Laboratorium VIII

Wiktor Zmiendak

Wyzwalacze

1. Szkolenie:

```
create schema tryger_demo;
use tryger_demo;

-- drop schema tryger_demo;

create table account (act_num int, amount decimal(10,2));

create trigger ins_sum before insert on account
for each row set @sum=@sum+new.amount;

set @sum=0;
insert into account values (137, 14.98), (141, 1937.50), (97, -100.00);
select @sum as 'Total amount inserted';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total amount inserted
1852.48

```
create trigger ins_transaction before insert on account
for each row precedes ins_sum
set @deposits = @deposits + if(new.amount>0, new.amount, 0),
@withdrawals = @withdrawals + if(new.amount<0, -new.amount,0);

set @deposits=0;
set @withdrawals=0;
insert into account values (137, 14.98), (141, 1937.50), (97, -100.00);
select @deposits as 'Total deposits', @withdrawals as 'Total withdrawls';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total deposits	Total withdrawls
1952.48	100.00

```
set @deposits=0;
set @withdrawals=0;
insert into account values (137, 14.98), (141, 1937.50), (97, -100.00);
insert into account values (1324, 134.98), (41, 1937.50), (9, -100.00);
insert into account values (12, 14.985), (1641, 1937.50), (7, -100.00);
select @deposits as 'Total deposits', @withdrawals as 'Total withdrawls';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total deposits	Total withdrawls
5977.45	300.00

```
update account set amount = -10 where act_num = 97;
update account set amount = -20 where act_num = 7;
select * from account;
select @deposits as 'Total deposits', @withdrawals as 'Total withdrawls';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

act_num	amount
137	14.98
141	1937.50
97	0.00
1324	134.98
41	1937.50
97	0.00
12	14.99
1641	1937.50
7	0.00

```
insert into test3 (a3) values
    (null), (null), (null), (null), (null),
    (null), (null), (null), (null), (null);

insert into test4 (a4) values
    (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);

insert into test1 values
    (1), (3), (1), (7), (1), (8), (4), (4);

select * from test1, test2, test3 ,test4;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

a1	a2	a3	a4	b4
4	3	6	2	0
4	3	5	2	0
4	3	2	2	0
1	1	10	2	0
1	1	9	2	0
1	1	6	2	0
1	1	5	2	0
1	1	2	2	0
3	1	10	2	0
3	1	9	2	0
3	1	6	2	0
3	1	5	2	0
3	1	2	2	0
1	1	10	2	0
1	1	9	2	0
1	1	6	2	0
1	1	5	2	0
1	1	2	2	0
7	1	10	2	0
7	1	9	2	0
7	1	6	2	0
7	1	5	2	0

Wnioski:

2. Utwórz co najmniej 2 wyzwalacze dla wcześniej utworzonej bazy danych według indywidualnego obszaru tematycznego. Zademonstruj wyniki pracy w postaci zrzutów ekranu skryptów i wyników ich wykonania:

```
create database airport5;
use airport5;

-- drop database airport5;

create table plane (
plane_id int(15) primary key,
company varchar(25),
weight int(15),
size int(15),
pasengers_slots int(15),
speed int(15));
```

```
create trigger ins_sum before insert on plane
for each row set @sum=@sum+new.weight;

create trigger ins_transaction before insert on plane
for each row precedes ins_sum
set @junk = @junk + if(new.weight>0, new.weight, 0),
@potentialPassengers = @potentialPassengers + if(new.pasengers_slots>0, new.pasengers_slots,0);

set @junk=0;
set @potentialPassengers=0;
insert into airport5.plane(plane_id, company, weight, size, pasengers_slots, speed)
values
('111', 'EasyJet', '1000', '50', '10', '500'),
('224', 'Ryaner', '1000', '50', '103', '500'),
('5253', 'Ryaner', '2500', '50', '200', '500'),
('4564', 'LOT', '1000', '50', '200', '500'),
('1', 'LOT', '1000', '50', '10', '600'),
('667', 'EasyJet', '3900', '50', '104', '550'),
('787', 'EasyJet', '1000', '50', '102', '550'),
('832', 'EasyJet', '9000', '50', '10', '550'),
('911', 'EasyJet', '8000', '50', '102', '600'),
('1110', 'Swishair', '200', '50', '10', '750');
```

