Mateusz Budzyński 210147
Arkadiusz Domrat 210166

Projekt Zaawansowane System Baz Danych lato 2017/18
Sprawozdanie
System bazodanowy parku zoologicznego

I Opis i założenia

Zaprojektowany system bazodanowy zapewnia sprawne obsługiwanie parku zoologicznego. Stworzono możliwość przechowywania i przetwarzania danych dotyczących bezpośrednio zoo (takich jak przebywające w nim zwierzęta, istniejące klatki i pawilony, zatrudnieni pracownicy), jaki i odnoszących się do ogólnych faktów nt. gatunków zwierząt oraz ich środowisk naturalnych. System oferuje szereg automatycznie wykonywanych działań oraz narządzi ułatwiających działania administratorskie. Przygotowano także rozwiązania umożliwiające uzyskiwanie informacji nieprecyzyjnych.

System składa się z następujących tabel:

1. „zwierzeta” opisuje każde zwierzę w zoo
   - „zwierze_id” - int NOT NULL, PRIMARY KEY
   - „gatunek_id” - int NOT NULL, foreign key references „gatunki”(„gatunek_id”)
   - „imie” - varchar(30)
   - „data_urodzenia” - date, is null or <= „data_przybycia”
   - „data_przybycia” - date, NOT NULL
   - „klatka_id” - int, NOT NULL, foreign key references „klatki”(„klatka_id”)
2. „gatunki” opisuje gatunki, którymi zoo się interesuje
   - „gatunek_id” – int, NOT NULL, PRIMARY KEY
   - „nazwa” - varchar(30), NOT NULL
   - „wiek_dojrzalosci” - int, NOT NULL
   - „biom_id” - int NOT NULL foreign key references „biomy”(„biom_id”)
3. „biom” opisuje biomy odpowiednie dla gatunków
   - „biom_id” – int, NOT NULL, PRIMARY KEY
   - „nazwa” – varchar(30), NOT NULL
   - „temperatura” - int, NOT NULL
   - „wilgotnosc” - int, NOT NULL
4. „pawilony” określa biom jaki został zaadaptowany w pawilonach zoo.
   - „pawilon_id” int, NOT NULL, PRIMARY KEY
   - „biom_id” - int, NOT NULL, foreign key references „biomy”(„biom_id”)
5. „klatki” opisuje każdą klatkę w zoo
   - „klatka_id” - int, NOT NULL, PRIMARY KEY
   - „pawilon_id” - int, NOT NULL, foreign key references „pawilony”(„pawilon_id”)
   - „gatunek_id” - int, NOT NULL, foreign key references „gatunki”(„gatunek_id”)
   - „liczba_zwierzat” – int, NOT NULL [maksymalna liczba zwierząt]
6. „pracownicy” opisuje każdego pracownika zoo
   - „pracownik_id” - int, NOT NULL, PRIMARY KEY

