239537
Illia Azler

Lista 2 (TN Sr 7:30-9:00)

Zad1.

```
create database aparaty;
use aparaty;
create user '239537'@'localhost' identified by '239537';
grant select, insert, update on aparaty.* to '239537'@'localhost';
flush privileges;
show grants for '239537'@'localhost';
```

Zad2.

```
create table aparat(
model varchar(30) not null primary key,
producent int unsigned not null,
matryca int unsigned not null,
obiektyw int unsigned not null,
typ enum('kompaktowy', 'lustrzanka', 'profesjonalny', 'inny') not null);

create table matryca(
id int unsigned not null auto_increment primary key,
przekatna decimal(4,2) unsigned not null,
rozdzielczosc decimal(3,1) unsigned not null,
typ varchar(10) not null);

create table obiektyw(
id int unsigned not null auto_increment primary key,
model varchar(30) not null,
minPrzeslona float unsigned not null,
maxPrzeslona float unsigned not null,
check (minPrzeslona<maxPrzeslona));

create table producent(
id int unsigned not null auto_increment primary key,
nazwa varchar(50),
kraj varchar(20));

alter table matryca auto_increment = 101;
alter table aparat add foreign key(matryca) references matryca(id);
alter table aparat add foreign key(obiektyw) references obiektyw(id);
alter table aparat add foreign key(producent) references producent(id);

show tables from aparaty;
select * from aparat;
select * from matryca;
select * from obiektyw;
select * from producent;
```

Zad3.

```
insert into aparat(model, producent, matryca, obiektyw, typ)
values   ('m11', floor(rand()*(15-1+1)+1), floor(rand()*(115-100+1)+100), floor(rand()*(15-1+1)+1),
'kompaktowy');

insert into matryca(przekatna,rozdzielczosc, typ)
 values ((round(rand() * 3.99 + 0.01, 2)), (round(rand() * 2.99 + 0.01, 1)), 'typ1');

insert into obiektyw(model, minPrzeslona, maxPrzeslona)
values ('model1', (round(rand() * 9.99 + 0.01, 2)), (round(rand() * 9.99 + 0.01, 2)));

insert into producent(nazwa, kraj)
values ('nazwa1', 'Chiny');
```

Zad4.

```
drop procedure if exists aparat_model;
DELIMITER $$
create procedure aparat_model(out prod int, out mat int, out obiek int, out typs varchar(20))
        begin
   declare x int;
        set x = 16;
        loop_label:  loop
                if  x = 116 then
                        leave  loop_label;
                else
                        select producent.id into prod from producent order by rand() limit 1;
                        select matryca.id into mat from matryca order by rand() limit 1;
                        select obiektyw.id into obiek from obiektyw order by rand() limit 1;
                        select aparat.typ into typs from aparat group by aparat.typ order by rand() limit
1;
                        insert into aparat(model, producent, matryca, obiektyw, typ)
                        values(concat("m", x), prod, mat, obiek, typs);
                        set x = x + 1;
                end if;
        end loop;
        end$$
DELIMITER ;
call aparat_model(@prod, @mat, @obiek, @typs);
```

Zad5.

```
drop procedure if exists producent_przekatna;
DELIMITER $$
create procedure producent_przekatna(in prod_id int, out wynik varchar(20))
begin
   select aparat.model into wynik from aparat
        join matryca on aparat.matryca=matryca.id and aparat.producent=prod_id and
matryca.przekatna =
```

```
	(select max(matryca.przekatna) from matryca join aparat on aparat.matryca=matryca.id  and
aparat.producent=prod_id);
end$$
DELIMITER ;
call producent_przekatna(1, @wynik);
select @wynik;

drop function if exists producent_przekatna;
delimiter $$
create function producent_przekatna (prod_id int)
returns varchar(20) deterministic
begin
	declare wynik varchar(30);

	select aparat.model into wynik from aparat
	join matryca on aparat.matryca=matryca.id and aparat.producent=prod_id and
matryca.przekatna =
	(select max(matryca.przekatna) from matryca join aparat on aparat.matryca=matryca.id  and
aparat.producent=prod_id);
   return wynik;
end$$
delimiter ;
select producent_przekatna(1);
```

