Materiały do zajęć laboratoryjnych
z przedmiotu:

Systemy baz danych

Opracowanie:

dr inż. Piotr Muryjas

Cel zajęć:

Zapoznanie studentów z instrukcjami języka SQL, umożliwiającymi:

- zarządzania obiektami bazy danych, w szczególności strukturami danych,
- ekstrakcję i przetwarzanie danych gromadzonych we współczesnych bazach danych.

Efekty kształcenia:

Nabycie umiejętności posługiwania się językiem SQL w zakresie:

- projektowania struktury danych w bazie danych z wykorzystaniem współczesnych modeli danych,
- projektowania struktur obiektów pomocniczych w bazie danych, wykorzystywanych w eksploracji i manipulowaniu danymi,
- eksploracji i przetwarzania danych z wykorzystaniem strukturalnego języka zapytań,
- wstawiania, modyfikowania i usuwania danych,
- zaawansowanego przetwarzania danych zgromadzonych w bazie danych z wykorzystaniem funkcjonalności języków programowania.

Zakres tematyczny zajęć laboratoryjnych:

1. Selekcja danych z pojedynczego zbioru danych. Eliminowanie powtarzających się wartości. Sortowanie wyświetlanych danych. Warunkowe wybieranie danych. Grupowanie danych. Funkcje analityczne. Wybór grup danych według zadanego kryterium.
2. Wybór danych z wielu źródeł. Rodzaje złączeń w zapytaniach. Podzapytania w instrukcji wyboru. Podzapytania skalarne, wierszowe i skorelowane.
3. Metody definiowania i modyfikacji struktury tabel. Definiowanie reguł poprawności danych dla kolumn i tabeli. Określanie aktywności więzów integralności.
4. Perspektywy w eksploracji danych. Manipulowanie danymi z wykorzystaniem perspektyw – możliwości i ograniczenia.
5. Ogólna budowa bloku PL/SQL. Deklaracje stałych i zmiennych skalarnych oraz złożonych. Deklaracja typów złożonych.
6. Struktury kontrolne IF...THEN...ELSE, CASE. Struktury kontrolne typu LOOP, WHILE-LOOP, FOR-LOOP.
7. Kolekcje w PL/SQL. Rodzaj kolekcji. Operatory zbiorów działających na kolekcjach.
8. Kursory jako technika przetwarzania wielowierszowego zbioru danych. Modyfikacja danych w tabelach z wykorzystaniem kursorów. Zastosowanie zmiennych kursora w przetwarzaniu danych.
9. Obsługa wyjątków. Rodzaje wyjątków. Deklarowanie, zagnieżdżanie i wywoływanie wyjątków. Propagacja obsługi wyjątków.
10. Obiektowość w PL/SQL. Rodzaje obiektów. Typy obiektowe. Wykorzystanie obiektów w definicji tabel i w przetwarzaniu danych.
11. Procedury i funkcje PL/SQL – deklarowanie, sposoby przekazywania parametrów, wywoływanie. Przeciążanie podprogramów.
12. Funkcje i procedury składowane. Pakiety standardowe oraz użytkownika. Klasyfikacja pakietów. Struktura pakietu. Zarządzanie funkcjami, procedurami oraz pakietami w bazie danych.
13. Wyzwalacze bazy danych jako technika identyfikacji i obsługi zdarzeń na poziomie serwera bazy danych. Klasyfikacja wyzwalaczy (DML, DDL, database events). Ograniczenia związane z wykorzystaniem wyzwalaczy bazy danych.
14. Tworzenie dynamicznego kodu SQL w bloku PL/SQL. Techniki uruchamiania dynamicznego SQL.

Literatura podstawowa:

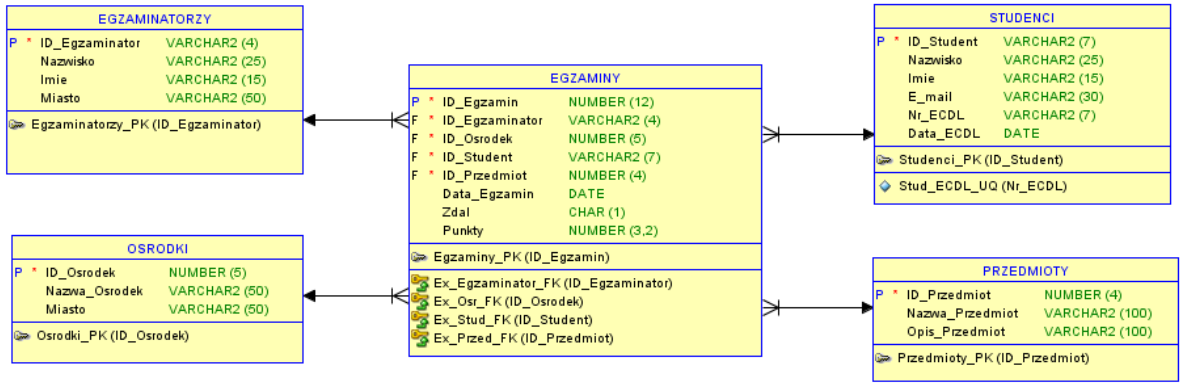
1. C.J. Date, Hugh Darwen: SQL - Omówienie standardu języka. WNT, 2008
2. Joe Celko: SQL. Zaawansowane techniki programowania. PWN, 2008
3. Danuta Mendrala, Marcin Szeliga: Praktyczny kurs SQL. Helion, 2008
4. Anthony Molinaro: SQL. Receptury. Helion, 2006
5. Bill Pribyl, Steven Feuerstein: Oracle PL/SQL. Wprowadzenie. Helion, 2002
6. Steven Feuerstein: Oracle PL/SQL Najlepsze praktyki. Mikom PWN, 2009
7. Michael McLaughlin: Oracle Database 11g. Programowanie w Języku PL/SQL. Helion 2009

Literatura uzupełniająca:

1. Oracle PL/SQL Language Reference: https://docs.oracle.com/en/database/oracle/oracle-database/20/lnpls/database-pl-sql-language-reference.pdf
2. Techonthenet: https://www.techonthenet.com/oracle/index.php

WPROWADZENIE

Realizacja zadań odbywać się będzie w oparciu o modelową bazę danych, której diagram związków encji przedstawia rysunek.



Baza danych składa się z następujących tabel:

- Studenci – zawiera informacje o osobach przystępujących do egzaminu,
- Egzaminatorzy – zawiera informacje o osobach przeprowadzających egzamin,
- Przedmioty – zawiera informacje o tematach przeprowadzanych egzaminów,
- Osrodki – zawiera informacje o miejscach przeprowadzania egzaminów,
- Egzaminy - zawiera informacje o procesie egzaminowania.

Założenia, na których oparto powyższy projekt bazy danych są następujące:

- każdy egzamin jest unikalnym faktem, identyfikowanym przez numer egzaminu;
- student zdaje egzaminy z poszczególnych przedmiotów;
- student może zdawać egzamin u każdego egzaminatora;
- każdy egzaminator może przeprowadzać egzamin z dowolnego przedmiotu;
- egzamin z przedmiotu może odbywać się w dowolnym ośrodku;
- wynikiem egzaminu nie jest ocena, lecz określa się wyłącznie fakt zdania;
- student może zdawać wielokrotnie egzamin z danego przedmiotu, lecz jest to już traktowane jako inny egzamin.

Temat 1.

Wybór danych z tabeli (instrukcja SELECT). Sortowanie danych (klauzula ORDER BY) i eliminowanie powtarzających się wartości (klauzula DISTINCT).

Przykłady treści zadań i ich rozwiązania

1. Wyświetlić podstawowe informacje o studentach (*Nazwisko*, *Imie*, *ID_Student*) posortowane malejąco wg nazwiska.

```
SELECT ID_Student, Nazwisko, Imie
FROM Studenci
ORDER BY Nazwisko DESC ;
```

2. Wyświetlić wszystkie dane o przedmiotach. Rezultat uporządkować wg nazwy przedmiotu.

```
SELECT *
FROM Przedmioty
ORDER BY Nazwa_Przedmiot ;
```

3. Wyświetlić podstawowe dane (*ID_Osrodek*, *Nazwa_Osrodek*, *Miasto*) o ośrodkach. Posortować wyświetlane dane wg identyfikatora (rosnąco) oraz miasta (malejąco). Kod pocztowy oraz nazwę miasta wyświetlić w jednej kolumnie, którą należy opisać aliasem *Miejsce*.

```
SELECT ID_Osrodek, Nazwa_Osrodek, Miasto AS Miejsce
FROM Osrodki
ORDER BY ID_Osrodek, Miasto DESC ;
```

4. Podać identyfikatory ośrodków, w których przeprowadzono już egzaminy. Uporządkować wyświetlane dane wg wartości identyfikatora.

```
SELECT DISTINCT ID_Osrodek
FROM Egzaminy
ORDER BY ID_Osrodek ;
```

5. Którzy studenci zdawali egzaminy u poszczególnych egzaminatorów. Podać identyfikator egzaminatora i identyfikator studenta. Uwzględnić tylko tych studentów, którzy przystapili już do egzaminu. Uporządkować wyświetlane dane wg wartości identyfikatora studenta oraz identyfikatora egzaminatora. Wynik ma być wyświetlany w następującej postaci: *Student o identyfikatorze ... zdawał egzamin u egzaminatora ...* Kolumnę wynikową nazwać *Informacja o egzaminie*.

```
SELECT DISTINCT ' Student o identyfikatorze ' || ID_Student || ' zdawał egzamin
u egzaminatora ' || ID_Egzaminator AS "Informacja o egzaminie"
FROM Egzaminy
ORDER BY 1 ;
```

6. Wyświetlić podstawowe dane (*Nazwisko, Imie, ID_Egzaminator*) o tych egzaminatorach, których nazwiska zaczynają się na literę M. Posortować wyświetlane dane wg miasta, w którym mieszkają egzaminatorzy.

```
SELECT Nazwisko, Imie, ID_Egzaminator
FROM egzaminatorzy
WHERE LOWER(Nazwisko) LIKE 'm%'
ORDER BY miasto ;
```

7. Podać identyfikatory przedmiotów, z których przeprowadzono egzaminy w ośrodkach o identyfikatorze 1 oraz 3. W odpowiedzi uwzględnić te przedmioty, z których egzamin odbył się przynajmniej w jednym z podanych ośrodków. Uporządkować rezultat wg ośrodka oraz przedmiotu.

```
SELECT DISTINCT ID_Osrodek, Id_przedmiot
FROM Egzaminy
WHERE ID_Osrodek IN (1, 3);
```

8. Podać nazwy miast, w których znajduje się ośrodek o nazwie CKMP. Uporządkować alfabetycznie wynik zapytania.

```
SELECT DISTINCT Miasto
FROM Osrodki
WHERE UPPER(Nazwa_Osrodek) = 'CKMP'
ORDER BY Miasto ;
```

9. Podać identyfikatory ośrodków, w których przeprowadzono egzaminy w okresie od 20 kwietnia 2012 do 20 kwietnia 2012. Dodatkowo wyświetlić datę tych egzaminów w postaci komunikatu: *Egzamin w ośrodku ... odbył się w dniu ....* Uporządkować wyświetlane informacje według ośrodka i daty egzaminu.

```
SELECT DISTINCT ID_Osrodek, 'Egzamin w ośrodku ' || ID_Osrodek || ' odbył się w
dniu: '  || Data_Egzamin
FROM Egzaminy
WHERE  Data_Egzamin BETWEEN '2012-04-20' AND '2012-04-20'
ORDER BY 1, 2 ;
```

10. Podać identyfikatory tych studentów, którzy w okresie od 01 stycznia 2010 do 31 grudnia 2010 zdali egzaminy. Uporządkować wyświetlane informacje według identyfikatora studenta.

```
SELECT ID_Student
FROM Egzaminy
WHERE  Data_Egzamin BETWEEN '2010-01-01' AND '2010-12-31' AND Zdal = 'Y'
ORDER BY ID_Student ;
```

Zadania do samodzielnego rozwiązania

11. Wyświetlić wszystkie dane o wszystkich ośrodkach. Uporządkować rosnąco wynik zapytania wg identyfikatora ośrodka oraz miasta, w którym znajduje się ośrodek.

12. W jakich miastach mieszkają studenci? Podać nazwę miasta oraz kod pocztowy. Wyświetlić dane wynikowe w jednej kolumnie, łącząc kod pocztowy z miastem. Kolumnę wynikową nazwać *Miasta studentów*. Posortować otrzymany rezultat wg nazwy miasta.

13. Którzy studenci przystąpili już do egzaminu? Podać ich identyfikator. Uporządkować rezultat według identyfikatora studenta.

14. Którzy egzaminatorzy przeprowadzili już egzaminy? Podać ich identyfikator. Uporządkować rezultat według identyfikatora egzaminatora.

15. W jakie dni przeprowadzono egzaminy? Przedstawić wynik w porządku malejącym.

16. W jakich miastach znajdują się ośrodki egzaminowania? Podać nazwy tych miast, uporządkowane alfabetycznie.

17. Podać informację o adresie email każdego egzaminatora. Wynik wyświetlić w następującej postaci: *Egzaminator ... ma następujący email: ...* . Nazwać kolumnę wynikową *Email egzaminatora*.

18. Dokonać identyfikacji sposobu określenia wyniku egzaminu. Informacja taka znajduje się w kolumnie *Zdal* w tabeli *Egzaminy*.

19. W jakich dniach przeprowadzono egzaminy w poszczególnych ośrodkach? Dla każdego ośrodka, opisanego przez jego identyfikator, podać datę przeprowadzonego w nim egzaminu. Uporządkować wyświetlane informacje wg ośrodka oraz daty egzaminu.

20. Kiedy poszczególni studenci zdawali swoje egzaminy? Podać identyfikator studenta oraz datę zdawania egzaminu. Wynik wyświetlić w nastepującej postaci: *Student o identyfikatorze ... zdawał egzamin w dniu ...* Kolumnę wynikową nazwać *Informacja o egzaminie studenta*. Uporządkować otrzymany wynik według identyfikatora studenta oraz daty zdawania egzaminu.

21. Którzy egzaminatorzy przeprowadzili egzaminy w poszczególnych ośrodkach? Dla każdego ośrodka, w którym odbył się egzamin podać identyfikator egzaminatora, prowadzącego ten egzamin. Uporządkować rezultat według identyfikatora ośrodka oraz identyfikatora egzaminatora.

22. Kiedy odbył się egzamin z poszczególnych przedmiotów w poszczególnych ośrodkach? Dla każdego ośrodka wskazać przedmioty, z których przeprowadzano egzamin i datę tego egzaminu. Dodatkowo uporządkować rezultat według daty zdawania egzaminu. Do identyfikacji ośrodka i przedmiotu należy posłużyć się ich identyfikatorami.

23. W których ośrodkach poszczególni egzaminatorzy egzaminowali poszczególnych studentów? Dla każdego egzaminatora wskazać studentów egzaminowanych przez niego oraz miejsce przeprowadzenia egzaminu. Do identyfikacji egzaminatora, studenta oraz ośrodka należy posłużyć się ich identyfikatorami. Uporządkować rezultat w taki sposób, aby można było zapewnić czytelność wyniku i możliwość spójnej i prawidłowej jego interpretacji.

24. Kiedy studenci zdawali egzaminy z poszczególnych przedmiotów? Uwzględnić tylko tych studentów, którzy przystąpili już do egzaminu. Dla każdego studenta należy wskazać przedmiot, który zdawał oraz określić datę egzaminu z tego przedmiotu. Do identyfikacji studenta oraz przedmiotu zastosować ich identyfikatory. Ponadto uporządkować malejąco wynik zapytania według daty egzaminu i zapewnić możliwość spójnej i poprawnej interpretacji otrzymanego wyniku.

Zadania do samodzielnego rozwiązania

25. Które ośrodki znajdują się w mieście o nazwie Lublin? Podać ich identyfikator, nazwę, ulicę i numer lokalu. Uporządkować wyświetlane informacje wg nazwy ośrodka.

26. Którzy studenci zdawali egzaminy w ośrodku o identyfikatorze 1? Podać identyfikator studenta. Uporządkować wyświetlane informacje wg identyfikatora studenta.

27. Którzy egzaminatorzy mają nazwiska zaczynające się na literę M oraz są z miasta o nazwie Lublin? Podać ich identyfikator, imię i Nazwisko. Uporządkować wyświetlane informacje wg nazwiska i Imienia egzaminatora.

28. Które przedmioty zostały zdane przez studenta '0000049' w ośrodku o identyfikatorze 1? Podać identyfikator przedmiotu oraz dodatkowo datę zdania egzaminu. Uporządkować otrzymany wynik wg daty zdania egzaminu.

29. Którzy studenci zdali egzamin u egzaminatora o identyfikatorze '0004' w okresie od 01 stycznia 2011 do 31 grudnia 2011? Uwzględnić tylko tych studentów, którzy już przystąpili do egzaminu i zdawali egzamin u podanego egzaminatora. W odpowiedzi podać identyfikator studenta oraz datę egzaminu. Uporządkować otrzymany wynik wg daty egzaminu oraz identyfikatora studenta.

30. Kiedy student o identyfikatorze '0000049' zdał egzaminy z poszczególnych przedmiotów? Uwzględnić tylko te przedmioty, które były już zdawane przez studenta. Do opisu przedmiotu zastosować jego identyfikator. Uporządkować wyświetlane informacje malejąco wg daty zdania egzaminu.

31. W których ośrodkach student o identyfikatorze '0000060' zdawał egzaminy z poszczególnych przedmiotów. Dla każdego przedmiotu, z którego ten student zdawał egzamin, określić ośrodek, podając jego identyfikator. Do identyfikacji przedmiotu użyć jego identyfikator. Uporządkować wyświetlane dane wg identyfikatora przedmiotu oraz ośrodka.

32. Którzy egzaminatorzy przeprowadzili egzaminy w poszczególnych ośrodkach w okresie od 01 lipca 2013 do 31 grudnia 2014? Dla każdego ośrodka, w którym odbył się egzamin podać identyfikator egzaminatora, prowadzącego ten egzamin oraz datę tego egzaminu. Uporządkować rezultat według identyfikatora ośrodka oraz daty egzaminu (malejąco).

33. Którzy studenci zdawali egzamin w ośrodkach o identyfikatorze 1 oraz 3 w okresie od 01 stycznia 2011do 31 grudnia 2011? Dla każdego ośrodka z osobna podać identyfikator studenta, który zdawał egzamin w danym ośrodku oraz datę przeprowadzonego egzaminu. Uporządkować wyświetlane dane wg identyfikatora ośrodka, identyfikatora studenta oraz daty egzaminu.

34. Podać identyfikatory ośrodków, w których studenci o identyfikatorze '0000060' oraz '0500324' zdawali swoje egzaminy z przedmiotu o identyfikatorze 5. Dla każdego studenta z osobna wskazać ośrodek, w którym dany student był na egzaminie z podanego przedmiotu. Uporządkować otrzymany wynik wg identyfikatora studenta oraz identyfikatora ośrodka.

Temat 2.

Łączenie wielu tabel. Zastosowanie różnych typów złączeń – INNER, LEFT, RIGHT, CROSS JOIN. Łączenie wielu instrukcji SELECT – operatory UNION, UNION ALL, INTERSECT, MINUS. Grupowanie danych. Wybór grup danych według zadanego kryterium.