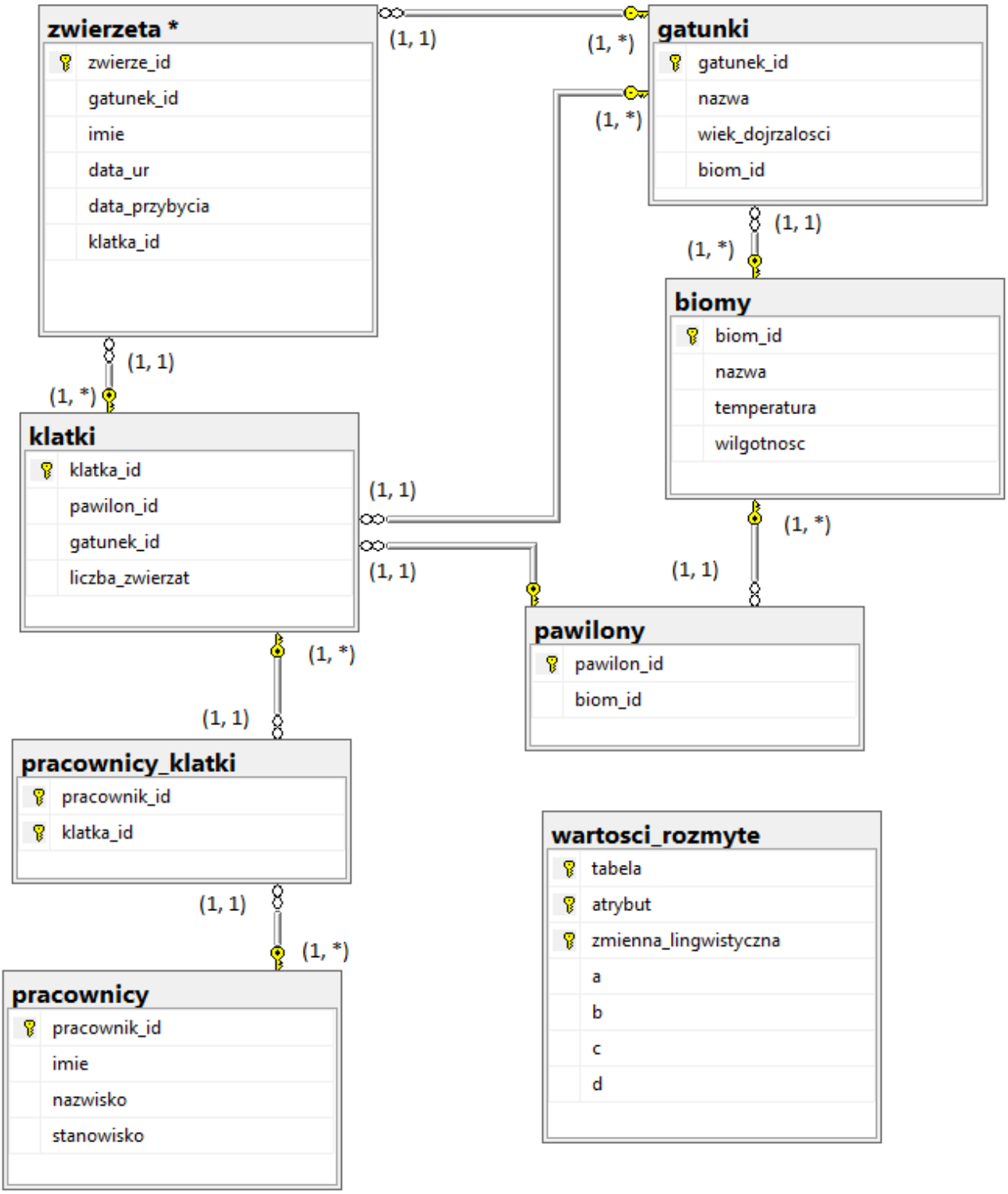
- „imie” - varchar(30), NOT NULL
- „nazwisko” – varchar(30), NOT NULL
- „ stanowisko” - varchar(30), NOT NULL

7. „pracownicy_klatki” określa przyporządkowania między pracownikami a klatkami.
   - „pracownik_id” - int, NOT NULL, PRIMARY KEY, foreign key references „pracownicy”(„pracownik_id”)
   - „klatka_id” - int, NOT NULL, PRIMARY KEY, foreign key references „klatki”(„klatka_id”)
8. „wartosci_rozmyte” określa i definiuje możliwe wartości rozmyte występujące w innych tabelach.
   - „tabela” - varchar(30), NOT NULL, PRIMARY KEY [tabela, której dotyczy zbiór rozmyty]
   - „atrybut” - varchar(30), NOT NULL, PRIMARY KEY [atrybut, którego dotyczy zbiór rozmyty]
   - „zmienna_lingwistyczna”- varchar(30) NOT NULL, PRIMARY KEY
   - „a” - int, NOT NULL, <=b [od jakiej wartości przynależność niezerowa]
   - „b” - int, NOT NULL, <=c [od jakiej wartości przynależność pełna]
   - „c” - int, NOT NULL,<=d [do jakiej wartości przynależność pełna]
   - „d” - int, NOT NULL [do jakiej wartości przynależność niezerowa]