```
select @junk as 'Total junk', @potentialPassengers as 'Total potential passengers';

set @sum=0;
insert into airport.plane(plane_id, company, weight, size, pasengers_slots, speed) values ('911', 'EasyJet', '8000', '50', '102', '600');
select @sum as 'Total amount inserted';
```

Result Grid | Filter Rows:

Total junk	Total potential passengers
28600	851

Total amount inserted
8000

Wnioski:

Procedury

1. Szkolenie:

```
5  create table employees(
6  id int auto_increment primary key,
7  job_id varchar(15) not null,
8  job_title varchar(45) not null,
9  min_salary int not null,
10 max_salary int not null
11 );
12
13 insert into company.employees(job_id, job_title, min_salary, max_salary)
14 values
15 ('ad_press', 'President', 20000, 40000),
16 ('ad_vp', 'Administration Vise President', 15000, 30000),
17 ('ad_asst', 'Administration Assistant', 3000, 6000),
18 ('fi_account', 'Accountant', 4200, 9000),
19 ('ac_mgr', 'Acounting Manager', 8200, 16000),
20 ('sa_mfn', 'Sales Manager', 10000, 20000),
21 ('sa_rep', 'Sales Repesentative', 6000, 12000),
22 ('st_clerk', 'Stok Clerk', 2000, 5000),
23 ('it_prog', 'Programer', 4000, 10000),
24 ('mk_man', 'Marketing Manager', 9000, 15000);
--
```

```
1 CREATE DEFINER=`root`@`localhost` PROCEDURE `my_proc_select`()
2 BEGIN
3 select * from employees;
4 END
```

```
26 call my_proc_select;
27
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	id	job_id	job_title	min_salary	max_salary
▶	1	ad_press	President	20000	40000
	2	ad_vp	Administration Vise President	15000	30000
	3	ad_asst	Administration Assistant	3000	6000
	4	fi_account	Accountant	4200	9000
	5	ac_mgr	Acounting Manager	8200	16000
	6	sa_mfn	Sales Manager	10000	20000
	7	sa_rep	Sales Repesentative	6000	12000
	8	st_clerk	Stok Clerk	2000	5000
	9	it_prog	Programer	4000	10000
	10	mk_man	Marketing Manager	9000	15000

```
1 CREATE DEFINER=`root`@`localhost` PROCEDURE `my_proc_out`(out highest_salary int)
2 BEGIN
3 select max(max_salary) into highest_salary from employees;
4 END
```

```
26 call my_proc_out(@m);
27 select @m;
28
```

Result Grid | Filter Rows:

	@m
▶	40000

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `my_proc_case`(inout no_employees int, in salary int)
BEGIN
case
when (salary>10000)
then (select count(job_id) into no_employees from employees
where min_salary > 10000);
when (salary<10000)
then (select count(job_id) into no_employees from employees
where min_salary<10000);
else (select count(job_id) into no_employees from employees
where min_salary=10000);
end case;
select no_employees;
END
```

```
26 • call my_proc_case(@c, 10001);
27
```

Result Grid | Filter Rows:

no_employees
2

```
26 • call my_proc_case(@c, 999);
27
```

Result Grid | Filter Rows:

no_employees
7

```
26 • call my_proc_case(@c, 1000);
27
```

Result Grid | Filter Rows:

no_employees
7

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `my_proc_select2`(in title varchar(45), out salary int)
BEGIN
select min_salary as salary from employees where job_title=title;
END
```

```
26 • call my_proc_select2('Programer', @s);
27
```

Result Grid | Filter Rows: | Export:

salary
4000

2. Utwórz co najmniej 2 funkcji oraz 3 procedury zapisane dla wcześniej utworzonej bazy danych według indywidualnego obszaru tematycznego. Zademonstruj ich skrypty i wyniki ich wykonania w postaci kopii ekranu:

```
create database airport6;
use airport6;

-- drop database airport6;

create table plane (
plane_id int(15) primary key,
company varchar(25),
weight int(15),
size int(15),
pasengers_slots int(15),
speed int(15));

insert into airport6.plane(plane_id, company, weight, size, pasengers_slots, speed)
values
('111', 'EasyJet', '1000', '50', '10', '500'),
('224', 'Ryaner', '1000', '50', '103', '500'),
('5253', 'Ryaner', '2500', '50', '200', '500'),
('4564', 'LOT', '1000', '50', '200', '500'),
('1', 'LOT', '1000', '50', '10', '600'),
('667', 'EasyJet', '3900', '50', '104', '550'),
('787', 'EasyJet', '1000', '50', '102', '550'),
('832', 'EasyJet', '9000', '50', '10', '550'),
('911', 'EasyJet', '8000', '50', '102', '600'),
('1110', 'Swishair', '200', '50', '10', '750');
```