Przykłady treści zadań i ich rozwiązania

35. Podać nazwy przedmiotów, z których przeprowadzono egzamin. Uporządkować wynik zapytania wg nazwy przedmiotu.

```
SELECT DISTINCT Nazwa_Przedmiot
FROM Przedmioty p
INNER JOIN Egzaminy e ON p.Id_przedmiot = e.Id_przedmiot
ORDER BY Nazwa_Przedmiot ;
```

36. Podać Nazwisko, imię i identyfikator studentów egzaminowanych w ośrodku o nazwie CKMP. Dla każdego ośrodka o podanej nazwie wskazać studentów, którzy byli na egzaminie w tym ośrodku. Posortować wyświetlane dane wg ośrodka oraz studenta.

```
SELECT DISTINCT o.ID_Osrodek, Nazwa_Osrodek, s.ID_Student, Nazwisko, Imie
FROM Osrodki o
INNER JOIN Egzaminy e ON o.ID_Osrodek = e.ID_Osrodek
INNER JOIN Studenci s ON e.ID_Student = s.ID_Student
WHERE UPPER(Nazwa_Osrodek) = 'CKMP'
ORDER BY o.ID_Osrodek, s.ID_Student ;
```

37. Z których przedmiotów nie przeprowadzono jeszcze egzaminu? Podać nazwę tego przedmiotu.

```
SELECT Nazwa_Przedmiot
FROM Przedmioty p
MINUS
SELECT DISTINCT Nazwa_Przedmiot FROM Przedmioty p
INNER JOIN Egzaminy e ON p.Id_przedmiot = e.Id_przedmiot ;
```

38. W których ośrodkach nie przeprowadzono jeszcze egzaminu? Podać identyfikator i nazwę takiego ośrodka. Uporządkować otrzymane dane wg identyfikatora ośrodka.

```
SELECT o.ID_Osrodek, Nazwa_Osrodek
FROM Osrodki o
LEFT JOIN Egzaminy e ON o.ID_Osrodek = e.ID_Osrodek
WHERE e.ID_Osrodek IS NULL ;
```

39. Którzy egzaminatorzy egzaminowali studentów o identyfikatorze '000000' oraz '0000010'? Podać identyfikator, Nazwisko oraz imię egzaminatora. Uwzględnić tylko tych egzaminatorów, którzy egzaminowali obydwie te osoby.

```
SELECT g.ID_Egzaminator, Nazwisko, Imie
FROM Egzaminy e
INNER JOIN Egzaminatorzy g ON e.ID_Egzaminator = g.ID_Egzaminator
WHERE e.ID_Student = '0000005'
INTERSECT
SELECT g.ID_Egzaminator, Nazwisko, Imie
FROM Egzaminy e
INNER JOIN Egzaminatorzy g ON e.ID_Egzaminator = g.ID_Egzaminator
WHERE e.ID_Student = '0000010' ;
```

40. W których ośrodkach przeprowadzono egzaminy z poszczególnych przedmiotów? Dla każdego przedmiotu, opisanego przez jego nazwę, podać ośrodek, identyfikując go jego identyfikatorem i nazwą. W odpowiedzi uwzględnić również te przedmioty, z których jeszcze nie przeprowadzono żadnego egzaminu. Uporządkować otrzymany wynik wg nazwy przedmiotu oraz identyfikatora ośrodka.

```
SELECT DISTINCT Nazwa_Przedmiot, o.ID_Osrodek, Nazwa_Osrodek
FROM Egzaminy e
INNER JOIN Osrodki o ON o.ID_Osrodek = e.ID_Osrodek
RIGHT JOIN Przedmioty p ON p.Id_przedmiot = e.Id_przedmiot
ORDER BY Nazwa_Przedmiot, o.ID_Osrodek ;
```

41. Którzy studenci nie zdawali jeszcze egzaminu z poszczególnych przedmiotów? Dla każdego przedmiotu, określonego przez jego nazwę, podać identyfikator, imię i Nazwisko takiego studenta.

```
SELECT Nazwa_Przedmiot, ID_Student, Nazwisko, Imie
FROM Przedmioty
CROSS JOIN Studenci
MINUS
SELECT Nazwa_Przedmiot, s.ID_Student, Nazwisko, Imie
FROM Przedmioty p
INNER JOIN Egzaminy e ON p.Id_przedmiot = e.Id_przedmiot
INNER JOIN Studenci s ON e.ID_Student = s. ID_Student
ORDER BY 1, 2 ;
```

42. Ile egzaminów przeprowadzono z każdego przedmiotu? Uwzględnić także te przedmioty, z których jeszcze nie było żadnego egzaminu. Podać nazwę przedmiotu oraz liczbę egzaminów. Nazwać odpowiednio kolumnę z informacją o liczbie egzaminów. Rezultat uporządkować wg nazwy przedmiotu.

```
SELECT Nazwa_Przedmiot, COUNT(e.ID_Egzamin) AS Liczba_egzaminow
FROM Przedmioty p
LEFT JOIN Egzaminy e ON p.Id_przedmiot = e.Id_przedmiot
GROUP BY Nazwa_Przedmiot
ORDER BY Nazwa_Przedmiot ;
```

43. Ilu studentów zdawało egzaminy w każdym ośrodku? Uwzględnić także te ośrodki, w których jeszcze nie było żadnego egzaminu. Podać identyfikator oraz nazwę ośrodka, a liczbę osób zdających w danym ośrodku odpowiednio opisać. Uporządkować rezultat zapytania wg nazwy ośrodka.

```
SELECT o.ID_Osrodek, Nazwa_Osrodek, COUNT(DISTINCT e.ID_Student) AS
Liczba_osob
FROM Osrodki o
LEFT JOIN Egzaminy e ON o.ID_Osrodek = e.ID_Osrodek
GROUP BY o.ID_Osrodek, Nazwa_Osrodek ;
```

44. Który egzaminator przeprowadził więcej niż 5 egzaminów w ośrodku o nazwie CKMP? Dla każdego ośrodka o podanej nazwie, w którym odbył się egzamin, wskazać egzaminatora, podając jego identyfikator, imię oraz Nazwisko. Dodatkowo wyświetlić liczbę egzaminów przeprowadzonych przez egzaminatora w danym ośrodku i odpowiednio nazwać kolumnę z tą informacją. Uporządkować otrzymany rezultat według identyfikatora ośrodka oraz identyfikatora egzaminatora.

```
SELECT o.ID_Osrodek, Nazwa_Osrodek, g.ID_Egzaminator, Nazwisko, Imie,
COUNT(e.ID_Osrodek) AS Liczba_egzaminow
FROM Osrodki o
INNER JOIN Egzaminy e ON o.ID_Osrodek = e.ID_Osrodek
INNER JOIN Egzaminatorzy g ON e.ID_Egzaminator = g.ID_Egzaminator
WHERE UPPER(Nazwa_Osrodek) = 'CKMP'
GROUP BY o.ID_Osrodek, Nazwa_Osrodek, g.ID_Egzaminator, Nazwisko, Imie
HAVING COUNT(e.ID_Osrodek) > 5
ORDER BY o.ID_Osrodek, g.ID_Egzaminator ;
```

45. Podać datę pierwszego i ostatniego egzaminu przeprowadzonego z każdego przedmiotu. Uwzględnić także te przedmioty, z których jeszcze nie przeprowadzono żadnego egzaminu. Nazwać odpowiednio kolumny z informacją o podanych datach. Uporządkować wynik zapytania wg nazwy przedmiotu.

```
SELECT Nazwa_Przedmiot, MIN(Data_Egzamin) AS Pierwszy, MAX(Data_Egzamin) AS
Ostatni
FROM Przedmioty p
LEFT JOIN Egzaminy e ON p.Id_przedmiot = e.Id_przedmiot
GROUP BY Nazwa_Przedmiot
ORDER BY Nazwa_Przedmiot ;
```

46. Który egzaminator egzaminował więcej niż 5 studentów? Podać identyfikator, Nazwisko i imię egzaminatora oraz liczbę egzaminowanych osób. Opisać czytelnie znaczenie kolumny z liczbę osób. Uporządkować rezultat zapytania wg nazwiska i Imienia egzaminatora.

```
SELECT g.ID_Egzaminator, Nazwisko, Imie, COUNT(DISTINCT ID_Student) AS
Liczba_osob
FROM Egzaminy e
INNER JOIN Egzaminatorzy g ON e.ID_Egzaminator = g.ID_Egzaminator
GROUP BY g.ID_Egzaminator, Nazwisko, Imie
HAVING COUNT(DISTINCT ID_Student) > 5
ORDER BY Nazwisko, Imie ;
```

Zadania do samodzielnego rozwiązania

47. Który egzaminator nie przeprowadził jeszcze żadnego egzaminu? Podać jego identyfikator, imię oraz Nazwisko. Uporządkować otrzymany wynik wg nazwiska i Imienia egzaminatora.

48. W których ośrodkach przeprowadzono egzaminy z przedmiotu o nazwie *Bazy danych* oraz *Arkusze kalkulacyjne*? W odpowiedzi uwzględnić te ośrodki, w których odbył się egzamin przynajmniej z jednego podanego przedmiotu. Dla każdego przedmiotu, opisanego przez jego nazwę, podać identyfikator oraz nazwę ośrodka. Rezultat uporządkować wg nazwy przedmiotu i identyfikatora ośrodka.

49. Którzy egzaminatorzy przeprowadzili egzaminy w poszczególnych ośrodkach? Dla każdego ośrodka, w którym odbył się egzamin podać identyfikator, imię i Nazwisko egzaminatora, prowadzącego ten egzamin. Uporządkować rezultat według identyfikatora ośrodka oraz identyfikatora egzaminatora.

50. W których ośrodkach student o identyfikatorze '0000049' zdał egzaminy z poszczególnych przedmiotów? Uwzględnić tylko te przedmioty, które były już zdawane przez studenta. Dla każdego przedmiotu wskazać ośrodek, w którym miał miejsce zdany egzamin. Do opisu przedmiotu zastosować jego nazwę, natomiast do opisu ośrodka – jego identyfikator oraz nazwę. Uporządkować wyświetlane informacje malejąco wg daty zdania egzaminu.

51. W których ośrodkach student o identyfikatorze '0000050' nie zdawał jeszcze żadnego egzaminu? Podać identyfikator ośrodka, jego nazwę oraz miasto. Uporządkować otrzymany wynik wg nazwy ośrodka.

52. Którzy studenci nie zdawali egzaminu w okresie od 20 maja 2000 r. do 20 maja 2002 r.? Uwzględnić także tych studentów, którzy jeszcze nie przystąpili do egzaminu. Podać ich identyfikator, Nazwisko oraz imię. Uporządkować otrzymany wynik wg identyfikatora studenta.

53. Którzy egzaminatorzy przeprowadzili egzaminy w ośrodkach o nazwie CKMP i LBS? Podać ich identyfikator, Nazwisko oraz imię. Uwzględnić tylko tych egzaminatorów, którzy przeprowadzali egzaminy w jednym i drugim ośrodku. Uporządkować otrzymany wynik wg identyfikatora egzaminatora.

54. W których ośrodkach nie przeprowadzono egzaminu z przedmiotu „Bazy danych”? Podać ich identyfikator, nazwę oraz miasto, w którym znajduje się ośrodek. Uwzględnić także te ośrodki, w których nie odbył się jeszcze żaden egzamin. Uporządkować otrzymany wynik wg nazwy ośrodka.

55. Którzy studenci nie zdawali egzaminu w ośrodku o nazwie CKMP i LBS? Uwzględnić tylko tych studentów, którzy nie zdawali egzaminów w jednym i drugim ośrodku. Podać ich identyfikator, imię i Nazwisko. Uporządkować otrzymany wynik wg nazwiska i Imienia studenta.

56. Którzy egzaminatorzy przeprowadzili egzaminy w ośrodku o nazwie CKMP? Dla każdego ośrodka o podanej nazwie wskazać egzaminatora, podając jego identyfikator, imię i Nazwisko. W odpowiedzi uwzględnić także te ośrodki, w których nie było jeszcze egzaminu. Do identyfikacji ośrodka użyć jego identyfikatora oraz nazwy. Uporządkować otrzymany rezultat wg identyfikatora ośrodka oraz identyfikatora egzaminatora.

57. Którzy studenci zdawali egzaminy z przedmiotu „Bazy danych” w ośrodkach o nazwie CKMP i LBS? Podać ich identyfikator, Nazwisko oraz imię. Uwzględnić tylko tych studentów, którzy zdawali egzaminy ze wskazanego przedmiotu w jednym i drugim ośrodku. Uporządkować otrzymany wynik wg identyfikatora studenta.

58. Którzy studenci zdawali egzaminy u egzaminatora o identyfikatorze '0001' oraz '0004'? Podać identyfikator, imię i Nazwisko studenta. Uwzględnić tylko tych studentów, którzy zdawali egzaminy zarówno u jednego jak i drugiego egzaminatora. Uporządkować otrzymany wynik wg identyfikatora studenta.

59. W których ośrodkach przeprowadzono egzaminy z przedmiotu o nazwie *Bazy danych,* a nie przeprowadzono egzaminu z przedmiotu o nazwie *Arkusze kalkulacyjne*? Podać identyfikator oraz nazwę ośrodka. Rezultat uporządkować wg nazwy ośrodka.

60. W których ośrodkach poszczególni egzaminatorzy egzaminowali poszczególnych studentów? Dla każdego egzaminatora wskazać studentów egzaminowanych przez niego oraz miejsce przeprowadzenia egzaminu. Do identyfikacji egzaminatora i studenta użyć ich identyfikatorów, imion i nazwisk. Natomiast do określenia ośrodka – identyfikatora i nazwy ośrodka. Uporządkować rezultat w taki sposób, aby można było zapewnić czytelność wyniku i możliwość spójnej i prawidłowej jego interpretacji.

61. Kiedy studenci zdawali egzaminy z poszczególnych przedmiotów? W odpowiedzi uwzględnić także tych studentów, którzy jeszcze nie przystąpili do egzaminu. Dla każdego studenta należy wskazać przedmiot oraz datę egzaminu z tego przedmiotu. Do identyfikacji studenta użyć identyfikatora, Imienia i nazwiska. Natomiast dla przedmiotu – nazwy przedmiotu. Ponadto uporządkować malejąco wynik zapytania według daty egzaminu i zapewnić możliwość spójnej i poprawnej interpretacji otrzymanego wyniku.

62. Którzy egzaminatorzy nie przeprowadzili egzaminów w poszczególnych ośrodkach? Dla każdego ośrodka podać identyfikator, imię i Nazwisko egzaminatora, który nie egzaminował w danym ośrodku. Uporządkować rezultat według identyfikatora ośrodka oraz identyfikatora egzaminatora.

63. W których ośrodkach przeprowadzono egzaminy z przedmiotu o nazwie *Bazy danych* i *Arkusze kalkulacyjne,* a nie przeprowadzono egzaminu z przedmiotu o nazwie *Przetwarzanie tekstów*? Podać identyfikator oraz nazwę ośrodka. Rezultat uporządkować wg nazwy ośrodka.

64. Który student zdawał egzaminy w okresie od 01 stycznia 2000 do 31 marca 2000, a nie zdawał egzaminów w okresie od 1 lipca 2002 do 31 grudnia 2002? Podać jego identyfikator, imię i Nazwisko. Otrzymany wynik uporządkować wg nazwiska i Imienia studenta.

65. Ile egzaminów przeprowadzono w każdym ośrodku? W odpowiedzi uwzględnić także te ośrodki, w których egzaminu jeszcze nie przeprowadzono. Podać identyfikator i nazwę ośrodka oraz liczbę egzaminów w każdym z nich. Nazwać odpowiednio kolumnę z informacją o liczbie egzaminów. Uporządkować otrzymany rezultat nazwy ośrodka.

66. Ile egzaminów zdali poszczególni studenci? W odpowiedzi uwzględnić tylko tych studentów, którzy przystąpili już do egzaminu. W odpowiedzi umieścić informację identyfikującą studenta (identyfikator, Nazwisko, imię) oraz licznie zdanych egzaminów. Nazwać odpowiednio kolumnę z informacją o liczbie egzaminów. Uporządkować otrzymany rezultat według nazwiska i Imienia studenta.

67. W których ośrodkach przeprowadzono z przedmiotu o nazwie *Bazy danych* więcej niż 2 egzaminy? Podać identyfikator ośrodka oraz jego nazwę. Uporządkować otrzymany rezultat według nazwy ośrodka.

68. Ile egzaminów przeprowadzono w ośrodkach o nazwie CKMP oraz LBS. Dla każdego ośrodka o podanej nazwie określić liczbę przeprowadzonych egzaminów. W odpowiedzi umieścić informację o identyfikatorze ośrodka, jego nazwie oraz liczbie egzaminów w każdym z w/w ośrodków. Nazwać odpowiednio kolumnę z informacją o liczbie egzaminów. Uporządkować otrzymany rezultat według nazwy ośrodka.

69. Podać datę pierwszego i ostatniego egzaminu przeprowadzonego przez każdego egzaminatora. W odpowiedzi uwzględnić także tych egzaminatorów, którzy jeszcze nie przystąpili do egzaminu. Nazwać odpowiednio kolumny z informacją o podanych datach. Uporządkować wynik zapytania wg nazwiska i Imienia egzaminatora.

70. W których ośrodkach przeprowadzono więcej niż 4 egzaminy? Podać identyfikator ośrodka, jego nazwę oraz miasto. Uporządkować rezultat według identyfikatora ośrodka.

71. W ilu ośrodkach student o identyfikatorze '0000009' zdawał swoje egzaminy? W odpowiedzi podać informację o Imieniu i nazwisku studenta oraz liczbie ośrodków. Nazwać odpowiednio kolumnę z informacją o liczbie ośrodków.

72. Którzy studenci zdawali egzaminy tylko w jednym ośrodku? Podać ich identyfikator, Nazwisko i imię. Uporządkować wynik zapytania wg nazwiska i Imienia studenta.

73. Podać datę pierwszego i ostatniego egzaminu z przedmiotu o nazwie *Bazy danych* w poszczególnych ośrodkach. Uwzględnić tylko te ośrodki, w których odbył się egzamin z podanego przedmiotu. W odpowiedzi umieścić informację o ośrodku (identyfikator, nazwa) oraz odpowiednich datach. Nazwać odpowiednio kolumny z informacją o podanych datach.

74. Ile przedmiotów zdawał student o identyfikatorze '0000005' w ośrodku o nazwie CKMP? Dla każdego ośrodka o podanej nazwie określić liczbę przedmiotów zdawanych przez tego studenta. W odpowiedzi umieścić identyfikator i nazwę ośrodka oraz liczbę zdawanych przedmiotów. Nazwać odpowiednio kolumnę z informacją o liczbie przedmiotów. Uporządkować wynik zapytania wg identyfikatora ośrodka.

75. Ilu studentów zdawało egzaminy z każdego przedmiotu? Uwzględnić także te przedmioty, z których jeszcze nie było żadnego egzaminu. Podać nazwę przedmiotu, a liczbę osób zdających dany przedmiot odpowiednio opisać. Uporządkować rezultat zapytania wg nazwy przedmiotu.

76. Ile egzaminów przeprowadzono w poszczególnych ośrodkach z poszczególnych przedmiotów? Dla każdego przedmiotu, opisanego przez jego nazwę i ośrodka, opisanego przez identyfikator i nazwę, podać liczbę egzaminów. W odpowiedzi uwzględnić również te przedmioty, z których jeszcze nie przeprowadzono żadnego egzaminu. Uporządkować otrzymany wynik wg nazwy przedmiotu oraz identyfikatora ośrodka.

77. Który egzaminator egzaminował więcej niż 5 osób w ośrodku o nazwie CKMP? Dla każdego ośrodka o podanej nazwie wskazać egzaminatora, podając jego identyfikator, imię oraz Nazwisko. Uporządkować otrzymany rezultat według identyfikatora ośrodka oraz nazwiska i Imienia egzaminatora.

78. Podać datę pierwszego i ostatniego egzaminu zdanego przez poszczególnych studentów. Uwzględnić tylko tych studentów, którzy przystąpili już do egzaminu. Podać ich identyfikator, imię i Nazwisko. Otrzymany wynik uporządkować wg nazwiska i Imienia studenta.

79. Z którego przedmiotu w ośrodku o nazwie CKMP przeprowadzono więcej niż 5 egzaminów? Dla każdego ośrodka o podanej nazwie wskazać przedmiot, podając jego nazwę. Uporządkować otrzymany rezultat według identyfikatora ośrodka oraz nazwy przedmiotu.

80. W których ośrodkach egzaminator o nazwisku *Muryjas* egzaminował więcej niż 3 osoby? Odpowiedź podać dla każdego egzaminatora o wymienionym nazwisku. W odpowiedzi należy umieścić informację o egzaminatorze (identyfikator, imię i Nazwisko), o ośrodku (identyfikator i nazwę) oraz liczbie egzaminowanych osób w danym ośrodku. Uporządkować otrzymany rezultat według identyfikatora egzaminatora oraz nazwy ośrodka.

81. Ile egzaminów zdawali i ile zdali poszczególni studenci? Dla każdego studenta, który przystąpił do egzaminu podać liczbę zdawanych przez niego egzaminów oraz liczbę zdanych przez niego egzaminów. W odpowiedzi umieścić informację o studencie (identyfikator, Nazwisko, imię), liczbie zdawanych i zdanych egzaminów. Opisać odpowiednio prezentowane liczby egzaminów.