Zad6.

```
drop trigger if exists add_producent;
delimiter $$
create trigger add_producent before insert on aparat
for each row
begin
	IF (select id From producent where id=NEW.producent) is null THEN
		insert into producent (id, nazwa, kraj) values (NEW.producent, 'unknown', 'unknown');
	END IF;
end$$
delimiter ;
```

Zad7.

```
drop function if exists model_martyca_count;
delimiter $$
create function model_martyca_count (matryca_id int)
returns int deterministic
begin
	declare wynik int;

	select count(model) into wynik from aparat
	group by matryca
	having matryca = matryca_id;

	return wynik;
end$$
delimiter ;
```

Zad8.

```
drop trigger if exists delete_matryca_byaparat;
delimiter $$
create trigger delete_matryca_byaparat after delete on aparat
for each row
begin
	IF (select count(*) from aparat where aparat.matryca=old.matryca)=0 THEN
		DELETE FROM aparaty.matryca WHERE (matryca.id = old.matryca);
	END IF;
end$$
delimiter ;
```

Zad9.

```
drop view if exists model_lustrzanka;
create view model_lustrzanka as
	select aparat.model, producent.nazwa, matryca.przekatna, matryca.rozdzielczosc,
	obiektyw.minprzeslona, obiektyw.maxprzeslona
	as model_lustrzanka
from (((aparat
	join producent on producent.id = aparat.producent)
	join matryca on matryca.id = aparat.matryca)
	join obiektyw on obiektyw.id = aparat.obiektyw)
where aparat.typ like 'lustrzanka' and producent.kraj not like 'Chiny';
```

Zad10.

```
drop view if exists model_skad;
create view model_skad as
select aparat.model, producent.nazwa, producent.kraj
	as model_skad
from (aparat
	join producent on producent.id = aparat.producent);

drop procedure if exists delete_chiny;
DELIMITER $$
create procedure delete_chiny()
begin
loop_label:  loop
		if  (select count(*) from aparat
				join producent on producent.id = aparat.producent
				where producent.kraj like 'Chiny')=0
		then
			leave  loop_label;
		else
			delete from aparaty.aparat where aparat.model =
				(select model from (select * from(aparat
				join producent on producent.id = aparat.producent)
				where producent.kraj like 'Chiny'
				limit 1)as a);
		end if;
```

```
        end loop;
end$$
DELIMITER ;
call delete_chiny();
```

Zad11.

```
alter table producent
add liczba_modeli int not null;

drop procedure if exists update_liczba_modeli;
DELIMITER $$
create procedure update_liczba_modeli()
begin
declare x int;
        set x = 1;
loop_label:  loop
                if  x >= (select count(id) from producent)+1
                then
                        leave  loop_label;
                else
                        if (select(select count(aparat.model) from aparat
                                                group by aparat.producent
                                                having producent = x) as a) is null
                        then
                                update producent
                                set liczba_modeli = 0
                                where producent.id = x;
                                set x=x+1;
                        else
                                update producent
                                set liczba_modeli =
                                        (select(select count(aparat.model) from aparat
                                                        group by aparat.producent
                                                        having producent = x) as a)
                                where producent.id = x;
                                set x = x + 1;
                        end if;
                end if;
        end loop;
end$$
DELIMITER ;

call update_liczba_modeli();

drop trigger if exists checkI_liczba_modeli;
delimiter $$
create trigger checkI_liczba_modeli after insert on aparat
for each row
begin
        call update_liczba_modeli();
```

```
end$$
delimiter ;

drop trigger if exists checkU_liczba_modeli;
delimiter $$
create trigger checkU_liczba_modeli after update on aparat
for each row
begin
	call update_liczba_modeli();
end$$
delimiter ;

drop trigger if exists checkD_liczba_modeli;
delimiter $$
create trigger checkD_liczba_modeli after delete on aparat
for each row
begin
	call update_liczba_modeli();
end$$
delimiter ;
```