Założenia:

1. w klatce mogą przebywać jedynie zwierzęta odgórnie określonego gatunku
2. klatką może zajmować się dowolna liczba pracowników, a pracownik może zajmować się dowolną liczbą klatek
3. data przybycia zwierzęcia do zoo jest znana, data urodzenia nie musi być
4. zwierze nie musi być nazwane
5. biom klatki określić można na podstawie pawilonu, w którym się ona znajduje
6. zwierzęta nie muszą przebywać w odpowiednim dla nich biomie
7. gatunek każdego zwierzęcia jest znany
8. biom każdego pawilonu jest znany
9. w klatce nie może przebywać więcej zwierząt niż jest to określonego
10. nie można usunąć klatki, w której znajduje się choć jedno zwierzę
11. stanowisko pracownika nie ogranicza jego obowiązków
12. zwierzęta różniące się wiekiem mogą przebywać w jednej klatce
13. każda klatka przebywa w jakimś pawilonie
14. zwierzęta mogą rodzić się w zoo

II Diagram


zwierzeta *
zwierze_id
gatunek_id
imie
data_ur
data_przybycia
klatka_id
gatunki
gatunek_id
nazwa
wiek_dojrzalosci
biom_id
biomy
biom_id
nazwa
temperatura
wilgotnosc
klatki
klatka_id
pawilon_id
gatunek_id
liczba_zwierzat
pawilony
pawilon_id
biom_id
pracownicy_klatki
pracownik_id
klatka_id
pracownicy
pracownik_id
imie
nazwisko
stanowisko
wartosci_rozmyte
tabela
atrybut
zmienna_lingwistyczna
a
b
c
d
(1, 1)
(1, *)

III Wyzwalacze

1. Wyzwalacz „dod_zwierze_trig” wywołuje się zamiast dodania nowego zwierzęta. Jeśli wybrana dl zwierzęta klatka jest pełna klatka, dodanie zwierzęcia nie nastąpi i wyświetlony zostanie stosowny komunikat.

```
CREATE TRIGGER dod_zwierze_trig
        ON zwierzeta
  INSTEAD OF INSERT
  as
        begin
                declare @klatka_id int
                select @klatka_id = klatka_id from  inserted
                if (SELECT COUNT(klatka_id) from zwierzeta where klatka_id = @klatka_id) <
(SELECT liczba_zwierzat from klatki where klatka_id = @klatka_id)
                        INSERT INTO zwierzeta SELECT * from inserted
                else
                        print('BŁĄD! W tej klatce nie ma już miejsca')
  END;
go
```

2. Wyzwalacz „usun_klatke_trig” wywołuje się zamiast usunięcia klatki. Jeśli wybrana do klatka nie jest pusta, jej usunięcie nie nastąpi i wyświetlony zostanie stosowny komunikat.

```
CREATE TRIGGER usun_klatke_trig
        ON klatki
  INSTEAD OF delete
  as
        begin
                declare @klatka_id int
                select @klatka_id = klatka_id from deleted
                if NOT EXISTS(SELECT zwierze_id from zwierzeta where klatka_id =
@klatka_id)
                        DELETE FROM klatki where klatka_id = (select klatka_id from deleted)
                else
                        print('BŁĄD! Nie można usunąć klatki, ponieważ są w niej zwierzęta')

  END;
```

3. Wyzwalacz „aktualizuj_imie_trig” wywołuje się po uaktualnieniu tabeli zwierzęta. Po wybraniu nazwy dla nienazwanego zwierzęcia o nieznanej dacie urodzenia, jako jego data urodzenia przyjęty zostanie aktualny dzień i miesiąc oraz rok poprzedzający rok przybycia do zoo.

```
CREATE TRIGGER aktualizuj_imie_trig
        ON zwierzeta
   AFTER UPDATE
   as
        begin
        if exists (select * from deleted where imie is null)
                update zwierzeta
                set data_urodzenia = DATEADD(YEAR, -1 * (1+DATEDIFF(YEAR, data_przybycia,
getdate())), getdate())
```

```
            where data_urodzenia is null and zwierze_id in (select zwierze_id from
deleted)
  END
```