Temat 3.

Wybór danych z wykorzystaniem podzapytań skalarnych, wierszowych i skorelowanych.

Przykłady treści zadań i ich rozwiązania

82. Podać identyfikator studenta, który zdawał egzamin w pierwszym dniu egzaminowania.

```
SELECT DISTINCT ID_Student
FROM Egzaminy
WHERE Data_Egzamin = ( SELECT MIN(Data_Egzamin)
                       FROM Egzaminy ) ;
```

83. Z którego przedmiotu przeprowadzono egzaminy po ostatnim egzaminie z przedmiotu *Bazy danych*? Podać nazwę tego przedmiotu.

```
SELECT DISTINCT Nazwa_Przedmiot
FROM Przedmioty p
INNER JOIN Egzaminy e ON p.Id_przedmiot = e.Id_przedmiot
WHERE Data_Egzamin > ( SELECT MAX(Data_Egzamin)
                       FROM Egzaminy e
                       INNER JOIN Przedmioty p ON p.Id_przedmiot = e.Id_przedmiot
                       AND UPPER(Nazwa_Przedmiot) = 'BAZY DANYCH' ) ;
```

84. Który egzaminator przeprowadził najmniej egzaminów? Podać jego identyfikator, Nazwisko oraz imię. Uwzględnić tylko tych egzaminatorów, którzy już przeprowadzili egzaminy.

```
SELECT g.ID_Egzaminator, Nazwisko, Imie
FROM Egzaminatorzy g
INNER JOIN Egzaminy e ON g.ID_Egzaminator = e.ID_Egzaminator
GROUP BY g.ID_Egzaminator, Nazwisko, Imie
HAVING COUNT(ID_Egzamin) = ( SELECT MIN(COUNT(ID_Egzamin))
                             FROM Egzaminy
                             GROUP BY ID_Egzaminator ) ;
```

85. W którym ośrodku do egzaminu przystąpiło najwięcej osób? Podać identyfikator i nazwę takiego ośrodka oraz liczbę egzaminowanych osób. Jeśli wynik zawiera wiele ośrodków, uporządkować rezultat według nazwy ośrodka.

```
SELECT o.ID_Osrodek, Nazwa_Osrodek, COUNT(DISTINCT ID_Student)
FROM Osrodki o
INNER JOIN Egzaminy e ON o.ID_Osrodek = e.ID_Osrodek
GROUP BY o.ID_Osrodek, Nazwa_Osrodek
HAVING COUNT(DISTINCT ID_Student) = ( SELECT MAX(COUNT(ID_Student))
                                     FROM Egzaminy
                                     GROUP BY ID_Osrodek ) ;
```

86. Z którego przedmiotu nie przeprowadzono jeszcze egzaminu? Podać nazwę przedmiotu.

```
SELECT Nazwa_Przedmiot
FROM Przedmioty
WHERE Id_przedmiot NOT IN ( SELECT DISTINCT Id_przedmiot
                            FROM Egzaminy ) ;
```

87. Który student nie przystąpił jeszcze do egzaminu? Podać jego identyfikator, imię oraz Nazwisko.

```
SELECT ID_Student, Imie, Nazwisko
FROM Studenci s
WHERE NOT EXISTS ( SELECT ID_Egzamin
                   FROM Egzaminy e
                   WHERE e.ID_Student = s.ID_Student ) ;
```

88. Którzy studenci byli na ostatnim egzaminie przeprowadzanym w poszczególnych ośrodkach? Uwzględnić tylko te ośrodki, w których odbyły się już egzaminy. Dla każdego ośrodka określonego przez jego identyfikator, podać identyfikator studenta oraz datę ostatniego egzaminu.

```
SELECT DISTINCT ID_Osrodek, ID_Student, Data_Egzamin
FROM Egzaminy e1
WHERE (ID_Osrodek, Data_Egzamin) = ( SELECT ID_Osrodek, MAX(Data_Egzamin)
                                     FROM Egzaminy e2
                                     WHERE e2.ID_Osrodek = e1.ID_Osrodek
                                     GROUP BY ID_Osrodek )
ORDER BY ID_Osrodek, ID_Student, Data_Egzamin ;
```

89. Podać trzy ostatnie dni, kiedy odbyły się egzaminy? Dni należy określić poprzez podanie pełnej daty tj. dnia, miesiąca oraz roku. Wynik uporządkować malejąco według daty egzaminu.

```
SELECT DISTINCT Data_Egzamin
FROM Egzaminy e1
WHERE 3 >= ( SELECT COUNT(DISTINCT e2.Data_Egzamin)
             FROM Egzaminy e2
             WHERE e2.Data_Egzamin > e1.Data_Egzamin )
ORDER BY Data_Egzamin DESC;
```

90. W których ośrodkach przeprowadzono najwięcej egzaminów z poszczególnych przedmiotów? Dla każdego przedmiotu, identyfikowanego przez nazwę, wskazać ośrodek, podając jego identyfikator oraz nazwę. Dodatkowo wyświetlić liczbę egzaminów.

```
SELECT Nazwa_Przedmiot, o.ID_Osrodek, Nazwa_Osrodek, COUNT(ID_Egzamin)
FROM Przedmioty p
INNER JOIN Egzaminy e1 ON p.Id_przedmiot = e1.Id_przedmiot
INNER JOIN Osrodki o ON o.ID_Osrodek = e1.ID_Osrodek
GROUP BY p.Id_przedmiot, Nazwa_Przedmiot, o.ID_Osrodek, Nazwa_Osrodek
HAVING COUNT(ID_Egzamin) = ( SELECT MAX(COUNT(ID_Egzamin))
                             FROM Egzaminy e2
                             WHERE e2.Id_przedmiot = p.Id_przedmiot
                             GROUP BY ID_Osrodek) ;
```

Zadania do samodzielnego rozwiązania

91. W których ośrodkach student o identyfikatorze *0000050* nie zdawał jeszcze żadnego egzaminu? Podać identyfikator ośrodka, jego nazwę oraz miasto.

92. Który student nie zdawał egzaminu z przedmiotu *Bazy danych*? Podać identyfikator, Nazwisko oraz imię studenta. Uwzględnić także tych studentów, którzy jeszcze nie zdawali żadnego egzaminu. Wynik uporządkować według identyfikatora studenta. Zadanie należy wykonać, wykorzystując podzapytanie.

93. W których ośrodkach nie przeprowadzono egzaminu z przedmiotu *Bazy danych*? Podać ich identyfikator, nazwę oraz miasto, w którym znajduje się ośrodek. Uwzględnić także te ośrodki, w których nie odbył się jeszcze żaden egzamin. Wynik uporządkować według identyfikatora ośrodka. Zadanie należy wykonać, wykorzystując podzapytanie.

94. Którzy studenci zdawali egzaminy z przedmiotu *Bazy danych* w ośrodkach o nazwie CKMP i LBS? Podać ich identyfikator, Nazwisko oraz imię. Uwzględnić tylko tych studentów, którzy zdawali egzaminy ze wskazanego przedmiotu w jednym i drugim ośrodku. Wynik uporządkować według identyfikatora studenta. Zadanie należy wykonać, wykorzystując podzapytanie.

95. W których ośrodkach student o nazwisku *Biegaj* nie zdawał jeszcze egzaminu? Dla każdego studenta o podanym nazwisku wskazać ośrodek, podając jego identyfikator oraz nazwę. Uporządkować rezultat zapytania według identyfikatora studenta oraz nazwy ośrodka.

96. Z którego przedmiotu przeprowadzono najwięcej egzaminów? Podać nazwę przedmiotu oraz liczbę egzaminów.

97. W którym ośrodku przeprowadzono najmniej egzaminów? Podać identyfikator, nazwę ośrodka oraz liczbę egzaminów w ośrodku. Uwzględnić tylko te ośrodki, w których odbyły się już egzaminy.

98. Który student zdał najmniej egzaminów w ośrodku (ośrodkach) o nazwie CKMP? Dla każdego ośrodka o podanej nazwie wskazać studenta, podając jego identyfikator, imię oraz Nazwisko. Dodatkowo wyświetlić liczbę zdanych egzaminów przez studenta, wskazanego w wyniku zapytania. Uporządkować wynik zapytania według identyfikatora ośrodka oraz identyfikatora studenta.

99. Który egzaminator egzaminował najwięcej osób? Podać identyfikator, imię i Nazwisko egzaminatora oraz liczbę egzaminowanych przez niego osób. Jeśli wynik zawiera wielu egzaminatorów, uporządkować rezultat według identyfikatora egzaminatora.

100. W których ośrodkach przeprowadzono ostatni egzamin z poszczególnych przedmiotów? Dla każdego przedmiotu, identyfikowanego przez nazwę, podać identyfikator oraz nazwę ośrodka. Dodatkowo wyświetlić datę egzaminu. Uporządkować wynik zapytania według nazwy przedmiotu.

101. W których ośrodkach egzaminator o nazwisku *Muryjas* nie przeprowadził egzaminu? Dla każdego egzaminatora o podanym nazwisku wskazać ośrodek bądź ośrodki, podając jego identyfikator oraz nazwę. Uwzględnić także te ośrodki, w których nie odbył się jeszcze żaden egzamin. Rezultat zapytania uporządkować według identyfikatora egzaminatora.

102. Z którego przedmiotu przeprowadzono najwięcej egzaminów w poszczególnych ośrodkach? Dla każdego ośrodka, opisanego przez jego identyfikator oraz nazwę, wskazać przedmiot, podając jego nazwę. Dodatkowo podać największą liczbę egzaminów przeprowadzonych z danego przedmiotu w kolejnych ośrodkach. Uporządkować rezultat zapytania według identyfikatora ośrodka oraz nazwy przedmiotu.

103. Którzy studenci zdawali egzamin z poszczególnych przedmiotów w pierwszym dniu egzaminowania z danego przedmiotu? Dla każdego przedmiotu, identyfikowanego przez jego nazwę, wskazać studenta, podając jego identyfikator, imię oraz Nazwisko. Dodatkowo wyświetlić datę pierwszego egzaminu z poszczególnych przedmiotów. Uporządkować wynik zapytania według nazwy przedmiotu.

104. Którzy studenci zdawali egzamin u egzaminatora o nazwisku *Muryjas* w pierwszym dniu egzaminowania przez tego egzaminatora? Dla każdego egzaminatora o podanym nazwisku wskazać studenta, podając jego identyfikator, Imie oraz Nazwisko. Ponadto wyświetlić datę pierwszego egzaminu dla każdego egzaminatora o podanym nazwisku. Wynik zapytania uporządkować według nazwiska i Imienia egzaminatora.

105. Z którego przedmiotu przeprowadzono najwięcej egzaminów w ośrodkach o nazwie CKMP? Dla każdego ośrodka o podanej nazwie wskazać przedmiot, opisując go jego nazwą, oraz podać liczbę egzaminów z przedmiotu, będącego wynikiem zapytania. Otrzymane rezultaty uporządkować według identyfikatora ośrodka.

106. Którzy studenci zdawali najwięcej egzaminów z poszczególnych przedmiotów? Dla każdego przedmiotu, opisanego przez jego nazwę, wskazać studenta, podając jego identyfikator, imię oraz Nazwisko. Uporządkować wynik zapytania według nazwy przedmiotu.

107. W których ośrodkach egzaminator o nazwisku *Muryjas* egzaminował więcej niż 2 studentów. Dla każdego egzaminatora o podanym nazwisku podać identyfikator ośrodka oraz jego nazwę. Rezultat zapytania uporządkować według nazwiska i Imienia egzaminatora oraz nazwy ośrodka.

108. Którzy studenci zdawali pierwszy oraz ostatni egzamin przeprowadzony w poszczególnych ośrodkach? Dla każdego ośrodka wskazać studenta, podając jego identyfikator, Nazwisko oraz imię. Do opisu ośrodka użyć jego identyfikator oraz nazwę. W odpowiedzi dodatkowo zamieścić datę pierwszego i ostatniego egzaminu w poszczególnych ośrodkach dla wybranych studentów. Uwzględnić tylko te ośrodki, w których odbył się już egzamin.

109. Z jakiego przedmiotu egzaminator o nazwisku *Muryjas* przeprowadził najwięcej egzaminów w ośrodku o nazwie CKMP? Informację o przedmiocie podać dla każdego egzaminatora o podanym nazwisku oraz ośrodka o wskazanej nazwie, w którym egzaminator ten przeprowadził egzaminy. Uporządkować otrzymany wynik według identyfikatora egzaminatora oraz identyfikatora ośrodka.

110. W którym ośrodku student o nazwisku *Biegaj* zdał najmniej egzaminów? Dla każdego studenta o podanym nazwisku, wskazać ośrodek, podając jego identyfikator oraz nazwę. Uwzględnić tylko tych studentów, którzy już przystąpili do egzaminu. Dodatkowo podać dla każdego wybranego studenta liczbę zdanych przez niego egzaminów. Uporządkować rezultat zapytania według nazwiska i Imienia studenta.

111. Który student ma najwięcej niezdanych egzaminów w poszczególnych ośrodkach? Dla każdego ośrodka wskazać studenta, podając jego identyfikator, Nazwisko, Imie. Uwzględnić tylko te ośrodki, w których odbył się już egzamin. Do opisu ośrodka użyć jego identyfikator oraz nazwę. Dodatkowo dla każdego wybranego studenta wyświetlić liczbę niezdanych przez niego egzaminów. Uporządkować wynika zapytania według nazwy ośrodka oraz nazwiska i Imienia studenta.

112. W których ośrodkach poszczególni egzaminatorzy przeprowadzili swój pierwszy i ostatni zdany egzamin? Dla każdego egzaminatora, opisanego identyfikatorem, nazwiskiem i Imieniem, wskazać ośrodek, podając jego identyfikator oraz nazwę. Dodatkowo w odpowiedzi wyraźnie podkreślić informację, iż dany ośrodek jest miejscem pierwszego lub ostatniego egzaminu. Uporządkować rezultat zapytania według nazwiska i Imienia egzaminatora.

113. W którym miesiącu i którego roku przeprowadzono najwięcej egzaminów w ośrodku o nazwie Atena? Dla każdego ośrodka o wskazanej nazwie wyznaczyć miesiąc oraz rok, podając pełną nazwę miesiąca i rok w formacie czterocyfrowym. Dodatkowo wyświetlić liczbę egzaminów przeprowadzonych w wybranych ośrodkach dla każdego wyznaczonego miesiąca i roku. Uporządkować wynik zapytania według identyfikatora ośrodka.

114. W którym miesiącu każdego roku przeprowadzono najwięcej egzaminów? Uwzględnić tylko te lata, w których odbywały się egzaminy. Dla każdego roku wskazać miesiąc, opisując go jego nazwą.

115. W którym miesiącu każdego roku przeprowadzono najwięcej egzaminów w ośrodku (ośrodkach) o nazwie CKMP? Dla każdego ośrodka o podanej nazwie, opisanego identyfikatorem i nazwą wskazać ten miesiąc roku, w którym odbyło się najwięcej egzaminów. Uporządkować wynik zapytania według identyfikatora ośrodka oraz roku.

116. W którym miesiącu każdego roku przeprowadzono najwięcej egzaminów? Uwzględnić tylko te lata, w których odbywały się egzaminy. Dla każdego roku wskazać miesiąc, opisując go jego nazwą. Ponadto podać dla każdego wyznaczonego miesiąca liczbę przeprowadzonych w nim egzaminów. Uporządkować rezultat zapytania według roku.

Temat 4.

Tworzenie i modyfikowanie struktury tabel (instrukcje CREATE TABLE, ALTER TABLE). Usuwanie tabel (instrukcja DROP TABLE). Definiowanie warunków integralności danych (PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK). Aktywacja i dezaktywacja więzów integralności.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

117. Utworzyć tabelę o nazwie *Przedmioty*. Zdefiniować w niej kolumny podane na diagramie.

```
CREATE TABLE Przedmioty ( Id_przedmiot  NUMBER(3),
                          Nazwa_Przedmiot      VARCHAR2(40),
                          Opis                 VARCHAR2(200) ) ;
```

118. Utworzyć tabelę o nazwie *KopiaEgzaminy*, która będzie wierną kopią tabeli *Egzaminy*.

```
CREATE TABLE  KopiaEgzaminy
        AS
        SELECT * FROM Egzaminy ;
```

119. Utworzyć tabelę o nazwie *KopiaStudenci* na podstawie struktury tabeli *Studenci*. W nowej tabeli utworzyć tylko kolumny, które będą zawierać identyfikator studenta, jego imię oraz Nazwisko. Kolumny te nazwać odpowiednio: *NrAlbumu, Nazwisko, Imie*. Spowodować, by polecenie tworzyło wyłącznie strukturę nowej tabeli, bez kopiowania do niej danych z tabeli *Studenci*.

```
CREATE TABLE KopiaStudenci (NrAlbumu, Nazwisko, Imie)
        AS
        SELECT ID_Student, Nazwisko, Imie
        FROM Studenci  WHERE 1 = 0 ;
```

120. Utworzyć tabelę o nazwie *KopiaEgzaminatorzy* na podstawie struktury tabeli *Egzaminatorzy*. W nowej tabeli utworzyć kolumny, które będą zawierać identyfikator egzaminatora, jego imię oraz Nazwisko. Kolumny te nazwać odpowiednio: *IDEgzaminator, Nazwisko, Imie*. Kolumna *IDEgzaminator* ma przechowywać dane typu liczbowego. Spowodować, by polecenie tworzyło wyłącznie strukturę nowej tabeli, bez kopiowania do niej danych z tabeli *Egzaminatorzy*.

```
CREATE TABLE KopiaEgzaminatorzy
        AS
        SELECT TO_NUMBER(ID_Egzaminator) AS IDEgzaminator, Nazwisko, Imie
        FROM Egzaminatorzy
        WHERE 'a' = 'b' ;
```

121. Utworzyć tabelę o nazwie *TMaxOsrodki*. Tabela będzie umożliwiać przechowywanie danych na temat studentów, którzy zdawali najwięcej egzaminów w poszczególnych ośrodkach. Zdefiniować polecenie tak, aby utworzyło ono strukturę tej tabeli oraz wstawiło do niej odpowiednie dane, uzyskane na podstawie istniejących tabel.

```
CREATE TABLE TMaxOsrodki  AS
SELECT e.ID_Osrodek, Nazwa_Osrodek, e.ID_Student, Nazwisko, Imie, COUNT
(ID_Egzamin) AS Liczba
FROM Egzaminy e JOIN Osrodki o
ON e.ID_Osrodek=o.ID_Osrodek
JOIN Studenci s ON e.ID_Student=s.ID_Student
GROUP BY e.ID_Osrodek, Nazwa_Osrodek, e.ID_Student, Nazwisko, Imie
HAVING COUNT(ID_Egzamin) = (
        SELECT MAX(COUNT(ID_Egzamin))
        FROM Egzaminy e1
        WHERE e1.ID_Osrodek = e.ID_Osrodek
        GROUP BY ID_Student ) ;
```

122. Do tabeli *Egzaminy* dodać kolumnę o nazwie *Punkty*. Kolumna ta będzie umożliwiać gromadzenie danych o liczbie punktów uzyskanych na egzaminie. Uwzględnić w typie danych kolumny fakt, że maksymalna liczba punktów do zdobycia wynosi 10.

```
ALTER TABLE Egzaminy
        ADD Punkty  NUMBER(2) ;
```

123. Zmodyfikować strukturę tabeli *Egzaminy* tak, aby możliwe było przyznanie za egzamin maksymalnie 100 punktów.

```
ALTER TABLE Egzaminy
        MODIFY Punkty  NUMBER(3) ;
```

124. W tabeli *Egzaminy* zmodyfikować definicję kolumny *Punkty* tak, aby dla każdego nowego rekordu w tej tabeli wartość w tej kolumnie wynosiła 1.

```
ALTER TABLE Egzaminy
        MODIFY Punkty  DEFAULT 1 ;
```

125. Usunąć tabelę o nazwie *TMaxOsrodki*. Tabela ta nie jest powiązana z żadną inną tabelą w bazie danych.

```
DROP TABLE TMaxOsrodki ;
```

126. Usunąć tabelę o nazwie *TMaxOsrodki*. Przyjąć założenie, że tabela ta jest powiązana z pewną inną tabelą w bazie danych.

```
DROP TABLE TMaxOsrodki CASCADE CONSTRAINTS ;
```