Funkcja 1:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `add_plane`(
    p_id INT,
    p_company VARCHAR(25),
    p_weight INT,
    p_size INT,
    p_pasenger_slots INT,
    p_speed INT
) RETURNS varchar(100) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
    DECLARE msg VARCHAR(100);

    IF EXISTS (SELECT * FROM plane WHERE plane_id = p_id) THEN
        SET msg = 'Samolot o podanym ID już istnieje.';
    ELSE
        INSERT INTO plane(plane_id, company, weight, size, pasengers_slots, speed)
        VALUES (p_id, p_company, p_weight, p_size, p_pasenger_slots, p_speed);
        SET msg = 'Nowy samolot został dodany.';
    END IF;

    RETURN msg;
END
```

```
SELECT add_plane(123, 'AirFrance', 3000, 60, 150, 600);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	add_plane(123, 'AirFrance', 3000, 60, 150, 600)
▶	Nowy samolot został dodany.

Funkcja dodaje do bazy samolot o zadanych parametrach. W przypadku gdy samolot o danym numerze id już istnieje to go nie dodaje.

Funkcja 2:

```
1  CREATE DEFINER=`root`@`localhost` FUNCTION `delete_plane`(
2      p_id INT
3  ) RETURNS varchar(100) CHARSET utf8mb4
4      READS SQL DATA
5  BEGIN
6      DECLARE msg VARCHAR(100);
7
8      IF EXISTS (SELECT * FROM plane WHERE plane_id = p_id) THEN
9          DELETE FROM plane WHERE plane_id = p_id;
10         SET msg = CONCAT('Samolot o ID ', p_id, ' został usunięty.');
11     ELSE
12         SET msg = 'Nie znaleziono samolotu o podanym ID.';
13     END IF;
14
15     RETURN msg;
16 END
```

```
28  SELECT delete_plane(1);
```

Result Grid | Filter Rows:

delete_plane(1)
Samolot o ID 1 został usunięty.

Funkcja usuwa samolot z bazy na podstawie numeru id. W przypadku gdy nie znajdzie zadanego samolotu to informuje o tym użytkownika.

Procedura 1:

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `select_all`()
2  BEGIN
3  select * from plane;
4  END
```

```
29  call select_all;
```

Result Grid | Filter Rows: | Export: | Wrap C

plane_id	company	weight	size	pasengers_slots	speed
13	AirFrance	3000	60	150	600
111	EasyJet	1000	50	10	500
123	AirFrance	3000	60	150	600
224	Ryaner	1000	50	103	500
667	EasyJet	3900	50	104	550
787	EasyJet	1000	50	102	550
832	EasyJet	9000	50	10	550
911	EasyJet	8000	50	102	600
1110	Swishair	200	50	10	750
4564	LOT	1000	50	200	500
5253	Ryaner	2500	50	200	500

Procedura wypisuje całą bazę danych dla tabeli plane.

Procedura 2:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `search_plane`(in p_id int)
BEGIN
select * from plane where plane_id = p_id;
END
```

```
call search_plane(667);
```

Result Grid | Filter Rows: | Export: | Wrap C

plane_id	company	weight	size	pasengers_slots	speed
667	EasyJet	3900	50	104	550

Procedura wyszukuje informacje na temat danego samolotu na podstawie numeru id.

Procedura 3:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_plane`(
    IN p_id INT,
    IN p_company VARCHAR(25),
    IN p_weight INT,
    IN p_size INT,
    IN p_pasengers_slots INT,
    IN p_speed INT)
BEGIN
update plane
set
company = p_company,
weight = p_weight,
size = p_size,
pasengers_slots = p_pasengers_slots,
speed = p_speed
where plane_id = p_id;
END
```

```
call update_plane(1110, 'NewPlane', 999, 22, 11, 20);
select * from plane;
```

Result Grid | Filter Rows: | Edit: | Export/

plane_id	company	weight	size	pasengers_slots	speed
13	AirFrance	3000	60	150	600
111	EasyJet	1000	50	10	500
123	AirFrance	3000	60	150	600
224	Ryaner	1000	50	103	500
667	EasyJet	3900	50	104	550
787	EasyJet	1000	50	102	550
832	EasyJet	9000	50	10	550
911	EasyJet	8000	50	102	600
1110	NewPlane	999	22	11	20

Procedura wyszukuje dany samolot na podstawie numeru id a następnie aktualizuje informacje o nim zgodnie z nowo wprowadzonymi w zapytaniu.

Kursory

1. Szkolenie:

```sql
create database company2;
use company2;
-- drop database company2;

create table employees(
id int auto_increment primary key,
job_id varchar(15) not null,
job_title varchar(45) not null,
min_salary int not null,
max_salary int not null
);

insert into company2.employees(job_id, job_title, min_salary, max_salary)
values
('ad_press', 'President', 20000, 40000),
('ad_vp', 'Administration Vise President', 15000, 30000),
('ad_asst', 'Administration Assistant', 3000, 6000),
('fi_account', 'Accountant', 4200, 9000),
('ac_mgr', 'Acounting Manager', 8200, 16000),
('sa_mfn', 'Sales Manager', 10000, 20000),
('sa_rep', 'Sales Repesentative', 6000, 12000),
('st_clerk', 'Stok Clerk', 2000, 5000),
('it_prog', 'Programer', 4000, 10000),
('mk_man', 'Marketing Manager', 9000, 15000);
--
```

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `my_proc_cursor`()
BEGIN

declare done int default false;
declare id varchar(15);
declare fn varchar(45);
declare bal int;
declare mycursor cursor for select job_id, job_title, min_salary from employees where min_salary<5000;
declare continue handler for not found set done=true;
open mycursor;
fetch_loop: loop
fetch mycursor into id, fn, bal;
if done then leave fetch_loop;
end if;
select id,fn,bal;
end loop;
close mycursor;
END
```

```sql
call my_proc_cursor;
```

Result Grid | Filter Rows:

id	fn	bal
it_prog	Programer	4000

Wyniki:

2. Utwórz co najmniej 2 procedury zapisane za pomocą kursorów dla wcześniej utworzonej bazy danych według indywidualnego obszaru tematycznego. Zademonstruj ich skrypty i ich wyniki w postaci kopii ekranu:

Procedura 1:

```
use airport7;
-- drop database airport7;

create table plane (
plane_id int(15) primary key,
company varchar(25),
weight int(15),
size int(15),
pasengers_slots int(15),
speed int(15));

insert into airport7.plane(plane_id, company, weight, size, pasengers_slots, speed)
values
('111', 'EasyJet', '1000', '50', '10', '500'),
('224', 'Ryaner', '1000', '50', '103', '500'),
('5253', 'Ryaner', '2500', '50', '200', '500'),
('4564', 'LOT', '1000', '50', '200', '500'),
('1', 'LOT', '1000', '50', '10', '600'),
('667', 'EasyJet', '3900', '50', '104', '550'),
('787', 'EasyJet', '1000', '50', '102', '550'),
('832', 'EasyJet', '9000', '50', '10', '550'),
('911', 'EasyJet', '8000', '50', '102', '600'),
('1110', 'Swishair', '200', '50', '10', '750');
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `display_planes`()
BEGIN
declare done int default false;
declare plane_id_val int;
declare company_val varchar(45);
declare weight_val int;
declare size_val int;
declare pasengers_slots_val int;
declare speed_val int;
declare mycursor cursor for select * from plane;
declare continue handler for not found set done=true;
open mycursor;
read_loop: loop
fetch mycursor into plane_id_val, company_val, weight_val, size_val, pasengers_slots_val, speed_val;
if done then leave read_loop;
end if;
select plane_id_val, company_val, weight_val, size_val, pasengers_slots_val, speed_val;
end loop;
close mycursor;
END
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

plane_id_val	company_val	weight_val	size_val	pasengers_slots_val	speed_val
5253	Ryaner	2500	50	200	500

Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Result 6 | Result 7 | Result 8 | Result 9 | Result 10

Procedura wypisuje przy pomocy kursora wszystkie dodane do bazy samoloty.

Procedura 2:

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `search_plane`(in p_id int)
2  BEGIN
3  declare plane_id_val int;
4  declare company_val varchar(45);
5  declare weight_val int;
6  declare size_val int;
7  declare pasengers_slots_val int;
8  declare speed_val int;
9  declare done int default false;
10 declare mycursor cursor for select * from plane where plane_id = p_id;
11 declare continue handler for not found set done=true;
12 open mycursor;
13 fetch mycursor into plane_id_val, company_val, weight_val, size_val, pasengers_slots_val, speed_val;
14 if done then
15     select 'Nie znaleziono samolotu o podanym ID';
16 else
17     select plane_id_val, company_val, weight_val, size_val, pasengers_slots_val, speed_val;
18 end if;
19 close mycursor;
20 END
```

```
26  call search_plane(787);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

plane_id_val	company_val	weight_val	size_val	pasengers_slots_val	speed_val
787	EasyJet	1000	50	102	550

Procedura do wyszukiwania samolotu po jego id. Ta procedura wykorzystuje kursor do iteracji po wynikach zapytań SQL i obsługi danych z bazy danych.

Transakcje

1. Szkolenie:

```
12 •  select * from train where start_station = 'Lviv';
13 •  rollback;
14 •  select * from train where start_station = 'Lviv';
15 •  insert into train values(760, 'Lviv', 'Kovel');
16 •  commit;
17 •  select * from train where start_station = 'Lviv';
```

Result Grid | Filter Rows: | Edit: | Ex

id	start_station	finish_station
NULL	NULL	NULL

```
10 •  start transaction;
11 •  insert into train values(760, 'Lwiw', 'Kowel');
12 •  select * from train where start_station = 'Lviv';
13 •  rollback;
14 •  select * from train where start_station = 'Lviv';
15 •  insert into train values(760, 'Lviv', 'Kovel');
16 •  commit;
17 •  select * from train where start_station = 'Lviv';
```

Result Grid | Filter Rows: | Edit: | Exp

id	start_station	finish_station
NULL	NULL	NULL

```
10 •  start transaction;
11 •  insert into train values(760, 'Lwiw', 'Kowel');
12 •  select * from train where start_station = 'Lviv';
13 •  rollback;
14 •  select * from train where start_station = 'Lviv';
15 •  insert into train values(760, 'Lviv', 'Kovel');
16 •  commit;
17 •  select * from train where start_station = 'Lviv';
```

Result Grid | Filter Rows: | Edit: | Exp

id	start_station	finish_station
760	Lviv	Kovel
NULL	NULL	NULL

```sql
create table account_bank (
account_number int not null primary key,
balans int
);
```

```sql
insert into account_bank
values
('12345678', 5000),
('42345255', 15000),
('46222774', 7000),
('83242746', 3400),
('76234213', 1000);
```

```sql
start transaction;
update account_bank set balans=balans-2500 where account_number = 46222774;
update account_bank set balans=balans+2500 where account_number = 83242746;
select * from account_bank;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

account_number	balans
12345678	5000
42345255	15000
46222774	4500
76234213	1000
83242746	5900
NULL	NULL

```sql
start transaction;
update account_bank set balans=balans-2500 where account_number = 46222774;
update account_bank set balans=balans+2500 where account_number = 83242746;
select * from account_bank;
rollback;
select * from account_bank;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

account_number	balans
12345678	5000
42345255	15000
46222774	7000
76234213	1000
83242746	3400
NULL	NULL

```sql
select * from account_bank;
update account_bank set balans=balans-2500 where account_number = 46222774;
update account_bank set balans=balans+2500 where account_number = 83242746;
select * from account_bank;
commit;
select * from account_bank;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Con

account_number	balans
12345678	5000
42345255	15000
46222774	4500
76234213	1000
83242746	5900
NULL	NULL

```sql
select * from account_bank;
commit;
select * from account_bank;
```

Result Grid | Filter Rows:

account_number	balans
12345678	5000
42345255	15000
46222774	4500
76234213	1000
83242746	5900
NULL	NULL

2. Skompiluj co najmniej 2 skrypty transakcyjne dla wcześniej utworzonej bazy danych według indywidualnego obszaru tematycznego. Pokaż skrypty i ich wyniki w postaci kopii ekranu:

```
create database airport8;
use airport8;
-- drop database airport8;

create table plane (
plane_id int(15) primary key,
company varchar(25),
weight int(15),
size int(15),
pasengers_slots int(15),
speed int(15));

insert into airport8.plane(plane_id, company, weight, size, pasengers_slots, speed)
values
('111', 'EasyJet', '1000', '50', '10', '500'),
('224', 'Ryaner', '1000', '50', '103', '500'),
('5253', 'Ryaner', '2500', '50', '200', '500'),
('4564', 'LOT', '1000', '50', '200', '500'),
('1', 'LOT', '1000', '50', '10', '600'),
('667', 'EasyJet', '3900', '50', '104', '550'),
('787', 'EasyJet', '1000', '50', '102', '550'),
('832', 'EasyJet', '9000', '50', '10', '550'),
('911', 'EasyJet', '8000', '50', '102', '600'),
('1110', 'Swishair', '200', '50', '10', '750');
```