IV Funkcje i procedury

1. Funkcja „zapytanie_rozmyte” oblicza stopień spełnienia przynależności danej wartości do danego zbioru rozmytego. Jako argumenty przyjmuje rozpatrywany atrybut, jego wartość, tabelę, z której pochodzi oraz zmienną lingwistyczną, określającą zbiór rozmyty. Funkcja oblicza, na podstawie danych zawartych w tabeli „wartosci_rozmyte” i zwraca liczbę rzeczywistą z przedziału [0,1] reprezentującą stopień spełnienia przynależności.

```
create function zapytanie_rozmyte(@tabela varchar(30), @atrybut varchar(30),
@zmienna_lingwistyczna varchar(30), @wartosc int)
        returns float
as
begin
        declare @a float, @b float, @c float, @d float
        select @a = a,@b = b,@c = c,@d =d
        from Zoo..wartosci_rozmyte
        where tabela = @tabela and atrybut = @atrybut and zmienna_lingwistyczna =
@zmienna_lingwistyczna
        if @wartosc <= @a or @wartosc >= @d
                return 0
        if @wartosc >= @b and @wartosc <= @c
                return 1
        if @wartosc > @a and @wartosc < @b
                return ROUND((@wartosc/(@a + @b)),2)

        return ROUND((@wartosc/(@c + @d)),2)

end
```

2. Funkcje „minimum” i „maksimum” zwracają odpowiednio mniejszą lub większą liczbę rzeczywistą spośród dwóch podanych jako argumenty.

```
create function minimum(@a float, @b float)
        returns float
as
begin
        if @a<@b
                begin
                        return @a
                end
        return @b
end

create function maksimum(@a float, @b float)
        returns float
```

```
as
begin
	if @a>@b
		begin
			return @a
		end
	return @b
end
```

3. Procedura „przenies_zwierzeta” służy do przeniesienia zwierząt między klatkami, których numery przyjmowane są jako argumenty. Procedura nie dokona przeniesienia i wypisze odpowiedni komunikat jeśli w klatce docelowej nie będzie wystarczająco miejsca lub jeśli przystosowano ona będzie dla innych gatunków niż klatka pierwotna.

```
create procedure przenies_zwierzeta @klatka_z int, @klatka_do int
as
declare
		@l_zwierzat int,
		@wolne_miejsce int
begin
	set @l_zwierzat = (select count(zwierze_id)
	from zwierzeta
	where klatka_id = @klatka_z)
	set @wolne_miejsce = (select liczba_zwierzat
					from klatki
					where klatka_id = @klatka_do) -
					(select count(zwierze_id)
					from zwierzeta
					where klatka_id = @klatka_do)
	if @l_zwierzat > @wolne_miejsce
		print('BŁĄD! W klatce, do której przynosisz zwierzęta nie ma wystarczajaco
wolnego miejsca.')
	else if (SELECT gatunek_id
		from klatki
		where klatka_id = @klatka_z) <>
		(SELECT gatunek_id
		from klatki
		where klatka_id = @klatka_do)
	print('BŁĄD! W klatce, z której przenosisz są trzymane zwierzęta innego gatunku niż w
klatce docelowej')
	else
	update zwierzeta
	set klatka_id = @klatka_do
	where klatka_id = @klatka_z

end
```

4. Funkcja „wyszukaj_imie” służy do znajdowania zwierząt, których imiona zaczynają się od podanej jako argumenty frazy. Zwracana jest tabela zawierająca pasujące imiona oraz gatunki danych zwierząt.

```
create function wyszukaj_imie(@fraza varchar(30))
	returns table
as
	return (select imie, nazwa as gatunek
			from zwierzeta, gatunki
			where zwierzeta.gatunek_id = gatunki.gatunek_id
				and imie is not null
				and imie like @fraza + '%')
```