127. W tabeli *Przedmioty* zdefiniować klucz podstawowy oparty na kolumnie *Id_przedmiot*. Przyjąć założenie, że tabela *Przedmioty* już istnieje w bazie danych.

```
ALTER TABLE Przedmioty
        ADD PRIMARY KEY(Id_przedmiot ) ;
```

128. W tabeli *Przedmioty* zdefiniować klucz podstawowy oparty na kolumnie *Id_przedmiot*. Operację wykonać podczas tworzenia tabeli *Przedmioty*.

```
CREATE TABLE Przedmioty ( Id_przedmiot  NUMBER(3)  PRIMARY KEY,
                          Nazwa_Przedmiot       VARCHAR2(40),
                          Opis                  VARCHAR2(200) ) ;
```

129. W tabeli *Przedmioty* zapewnić unikalność wartości w kolumnie z nazwą przedmiotu. Przyjąć założenie, że tabela *Przedmioty* już istnieje w bazie danych.

```
ALTER TABLE Przedmioty
        ADD UNIQUE (Nazwa_Przedmiot) ;
```

130. W tabeli *Przedmioty* zapewnić unikalność oraz określoność wartości w kolumnie z nazwą przedmiotu. Operację wykonać podczas tworzenia tabeli *Przedmioty*.

```
CREATE TABLE Przedmioty ( Id_przedmiot  NUMBER(3)  PRIMARY KEY,
                          Nazwa_Przedmiot       VARCHAR2(40) NOT NULL,
                          Opis                  VARCHAR2(200),
                          UNIQUE (Nazwa_Przedmiot)) ;
```

131. W tabeli *Osrodki* spowodować, by wartości w kolumnie z nazwą ośrodka były zawsze określone. Przyjąć założenie, że tabela *Osrodki* już istnieje w bazie danych.

```
ALTER TABLE Osrodki
        MODIFY Nazwa_Osrodek NOT NULL ;
```

132. W tabeli *Egzaminy* zapewnić, by wartości w kolumnie *Zdal* wynosiły 'Y' lub 'N'. Nazwać tak zdefiniowane więzy integralności.

```
ALTER TABLE Egzaminy
        ADD CONSTRAINT ZdalNN CHECK (Zdal IN ('Y', 'N'));
```

133. Zdefiniować powiązanie między tabelami *Studenci* i *Egzaminy*, wykorzystując kolumnę *ID_Student* jako kolumnę łączącą te tabele. Nazwać tak powstałe więzy integralności.

```
ALTER TABLE Egzaminy
        ADD CONSTRAINT EgzaminyStudenciFK FOREIGN KEY (ID_Student)
        REFERENCES Studenci(ID_Student) ;
```

134. Zdefiniować powiązanie między tabelami *Osrodki* i *Egzaminy*, wykorzystując kolumnę *ID_Osrodek* jako kolumnę łączącą te tabele. Nazwać tak powstałe więzy integralności. Dodatkowo zapewnić możliwość usunięcia rekordów z tabeli *Egzaminy* w przypadku usunięcia rekordu z tabeli *Osrodki*.

```
ALTER TABLE Egzaminy
        ADD CONSTRAINT EgzaminyOsrodkiFK FOREIGN KEY(ID_Osrodek)
        REFERENCES Osrodki(ID_Osrodek ON DELETE CASCADE ;
```

135. W tabeli *Osrodki* usunąć klucz podstawowy. Należy uwzględnić fakt, że tabela *Osrodki* jest powiązana z tabelą *Egzaminy*. W związku z tym usunięcie klucza podstawowego w tabeli *Osrodki* musi spowodować automatyczne usunięcie odpowiedniego klucza obcego w tabeli *Egzaminy*.

```
ALTER TABLE Osrodki
        DROP PRIMARY KEY CASCADE CONSTRAINTS ;
```

136. W tabeli *Egzaminy* wyłączyć więzy integralności odpowiedzialne za kontrolę wartości wprowadzanych do kolumny *Zdal*. Przyjąć założenie, że tabela *Egzaminy* już istnieje w bazie danych.

```
ALTER TABLE Egzaminy
        DISABLE CONSTRAINT ZdalNN ;
```

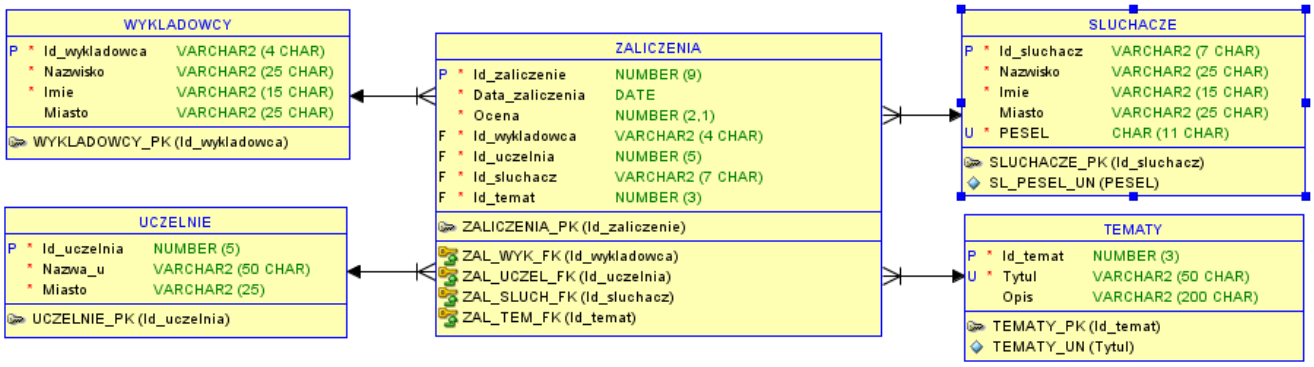
137. W tabeli *Przedmioty* wyłączyć więzy integralności odpowiedzialne za unikalność nazw przedmiotów. Przyjąć założenie, że tabela *Przedmioty* już istnieje w bazie danych.

```
ALTER TABLE Przedmioty
        DISABLE UNIQUE (Nazwa_Przedmiot) ;
```

138. W tabeli *Osrodki* dezaktywować klucz podstawowy. Należy uwzględnić fakt, że tabela *Osrodki* jest powiązana z tabelą *Egzaminy*. W związku z tym dezaktywacja klucza podstawowego w tabeli *Osrodki* musi spowodować automatyczną dezaktywację odpowiedniego klucza obcego w tabeli *Egzaminy*.

```
ALTER TABLE Osrodki
        DISABLE  PRIMARY KEY CASCADE CONSTRAINTS ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony na rysunku)



139. Utworzyć tabelę *Wykladowcy* według schematu podanego na rysunku. Nie definiować warunków integralności danych.

140. Utworzyć tabelę *Sluchacze* według schematu podanego na rysunku. Nie definiować warunków integralności danych.

141. Utworzyć tabelę *Uczelnie* według schematu podanego na rysunku. Nie definiować warunków integralności danych.

142. Utworzyć tabelę *Tematy* według schematu podanego na rysunku. Nie definiować warunków integralności danych.

143. Utworzyć tabelę *Zaliczenia* według schematu podanego na rysunku. Nie definiować warunków integralności danych.

144. Do tabeli *Zaliczenia* dodać kolumnę o nazwie *Ocena*. Kolumna powinna umożliwiać przechowywanie informacji o ocenie uzyskanej z egzaminu. Zbiór ocen zdefiniowany jest jako 2, 3, 3.5, 4, 4.5 i 5.

145. W tabeli *Zaliczenia* zmodyfikować definicję kolumny *Ocena* tak, aby dla każdego nowego egzaminu przyjmowała ona wartość równą 3.

146. W tabeli *Zaliczenia* usunąć kolumnę *Zdal*.

147. W tabeli *Wykladowcy* dodać nową kolumnę o nazwie *Pensja*. Kolumna powinna umożliwić przechowywanie danych o wynagrodzeniu egzaminatora za każdy przeprowadzony egzamin. Wynagrodzenie za jeden egzamin dla każdego egzaminatora wynosi domyślnie 15,50 zł.

148. Utworzyć tabelę o nazwie *TSuperSluchacz*. Jej struktura powinna umożliwić przechowywanie danych o tych słuchaczach, którzy mają najwyższą średnią z zaliczeń przeprowadzonych na poszczególnych uczelniach. Dla każdej uczelni należy wskazać słuchacza, który uzyskał najwyższą średnią. W tabeli należy utworzyć kolumny opisujące uczelnię, słuchacza oraz średnią ocen słuchacza. Nazwać odpowiednio kolumny w tabeli, by zapewnić czytelność gromadzonych danych.

149. Utworzyć tabelę o nazwie *TSuperWykladowca*. Jej struktura powinna umożliwić przechowywanie danych o tych wykładowcach, którzy zarobili najwięcej na przeprowadzaniu zaliczeń w poszczególnych uczelniach. Dla każdej uczelni należy wskazać wykładowcę, który zarobił najwięcej w tej uczelni na przeprowadzaniu zaliczeń. W tabeli należy utworzyć kolumny opisujące uczelnię, wykładowcę oraz sumaryczną kwotę uzyskaną przez wykładowcę. Nazwać odpowiednio kolumny w tabeli, by zapewnić czytelność gromadzonych danych.

150. Utworzyć tabelę o nazwie *TUczelnieMax*, w której znajdą się informacje nt. miesiąca i roku, w których przeprowadzono najwięcej zaliczeń na poszczególnych uczelniach. Dla każdej uczelni, w której przeprowadzono zaliczenia, należy wskazać miesiąc i rok oraz liczbę przeprowadzonych w uczelni zaliczeń.

151. W tabeli *Wykladowcy* zdefiniować:

- klucz podstawowy oparty na kolumnie *Id_wykladowca*;
- warunki integralności dla pól *Imie* i *Nazwisko*, zapewniające wartość określoną w tych kolumnach.

Nazwać wszystkie warunki integralności.

152. W tabeli *Sluchacze* zdefiniować i nazwać warunki integralności dotyczące:
    - klucza podstawowego, opartego na kolumnie *Id_sluchacz*;
    - unikalności wartości w polu *PESEL*;
    - określoności wartości w polach *Imie* i *Nazwisko*.

    Nazwać wszystkie warunki integralności.

153. W tabeli *Uczelnie* zdefiniować i nazwać warunki integralności dotyczące:
    - klucza podstawowego, opartego na kolumnie *Id_uczelnia*;
    - określoności wartości dla pola *Nazwa_u*.

    Nazwać wszystkie warunki integralności.

154. W tabeli *Tematy* zdefiniować i nazwać warunki integralności dotyczące:
    - klucza podstawowego, opartego na kolumnie *Id_temat*;
    - określoności i unikalności wartości dla pola *Tytul*.

    Nazwać wszystkie warunki integralności.

155. W tabeli *Zaliczenia* zdefiniować i nazwać warunki integralności dotyczące:
    - klucza podstawowego, opartego na kolumnie *Nr_zal*;
    - możliwości wprowadzenia do pola *Ocena* tylko wartości ze zbioru: 2, 3, 3.5, 4, 4.5 i 5.

156. Utworzyć powiązanie pomiędzy tabelą *Zaliczenia* i *Wykladowcy*, wykorzystując kolumnę *Id_wykladowca* jako kolumnę łączącą te tabele. Nazwać tak powstałe więzy integralności.

157. Utworzyć powiązanie pomiędzy tabelą *Zaliczenia* i *Sluchacze*, wykorzystując kolumnę *Id_sluchacz* jako kolumnę łączącą te tabele. Nazwać tak powstałe więzy integralności.

158. Utworzyć powiązanie pomiędzy tabelą *Zaliczenia* i *Tematy*, wykorzystując kolumnę *Id_temat* jako kolumnę łączącą te tabele. Nazwać tak powstałe więzy integralności.

159. Utworzyć powiązanie pomiędzy tabelą *Zaliczenia* i *Uczelnie*, wykorzystując kolumnę *Id_uczelnia* jako kolumnę łączącą te tabele. Nazwać tak powstałe więzy integralności.

160. W tabeli *Wykladowcy* zmodyfikować definicję kolumny *Pensja* tak, aby przyjmowała ona wartość równą z przedziału od 15,50 zł do 20,50 zł. Nazwać tak utworzony warunek integralności.

161. W tabeli *Wykladowcy* zmodyfikować definicję kolumny *Pensja* tak, aby przyjmowała ona wartość domyślną równą 17,50 zł.

162. W tabeli *Wykladowcy* wyłączyć kontrolę poprawności danych wprowadzanych do kolumny *Pensja*. Następnie dokonać sprawdzenia poprawności wykonania tego polecenia (określić sposób kontroli, który pozwoli stwierdzić, że więzy integralności odpowiedzialne za poprawność danych w kolumnie *Pensja* są nieaktywne).

163. W tabeli *Tematy* wyłączyć kontrolę unikalności danych wprowadzanych do kolumny *Tytul*. Następnie dokonać sprawdzenia poprawności wykonania tego polecenia (określić sposób kontroli, który pozwoli stwierdzić, że więzy integralności odpowiedzialne za unikalność danych w kolumnie *Tytul* są nieaktywne).

164. W tabeli *Uczelnie* wyłączyć kontrolę poprawności danych wprowadzanych do kolumny *Id_uczelnia* (klucz podstawowy). Należy pamiętać, że tabela *Uczelnie* jest powiązana z tabelą *Zaliczenia*. Następnie dokonać sprawdzenia poprawności wykonania tego polecenia (określić sposób kontroli, który pozwoli stwierdzić, że klucz podstawowy w tabeli *Uczelnie* jest nieaktywny).

165. W tabeli *Wykladowcy* włączyć kontrolę poprawności danych wprowadzanych do kolumny *Pensja*. Następnie dokonać sprawdzenia poprawności wykonania tego polecenia (określić sposób kontroli, który pozwoli stwierdzić, że więzy integralności odpowiedzialne za poprawność danych w kolumnie *Pensja* są ponownie aktywne).

166. W tabeli *Tematy* włączyć kontrolę unikalności danych wprowadzanych do kolumny *Tytul*. Następnie dokonać sprawdzenia poprawności wykonania tego polecenia (określić sposób kontroli, który pozwoli stwierdzić, że więzy integralności odpowiedzialne za unikalność danych w kolumnie *Tytul* są ponownie aktywne).

Temat 5.

Definiowanie perspektyw. Wykorzystanie perspektyw w zapytaniach i do realizacji operacji DML.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

167. Utworzyć perspektywę o nazwie *VStudenci*, która ograniczy widok danych o studentach tylko do danych o identyfikatorze, nazwisku i Imieniu studenta. Zapewnić możliwość wykonywania operacji DML na tabeli *Studenci* z wykorzystaniem tej perspektywy.

```
CREATE VIEW VStudenci
        AS
        SELECT ID_Student, Nazwisko, Imie
        FROM Studenci;
```

168. Utworzyć perspektywę o nazwie *VEgzaminyCKMP*, która ograniczy widok danych tylko do egzaminów przeprowadzonych w ośrodku o nazwie CKMP. W perspektywie należy umieścić te kolumny, które pozwolą określić miejsce egzaminu, osobę egzaminowaną, przedmiot oraz datę egzaminu.

```
CREATE VIEW VEgzaminyCKMP
        AS
        SELECT o.ID_Osrodek, Nazwa_Osrodek, s.ID_Student, Nazwisko, Imie,
        Nazwa_Przedmiot, Data_Egzamin
        FROM Osrodki o
        INNER JOIN Egzaminy e ON e.ID_Osrodek = o.ID_Osrodek
        INNER JOIN Studenci s ON s.ID_Student = e.ID_Student
        INNER JOIN Przedmioty p ON p.Id_przedmiot = e.Id_przedmiot ;
```

169. Utworzyć perspektywę o nazwie *VEgzaminatorStudent*. Jej zadaniem jest dostarczenie danych nt. studenta, który zdawał najwięcej egzaminów u poszczególnych egzaminatorów oraz liczby egzaminów zdawanych przez studenta egzaminatora. W perspektywie należy umieścić takie kolumny, które zapewnią jednoznaczną identyfikację egzaminatora oraz studenta.

```
CREATE VIEW VEgzaminatorStudent (ID_Egzaminator, EgzaminatorNazwisko,
EgzaminatorImie, ID_Student, StudentNazwisko, StudentImie, LiczbaMax)
        AS
        SELECT g.ID_Egzaminator, g.Nazwisko, g.Imie, s.ID_Student, s.Nazwisko,
        s.Imie, COUNT(ID_Egzamin)
        FROM Egzaminatorzy g
        INNER JOIN Egzaminy e ON e.ID_Egzaminator = g.ID_Egzaminator
        INNER JOIN Studenci s ON s.ID_Student = e.ID_Student
        GROUP BY  g.ID_Egzaminator, g.Nazwisko, g.Imie, s.ID_Student, s.Nazwisko,
        s.Imie
        HAVING COUNT(ID_Egzamin) = ( SELECT MAX(COUNT(ID_Egzamin))
                                     FROM Egzaminy e2
                                     WHERE e2.ID_Egzaminator = g.ID_Egzaminator
                                     GROUP BY ID_Student ) ;
```

170. Utworzyć perspektywę o nazwie *VOsrodkiLublin*. Jej zadaniem jest dostarczenie danych o ośrodkach znajdujących się w mieście o nazwie Lublin. Zapewnić możliwość modyfikacji danych w tabeli *Osrodki* z użyciem tej perspektywy, ograniczając operacje DML tylko do ośrodków z miasta o podanej nazwie.

```
CREATE VIEW VOsrodkiLublin
        AS
        SELECT *
        FROM Osrodki
        WHERE UPPER(Miasto) = 'LUBLIN'
        WITH CHECK OPTION ;
```

171. Wykorzystując perspektywę o nazwie *VOsrodkiLublin*, wyświetlić wszystkie dane o ośrodkach znajdujących się w mieście o nazwie Lublin. Uporządkować wynik według nazwy ośrodka.

```
SELECT *
FROM VOsrodkiLublin
ORDER BY Nazwa_Osrodek ;
```

172. Wykorzystując perspektywę o nazwie *VStudenci*, wstawić dane o nowym studencie do tabeli *Studenci*. Wprowadzić następujące dane: identyfikator – 0101011, Nazwisko – Kuraś, imię – Jan.

```
INSERT INTO VStudenci
VALUES ('0101011', ' Kuraś ', ' Jan') ;
```

173. Wykorzystując perspektywę o nazwie *VStudenci*, zmienić imię studenta o identyfikatorze 0101011 na Marian.

```
UPDATE VStudenci
SET Imie = ' Marian'
WHERE ID_Student = '0101011' ;
```

174. Wykorzystując perspektywę o nazwie *VOsrodkiLublin* oraz tabelę *Egzaminy*, wyświetlić dane o tych studentach, którzy zdawali egzaminy w ośrodkach znajdujących się w mieście o nazwie Lublin. Uporządkować wynik według nazwiska i Imienia studenta.

```
SELECT DISTINCT s.ID_Student, Nazwisko, Imie
FROM Studenci s
INNER JOIN Egzaminy e ON e.ID_Student = s.ID_Student
INNER JOIN VOsrodkiLublin v ON v.ID_Osrodek = e.ID_Osrodek
ORDER BY Nazwisko, Imie ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

175. Zdefiniować perspektywę o nazwie *VEgzaminy*, która umożliwi dostęp do danych z tabeli *Egzaminy*, z wyjątkiem danych o dacie i wyniku egzaminu.

176. Wykorzystując perspektywę o nazwie *VEgzaminy* oraz tabelę *Osrodki*, wyznaczyć liczbę egzaminów przeprowadzonych w poszczególnych ośrodkach. Do opisu ośrodka zastosować jego identyfikator oraz nazwę.

177. Wykorzystując perspektywę o nazwie *VEgzaminy*, wstawić do tabeli *Egzaminy* informacje o nowym egzaminie. Zestaw danych do wstawienia jest następujący: *ID_Egzamin* – 111, *ID_Student* – 0500324, *ID_Egzaminator* – 0004, *ID_Osrodek* – 2, *Id_przedmiot* - 7.

