PROJEKT BAZY DANYCH – ZOO

TABELE:

Stan gatunku

```
#tabela 1
create or replace TABLE stan_gatunku (
    ID INT PRIMARY KEY,
    zagrozenie varchar(50) unique not null
);
```

Obszary zoo

```
#tabela 2
create or replace TABLE obszary_zoo (
    ID INT PRIMARY KEY,
    nazwa_obszaru varchar(50) unique not null
);
```

Opiekunowie

```
#tabela 3
create or replace TABLE opiekunowie (
    ID INT auto_increment PRIMARY KEY,
    imie varchar(50) not null,
    nazwisko varchar(50) not null,
    email varchar(50) unique
);
```

Sponsorzy

```
#tabela 4
create or replace TABLE sponsorzy (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    org_name varchar(50) not null,
    donate_date date not null,
    amount varchar(50) not null,
    animal_name varchar(50)
);
```

Opiekun

```
#tabela 5
create or replace TABLE opiekun (
    ID INT,
    data_zatrudnienia date,
    zwierze_opieka varchar(50),
    okres_opieki date,
    foreign key (id) references opiekunowie(id) on update cascade
);
```

Zwierzęta

```
#tabela główna (6)
create or replace TABLE zwierzeta (
    ID INT auto_increment,
    imie varchar(20) unique,
    gatunek varchar(50) not null,
    wiek int not null,
    opiekun int not null,
    dotacje INT,
    czesc_zoo int not null,
    stan_gatunku int not null,
    data_przybycia date not null,
    primary key (ID),
    FOREIGN KEY (opiekun) REFERENCES opiekunowie(id) on update cascade,
    FOREIGN KEY (dotacje) REFERENCES sponsorzy(id) on update cascade,
    FOREIGN KEY (czesc_zoo) REFERENCES obszary_zoo(id) on update cascade,
    FOREIGN KEY (stan_gatunku) REFERENCES stan_gatunku(id) on update cascade
);
```

Gdzie zwierzę

```
#tabela 7
create or replace TABLE gdzie_zwierze (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    nazwa_obszaru varchar(50) not null,
    zwierze varchar(50) not null,
    data_pobytu date not null,
    FOREIGN KEY (nazwa_obszaru) REFERENCES obszary_zoo(nazwa_obszaru) on update cascade,
    FOREIGN KEY (zwierze) REFERENCES zwierzeta(imie) on update cascade
);
```

Zwierzę widok

```
#tabela 8
create or replace TABLE zwierze_widok (
    ID INT auto_increment primary key,
    nazwa_zwierzecia varchar(50) unique,
    data_przyjecia date,
    opiekun int,
    okres_opieki date,
    wiek int,
    pozywienie varchar(50) not null,
    foreign key (nazwa_zwierzecia) references zwierzeta(imie) on update cascade,
    foreign key (opiekun) references opiekun(id) on update cascade
);
```

Karta przyjęcia + partycjonowanie

```
#tabela 9
create or replace table karta_przyjecia (
        id int auto_increment,
        data_przyjecia date,
        stan int,
        kraj_pochodzenia varchar(20),
        primary key (id, stan)
)
partition by list(stan) (
        partition wymarle_na_wolnosci values in (0),
        partition zagrozone_wyginieciem values in (1,2,3),
        partition bliskie_zagrozenia values in (4),
        partition najmniejszej_troski values in (5)
);
```

Dodanie kluczów obcych do partycowanej tabeli:

```
#insert kluczy obcych do zpartycjonowanej tabeli
insert into karta_przyjecia (id, stan)
select zwierzeta.id, stan_gatunku.id
from zwierzeta
join stan_gatunku on zwierzeta.stan_gatunku = stan_gatunku.id;
```

Indeksowanie:

```
#indeksowanie - indeksowanie zwierzat po imieniu i gatunku
create or replace unique index zwierze on zwierzeta(imi   gatunek);
```

FUNKCJE:

1. Liczenie ile zwierząt jest w danym obszarze ZOO

```
#funkcja licząca ile jest zwierząt w danym obszarze ZOO
CREATE FUNCTION IleZwierzatnaobszar (
    NazwaObszaru VARCHAR(50)
)
RETURNS INT
BEGIN
    DECLARE LiczbaZwierzat INT;

    SELECT COUNT(*)
    INTO LiczbaZwierzat
    FROM zwierzeta z
    INNER JOIN obszary_zoo o ON z.czesc_zoo = o.ID
    WHERE o.nazwa_obszaru = NazwaObszaru;

    RETURN LiczbaZwierzat;
end;
```

2. Wypisywanie, które gatunki są zagrożone lub wyginęły

```
#Funckja wypisująca,, które gatunki są bardzo zagrożone lub wyginęły
CREATE FUNCTION GatunkiZagrożone()
RETURNS VARCHAR(255)
BEGIN
    DECLARE lista_gatunkow VARCHAR(255);

    SELECT GROUP_CONCAT(DISTINCT gatunek)
    INTO lista_gatunkow
    FROM zwierzeta
    WHERE stan_gatunku = 1 or stan_gatunku = 0;

    IF lista_gatunkow IS NULL THEN
        SET lista_gatunkow = 'Brak zagrożonych gatunków.';
    END IF;
```

