modyfikując procedurę dodaj_etat, zabezpieczyłem ją przed dodaniem duplikujących się rekordów, co przyczyniło się do poprawy integralności danych.

Dodatkowo, zaproponowałem funkcje takie jak srednia_pensja, zmien_prowizje, ilosc_pracownicy_z_pensja oraz zwieksz_pensje, które umożliwiły mi wykonywanie różnorodnych operacji na danych pracowników zgodnie z założeniami biznesowymi.

Pracując nad rozwojem funkcjonalności, wprowadziłem także definicje typu typ_etat oraz tabela_etat_typ, co umożliwiło mi przechowywanie i manipulowanie danymi dotyczącymi etatów w bardziej strukturalny sposób.

W rezultacie przeprowadzonych działań, zdobyłem nowe umiejętności w obszarze zarządzania bazami danych oraz programowania w języku SQL. Wdrażając te rozwiązania, podjąłem skuteczne kroki w kierunku optymalizacji procesów związanych z zarządzaniem danymi w organizacji.