178. Zdefiniować perspektywę o nazwie *VOsrodkiPrzedmioty*, która umożliwi wyświetlenie informacji na temat tych przedmiotów, z których przeprowadzono najwięcej egzaminów w poszczególnych ośrodkach? Dla każdego ośrodka, w którym odbył się egzamin, należy wskazać przedmiot oraz liczbę egzaminów zdanego przedmiotu w tym, ośrodku.

179. Zdefiniować perspektywę o nazwie *VStudenciL*, która umożliwi wyświetlenie szczegółowych informacji o liczbie egzaminów zdawanych przez poszczególnych studentów. Proszę uwzględnić także tych studentów, którzy nie zdawali jeszcze żadnego egzaminu. Kolumny perspektywy należy nazwać odpowiednio: *NrAlbumu*, *Nazwisko*, *Imie*, *LiczbaEgzaminow*.

180. Zdefiniować perspektywę o nazwie *VOsrodkiMinMax*, która umożliwi wyświetlenie informacji o tych ośrodkach, w których przeprowadzono najmniej i najwięcej egzaminów. Podać identyfikator i nazwę ośrodka oraz liczbę egzaminów przeprowadzonych w ośrodku. Uwzględnić tylko te ośrodki, w których odbyły się już egzaminy.

181. Wykorzystując perspektywę *VEgzaminy*, usunąć z tabeli *Egzaminy* dane o egzaminie, którego numer wynosi 111.

182. Usunąć perspektywę o nazwie *VOsrodkiMinMax*.

183. Usunąć perspektywę o nazwie *VStudenciL*.

Temat 6.

Wykorzystanie instrukcji sterujących, zmiennych i kursorów jawnych do przetwarzania danych pobranych z tabeli.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

184. Wyświetlić liczby parzyste z przedziału od 0 do 100.

```
BEGIN
        FOR i IN 1..100 LOOP
                IF MOD(i, 2) = 0 THEN
                        Dbms_output.put_line(i) ;
                END IF ;
        END LOOP ;
END ;
```

185. Wyświetlić listę wszystkich studentów. Lista powinna zawierać 4 kolumny: Lp, identyfikator, Nazwisko oraz imię studenta. Listę uporządkować alfabetycznie według nazwiska i Imienia.

```
DECLARE
        i NUMBER DEFAULT 0 ;
        CURSOR CStudent IS SELECT ID_Student, Nazwisko, Imie FROM Studenci
        ORDER BY Nazwisko, Imie ;
BEGIN
        FOR dane IN CStudent LOOP
                i := i + 1 ;
Dbms_output.put_line(i || ' ' || dane.ID_Student || ' ' || dane.Nazwisko || ' ' ||  dane.Imie) ;
        END LOOP ;
END ;
```

186. Wskazać trzy ostatnie dni, w których przeprowadzono egzamin.

```
DECLARE
        vData  DATE ;
        CURSOR CEgzaminy IS SELECT DISTINCT Data_Egzamin FROM Egzaminy
        ORDER BY Data_Egzamin DESC ;
BEGIN
        OPEN CEgzaminy ;
        IF CEgzaminy%ISOPEN THEN
                FOR i IN 1..3 LOOP
                        FETCH CEgzaminy INTO vData ;
                        EXIT WHEN CEgzaminy%NOTFOUND ;
                        Dbms_output.put_line(vData) ;
                END LOOP ;
        CLOSE CEgzaminy ;
        END IF ;
END ;
```

187. Utworzyć trigger, który automatycznie będzie wstawiał identyfikator przedmiotu w tabeli *Przedmioty* w momencie utworzenia nowego rekordu w tej tabeli. Wartość identyfikatora ma być o jeden większa od największej wartości już istniejącej w tej kolumnie.

```
CREATE TRIGGER BI_Przedmioty BEFORE INSERT ON Przedmioty FOR EACH ROW
        DECLARE
                imax  Przedmioty.Id_przedmiot%TYPE ;
        BEGIN
                SELECT MAX(Id_przedmiot) INTO imax FROM Przedmioty ;
                :new.Id_przedmiot := imax + 1 ;
        END ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

188. Wyświetlić listę egzaminatorów, która będzie zawierała 4 kolumny: Lp, Nazwisko, imię oraz identyfikator. Uporządkować wyświetlane dane według nazwiska i Imienia egzaminatora.

189. Wyświetlić listę, zawierającą informacje o liczbie egzaminów w poszczególnych ośrodkach. Lista powinna składać się z trzech sekcji. W pierwszej sekcji należy wyświetlić ośrodki, w których liczba egzaminów jest mniejsza od 5. W drugiej sekcji powinny znaleźć się ośrodki, w których liczba egzaminów znajduje się w przedziale od 5 do 10 (włącznie). W trzeciej sekcji należy wyświetlić ośrodki, w których liczba egzaminów jest większa od 10. Każda sekcja powinna być oddzielona od poprzedniej linią przerywaną.

190. Wskazać trzy przedmioty, z których przeprowadzono najwięcej egzaminów. W odpowiedzi umieścić nazwę przedmiotu oraz liczbę egzaminów.

191. Wskazać tych studentów, którzy zdawali egzaminy w ciągu trzech ostatnich dni egzaminowania. W odpowiedzi umieścić datę egzaminu oraz dane identyfikujące studenta tj. identyfikator, imię i Nazwisko.

192. Dla każdego ośrodka o nazwie CKMP wskazać tych dwóch studentów, którzy zdawali w nim najwięcej egzaminów. W odpowiedzi umieści dane identyfikujące ośrodek tj. jego identyfikator i nazwę oraz dane identyfikujące studenta tj. jego identyfikator, Nazwisko i imię.

193. Wstawić do tabeli *Przedmioty* nowy wiersz, który będzie zawierał następujące dane o przedmiocie: identyfikator przedmiotu – *10,* nazwa przedmiotu – *Hurtownie danych*. Jeżeli przedmiot o podanym identyfikatorze już istnieje, wyświetlić komunikat "*Podany przedmiot już istnieje*". Zadanie rozwiązać dwoma sposobami: z użyciem kursora oraz zastosowanie wyjątków.

194. W tabeli *Egzaminy* w kolumnie *Punkty* wygenerować wartości numeryczne. Kolumna będzie przechowywać informację o liczbie punktów zdobytych na egzaminie przez studenta. Następnie każdemu studentowi, który zdał egzamin należy przyznać liczbę punktów z przedziału 3 do 5 (z dokładnością do dwóch miejsc po przecinku), a studentowi, który nie zdał egzaminu – liczbę punktów z przedziału 2 do 2.99 (również z dokładnością do dwóch miejsc po przecinku).

195. Wyświetlić informację o liczbie egzaminów zdawanych z poszczególnych przedmiotów przez poszczególnych studentów. Rezultat należy wyświetlić w takiej postaci, aby na początku wyświetlały się dane o studencie (identyfikator, Nazwisko, imię), a następnie informacje o liczbie egzaminów z poszczególnych przedmiotów.

196. Dla tabeli *Egzaminy* zdefiniować wyzwalacz, który będzie wstawiał wartość w polu *Punkty* w momencie wstawienia nowego rekordu lub aktualizacji rekordu istniejącego. Kolumna będzie przechowywać informację o liczbie punktów zdobytych na egzaminie przez studenta. Każdemu studentowi, który zdał egzamin, należy przyznać liczbę punktów z przedziału 3 do 5 (z dokładnością do dwóch miejsc po przecinku), a studentowi, który nie zdał egzaminu – liczbę punktów z przedziału 2 do 2.99 (również z dokładnością do dwóch miejsc po przecinku).

197. Utworzyć tabelę o nazwie *TLOEgzaminy*, która będzie zawierać informacje o liczbie egzaminów przeprowadzonych w poszczególnych ośrodkach. Następnie zdefiniować odpowiedni wyzwalacz dla tabeli *Egzaminy*, który w momencie wstawienia nowego egzaminu spowoduje aktualizację danych w tabeli *TLOEgzaminy*.

198. Utworzyć tabelę o nazwie *TLMOsrodki*, która będzie zawierała informacje o liczbie egzaminów w poszczególnych ośrodkach w kolejnych miesiącach poszczególnych lat. Tabela powinna zawierać pięć kolumn: *Rok, Miesiac, ID_Osrodek, Nazwa_Osrodek, Liczba*. Należy zdefiniować odpowiedni kod PL/SQL, który spowoduje wypełnienie tej tabeli odpowiednimi danymi.

199. Dla tabeli *Osrodki* zdefiniować odpowiedni trigger, który podczas operacji usuwania ośrodka z tej tabeli będzie kontrolował, czy w tabeli *Egzaminy* istnieją egzaminy powiązane z tym ośrodkiem. Jeśli takie egzaminy istnieją, należy wyświetlić komunikat "*Nie można usunąć ośrodka, gdyż istnieją dla niego powiązane egzaminy*".

200. Dla tabeli *Egzaminy* zdefiniować odpowiednie triggery, które będą kontrolować integralność referencyjną podczas wstawiania i modyfikacji danych o egzaminie. Integralność referencyjna ma dotyczyć danych o studencie, który zdawał egzamin.

201. Na podstawie danych znajdujących się w tabelach *Studenci*, *Przedmioty*, *Osrodki*, *Egzaminatorzy*, wygenerować dane o egzaminach i wstawić je do tabeli *Egzaminy*. Dane o poszczególnych egzaminach należy utworzyć jako iloczyn kartezjański danych z poszczególnych tabel. Egzaminy przeprowadzano w okresie od 1. stycznia 2005 roku do 15.maja 2011 roku. Każdego dnia przeprowadzono maksymalnie 25 egzaminów. Co dziesiąty egzamin był egzaminem niezdanym.

Temat 7.

Obsługa wyjątków systemowych. Deklarowanie, wywoływanie i obsługa wyjątków użytkownika.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

202. Sprawdzić, czy z przedmiotu *Bazy Danych* odbył się już egzamin. Wyświetlić odpowiedni komunikat informujący o fakcie przeprowadzenia lub nie takiego egzaminu. Zadanie wykonać z użyciem wyjątku systemowego.

```
DECLARE
        vPrzedmiot  Przedmioty.Id_przedmiot%TYPE  ;
BEGIN
        SELECT DISTINCT p.Id_przedmiot INTO vPrzedmiot FROM Przedmioty p
                JOIN Egzaminy e  ON p.Id_przedmiot = e.Id_przedmiot
                WHERE UPPER(Nazwa_Przedmiot) = 'BAZY DANYCH' ;
        Dbms_output.Put_line('Z baz danych odbył się już egzamin') ;
EXCEPTION
        WHEN NO_DATA_FOUND THEN
                Dbms_output.Put_line('Z baz danych nie było jeszcze egzaminu') ;
END ;
```

```
DECLARE
        vPrzedmiot  Przedmioty.Id_przedmiot%TYPE  ;
        vP   Egzaminy.Id_przedmiot%TYPE  ;
        CURSOR CPrzedmiot IS SELECT Id_przedmiot FROM Przedmioty
                        WHERE UPPER(Nazwa_Przedmiot) = 'BAZY DANYCH' ;
BEGIN
        OPEN CPrzedmiot ;
        IF CPrzedmiot%ISOPEN THEN
                FETCH CPrzedmiot  INTO vPrzedmiot  ;
                IF CPrzedmiot%FOUND THEN
```

```
                    SELECT DISTINCT Id_przedmiot INTO vP
                    FROM EGZAMINY
                    WHERE Id_Przedmiot = vPrzedmiot ;
                    Dbms_output.Put_line('Z baz danych odbył się już egzamin') ;
            ELSE
                    Dbms_output.Put_line('Nie ma takiego przedmiotu') ;
            END IF ;
            CLOSE CPrzedmiot ;
    END IF ;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            Dbms_output.Put_line('Z baz danych nie było jeszcze egzaminu') ;
            CLOSE CPrzedmiot ;
END ;
```

203. Który egzaminator przeprowadził więcej niż jeden egzamin? Podać jego identyfikator, Nazwisko oraz imię. Zadanie wykonać z użyciem wyjątku systemowego.

```
DECLARE
vX  Egzaminy.ID_Egzaminator%TYPE ;
 CURSOR CurEgzam IS SELECT ID_Egzaminator, Nazwisko, Imie
 FROM Egzaminatorzy ;
 VRekEgzam CurEgzam%ROWTYPE ;
BEGIN
OPEN CurEgzam ;   FETCH CurEgzam INTO vRekEgzam ;
WHILE CurEgzam%FOUND LOOP
    BEGIN
        SELECT ID_Egzaminator INTO vX FROM Egzaminy
        WHERE ID_Egzaminator = vRekEgzam.ID_Egzaminator ;
    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            Dbms_output.Put_line(vRekEgzam.ID_Egzaminator|| ' ' ||
```

```
                                    vRekEgzam.Nazwisko || ' ' || vRekEgzam.Imie);
            WHEN NO_DATA_FOUND THEN   NULL ;
        END  ;
        FETCH CurEgzam INTO vRekEgzam ;
END LOOP ;
CLOSE CurEgzam ;
END ;
```

204. Podać informację, w których ośrodkach nie przeprowadzono egzaminu. Wyświetlić identyfikator oraz nazwę ośrodka. Uporządkować wyświetlane informacje wg identyfikatora ośrodka. Zadanie wykonać wykorzystując wyjątek systemowy.

```
DECLARE
        vX   VARCHAR(1) ;
        CURSOR COsrodek IS SELECT ID_Osrodek, Nazwa_Osrodek  FROM Osrodki
                                                ORDER BY ID_Osrodek;
BEGIN
        FOR vCur IN COsrodek LOOP
                BEGIN
                        SELECT DISTINCT 'X' INTO vX FROM EGZAMINY
                                        WHERE ID_Osrodek = vCur.ID_Osrodek ;
                EXCEPTION
                        WHEN NO_DATA_FOUND THEN
                                Dbms_output.Put_line(vCur.ID_Osrodek || ' ' ||
                                vCur.Nazwa_Osrodek) ;
                END ;
        END LOOP ;
END ;
```

205. Podać informację, który student nie przystąpił do egzaminu. Podać identyfikator, Nazwisko i imię studenta. Uporządkować wyświetlane informacje wg nazwiska studenta. Zadanie wykonać wykorzystując wyjątek użytkownika.

```
DECLARE
        Uwaga  EXCEPTION ;
        CURSOR CStudent IS SELECT s.ID_Student, Nazwisko, Imie,
        COUNT(ID_Egzamin)  AS Liczba  FROM Studenci s LEFT JOIN Egzaminy e
        ON s.ID_Student = e.ID_Student   GROUP BY s.ID_Student, Nazwisko, Imie ;
BEGIN
        FOR vCur IN CStudent LOOP
                BEGIN
                        IF vCur.Liczba = 0 THEN RAISE Uwaga ;
                        END IF ;
                EXCEPTION
                        WHEN Uwaga THEN
                                Dbms_output.Put_line(vCur.ID_Student || ' ' ||
                                vCur.Nazwisko || ' ' || vCur.Imie) ;
                END ;
        END LOOP ;
END ;
```

Zadania do samodzielnego rozwiązania

206. Podać informację, z których przedmiotów nie przeprowadzono egzaminu. Wyświetlić nazwę przedmiotu. Uporządkować wyświetlane informacje wg nazwy przedmiotu. Zadanie wykonać wykorzystując wyjątek systemowy.

207. Który egzaminator i kiedy egzaminował więcej niż 5 osób w ciągu jednego dnia? Podać identyfikator, Nazwisko i imię egzaminatora, a także informacje o liczbie egzaminowanych osób oraz dniu, w których takie zdarzenie miało miejsce. Zadanie wykonać wykorzystując wyjątek użytkownika.

208. Określić liczbę egzaminów przeprowadzonych przez egzaminatora (egzaminatorów) o nazwisku *Muryjas*. Uwzględnić każdego egzaminatora, również tych, którzy jeszcze nie prowadzili egzaminów. Dla każdego egzaminatora o tym nazwisku podać jego identyfikator oraz liczbę egzaminów. Uwzględnić przypadek, iż egzaminator o tym nazwisku może nie istnieć w bazie danych. Wówczas wyświetlić komunikat "*Egzaminator o podanym nazwisku nie istnieje*". Wykorzystać wyjątki do rozwiązania zadania.

209. Przeprowadzić kontrolę, czy w ośrodku (ośrodkach) o nazwie LBS przeprowadzono egzaminy. Dla każdego ośrodka o podanej nazwie, w którym odbył się egzamin, wyświetlić odpowiedni komunikat podający liczbę egzaminów. Jeśli nie ma ośrodka o podanej nazwie, wyświetlić komunikat o treści "*Ośrodek o podanej nazwie nie istnieje*". Jeśli w ośrodku nie było egzaminu, należy wyświetlić komunikat "*Ośrodek nie uczestniczył w egzaminach*". Do rozwiązania zadania wykorzystać wyjątki systemowe i/lub wyjątki użytkownika.

210. Utworzyć kopię tabeli *Przedmioty* i nazwać ją *Kopia_Przedmioty*. Następnie dla tej tabeli zdefiniować trigger o nazwie *Trg_BI_KPrzedmioty*, który będzie kontrolował poprawność operacji wstawiania rekordów do tej tabeli. Jeśli wstawiana wartość identyfikatora istnieje już w tabeli, należy wygenerować nową wartość, większą o 1 od wartości największej. Jeśli podana nazwa nowego przedmiotu istnieje w tabeli, należy ją zastąpić ciągiem znaków *'Nazwa nieokreślona'*. Sprawdzić poprawność działania wyzwalacza.

211. Utworzyć tabelę *Wynik_Synchro*, której struktura jest zgodna ze strukturą tabeli *Przedmioty*. Następnie wstawić do tabeli *Wynik_Synchro* dodatkowe pole o nazwie *Miejsce*. Pole to będzie zawierało daną informującą o miejscu występowania rekordu, brakującego w drugiej z synchronizowanych tabel (wartości dopuszczalne w tym polu to *P* – jeśli rekord występuje tylko w tabeli *Przedmioty* lub *K* – jeśli rekord występuję tylko w tabeli *Kopia_Przedmioty*). Wygenerować informacje niezbędne do synchronizacji zawartości tabel *Przedmioty* i *Kopia_Przedmioty*, przy wykorzystaniu kodu PL/SQL (wyjątki, kursory, itp.). Jako podstawę identyczności rekordów przyjąć równość identyfikatorów porównywanych przedmiotów.

212. Zmodyfikować kod źródłowy z zadania poprzedniego (tj. 66) tak, aby podczas porównywania rekordów z tabel *Przedmioty* i *Kopia_Przedmioty* były brane pod uwagę wartości z dwóch pól tj. identyfikator i nazwa przedmiotu. Warunkiem identyczności dwóch rekordów w tych tabelach będzie równość identyfikatorów oraz nazw przedmiotów. Wynik porównania zapisać w tabeli *Wynik_Synchro* (przed wykonaniem zadania usunąć dane z tej tabeli).

213. Przeprowadzić aktualizację zawartości tabeli *Kopia_Przedmioty* w oparciu o dane znajdujące się w tabeli *Wynik_Synchro*. Wykonać kopiowanie przyrostowe tzn. nie usuwać rekordów już istniejących, a dodać do tabeli *Kopia_Przedmioty* tylko te wiersze, które występują w tabeli *Przedmioty*, a nie ma ich w tabeli *Kopia_Przedmioty*.

Temat 8.

Definiowanie kolekcji. Posługiwanie się kolekcjami w przetwarzaniu danych.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

214. Utworzyć tabelę zagnieżdżoną o nazwie *NT_Lista*, której elementy będą liczbami. Następnie zainicjować 3 elementy, nadając im odpowiednio wartość 10, 50, 100.

```
DECLARE
        TYPE Typ_NT IS TABLE OF NUMBER ;
        NT_Lista  Typ_NT := Typ_NT (10, 50, 100) ;
BEGIN
        NULL ;
END ;
```

215. Do tabeli zagnieżdżonej *NT_Lista*, utworzonej w zadaniu 80, dodać nowy element, który będzie miał wartość 150.

```
DECLARE
        TYPE Typ_NT IS TABLE OF NUMBER ;
        NT_Lista  Typ_NT := Typ_NT (10, 50, 100) ;
BEGIN
        NT_Lista.EXTEND ;
        NT_Lista(4) := 150  ;
END ;
```

216. Utworzyć tabelę zagnieżdżoną *NT_Przedmioty*, której elementami będą nazwy przedmiotów pobrane z tabeli *Przedmioty*.