Skrypt transakcyjny 1:

```
start transaction;
update plane set speed=speed-400 where plane_id = 111;
update plane set speed=speed+300 where plane_id = 667;
select * from plane;
```

Result Grid | Filter Rows: | Edit: | Export/Ir

plane_id	company	weight	size	pasengers_slots	speed
1	LOT	1000	50	10	600
111	EasyJet	1000	50	10	100
224	Ryaner	1000	50	103	500
667	EasyJet	3900	50	104	850
787	EasyJet	1000	50	102	550
832	EasyJet	9000	50	10	550
911	EasyJet	8000	50	102	600
1110	Swishair	200	50	10	750
4564	LOT	1000	50	200	500
5253	Ryaner	2500	50	200	500
NULL	NULL	NULL	NULL	NULL	NULL

```
start transaction;
update plane set speed=speed-400 where plane_id = 111;
update plane set speed=speed+300 where plane_id = 667;
select * from plane;
rollback;
select * from plane;
```

Result Grid | Filter Rows: | Edit: | Export/In

plane_id	company	weight	size	pasengers_slots	speed
1	LOT	1000	50	10	600
111	EasyJet	1000	50	10	500
224	Ryaner	1000	50	103	500
667	EasyJet	3900	50	104	550
787	EasyJet	1000	50	102	550
832	EasyJet	9000	50	10	550
911	EasyJet	8000	50	102	600
1110	Swishair	200	50	10	750
4564	LOT	1000	50	200	500
5253	Ryaner	2500	50	200	500
NULL	NULL	NULL	NULL	NULL	NULL

```
update plane set speed=speed-400 where plane_id = 111;
update plane set speed=speed+300 where plane_id = 667;
select * from plane;
```

Result Grid | Filter Rows: | Edit: | Export/I

plane_id	company	weight	size	pasengers_slots	speed
1	LOT	1000	50	10	600
111	EasyJet	1000	50	10	100
224	Ryaner	1000	50	103	500
667	EasyJet	3900	50	104	850
787	EasyJet	1000	50	102	550
832	EasyJet	9000	50	10	550
911	EasyJet	8000	50	102	600
1110	Swishair	200	50	10	750
4564	LOT	1000	50	200	500
5253	Ryaner	2500	50	200	500
NULL	NULL	NULL	NULL	NULL	NULL

```
select * from plane;
commit;
select * from plane;
```

Result Grid | Filter Rows: | Edit:

plane_id	company	weight	size	pasengers_slots	speed
1	LOT	1000	50	10	600
111	EasyJet	1000	50	10	100
224	Ryaner	1000	50	103	500
667	EasyJet	3900	50	104	850
787	EasyJet	1000	50	102	550
832	EasyJet	9000	50	10	550
911	EasyJet	8000	50	102	600
1110	Swishair	200	50	10	750
4564	LOT	1000	50	200	500
5253	Ryaner	2500	50	200	500
NULL	NULL	NULL	NULL	NULL	NULL

Skrypt transakcyjny 2:

```
start transaction;
insert into plane values('113', 'Swishair', '220', '150', '11', '550');
select * from plane where company = 'Swishair';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

plane_id	company	weight	size	pasengers_slots	speed
113	Swishair	220	150	11	550
1110	Swishair	200	50	10	750
NULL	NULL	NULL	NULL	NULL	NULL

```
start transaction;
insert into plane values('113', 'Swishair', '220', '150', '11', '550');
select * from plane where company = 'Swishair';
rollback;
select * from plane where company = 'Swishair';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

plane_id	company	weight	size	pasengers_slots	speed
1110	Swishair	200	50	10	750
NULL	NULL	NULL	NULL	NULL	NULL

```
start transaction;
insert into plane values('113', 'Swishair', '220', '150', '11', '550');
select * from plane where company = 'Swishair';
rollback;
select * from plane where company = 'Swishair';
insert into plane values('113', 'Swishair', '220', '150', '11', '550');
commit;
select * from plane where company = 'Swishair';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

plane_id	company	weight	size	pasengers_slots	speed
113	Swishair	220	150	11	550
1110	Swishair	200	50	10	750
NULL	NULL	NULL	NULL	NULL	NULL