5. Procedura „usun_obowiazki_pracownika” usuwa najstarsze przypisania klatek do pracowników, którzy zajmują się liczbą klatek większą od podawanej jako argument. Procedura wykorzystuje kursor oraz domyślną wartość argumentu.

```
create procedure usun_obowiazki_pracownika @max_liczba_klatek int=5
as
begin
declare @liczba_klatek int,
	@id_pracownik int
DECLARE
	cur cursor for select pracownik_id, count(klatka_id) as liczba_klatek from
pracownicy_klatki group by pracownik_id

	OPEN cur;
	FETCH NEXT FROM cur INTO @id_pracownik, @liczba_klatek;
	WHILE @@FETCH_STATUS=0
	BEGIN
		if(@liczba_klatek>@max_liczba_klatek)
		begin
			delete pracownicy_klatki
			where klatka_id in (select top (@liczba_klatek-@max_liczba_klatek)
klatka_id from pracownicy_klatki where @id_pracownik=pracownik_id ) and
pracownik_id=@id_pracownik
		end
		FETCH NEXT FROM cur INTO @id_pracownik, @liczba_klatek;
	END
	CLOSE cur
	DEALLOCATE cur
```

6. Procedura „dodaj_pawilon” tworzy pawilon o biomie, którego nazwa podawana jest jako argument. Poprzez efekt uboczny procedura zwraca zakodowany wynik działania (powodzeniu lub nie).

```
create procedure dodaj_pawilon @nazwa_biomu varchar(30), @kod_bledu int output
as
declare @pawilon_id int,
		@biom_id int
begin
	if(not exists (select * from biomy where nazwa=@nazwa_biomu))
		begin
		set @kod_bledu = 1
		return
		end
	select @pawilon_id=max(pawilon_id)+1 from pawilony
	select @biom_id=biom_id from biomy where nazwa = @nazwa_biomu
	insert into pawilony (pawilon_id, biom_id) values (@pawilon_id, @biom_id)
	set @kod_bledu = 0

end
```

V Zapytania

W celu przedstawienia zamieszczonych w bazie danych, zrealizowane zostały następujące zapytania (część wyników nie została przedstawiona w całości):

1. Biomy o "przeciętnej" wilgotności i "ciepłej" temperaturze:

```
SELECT nazwa,
    wilgotnosc,
    temperatura,
    dbo.minimum(dbo.zapytanie_rozmyte('biomy', 'wilgotnosc', 'przeciętny', wilgotnosc), dbo.zapytanie_rozmyte('biomy', 'temperatura', 'ciepły', temperatura)) AS przynaleznosc
    FROM biomy WHERE dbo.minimum(dbo.zapytanie_rozmyte('biomy', 'wilgotnosc', 'przeciętny', wilgotnosc), dbo.zapytanie_rozmyte('biomy', 'temperatura', 'ciepły', temperatura))>0
```

	nazwa	wilgotnosc	temperatura	przynaleznosc
1	Sawanna	25	30	1
2	Las deszczowy	80	24	0,53

2. Klatki, w których panuje "wilgotna" wilgotności lub "zimna" temperatura

```
SELECT klatka_id,
    wilgotnosc,
    temperatura,
    dbo.maksimum(dbo.zapytanie_rozmyte('biomy', 'wilgotnosc', 'wilgotny', wilgotnosc), dbo.zapytanie_rozmyte('biomy', 'temperatura', 'zimny', temperatura)) AS przynaleznosc
FROM biomy,
    klatki,
    pawilony WHERE klatki.pawilon_id = pawilony.pawilon_id
AND pawilony.biom_id = biomy.biom_id
    AND dbo.maksimum(dbo.zapytanie_rozmyte('biomy', 'wilgotnosc', 'wilgotny', wilgotnosc), dbo.zapytanie_rozmyte('biomy', 'temperatura', 'zimny', temperatura))>0
```

	klatka_id	wilgotnosc	temperatura	przynaleznosc
1	6	80	24	1
2	7	80	24	1
3	8	80	24	1
4	9	80	24	1
5	13	13	-20	1
6	14	13	-20	1
7	15	13	-20	1
8	16	13	-20	1