```
DECLARE
        TYPE Typ_NTP IS TABLE OF Przedmioty.Nazwa_Przedmiot%TYPE ;
        NT_Przedmioty  Typ_NTP  := Typ_NTP() ;
        i NUMBER := 0 ;
        CURSOR C_Przedmioty IS SELECT Nazwa_Przedmiot FROM Przedmioty
                        ORDER BY Nazwa_Przedmiot;
BEGIN
        FOR vCur_P IN C_Przedmioty LOOP
                i := i + 1 ;
                NT_Przedmioty.EXTEND ;
                NT_Przedmioty(i) :=  vCur_P.Nazwa_Przedmiot ;
        END LOOP ;
END ;
```

217. Utworzyć tabelę o nazwie *Harmonogram*, która będzie zawierała informacje na temat egzaminów, jakie odbędą się w poszczególnych ośrodkach. Informacje te obejmują dane o ośrodku (identyfikator), nazwę przedmiotu oraz dzień, w którym odbędzie się egzamin z danego przedmiotu. Zaprojektować strukturę danych tak, aby dane nt. przedmiotu i daty były umieszczone w tabeli zagnieżdżonej, będącej jedną z kolumn tabeli *Harmonogram*.

```
CREATE OR REPLACE TYPE  Typ_Harm_Obj
            AS OBJECT (Nazwa_Przedmiot VARCHAR2(40), Data_Egzamin DATE) ;
CREATE OR REPLACE TYPE  Typ_Harm IS TABLE OF Typ_Harm_Obj ;
CREATE TABLE Harmonogram (  ID_Osrodek  NUMBER(3) PRIMARY KEY,
                                        Plan  Typ_Harm )
        NESTED TABLE Plan STORE AS Plan_tabela ;
```

218. Wykorzystując tabelę o nazwie *Harmonogram*, utworzoną w poprzednim zadaniu, wprowadzić do niej wiersze, zawierające dane dotyczące planowanych terminów egzaminów z poszczególnych przedmiotów w poszczególnych ośrodkach. Ośrodki oraz przedmioty wybrać spośród umieszczonych w tabeli *Osrodki* oraz *Przedmioty*.

```
DECLARE
  vHarm Typ_Harm := Typ_Harm(Typ_Harm_Obj('Hurtownie danych', '12-01-15')) ;
BEGIN
  INSERT INTO Harmonogram VALUES  ( 3,
                Typ_Harm(Typ_Harm_Obj('Bazy danych', '12-01-05'),
                              Typ_Harm_Obj('Języki baz danych', '12-01-20') ) ) ;
  INSERT INTO Harmonogram VALUES ( 1, vHarm) ;
END ;
```

219. W tabeli *Harmonogram* w kolumnie zagnieżdżonej *Plan* wstawić nowy wiersz. Będzie on zawierał, dla ośrodka o identyfikatorze 1, datę egzaminu równą *2012-02-15* dla przedmiotu *Business Intelligence*.

```
INSERT INTO THE(SELECT Plan FROM Harmonogram
                        WHERE ID_Osrodek = 1)
VALUES (Typ_Harm_Obj('Business Intelligence', '12-02-05')) ;
```

220. W tabeli *Harmonogram* dla ośrodka o identyfikatorze 1 w kolumnie zagnieżdżonej *Plan* usunąć wiersz, zawierający informację o terminach egzaminów z przedmiotu *Hurtownie danych*.

```
DELETE FROM THE(SELECT Plan FROM Harmonogram
                        WHERE ID_Osrodek = 1)
      WHERE UPPER(Nazwa_Przedmiot) = 'HURTOWNIE DANYCH' ;
```

221. Utworzyć tabelę o zmiennym rozmiarze o nazwie *VT_Osrodki*. Tabela powinna zawierać elementy, opisujące ośrodek (identyfikator, nazwa). Wartości elementów należy określić na podstawie tabeli *Osrodki*. Po zainicjowaniu tabeli *VT_Osrodki* wyświetlić zawartość jej elementów.

```
DECLARE
        TYPE Typ_Rec_Osr IS RECORD
                        (Id    Osrodki.ID_Osrodek%TYPE,
                        Nazwa   Osrodki.Nazwa_Osrodek%TYPE) ;
        TYPE Typ_VTO IS VARRAY(999) OF Typ_Rec_Osr ;
        VT_Osrodki  Typ_VTO := Typ_VTO();
        i  NUMBER := 0 ;
        CURSOR c_Osrodki IS SELECT ID_Osrodek, Nazwa_Osrodek FROM Osrodki ;
BEGIN
        FOR v_Rec_Osr  IN c_Osrodki LOOP
                i := c_Osrodki%ROWCOUNT  ;
                VT_Osrodki.EXTEND ;
                VT_Osrodki(i) := v_rec_Osr ;
                Dbms_Output.Put_Line(VT_Osrodki(i).Id || ' ' || VT_Osrodki(i).Nazwa) ;
        END LOOP ;
EXCEPTION
        WHEN SUBSCRIPT_BEYOND_COUNT THEN
                Dbms_Output.Put_Line ('Licznik poza zakresem') ;
END;
```

222. Utworzyć tabelę w bazie danych o nazwie *Studenci_Terminy*. Tabela powinna zawierać dwie kolumny: identyfikator studenta oraz tabelę o zmiennej długości, zawierającą daty egzaminów każdego studenta. Następnie wstawić do tabeli *Studenci_Terminy* rekordy na podstawie danych z tabeli *Egzaminy* i *Studenci*. Przed wstawieniem danych do tabeli, należy je wyświetlić, porządkując wg identyfikatora studenta.

```
CREATE OR REPLACE TYPE Typ_VT_Terminy IS VARRAY(365) OF DATE ;
CREATE  TABLE  studenci_Terminy ( ID_Student  VARCHAR2(7),
                                        Terminy  Typ_VT_Terminy ) ;
DECLARE
        i NUMBER := 0 ;
  VT_Terminy  Typ_VT_Terminy := Typ_VT_Terminy();
  CURSOR c_Studenci IS SELECT ID_Student FROM Studenci ;
```

```
  CURSOR c_Terminy (p_IdStud VARCHAR2) IS SELECT DISTINCT Data_Egzamin
            FROM Egzaminy WHERE ID_Student = p_IdStud ;
BEGIN
  FOR v_Rec_Stud IN c_Studenci LOOP
      DBMS_OUTPUT.PUT_LINE( '------------') ;
      DBMS_OUTPUT.PUT_LINE( 'Student: ' || v_Rec_Stud.ID_Student) ;
      FOR v_Rec_Terminy IN c_Terminy (v_Rec_Stud.ID_Student) LOOP
            i := c_Terminy%ROWCOUNT  ;
            VT_Terminy.EXTEND ;
            VT_Terminy(i) := v_Rec_Terminy.Data_Egzamin ;
            DBMS_OUTPUT.PUT_LINE( VT_Terminy(i)) ;
      END LOOP ;
      INSERT INTO Studenci_Terminy VALUES (v_Rec_Stud.ID_Student , VT_Terminy)
;
      VT_Terminy := Typ_VT_Terminy () ;
  END LOOP ;
END ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

223. Utworzyć tabelę zagnieżdżoną o nazwie *NT_Osrodki*, której elementy będą rekordami. Każdy rekord zawiera dwa pola: Id oraz Nazwa, odnoszące się odpowiednio do identyfikatora i nazwy ośrodka. Następnie zainicjować tabelę, wprowadzając do jej elementów kolejne ośrodki z tabeli *Osrodki*. Po zainicjowaniu wartości elementów należy wyświetlić ich wartości. Dodatkowo określić i wyświetlić liczbę elementów powstałej tabeli zagnieżdżonej.

224. Zmodyfikować kod źródłowy w poprzednim zadaniu tak, aby po zainicjowaniu tabeli zagnieżdżonej usunąć z niej elementy, zawierające ośrodki, w których nie przeprowadzono egzaminu. Dokonać sprawdzenia poprawności wykonania zadania, wyświetlając elementy tabeli po wykonaniu operacji usunięcia. Zadanie rozwiązać z wykorzystaniem podprogramów PL/SQL.

225. Utworzyć tabelę bazy danych o nazwie *Indeks*. Tabela powinna zawierać informacje o studencie (identyfikator, Nazwisko, imię), przedmiotach (nazwa przedmiotu), z których student zdał już swoje egzaminy oraz datę zdanego egzaminu. Lista przedmiotów wraz z datami dla danego studenta powinna być kolumną typu tabela zagnieżdżona. Dane w tabeli *Indeks* należy wygenerować na podstawie zawartości tabeli *Egzaminy*, *Studenci* oraz *Przedmioty*.

226. Wyświetlić z tabeli *Indeks* informacje, jakie przedmioty i kiedy zostały zdane przez poszczególnych studentów. Uporządkować wyświetlane dane wg nazwiska studenta.

227. W tabeli *Indeks* dla studenta o identyfikatorze 0000060 zmodyfikować datę egzaminu z przedmiotu *Bazy danych* tak, by była ona równa bieżącej dacie systemowej.

228. Utworzyć tabelę bazy danych o nazwie *Analityka*. Tabela powinna zawierać informacje o liczbie egzaminów poszczególnych egzaminatorów w poszczególnych ośrodkach. W tabeli utworzyć kolumny opisujące ośrodek (identyfikator oraz nazwa), egzaminatora (identyfikator, imię i Nazwisko) oraz liczbę egzaminów egzaminatora w danym ośrodku. Dane dotyczące egzaminatora i liczby jego egzaminów należy umieścić w kolumnie, będącej tabelą zagnieżdżoną. Wprowadzić dane do tabeli *Analityka* na podstawie danych zgromadzonych w tabelach *Egzaminy*, *Osrodki* i *Egzaminatorzy*.

229. Wyświetlić z tabeli *Analityka* informacje, ile egzaminów przeprowadzili poszczególni egzaminatorzy w poszczególnych ośrodkach. Uporządkować wyświetlane dane wg nazwy ośrodka.

230. Utworzyć tabelę bazy danych o nazwie *Analityka_Studenci*. Tabela powinna zawierać informacje o liczbie egzaminów każdego studenta w każdym z ośrodków. W tabeli utworzyć kolumny opisujące studenta (identyfikator, Nazwisko i imię), ośrodek (identyfikator i nazwa) oraz liczbę egzaminów studenta w danym ośrodku. Dane dotyczące ośrodka i liczby egzaminów należy umieścić w kolumnie, będącej tabelą zagnieżdżoną. Wprowadzić dane do tabeli *Analityka_Studenci* na podstawie danych zgromadzonych w tabelach *Egzaminy*, *Osrodki* i *Studenci*.

231. Wykorzystując tabelę *Analityka_Studenci*, utworzoną w poprzednim zadaniu, usunąć z niej te elementy, dla których liczba egzaminów jest równa 0.

232. Utworzyć tabelę o zmiennym rozmiarze i nazwać ją *VT_Studenci*. Tabela powinna zawierać elementy opisujące liczbę egzaminów każdego studenta. Zainicjować wartości elementów na podstawie danych z tabel *Studenci* i *Egzaminy*. Zapewnić, by studenci umieszczeni w kolejnych elementach uporządkowani byli wg liczby zdawanych egzaminów, od największej do najmniejszej. Po zainicjowaniu tabeli, wyświetlić wartości znajdujące się w poszczególnych jej elementach.

233. Zmodyfikować kod źródłowy poprzedniego zadania tak, aby usunąć z tabeli *VT_Studenci* elementy, odpowiadające studentom, którzy nie przystąpili jeszcze do egzaminu.

234. Utworzyć tabelę w bazie danych o nazwie *Przedmioty_Terminy*. Tabela powinna zawierać dwie kolumny: nazwę przedmiotu oraz tabelę o zmiennej długości, zawierającą daty egzaminów z każdego przedmiotu. Następnie wstawić do tabeli *Przedmioty_Terminy* rekordy na podstawie danych z tabeli *Egzaminy* i *Przedmioty*. Przed wstawieniem danych do tabeli, należy je wyświetlić, porządkując wg nazwy przedmiotu.

235. W tabeli *Przedmioty_Terminy* usunąć te przedmioty, z których nie przeprowadzono żadnego egzaminu. Po wykonaniu zadania sprawdzić zawartość tabeli *Przedmioty_Terminy*.

236. Utworzyć tabelę bazy danych o nazwie *Analityka_Egzaminy*. Tabela powinna zawierać informacje o liczbie niezdanych egzaminów poszczególnych studentów z poszczególnych przedmiotów oraz ogólnej liczbie egzaminów studenta z danego przedmiotu. W tabeli utworzyć kolumny opisujące studenta (identyfikator, Nazwisko i imię), przedmiot (nazwa), liczbę niezdanych egzaminów z przedmiotu oraz liczbę wszystkich egzaminów studenta z danego przedmiotu. Dane dotyczące przedmiotu oraz poszczególnych liczb należy umieścić w kolumnie, będącej tabelą zagnieżdżoną. Wprowadzić dane do tabeli *Analityka_Egzaminy* na podstawie danych zgromadzonych w tabelach *Egzaminy*, *Przedmioty* i *Studenci*.

237. Wykorzystując tabelę bazy danych o nazwie *Analityka_Egzaminy*, dla poszczególnych studentów wskazać te przedmioty, z których zdaniem mieli oni największe problemy (tzn. zdawali najwięcej razy taki przedmiot; jeśli przedmiot był zdany za pierwszym razem, wówczas należy uznać, że problemu z jego zdaniem nie było).

Temat 9.

Deklarowanie typów obiektowych. Implementacja ciała obiektu. Zastosowanie metod do pobierania i ustawiania wartości.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

238. Utworzyć typ obiektowy o nazwie *T_Student_Obj*, który umożliwi opisanie studenta. Jako atrybuty przyjąć: *ID_Student*, *Nazwisko*, *Imie*. Następnie utworzyć obiekt o nazwie *Student_Obj* typu *T_Student_Obj*. Zainicjować wartości poszczególnych atrybutów obiektu.

```
CREATE OR REPLACE TYPE T_Student_Obj  AS OBJECT (
                                    ID_Student   VARCHAR2(7) ,
                                    Nazwisko     VARCHAR2(40) ,
                                    Imie         VARCHAR2(20) ) ;
DECLARE
        Student_Obj   T_Student_Obj  :=  T_Student_Obj('0000007', 'Muryjas', 'Mateusz') ;
BEGIN
        NULL ;
END ;
```

239. Utworzyć typ obiektowy o nazwie *T_Osrodek_Obj*, który umożliwi opisanie ośrodka. Jako atrybuty przyjąć: *ID_Osrodek*, *Nazwa_Osrodek* i *Nazwa_s*. Definicja niniejszego typu obiektowego powinna również zawierać definicję metody o nazwie *Gen_Skr_Osr*, która utworzy skróconą nazwę ośrodka, składającą się z trzech pierwszych znaków nazwy ośrodka oraz liczby wygenerowanej z sekwencji *Sq_Osrodek*. Następnie w bloku PL/SQL utworzyć dwa obiekty typu *T_Osrodek_Obj* i zainicjować wartości ich atrybutów.

```
CREATE OR REPLACE TYPE T_Osrodek_Obj  AS OBJECT (
                                    ID_Osrodek   NUMBER(3) ,
                                    Nazwa_Osrodek      VARCHAR2(40) ,
                                    Nazwa_s      VARCHAR2(5),
        MEMBER PROCEDURE Gen_Skr_Osr (Nazwa_Osrodek VARCHAR2) ) ;
CREATE OR REPLACE TYPE BODY T_Osrodek_Obj  AS
        MEMBER PROCEDURE Gen_Skr_Osr (Nazwa_Osrodek VARCHAR2) IS
```

```
        BEGIN
           IF Nazwa_Osrodek IS NOT NULL THEN
                SELF.Nazwa_s := SUBSTR(Nazwa_Osrodek, 1, 2) ||
                TO_CHAR(Sq_Osrodek.NEXTVAL);
           END IF ;
        END Gen_Skr_Osr ;
END ;

DECLARE
        V_OsrObj_1 T_Osrodek_Obj := T_Osrodek_Obj (1, 'CKMP', NULL);
        V_OsrObj_2 T_Osrodek_Obj := T_Osrodek_Obj (2, 'WSIZ', NULL);
BEGIN
        V_OsrObj_1.Gen_Skr_Osr ;
        DBMS_OUTPUT.PUT_LINE(V_OsrObj_1.Nazwa_s) ;
        V_OsrObj_2.Gen_Skr_Osr;
        DBMS_OUTPUT.PUT_LINE(V_OsrObj_2.Nazwa_s) ;
END;
```

240. Utworzyć tabelę o nazwie *Studenci_Obj*. Tabela będzie zawierała jedną kolumnę typu obiektowego, opisującego studentów. Taki obiekt kolumnowy powinien zawierać trzy atrybuty: *ID_Student, Nazwisko* i *Imie*. Następnie do tabeli *Studenci_Obj* wstawić trzy obiekty (wartości atrybutów dobrać samodzielnie). Po wstawieniu danych należy wyświetlić wartości atrybutów dla poszczególnych obiektów.

```
CREATE OR REPLACE TYPE T_Student_Obj AS OBJECT
                                (ID_Student  NUMBER(5),
                                Nazwisko VARCHAR2(40),
                                Imie VARCHAR2(20)) ;
CREATE TABLE Studenci_Obj (Student  T_Student_Obj) ;
DECLARE
        v_Stud          T_Student_Obj ;
        CURSOR c_Stud IS SELECT Student FROM Studenci_Obj ;
BEGIN
        INSERT INTO Studenci_Obj VALUES(T_Student_Obj(1, 'Muryjas', 'Mateusz')) ;
        INSERT INTO Studenci_Obj VALUES(T_Student_Obj(2, 'Barć', 'Michał')) ;
```

```
        INSERT INTO Studenci_Obj VALUES(T_Student_Obj(3, 'Muryjas', 'Anna')) ;
        OPEN c_Stud ;
        LOOP
                FETCH c_Stud INTO v_Stud ;
                EXIT WHEN c_Stud%NOTFOUND ;
                DBMS_OUTPUT.PUT_LINE (v_Stud.ID_Student || ' – ' || v_Stud.Nazwisko
||
                                                ' ' || v_Stud.Imie) ;
        END LOOP ;
        CLOSE c_Stud ;
END ;
```

