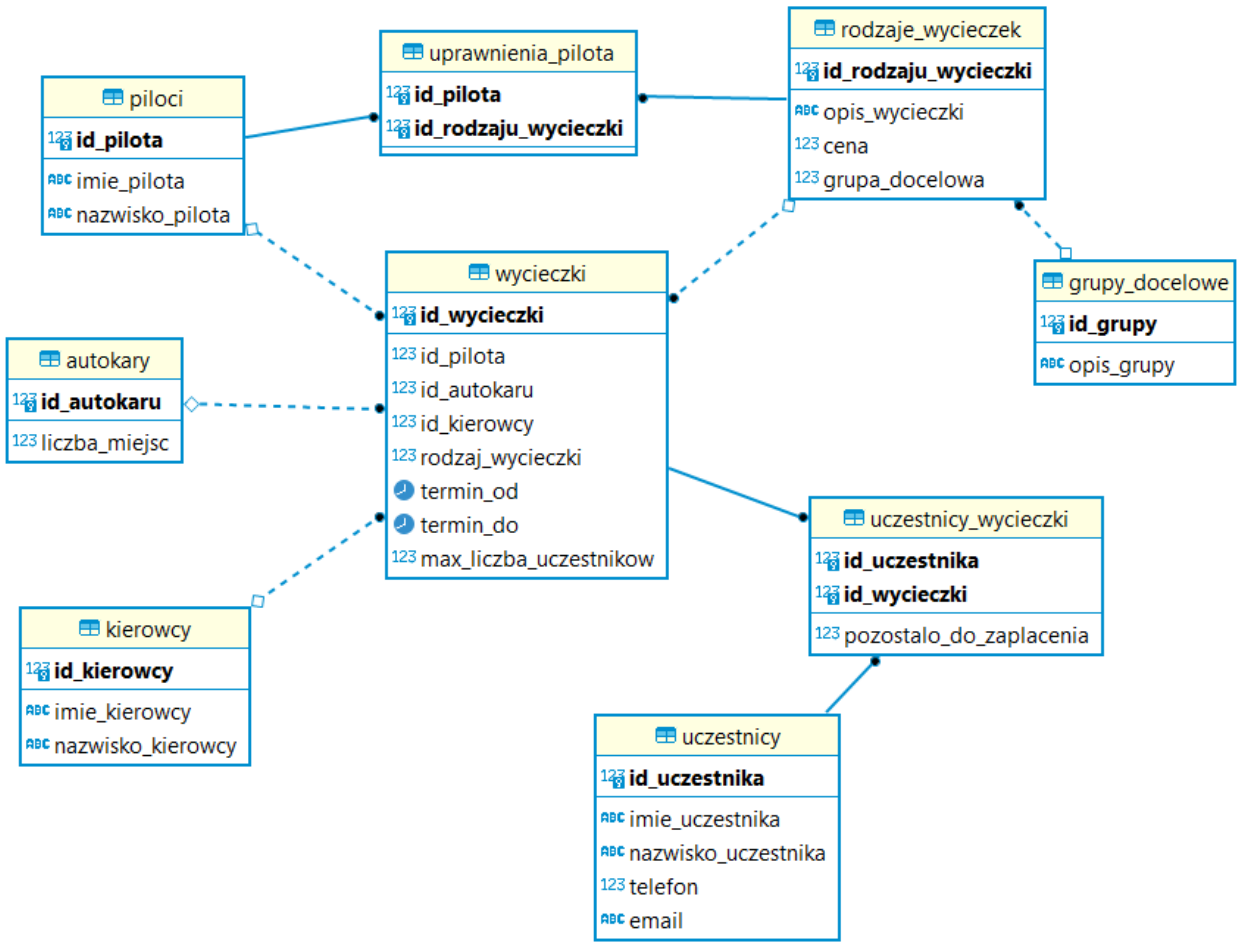
Katarzyna Prasoł, Łukasz Rębisz

Bazy danych – opis projektu

Projekt przedstawia bazę danych biura podróży organizującego wielodniowe wycieczki autokarowe z przewodnikiem. Każda wycieczka ma przypisanego pilota oraz autokar. Dany pilot posiada uprawnienia do prowadzenia określonych rodzajów wycieczek. Dla każdej wycieczki tworzona jest imienna lista uczestników.