3. Liczenie ile jest zwierząt danego gatunku

```
#funkcja licząca ile jest zwierząt danego gatunku
create function liczba_zwierzat_gatunku(
    gatunek_param VARCHAR(50)
)
returns INT
begin
    declare ilosc INT;

    select COUNT(*) into ilosc
    from zwierzeta
    where gatunek = gatunek_param;

    return ilosc;
end;
```

TRIGGERY:

1. Automatyczne tworzenie maila dla opiekuna

```
#trigger 1 - automatyczne tworzenie maila dla opiekuna
create trigger before_insert_opiekun
before insert on opiekunowie
for each row
begin
        set new.email = concat(lower(new.imie),'.',lower(new.nazwisko),'@zoo.pl');
end;
```

2. Aktualizacja tabeli opiekunowie po dodaniu nowego członka do tabeli opiekun

```
#trigger2 - aktualizacja tabeli opiekunowie po dodaniu do opiekun nowego członka
CREATE TRIGGER AktualizujTabeleOpiekunowie
AFTER INSERT ON opiekun
FOR EACH ROW
BEGIN
    INSERT INTO opiekunowie (ID, data_zatrudnienia, data_dodania)
    VALUES (NEW.ID, NOW(), NOW(), NEW.zwierze_opieka, NEW.okres_opieki)
    ON DUPLICATE KEY UPDATE
    ID = NEW.ID,
    okres_zatrudnienia = NOW(),
    zwierze_opieka = new.zwierze_opieka,
   okres_opieki = NOW();
end ;
```

PROCEDURY:

1. Sumowanie dotacji od sponsorów

```
#procedura 1 - suma kwot dotacji
CREATE PROCEDURE suma_dotacji()
BEGIN
    SELECT org_name, SUM(amount) AS suma_dotacji
    FROM sponsorzy
    GROUP BY org_name;
END;
```

2. Zmiana opiekuna danego zwierzęcia

```
#procedura 2 - zmiana opiekuna danego zwierzęcia
CREATE PROCEDURE zmiana_opiekuna(IN nowe_zwierze VARCHAR(50), IN nowy_opiekun INT)
BEGIN
    DECLARE stare_opiekun_id INT;

    SELECT ID INTO stare_opiekun_id
    FROM opiekun
    WHERE zwierze_opieka = nowe_zwierze
    LIMIT 1;

    IF stare_opiekun_id IS NOT NULL THEN
        UPDATE opiekun
        SET ID = nowy_opiekun
        WHERE zwierze_opieka = nowe_zwierze;
    END IF;
END;
```

3. Dodawanie sponsorów do tabeli sponsorzy (aktualizacja listy)

```
#procedura 3 aktualizowanie sponsorów w tabeli sponsorzy (dodawanie sponsorów)
CREATE PROCEDURE DodajSponsora (
    IN orgName VARCHAR(50),
    IN donateDate DATE,
    IN amount DECIMAL(10,2),
    IN animalName VARCHAR(50)
)
BEGIN
    INSERT INTO sponsorzy (org_name, donate_date, amount, animal_name)
    VALUES (orgName, donateDate, amount, animalName);
END;
```

WIDOKI:

1. Widok zwierząt z podziałem zwierząt na stan gatunku

```
#Widok zwierząt z podziałem na stan gatunku
CREATE VIEW Zwierzeta_StanGatunku AS
SELECT *,
       CASE
           when stan_gatunku = 0 then 'Zanikające'
           when stan_gatunku = 1 then 'Duże ryzyko wyginięcie'
           when stan_gatunku = 2 then 'Duże ryzyko wyginięcie'
           when stan_gatunku = 3 then 'Zagrożone'
           when stan_gatunku = 4 then 'Zagrożone'
           when stan_gatunku = 5 then 'Powszechnie występujące'
           else 'Nieznany'
       end as status_gatunku
FROM zwierzeta;
```

2. Widok, przedstawiający jaki opiekun zajmował się danym zwierzęciem

```
#Widok, jaki opiekun zajmował się jakim zwierzęciem
CREATE VIEW WidokZwierzatOpiekuna AS
SELECT opiekun, GROUP_CONCAT(CONCAT(imie, ' (', gatunek, ')') ORDER BY imie ASC) AS zwierzeta_opiekuna
FROM zwierzeta
WHERE opiekun IS NOT NULL
GROUP BY opiekun;
```

3. Widok na jakim obszarze ZOO które zwierzęta przebywały

```
#Widok na jakim obszarze zoo jakie zwierzęta przebywały
create view zwierzeta_obszary AS
SELECT czesc_zoo, GROUP_CONCAT(CONCAT(imie, ' (', gatunek, ')') ORDER BY imie ASC) AS zwierzeta_w_obszarze
FROM zwierzeta
GROUP BY czesc_zoo;
```