241. Utworzyć tabelę o nazwie *Egzaminatorzy_Obj*, która zawierała obiekty wierszowe, opisujące egzaminatorów. Obiekt wierszowy będzie składał się z trzech atrybutów: *ID_Egzaminator*, *Nazwisko* oraz *Imie*. Następnie do tabeli *Egzaminatorzy_Obj* wstawić trzy obiekty (wartości atrybutów dobrać samodzielnie). Po wstawieniu danych należy wyświetlić wartości atrybutów dla poszczególnych obiektów wierszowych.

```
CREATE OR REPLACE TYPE T_Egzaminator_Obj AS OBJECT
                        (ID_Egzaminator  NUMBER(5),
                        Nazwisko VARCHAR2(40),
                        Imie VARCHAR2(20)) ;

CREATE TABLE Egzaminatorzy_Obj OF T_Egzaminator_Obj ;

INSERT INTO Egzaminatorzy_Obj VALUES(T_Egzaminator_Obj(1, 'Muryjas', 'Mateusz')) ;
INSERT INTO Egzaminatorzy_Obj VALUES(T_Egzaminator_Obj(2, 'Barć', 'Piotr')) ;
INSERT INTO Egzaminatorzy_Obj VALUES(T_Egzaminator_Obj(3, 'Muryjas', 'Piotr')) ;

DECLARE
        v_Egzaminator T_Egzaminator_Obj ;
CURSOR c_Egzaminator IS
        SELECT VALUE(Egzam) FROM Egzaminatorzy_Obj  Egzam;
BEGIN
        OPEN c_Egzaminator ;
```

```
LOOP
        FETCH c_Egzaminator INTO v_Egzaminator ;
        EXIT WHEN c_Egzaminator%NOTFOUND ;
   DBMS_OUTPUT.PUT_LINE (v_Egzaminator.ID_Egzaminator || ' – ' ||
   v_Egzaminator.Nazwisko || ' ' || v_Egzaminator.Imie) ;
END LOOP ;
CLOSE c_Egzaminator ;
END ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

242. Utworzyć typ obiektowy o nazwie *T_Przedmiot_Obj*, opisujący przedmiot przy pomocy atrybutów *Id_przedmiot* – NUMBER(3), *Nazwa_Przedmiotelna* – VARCHAR2(100), *Nazwa_skrot* – VARCHAR2(6) oraz metody *Gen_Skrot*, generującej skrót nazwy przedmiotu na podstawie pierwszych liter z wyrazów tworzących nazwę przedmiotu.

243. Wykorzystując typ obiektowy *T_Przedmiot_Obj* z poprzedniego zadania, zdefiniować w bloku PL/SQL trzy obiekty, opisujące następujące przedmioty: *Bazy danych*, *Języki baz danych* i *Hurtownie danych*. Jako identyfikatory przedmiotów przyjąć wartości 101, 102 i 103. Po zainicjowaniu wartości atrybutów obiektów, wyświetlić te wartości.

244. Utworzyć tabelę o nazwie *Przedmioty_Obj*. Tabela będzie zawierała jedną kolumnę typu obiektowego *T_Przedmiot_Obj*, opisującego przedmioty. Wstawić do tabeli *Przedmioty_Obj* trzy obiekty, opisujące następujące przedmioty: *Zaawansowane* b*azy danych*, *Języki baz danych* i *Hurtownie danych*. Jako identyfikatory przedmiotów przyjąć wartości 101, 102 i 103. Po wstawieniu danych do tabeli wyświetlić jej zawartość.

245. Do tabeli *Przedmioty_Obj* wstawić nowe obiekty. Każdy z tych obiektów będzie odpowiadał przedmiotowi pobranemu z tabeli *Przedmioty*. Po wykonaniu zadania wyświetlić zawartość tabeli *Przedmioty_Obj*.

246. Utworzyć typ obiektowy o nazwie *T_Osrodek_Obj*, który będzie zawierał dwa atrybuty: *ID_Osrodek* (NUMBER) *Nazwa* (VARCHAR2(40)). Utworzyć typ obiektowy o nazwie *T_Adres_Obj*, zawierający następujące atrybuty: *Ulica* (VARCHAR2(30), *Numer* (VARCHAR2(8)), *Kod_poczta* (VARCHAR2(5)) i *Miasto* (VARCHAR2(35)).

247. Utworzyć tabelę o nazwie *Osrodki_Obj*, która będzie zawierała dwie kolumny: *Osrodek* typu *T_Osrodek_Obj* oraz *Adres* typu *T_Adres_Obj*. Wykorzystując tabelę relacyjną *Osrodki*, wstawić do tabeli *Osrodki_Obj* odpowiednie dane do kolumn *Osrodek* oraz *Adres*. Do kolumny *Osrodek* należy wstawić dane z pól *ID_Osrodek* oraz *Nazwa_Osrodek*, natomiast wartości z pozostałych pól tabeli relacyjnej *Osrodki* – do kolumny *Adres*. Po zakończeniu operacji wstawiania danych do tabeli *Osrodki_Obj*, należy wyświetlić jej zawartość.

248. Utworzyć typ obiektowy o nazwie *T_Egzamin_Obj*, opisujący egzamin. Typ ten będzie zawierał atrybuty identyczne jak kolumny tabeli *Egzaminy* (zgodność nazw i typów danych). Dodatkowo utworzyć metodę *Ustaw_date*, która będzie wstawiała datę systemową jako wartość atrybutu *Data_Egzamin*.

249. Wykorzystując typ obiektowy *T_Egzamin_Obj*, zdefiniowany w poprzednim zadaniu, utworzyć tabelę o nazwie *Egzaminy_Obj*, która umożliwi przechowywanie obiektów wierszowych, opisujących egzaminy. Następnie utworzyć obiekty tej tabeli, z których każdy będzie opisywał jeden egzamin. Wartości atrybutów kolejnych obiektów mają być zgodne z wartościami znajdującymi się w rekordach tabeli *Egzaminy*.

250. Zmodyfikować zawartość tabeli *Egzaminy_Obj* tak, aby wszystkie egzaminy miały wynik negatywny tj. wartość atrybutu *Zdal* dla wszystkich obiektów była równa *N*.

251. Z tabeli *Osrodki_Obj* usunąć te ośrodki, w których nie odbyły się jeszcze egzaminy. Do rozwiązania zadania wykorzystać także tabelę *Egzaminy_Obj*, w której należy kontrolować istnienie egzaminu w ośrodku przed jego usunięciem.

252. Do tabeli *Egzaminy_Obj* wstawić nowy egzamin. Podczas wstawiania obiektu wierszowego dokonać kontroli istnienia ośrodka na podstawie tabeli *Osrodki_Obj*, a przedmiotu – na podstawie tabeli *Przedmioty_Obj*. Kontrolę identyfikatorów studenta oraz egzaminatora przeprowadzić odpowiednio w oparciu o tabele relacyjne *Studenci* oraz *Egzaminatorzy*.

Temat 10.

Deklarowanie i zastosowanie funkcji oraz procedur PL/SQL do manipulowania danymi.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

253. Zdefiniować procedurę o nazwie *PDodStudent*, która pozwoli wstawić do tabeli *Studenci* dane o nowym studencie, tj. identyfikator, imię i Nazwisko studenta. Powyższe dane należy przekazać jako parametry procedury. W przypadku, gdy podany identyfikator już istnieje, należy wyświetlić odpowiedni komunikat.

```
DECLARE
        PROCEDURE PDodStudent (pId VARCHAR2, pNazwisko VARCHAR2, pImie
            VARCHAR2) IS
        BEGIN
                INSERT INTO Studenci (ID_Student, Nazwisko, Imie) VALUES (pId,
                pNazwisko, pImie) ;
        EXCEPTION
                WHEN DUP_VAL_ON_INDEX THEN
                        Dbms_output.put_line('Istnieje student o podanym
                        identyfikatorze') ;
        END PDodStudent ;
BEGIN
        PDodStudent('0000077', 'Muryjas', 'Mateusz') ;
END ;
```

254. Zdefiniować funkcję o nazwie *PJestOsrodek*, która pozwoli sprawdzić, czy istnieje ośrodek o podanym identyfikatorze. Parametrem funkcji będzie identyfikator ośrodka. W przypadku, gdy podany ośrodek już istnieje, należy wyświetlić odpowiedni komunikat. Jeśli ośrodka o takim identyfikatorze nie ma, również należy wyświetlić odpowiedni komunikat. Zadanie należy rozwiązać dwoma metodami tzn. wykorzystując kursor oraz wyjątki.

```
DECLARE
        FUNCTION PJestOsrodek (pId NUMBER) RETURN BOOLEAN IS
                istnieje BOOLEAN DEFAULT FALSE ;
                x VARCHAR2(1) ;
                CURSOR OJest IS SELECT 'X' FROM Osrodki WHERE ID_Osrodek = pId ;
        BEGIN
                OPEN OJest ;
                IF OJest%ISOPEN THEN
                        FETCH OJest INTO x ;
                        IF OJest%FOUND THEN
                                istnieje := TRUE ;
                        END IF ;
                        CLOSE OJest ;
                END IF ;
                RETURN istnieje ;
        END PJestOsrodek ;
BEGIN
        IF PJestOsrodek(1) THEN
                Dbms_output.put_line('Istnieje ośrodek o podanym identyfikatorze') ;
        ELSE
                Dbms_output.put_line('Nie ma ośrodka o podanym identyfikatorze') ;
        END IF ;
END ;
```

```
DECLARE
        FUNCTION PJestOsrodek (pId NUMBER) RETURN BOOLEAN IS
                istnieje BOOLEAN DEFAULT FALSE ;
                x VARCHAR2(1) ;
        BEGIN
                SELECT 'X' INTO x FROM Osrodki WHERE ID_Osrodek = pId ;
                istnieje := TRUE ;
                RETURN istnieje ;
        EXCEPTION
                WHEN NO_DATA_FOUND THEN
                        RETURN istnieje ;
        END PJestOsrodek ;
BEGIN
        IF PJestOsrodek(1) THEN
                Dbms_output.put_line('Istnieje ośrodek o podanym identyfikatorze') ;
        ELSE
                Dbms_output.put_line('Nie ma ośrodka o podanym identyfikatorze') ;
        END IF ;
END ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

255. W bloku PL/SQL utworzyć procedurę o nazwie *PMOsrodek*. Procedura ma za zadanie modyfikować dane dotyczące nazwy ośrodka oraz miasta, w którym on się znajduje. Identyfikator ośrodka należy przekazać jako parametr procedury. Jeśli ośrodek o podanym identyfikatorze nie istnieje, należy wyświetlić odpowiedni komunikat.

256. Utworzyć procedurę, umożliwiającą wyświetlenie identyfikatora i nazwy ośrodka, w których odbyły się już egzaminy. Następnie sprawdzić poprawność jej działania. Zadanie należy zrealizować, wykorzystując kod PL/SQL.

257. Wyświetlić informację o liczbie punktów uzyskanych z egzaminów przez każdego studenta. W odpowiedzi należy uwzględnić również tych studentów, którzy jeszcze nie zdawali egzaminów. Liczbę punktów należy wyznaczyć używając funkcji. Jeżeli student nie zdawał egzaminu, należy wyświetlić odpowiedni komunikat. Zadanie należy zrealizować, wykorzystując kod PL/SQL.

258. Dla tabeli *Egzaminy* utworzyć odpowiedni wyzwalacz, który w przypadku zmiany wartości w kolumnie *Zdal* z Y na N lub N na Y spowoduje automatyczną zmianę liczby punktów w kolumnie *Punkty* (dla Y – wartość od 3 do 5 punktów, dla N – wartość 2 do 2.99). Powyższą operację należy przeprowadzić w podprogramie (samodzielnie zdecydować, czy będzie to funkcja czy procedura).

259. Utworzyć tabelę o nazwie *TLSEgzaminy*, która będzie zawierać informacje o liczbie egzaminów zdawanych przez poszczególnych studentów. Tabela powinna zawierać 4 kolumny (identyfikator studenta, Nazwisko, imię, liczba egzaminów). Następnie zdefiniować odpowiedni wyzwalacz dla tabeli *Egzaminy*, który w momencie wstawienia nowego egzaminu spowoduje aktualizację danych w tabeli *TLSEgzaminy*. Aktualizacja danych w tabeli *TLSEgzaminy* powinna odbywać się z wykorzystaniem procedury.

260. Utworzyć tabelę o nazwie *TAZEgzaminy*, która będzie zawierała informację o liczbie zdanych i niezdanych egzaminów w poszczególnych ośrodkach. Tabela powinna zawierać cztery kolumny (identyfikator ośrodka, nazwa ośrodka, liczba zdanych i liczba niezdanych egzaminów). Następnie dla tabeli *Egzaminy* zdefiniować odpowiedni wyzwalacz, który w przypadku modyfikacji lub wstawienia nowego egzaminu spowoduje aktualizację zawartości tabeli *TAZEgzaminy*.

261. W tabeli *Studenci* dokonać aktualizacji danych w kolumnie *Nr_ECDL* oraz *Data_ECDL*. Wartość *Nr_ECDL* powinna być równa identyfikatorowi studenta, a *Data_ECDL* – dacie ostatniego zdanego egzaminu. Wartości te należy wstawić tylko dla tych studentów, którzy zdali już wszystkie przedmioty. W rozwiązaniu zastosować podprogramy typu funkcja i procedura (samodzielnie określić strukturę kodu źródłowego w PL/SQL).

262. Dla tabeli *Egzaminy* zdefiniować odpowiedni wyzwalacz, który zrealizuje aktualizację danych, opisaną w zadaniu 262. Modyfikacja danych w tabeli *Studenci* powinna odbyć się automatycznie po zdaniu przez studenta ostatniego egzaminu i wprowadzeniu danych na ten temat.

263. Utworzyć procedurę o nazwie *SprawdzRefEgzaminy*. Zadaniem procedury będzie kontrola poprawności wartości identyfikatorów: przedmiotu, ośrodka, studenta oraz egzaminatora, wprowadzanych do tabeli *Egzaminy*. Wartości identyfikatorów powinny odpowiadać istniejącym wartościom w odpowiednich tabelach. Następnie zdefiniować zmienne, których wartości będą równe wartościom poszczególnych identyfikatorów i użyć tych zmiennych w instrukcji wstawienia rekordu do tabeli *Egzaminy*. Przed wykonaniem wstawienia sprawdzić poprawność wartości zmiennych przy pomocy procedury, przekazując jako parametry poszczególne zmienne. W przypadku, gdy wartość identyfikatora nie istnieje, należy zgłosić wyjątek i obsłużyć go, wyświetlając odpowiedni komunikat.

Temat 11.

Tworzenie i modyfikacja funkcji i procedur składowanych w bazie danych. Budowanie ciała i specyfikacji pakietów. Zarządzanie procedurami, funkcjami i pakietami.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

264. Utworzyć procedurę składowaną, która wstawia nowy wiersz do tabeli *Studenci*. Parametrami procedury będą: identyfikator studenta, jego imię i Nazwisko. W procedurze należy przeprowadzić kontrolę unikalności wprowadzanej wartości identyfikatora.

```
CREATE PROCEDURE Dodaj_Studenta (PId VARCHAR2,
                        PImie VARCHAR2, PNazwisko VARCHAR2) IS
        FUNCTION Check_ID (PIdS VARCHAR2) RETURN BOOLEAN IS
                x VARCHAR2(1) ;
                BEGIN
                        SELECT 'X' INTO x FROM Studenci WHERE ID_Student = PIdS ;
                         RETURN  FALSE ;
                EXCEPTION
                        WHEN NO_DATA_FOUND THEN  RETURN TRUE ;
        END Check_ID ;

BEGIN

        IF Check_ID(PId) THEN
                INSERT INTO Studenci(ID_Student, Nazwisko, Imie)
                VALUES (PId, PNazwisko, PImie) ;
                COMMIT ;
        ELSE
                DBMS_OUTPUT.PUT_LINE('Student o podanym Id już istnieje') ;
        END IF ;
END Dodaj_Studenta ;
```

265. Utworzyć procedurę składowaną, która dokona weryfikacji poprawności daty egzaminu. Proces ten polegać będzie na sprawdzeniu, czy data ta jest większa od bieżącej daty systemowej. Jeśli tak, wówczas należy zmodyfikować taką wartość, wstawiając bieżącą datę systemową do tabeli *Egzaminy*.

```
CREATE PROCEDURE Wer_Daty_Egzaminu  IS
        CURSOR CEgzaminy IS SELECT Data_Egzamin FROM Egzaminy
                        FOR UPDATE OF Data_Egzamin ;
BEGIN
        FOR vData IN CEgzaminy LOOP
              IF vData.Data_Egzamin > SYSDATE THEN
                   UPDATE Egzaminy SET Data_Egzamin = SYSDATE
                              WHERE CURRENT OF CEgzaminy ;
                   COMMIT ;
              END IF ;
        END LOOP ;
END Wer_Daty_Egzaminu  ;
```

266. Utworzyć pakiet o nazwie *Pcg_Przedmioty*. Pakiet powinien zawierać dwie funkcje. Pierwsza z nich będzie generować kolejną wartość identyfikatora (na podstawie sekwencji o nazwie *Sq_Przedmioty*), druga – skrót nazwy przedmiotu, powstały z pierwszych znaków wyrazów tworzących tę nazwę.

```
CREATE  PACKAGE Pcg_Przedmioty AS
        FUNCTION IdGenerator RETURN NUMBER ;
        FUNCTION SkrotGenerator (Nazwa VARCHAR) RETURN VARCHAR2 ;
END Pcg_Przedmioty ;

CREATE OR REPLACE PACKAGE BODY Pcg_Przedmioty AS
FUNCTION IdGenerator RETURN NUMBER IS
         IdGen   NUMBER ;
BEGIN
        SELECT Sq_Przedmioty.NEXTVAL INTO IdGen FROM DUAL ;
        RETURN IdGen ;
```

```
END IdGenerator ;

FUNCTION SkrotGenerator (Nazwa VARCHAR) RETURN VARCHAR2 IS
        Skrot    VARCHAR2 (10) ;
        poz_spacji NUMBER(2);
        nr_spacji NUMBER(2) ;
        LNazwa VARCHAR2(50) ;
BEGIN
       LNazwa := UPPER(TRIM(Nazwa)) ;
       poz_spacji := INSTR(LNazwa, ' ') ;
       IF poz_spacji = 0 THEN
              Skrot := SUBSTR(LNazwa, 1, 1) ;
       ELSE
              nr_spacji := 1 ;
              Skrot := SUBSTR(LNazwa, 1, 1) ;
              WHILE poz_spacji  != 0 LOOP
                     Skrot := Skrot ||
                     SUBSTR(LNazwa, INSTR(LNazwa, ' ', 1, nr_spacji) +1, 1) ;
                     nr_spacji := nr_spacji + 1 ;
                     poz_spacji := INSTR(LNazwa, ' ', 1, nr_spacji) ;
              END LOOP ;
         END IF ;
         RETURN Skrot ;
END SkrotGenerator ;
END Pcg_Przedmioty ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

267. Utworzyć procedurę składowaną, która dokona weryfikacji poprawności daty ECDL w tabeli *Studenci*. Proces ten polegać będzie na sprawdzeniu, czy data ta jest większa od bieżącej daty systemowej. Jeśli tak, wówczas należy zmodyfikować taką wartość, wstawiając bieżącą datę systemową do tabeli *Studenci*.

268. Utworzyć funkcję składowaną o nazwie *Check_PESEL*, która dokona sprawdzenia poprawności wartości PESEL w tabeli *Studenci*. Proces ten powinien składać się z dwóch etapów. Pierwszy to kontrola długości ciągu znaków. Drugi – poprawny rodzaj znaków tj. tylko cyfry. Jeśli kontrolowana wartość nie spełnia powyższych warunków, funkcja powinna zwrócić wartość FALSE.

269. Utworzyć procedurę składowaną, która dokona weryfikacji poprawności wartości pola *Nr_ECDL* w tabeli *Studenci*. Weryfikacja tej wartości jest dwuetapowa. Pierwszy etap to sprawdzenie, czy wartość ta może wystąpić. Warunkiem istnienia tej wartości jest zdanie przez studenta egzaminów ze wszystkich przedmiotów, które znajdują się w tabeli *Przedmioty*. Drugi etap polega na sprawdzeniu, czy wartość występująca w tym polu w danym wierszu jest równa identyfikatorowi studenta, którego ten wiersz dotyczy.

270. Utworzyć funkcję składowaną, która będzie kontrolowała proces wprowadzania danych do tabeli *Egzaminy*. Funkcja powinna zwrócić wartość FALSE, jeśli podjęto próbę wprowadzenia egzaminu z przedmiotu, który został już zdany przez studenta. Jako parametry funkcji przyjąć identyfikator studenta, identyfikator przedmiotu oraz wynik egzaminu.

271. Utworzyć pakiet o nazwie *Pcg_Conversion*. Pakiet powinien zawierać funkcje, które będą realizować konwersję danych znakowych do postaci dużych znaków. Konwersja będzie dotyczyć następujących danych: *Studenci (Nazwisko, Imie, Miejsce, Miasto, Ulica), Egzaminatorzy (Nazwisko, Imie, Miasto, Ulica, E_mail), Osrodki (Nazwa_Osrodek, Miasto, Ulica), Przedmioty (Nazwa_Przedmiot).*

272. Utworzyć pakiet o nazwie *Pcg_Intg_Egzaminy*, który umożliwi kontrolę integralności danych wprowadzanych do tabeli *Egzaminy*. Pakiet powinien zawierać dwa podprogramy. Pierwszy z nich będzie realizował funkcjonalność związaną z kontrolą daty egzaminu (nie może być większa niż bieżąca data systemowa). Drugi podprogram powinien zapewnić, by data zdanego egzaminu przez studenta z danego przedmiotu nie była wcześniejsza niż data egzaminu niezdanego, w przypadku wystąpienia takiego egzaminu. W oby przypadkach należy przyjąć datę systemową jako datę egzaminu, jeśli została naruszona poprawność danych.

Temat 12.

Tworzenie i modyfikacja wyzwalaczy DML, DDL oraz dla wybranych zdarzeń w bazie danych. Obsługa zdarzeń zachodzących na poziomie serwera bazy danych.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

273. Dla tabeli *Egzaminy* zdefiniować trigger, związany z operacją wstawiania nowego rekordu. Będzie on automatycznie wstawiał wartość Y w polu *Zdal* w przypadku, gdy wartości takiej nie podano w instrukcji INSERT.