3. Zwierzęta, które nie mieszkają we właściwym biomie

```
SELECT zwierze_id,
    imie,
    gatunki.nazwa,
```

```
        zwierzeta.klatka_id
FROM zwierzeta,
    gatunki,
    klatki,
    pawilony WHERE gatunki.gatunek_id = zwierzeta.gatunek_id
AND zwierzeta.klatka_id = klatki.klatka_id
AND klatki.pawilon_id = pawilony.pawilon_id
AND pawilony.biom_id <> gatunki.biom_id
```

	zwierze_id	imie	nazwa	klatka_id
1	4	Alex	Lew	9
2	8	NULL	Lew	9
3	9	NULL	Lew	9
4	46	Mruczek	Ryś	16

4. Pracownicy nie zajmujący się żadną klatką

```
SELECT *
FROM pracownicy WHERE NOT EXISTS
  (SELECT *
   FROM pracownicy_klatki
   WHERE pracownicy_klatki.pracownik_id = pracownicy.pracownik_id)
```

	pracownik_id	imie	nazwisko	stanowisko
1	3	Piotr	Wiśniewski	Treser
2	4	Paweł	Kowalski	Pomocnik

5. Pracownicy obsługujący wszystkie klatki z lwami

```
SELECT *
 FROM pracownicy WHERE NOT EXISTS
  ( SELECT *
   FROM klatki
   WHERE gatunek_id =
     (SELECT gatunek_id
      FROM gatunki
      WHERE nazwa = 'Lew')
    AND NOT EXISTS
     (SELECT *
```

```
        FROM pracownicy_klatki
        WHERE pracownik_id = pracownicy.pracownik_id
          AND klatka_id = klatki.klatka_id))
```

	pracownik_id	imie	nazwisko	stanowisko
1	1	Jan	Kowalski	Pomocnik
2	2	Jan	Nowak	Specjalista

6. Liczba zwierząt, którą opiekuje się każdy pracownik posortowana malejąco

```
SELECT pracownik_id,
        COUNT(zwierze_id) AS liczba_zwierzat
  FROM zwierzeta,
       pracownicy,
       klatki WHERE zwierzeta.klatka_id = klatki.klatka_id
  AND klatki.klatka_id IN
    (SELECT klatka_id
     FROM pracownicy_klatki
     WHERE pracownik_id = pracownicy.pracownik_id)
GROUP BY pracownik_id
ORDER BY liczba_zwierzat DESC
```

	pracownik_id	liczba_zwierzat
1	1	15
2	2	12
3	5	11
4	10	10
5	7	8
6	8	5
7	6	4
8	9	1

7. Zwierzęta, które nie mają imion

```
SELECT zwierze_id,
       nazwa AS gatunek,
       klatka_id
FROM zwierzeta,
    gatunki
```

```
WHERE zwierzeta.gatunek_id = gatunki.gatunek_id
 AND imie IS NULL
```

	zwierze_id	gatunek	klatka_id
1	8	Lew	9
2	9	Lew	9
3	10	Antylopa gnu	3
4	11	Antylopa gnu	3
5	12	Antylopa gnu	3
6	13	Antylopa gnu	3
7	19	Żyrafa	4
8	20	Żyrafa	4

8. Gatunki zwierząt, które są w więcej niż jednym pawilonie

```
SELECT nazwa AS gatunek
 FROM gatunki WHERE
  (SELECT COUNT(gatunek_id)
   FROM klatki
   WHERE gatunek_id = gatunki.gatunek_id) > 1
```

	gatunek
1	Lew
2	Żyrafa
3	Krokodyl
4	Lis
5	Ryś

9. Ile zwierząt przybyło do zoo w każdym roku, posortowane po roku malejąco

```
SELECT YEAR(data_przybycia) AS rok,
       COUNT(zwierze_id) AS liczba_zwierząt
 FROM zwierzeta
GROUP BY YEAR(data_przybycia)
ORDER BY rok DESC
```

	rok	liczba_zwierząt
1	2018	2
2	2017	2
3	2016	2
4	2015	2
5	2012	3
6	2011	1
7	2010	9
8	2009	1