```
CREATE OR REPLACE TRIGGER BI_Egzaminy  BEFORE INSERT ON Egzaminy
        FOR EACH ROW
BEGIN
        IF :new.Zdal IS NULL THEN
                :new.Zdal := 'Y' ;
        END IF ;
END ;
```

274. Dla tabeli *Studenci* zdefiniować trigger, związany z operacją modyfikacji wartości w polu *ID_Student*. Jeżeli w tabeli Egzaminy istnieją powiązane rekordy z rekordem modyfikowanym, należy uprzednio dokonać aktualizacji wartości w polu *ID_Student* w tabeli *Egzaminy* tak, aby była ona równa nowej wartości pola *ID_Student* z tabeli *Studenci*.

```
CREATE OR REPLACE TRIGGER BU_Studenci
BEFORE UPDATE OF ID_Student ON Studenci FOR EACH ROW
DECLARE
        CURSOR cEgzaminy IS SELECT ID_Student FROM Egzaminy e
        WHERE e.ID_Student = :old.ID_Student  FOR UPDATE OF ID_Student;
BEGIN
        IF :new.ID_Student  != :old.ID_Student THEN
                FOR vRek IN cEgzaminy LOOP
                        UPDATE Egzaminy SET ID_Student = :new.ID_Student
```

```
                                WHERE CURRENT OF cEgzaminy ;
            END LOOP ;
        END IF ;
END ;
```

275. Zdefiniować we własnym schemacie wyzwalacz, który będzie monitorował proces tworzenia nowych obiektów w tym schemacie. W tym celu należy utworzyć tabelę *Monitor_Create*, która zawiera pola: *Object_Type* (rodzaj tworzonego obiektu) typu VARCHAR2(20), *Object_Name* (nazwa tworzonego obiektu) typu VARCHAR2(30), *Create_Date* (data utworzenia obiektu) typu DATE, *Object_Creator* (użytkownik, który utworzył obiekt) typu VARCHAR2(30).

```
CREATE OR REPLACE TRIGGER OBJ_Create   AFTER CREATE  ON  SCHEMA
        BEGIN
            INSERT INTO Monitor_Create
            VALUES (ora_dict_obj_type, ora_dict_obj_name, SYSDATE, ora_login_user ) ;
        END ;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

276. Dla tabeli *Egzaminy* zdefiniować trigger dla operacji wstawiania nowego rekordu. Będzie on automatycznie wstawiał datę systemową w polu *Data_Egzamin* w przypadku, gdy wartości takiej nie podano w instrukcji INSERT.

277. Dla tabeli *Osrodki* zdefiniować wyzwalacz, który podczas wstawiania lub modyfikacji danych dokona konwersji na duże znaki wartości wprowadzonej do pola *Miasto*.

278. Dla tabeli *Osrodki* zdefiniować odpowiedni trigger, który podczas operacji usuwania ośrodka z tej tabeli będzie kontrolował, czy w tabeli *Egzaminy* istnieją egzaminy powiązane z tym ośrodkiem. Jeśli takie egzaminy istnieją, należy wyświetlić komunikat „*Nie można usunąć ośrodka, gdyż istnieją dla niego powiązane egzaminy*”.

279. Dla tabeli *Egzaminy* zdefiniować odpowiednie triggery, które będą kontrolować integralność referencyjną podczas wstawiania i modyfikacji danych o egzaminie. Integralność referencyjna ma dotyczyć danych o studencie, który zdawał egzamin oraz przedmiotu, z którego przeprowadzono egzamin.

280. Dla tabeli *Studenci* zdefiniować odpowiedni trigger, który uniemożliwi operacje DML w tej tabeli w okresie od soboty (godz. 00:00) do niedzieli (godz. 23:56).

281. Dla tabeli *Egzaminy* zdefiniować odpowiedni wyzwalacz, który zrealizuje aktualizację danych w tabeli *Studenci* w kolumnie *Nr_ECDL* oraz *Data_ECDL*. Wartość *Nr_ECDL* powinna być równa identyfikatorowi studenta, a *Data_ECDL* – dacie ostatniego zdanego egzaminu. Wartości te należy wstawić tylko dla tych studentów, którzy zdali już wszystkie przedmioty (tj. te, które znajdują się w tabeli *Przedmioty*). Modyfikacja danych w tabeli *Studenci* powinna odbyć się automatycznie po zdaniu przez studenta ostatniego egzaminu i wprowadzeniu danych na ten temat. W rozwiązaniu zastosować podprogramy typu funkcja i procedura (samodzielnie określić strukturę kodu źródłowego w PL/SQL).

282. Dla tabeli *Egzaminy* zdefiniować wyzwalacz, który będzie wstawiał wartość w polu *Punkty* w momencie wstawienia nowego rekordu lub aktualizacji rekordu istniejącego. Wstawienie wartości 2 lub 5 w polu *Punkty* będzie odbywało się na podstawie danej z kolumny *Zdal*. Jeżeli wartość w kolumnie *Zdal* jest równa Y, wówczas należy przyznać 5 punktów. Jeżeli wartość w kolumnie *Zdal* jest równa N – 2 punkty. Jeśli wartość w polu *Zdal* jest równa NULL, przyjąć założenie, że egzamin był niezdany.

283. Utworzyć tabelę o nazwie *TLOEgzaminy*, która będzie zawierać informacje o liczbie egzaminów przeprowadzonych w poszczególnych ośrodkach. Następnie zdefiniować odpowiedni wyzwalacz dla tabeli *Egzaminy*, który w momencie wstawienia nowego egzaminu spowoduje aktualizację danych w tabeli *TLOEgzaminy*.

284. Utworzyć tabelę o nazwie *TLSEgzaminy*, która będzie zawierać informacje o liczbie egzaminów zdawanych przez poszczególnych studentów. Tabela powinna zawierać 4 kolumny (identyfikator studenta, Nazwisko, imię, liczba egzaminów). Następnie zdefiniować odpowiedni wyzwalacz dla tabeli *Egzaminy*, który w momencie wstawienia nowego egzaminu spowoduje aktualizację danych w tabeli *TLSEgzaminy*. Aktualizacja danych w tabeli *TLSEgzaminy* powinna odbywać się z wykorzystaniem procedury.

285. Utworzyć tabelę o nazwie *TAZEgzaminy*, która będzie zawierała informację o liczbie zdanych i niezdanych egzaminów w poszczególnych ośrodkach. Tabela powinna zawierać cztery kolumny (identyfikator ośrodka, nazwa ośrodka, liczba zdanych i liczba niezdanych egzaminów). Następnie dla tabeli *Egzaminy* zdefiniować odpowiedni wyzwalacz, który w przypadku modyfikacji lub wstawienia nowego egzaminu spowoduje aktualizację zawartości tabeli *TAZEgzaminy*.

286. Zdefiniować we własnym schemacie wyzwalacz, który będzie monitorował proces logowania się do bazy danych. W tym celu należy utworzyć tabelę *Monitor_User*, która zawiera pola: *User_Name* (nazwa logującego się użytkownika) typu VARCHAR2(30), *Login_Date* (data logowania się do bazy danych) typu DATE, *Login_IP* (adres IP, z którego nastąpiło logowanie do bazy danych) typu VARCHAR2.

287. Dla tabeli *Egzaminy* zdefiniować odpowiedni trigger, który umożliwi monitorowanie operacji DML w tej tabeli. Monitorowanie będzie polegać na identyfikacji rodzaju zdarzenia DML, czasie jego zaistnienia (dzień, godzina), zakresu modyfikacji (jakie pole miało zmienioną wartość). Ponadto konieczne jest określenie użytkownika, który wykonał taką operację. Dane opisujące przedstawiony monitoring zapisać do tabeli o nazwie *Monitor_DML*, którą należy samodzielnie utworzyć. Struktura tej tabeli jest następująca: *Event_User* (użytkownik wykonujący operację DML) – VARCHAR2(30), *Event_Type* (rodzaj zdarzenia) – VARCHAR2(6), *DML_Date* (data zdarzenia) – DATE, *DML_Time* (godzina i minuty zdarzenia) – TIMESTAMP, *Modyfied_Column* (zmodyfikowana kolumna, tylko dla zdarzeń typu INSERT lub UPDATE) – VARCHAR2(30).

Temat 13.

Tworzenie zapytań ad-hoc z użyciem instrukcji dynamicznego SQL-a. Wykorzystanie pakietów wbudowanych w konstrukcji zapytań ad-hoc.

Przykłady treści zadań i ich rozwiązania (w oparciu o diagram przedstawiony we wprowadzeniu)

288. Wykorzystując natywny dynamiczny SQL, wstawić do tabeli *Przedmioty* rekord z danymi o następującym przedmiocie: *Id_przedmiot* – 19, *Nazwa_Przedmiot* – Hurtownie danych, *Opis* – NULL.

```
DECLARE
        v_Komenda  VARCHAR2(255) ;
BEGIN
        v_Komenda := 'INSERT INTO Przedmioty VALUES (:vc1, :vc2, :vc3)' ;
        EXECUTE IMMEDIATE v_komenda USING 19, 'Hurtownie danych', '' ;
        COMMIT ;
END;
```

289. Zdefiniować procedurę wbudowaną *P_Create_Tab*, pozwalającą utworzyć tabelę, której nazwa oraz struktura będzie określona dynamicznie przez parametry procedury. Strukturę tabeli mogą tworzyć maksymalnie dwie kolumny. Wykorzystać natywny dynamiczny kod SQL.

```
CREATE OR REPLACE
        PROCEDURE P_Create_Tab (p_Tab VARCHAR2, p_Kol1 VARCHAR2, p_Typ1
        VARCHAR2, p_Kol2 VARCHAR2 DEFAULT NULL, p_Typ2 VARCHAR2 DEFAULT
        NULL) IS
        v_Komenda  VARCHAR2(255) ;
BEGIN
   IF p_Tab IS NOT NULL AND p_Kol1 IS NOT NULL AND p_Typ1 IS NOT NULL
        THEN
                v_Komenda := 'CREATE TABLE ' || p_Tab || '( ' || p_Kol1 || '  ' || p_Typ1 || ' )' ;

        IF p_Kol2 IS NOT NULL AND p_Typ2 IS NOT NULL THEN
                        v_Komenda := 'CREATE TABLE ' || p_Tab || '( ' || p_Kol1 || '  ' ||
```

```
                p_Typ1 || ' , ' || p_Kol2 || '  ' || p_Typ2 || ' )' ;
        END IF ;
    END IF ;
        IF v_Komenda IS NOT NULL THEN
                EXECUTE IMMEDIATE v_Komenda ;
        END IF ;
END;
```

290. Wykorzystując natywny dynamiczny SQL, zdefiniować blok PL/SQL, zawierający dynamiczną instrukcję SELECT, pozwalającą określić liczbę egzaminów poszczególnych studentów w ośrodku o identyfikatorze 1.

```
DECLARE
        TYPE Typ_Cur  IS REF CURSOR ;
        v_Cur  Typ_Cur ;    v_Student   Egzaminy.ID_Student%TYPE ;
        liczba  NUMBER ;    v_Komenda  VARCHAR2(255) ;
BEGIN
        v_Komenda := 'SELECT ID_Student , COUNT(ID_Egzamin) FROM Egzaminy
        WHERE ID_Osrodek = :ido GROUP BY ID_Student' ;
        OPEN v_Cur FOR v_Komenda USING 1 ;
        LOOP
                FETCH v_Cur INTO v_Student, liczba ;
                EXIT WHEN v_Cur%NOTFOUND ;
                CASE
                        WHEN liczba = 1 THEN
Dbms_Output.Put_Line('Student ' || v_student || ' zdawał '|| liczba || ' egzamin') ;
                        WHEN liczba BETWEEN 2 AND 4 THEN
Dbms_Output.Put_Line('Student ' || v_student || ' zdawał '|| liczba || ' egzaminy') ;
                        ELSE
Dbms_Output.Put_Line('Student ' || v_student || ' zdawał '|| liczba || ' egzaminów') ;
                END CASE ;
        END LOOP ;
END ;
```

291. Wykorzystując pakiet DBMS_SQL, wstawić do tabeli *Przedmioty* rekord z danymi o następującym przedmiocie: *Id_przedmiot* – 20, *Nazwa_Przedmiot* – Business Intelligence, *Opis* – NULL.

```
DECLARE
        v_Komenda  VARCHAR2(255) ;
        v_CurID    NUMBER ;
        v_Id  Przedmioty.Id_przedmiot%TYPE ;
        v_Nazwa  Przedmioty.Nazwa_Przedmiot%TYPE ;
        v_RecIns  NUMBER ;
BEGIN
        v_Id  := 20 ;
        v_Nazwa := 'Business Intelligence' ;
        v_Komenda := 'INSERT INTO Przedmioty (Id_Przedmiot, Nazwa_Przedmiot)
                                VALUES (:vc1, :vc2)' ;
        v_CurID := DBMS_SQL.OPEN_CURSOR ;
        DBMS_SQL.PARSE(v_CurID, v_Komenda, 1) ;
        DBMS_SQL.BIND_VARIABLE(v_CurID, ':vc1', v_Id) ;
        DBMS_SQL.BIND_VARIABLE(v_CurID, ':vc2', v_Nazwa) ;
        v_RecIns := DBMS_SQL.EXECUTE(v_CurID) ;
        DBMS_SQL.CLOSE_CURSOR(v_CurID) ;
        COMMIT ;
END;
```

292. Wykorzystując pakiet DBMS_SQL, utworzyć tabelę o nazwie *Temp*, która będzie zawierała kolumnę *Id_Temp* typu NUMBER.

```
DECLARE
         v_Komenda  VARCHAR2(255) ;
        v_CurID        NUMBER ;
        v_Tab          VARCHAR2(30) ;
        v_Kol          VARCHAR2(30)  ;
        v_Kol_Typ      VARCHAR2(30) ;
        v_RecIns       NUMBER ;
BEGIN
```

```
        v_Tab := 'Temp' ;
        v_Kol  := 'Id_Temp' ;
        v_Kol_Typ := 'NUMBER' ;
        v_Komenda := 'CREATE TABLE ' || v_Tab || ' ( ' || v_Kol || '  ' || v_Kol_Typ || ' )' ;
        v_CurID := DBMS_SQL.OPEN_CURSOR ;
        DBMS_SQL.PARSE(v_CurID, v_Komenda, 1) ;
        DBMS_SQL.CLOSE_CURSOR(v_CurID) ;
END;
```

293. Wykorzystując pakiet DBMS_SQL, utworzyć zapytanie dynamiczne, umożliwiające wyświetlenie wyników analizy danych zagregowanych w tabeli *Egzaminy*. Kryterium grupowania oraz rodzaj funkcji agregującej należy określić dynamicznie na etapie wykonania zapytania. Powstały kod PL/SQL umieścić w procedurze PL/SQL o nazwie *P_Analytics*, której parametrami będą kryterium grupowania oraz rodzaj funkcji agregującej.

```
CREATE OR REPLACE PROCEDURE P_Analytics  (p_Kol VARCHAR2,
                p_F_Agg  VARCHAR2) IS
        v_Komenda  VARCHAR2(255) ;
        v_CurID        NUMBER ;
        v_Id             VARCHAR2(255) ;
        v_Liczba       NUMBER ;
        v_RecSel       NUMBER ;
BEGIN
        v_Komenda := 'SELECT ' || p_Kol || ', ' || p_F_Agg || ' FROM Egzaminy GROUP BY '
        || p_kol ;
        v_CurID := DBMS_SQL.OPEN_CURSOR ;
        DBMS_SQL.PARSE(v_CurID, v_Komenda, 1) ;
        DBMS_SQL.DEFINE_COLUMN(v_CurID, 1, v_Id, 255) ;
        DBMS_SQL.DEFINE_COLUMN(v_CurID, 2, v_Liczba) ;
        v_RecSel := DBMS_SQL.EXECUTE (v_CurID) ;
        LOOP
                IF DBMS_SQL.FETCH_ROWS(v_CurID) > 0 THEN
                        DBMS_SQL.COLUMN_VALUE(v_CurID, 1, v_Id) ;
                        DBMS_SQL.COLUMN_VALUE(v_CurID, 2, v_Liczba) ;
```

```
                DBMS_OUTPUT.PUT_LINE(v_Id || ' liczba egzaminów: ' || v_Liczba) ;
            ELSE
                EXIT ;
            END IF ;
        END LOOP ;
        DBMS_SQL.CLOSE_CURSOR(v_CurID) ;
END;
```

Zadania do samodzielnego rozwiązania (należy wykorzystać diagram przedstawiony we wprowadzeniu)

294. Zdefiniować dynamiczne zapytanie, w którym nazwa tabeli oraz nazwy kolumn, zawierające wybierane dane, zostaną podane w momencie wykonywania instrukcji wyboru. Zapytanie może zawierać maksymalnie 5 kolumn. Następnie podczas wykonywania tego zapytania przyjąć następujące wartości zmiennych dowiązanych: nazwa tabeli – *Osrodki*, kolumny – *ID_Osrodek, Nazwa_Osrodek*. Wykorzystać natywny dynamiczny kod SQL.

295. Utworzyć procedurę wbudowaną o nazwie *P_Query_Simple*, która będzie wykonywała operację selekcji i analizy danych na jednej tabeli. Nazwa tabeli, nazwy kolumn oraz rodzaj analizy będą określone dynamicznie na etapie wykonywania programu (np. w bloku PL/SQL). Rodzaj analizy zostanie zdefiniowany przez nazwę funkcji agregującej oraz kolumnę, na której będzie ona wykonywała obliczenia. Wykorzystać natywny dynamiczny kod SQL.

296. Utworzyć procedurę wbudowaną o nazwie *P_Query_Complex*, która będzie umożliwiała wykonanie dynamicznego zapytania, wykorzystującego dwie tabele powiązane ze sobą. Nazwy tabel, rodzaj złączenia, kolumny definiujące relację między tabelami oraz kolumny wybierane z tabel zostaną określone dopiero podczas wywołania procedury. Wykorzystać natywny dynamiczny kod SQL.

297. Utworzyć procedurę wbudowaną o nazwie *P_Query_All*, która umożliwi wykonanie zapytania dynamicznego, zawierającego wszelkie klauzule możliwe do wykorzystania w instrukcji SELECT. Jako parametry procedury zdefiniować kolejno: nazwę tabeli, nazwy kolumn, eliminowanie powtarzalnych wartości w zbiorze wynikowym, warunek wybierania rekordów, kryteria grupowania, rodzaj funkcji agregującej, warunek wyboru grup rekordów oraz kryteria sortowania.

298. Zdefiniować pakiet o nazwie *Pcg_Tab_DDL1*, umożliwiający tworzenie oraz usuwanie tabeli. Pakiet powinien zawierać dwa podprogramy. Pierwszy z nich będzie realizował operację tworzenia tabeli, drugi – usuwania tabeli. Parametrami przekazywanymi do odpowiednich podprogramów pakietu będą nazwa tabeli, nazwy kolumn tabeli oraz typy danych kolumn. Uwzględnić sytuację, w której tabela o podanej nazwie już istnieje (w przypadku instrukcji CREATE TABLE) oraz gdy tabela nie istnieje (dla instrukcji DROP TABLE). Wykorzystać natywny dynamiczny kod SQL.

299. Wykorzystując pakiet DBMS_SQL, wstawić do tabeli *Osrodki* rekord z danymi o następującym ośrodku: *ID_Osrodek* – 33, *Nazwa_Osrodek* – WSIZ, *Miasto* – Rzeszów. Pozostałe pola pozostawić puste. Uwzględnić sytuację, w której ośrodek o podanym identyfikatorze już istnieje w tabeli.

300. Wykorzystując pakiet DBMS_SQL, zmodyfikować w tabeli *Przedmioty* rekord zawierający przedmiot o nazwie *Business Intelligence*. Modyfikacja polegać będzie na wstawieniu w polu *Opis* wartości *'Analizy biznesowe'*. Uwzględnić sytuację, w której przedmiot taki nie istnieje.

301. Zdefiniować dynamiczne zapytanie, które umożliwi określenie liczby egzaminów w ośrodku (ośrodkach) o podanej nazwie. Zapytanie to umieścić w procedurze PL/SQL, której parametrem będzie nazwa ośrodka. Wartość parametru zostanie podana w momencie wywołania procedury. Wykorzystać pakiet DBMS_SQL do wykonania zadania.