10. Najstarsze zwierze

```
SELECT zwierze_id,
       imie,
       datediff(YEAR, data_urodzenia, getdate()) AS wiek
FROM zwierzeta
WHERE data_urodzenia IS NOT NULL
  AND data_urodzenia =
    (SELECT MIN(data_urodzenia)
     FROM zwierzeta
     WHERE data_urodzenia IS NOT NULL)
```

	zwierze_id	imie	wiek
1	46	Mruczek	23

11. Zwierzęta, które są starsze niż wynosi średni wiek dla ich gatunku

```
SELECT zwierze_id,
        nazwa AS gatunek,
        datediff(YEAR, data_urodzenia, getdate()) AS wiek
  FROM zwierzeta,
       gatunki WHERE zwierzeta.gatunek_id = gatunki.gatunek_id
  AND data_urodzenia IS NOT NULL
  AND datediff(YEAR, data_urodzenia, getdate()) >
    (SELECT AVG(datediff(YEAR, data_urodzenia, getdate()))
     FROM zwierzeta
     WHERE gatunek_id = gatunki.gatunek_id
       AND data_urodzenia IS NOT NULL)
```

	zwierze_id	gatunek	wiek
1	2	Lew	14
2	4	Lew	19
3	5	Lew	16
4	7	Lew	17
5	10	Antylopa gnu	21
6	21	Krokodyl	11
7	25	Goryl	17
8	27	Lis	3

12. Klatki z wolnymi miejscami i liczba miejsc w nich

```
SELECT klatka_id,
       pawilon_id,
       nazwa AS gatunek,
       liczba_zwierzat -
  (SELECT COUNT(zwierze_id)
   FROM zwierzeta
   WHERE klatka_id = klatki.klatka_id) AS liczba_miejsc
FROM klatki,
     gatunki WHERE klatki.gatunek_id = gatunki.gatunek_id
AND liczba_zwierzat >
  (SELECT COUNT(zwierze_id)
   FROM zwierzeta
   WHERE klatka_id = klatki.klatka_id)
```

	klatka_id	pawilon_id	gatunek	liczba_miejsc
1	2	1	Lew	1
2	3	1	Antylopa gnu	4
3	4	1	Żyrafa	5
4	5	2	Żyrafa	10
5	6	3	Krokodyl	11
6	7	3	Goryl	1
7	8	3	Krokodyl	7
8	9	3	Lew	1

13. Klatki, w których występuje zwierze przybyłe w tym roku

```
SELECT klatki.klatka_id,
       nazwa AS gatunek
FROM klatki,
     gatunki,
```

```
    zwierzeta WHERE klatki.gatunek_id = gatunki.gatunek_id
AND klatki.klatka_id = zwierzeta.klatka_id
AND YEAR(data_przybycia) = YEAR(getdate())
```

	klatka_id	gatunek
1	7	Goryl
2	18	Niedźwiedź brunatny

14. Zwierzęta przebywające w zoo od dnia urodzenia

```
SELECT zwierze_id,
       imie,
       nazwa AS gatunek
FROM zwierzeta,
    gatunki WHERE zwierzeta.gatunek_id = gatunki.gatunek_id
AND data_urodzenia IS NOT NULL
AND datediff(DAY, data_przybycia, data_urodzenia) = 0
```

	zwierze_id	imie	gatunek
1	3	Simba	Lew
2	5	Mufasa	Lew
3	12	NULL	Antylopa gnu
4	22	NULL	Krokodyl
5	42	NULL	Niedźwiedź polarny
6	48	NULL	Rosomak

15. Liczba pawilonów odtwarzających każdy z biomów

```
SELECT nazwa,
       count(pawilon_id) AS liczba_pawilonow
FROM pawilony,
    biomy WHERE pawilony.biom_id = biomy.biom_id
GROUP BY nazwa
```

	nazwa	liczba_pawilonow
1	Arktyka	2
2	Las deszczowy	1
3	Sawanna	2
4	Step	1
5	Tajga	1