1. Schemat bazy danych



2. Kod SQL

2.1 Tworzenie tabel wraz z kluczami

```
DROP TABLE IF EXISTS kierowcy CASCADE;
CREATE TABLE kierowcy(
    id_kierowcy INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    imie_kierowcy TEXT NOT NULL,
    nazwisko_kierowcy TEXT NOT NULL
);

DROP TABLE IF EXISTS autokary CASCADE;
CREATE TABLE autokary(
    id_autokaru INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    liczba_miejsc INTEGER NOT NULL
);

DROP TABLE IF EXISTS piloci CASCADE;
CREATE TABLE piloci(
    id_pilota INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    imie_pilota TEXT NOT NULL,
    nazwisko_pilota TEXT NOT NULL
);

DROP TABLE IF EXISTS grupy_docelowe CASCADE;
CREATE TABLE grupy_docelowe(
    id_grupy INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    opis_grupy TEXT NOT NULL
);

DROP TABLE IF EXISTS rodzaje_wycieczek CASCADE;
CREATE TABLE rodzaje_wycieczek(
    id_rodzaju_wycieczki INTEGER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
    opis_wycieczki TEXT NOT NULL,
    cena INTEGER NOT NULL,
    grupa_docelowa INTEGER REFERENCES grupy_docelowe(id_grupy)
      ON DELETE RESTRICT ON UPDATE CASCADE
);

DROP TABLE IF EXISTS uprawnienia_pilota CASCADE;
CREATE TABLE uprawnienia_pilota(
```

```
    id_pilota INTEGER REFERENCES piloci(id_pilota)
      ON DELETE CASCADE ON UPDATE CASCADE,
    id_rodzaju_wycieczki INTEGER REFERENCES
rodzaje_wycieczek(id_rodzaju_wycieczki )
      ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY(id_pilota , id_rodzaju_wycieczki)
);

DROP TABLE IF EXISTS uczestnicy CASCADE;
CREATE TABLE uczestnicy(
    id_uczestnika INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    imie_uczestnika TEXT NOT NULL,
    nazwisko_uczestnika TEXT NOT NULL,
    telefon INTEGER NOT NULL,
    email TEXT NOT NULL
);

DROP TABLE IF EXISTS wycieczki CASCADE;
CREATE TABLE wycieczki (
    id_wycieczki INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    id_pilota INTEGER REFERENCES piloci(id_pilota)
      ON DELETE RESTRICT ON UPDATE CASCADE,
    id_autokaru INTEGER REFERENCES autokary(id_autokaru)
      ON DELETE RESTRICT ON UPDATE CASCADE,
    id_kierowcy INTEGER REFERENCES kierowcy(id_kierowcy)
      ON DELETE RESTRICT ON UPDATE CASCADE,
    rodzaj_wycieczki INTEGER REFERENCES
rodzaje_wycieczek(id_rodzaju_wycieczki)
      ON DELETE RESTRICT ON UPDATE CASCADE,
    termin_od DATE NOT NULL,
    termin_do DATE NOT NULL,
    max_liczba_uczestnikow INTEGER NOT NULL
);

DROP TABLE IF EXISTS uczestnicy_wycieczki CASCADE;
CREATE TABLE uczestnicy_wycieczki (
    id_uczestnika INTEGER REFERENCES uczestnicy(id_uczestnika)
      ON DELETE CASCADE ON UPDATE CASCADE,
    id_wycieczki INTEGER REFERENCES wycieczki(id_wycieczki)
      ON DELETE CASCADE ON UPDATE CASCADE,
    pozostalo_do_zaplacenia INTEGER,
    PRIMARY KEY(id_uczestnika, id_wycieczki)
);
```

2.2 Tworzenie więzów integralności typu CHECK

```
ALTER TABLE wycieczki ADD CHECK (termin_do >= termin_od);
```

2.3 Tworzenie wyzwalaczy

- czy liczba miejsc wybranego autokaru jest >= maksymalnej liczbie uczestników wycieczki

```
CREATE OR REPLACE FUNCTION sprawdz_liczbe_miejsc_autokaru() RETURNS
TRIGGER AS $$
DECLARE
    liczba_miejsc_autokar INTEGER;
BEGIN
    SELECT DISTINCT a.liczba_miejsc INTO liczba_miejsc_autokar
    FROM wycieczki w JOIN autokary a USING(id_autokaru)
    WHERE a.id_autokaru=NEW.id_autokaru;

    IF (liczba_miejsc_autokar < NEW.max_liczba_uczestnikow) THEN
            RAISE EXCEPTION 'Wybrany autokar ma zbyt mało miejsc na tę
wycieczkę!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

DROP TRIGGER IF EXISTS sprawdz_liczbe_miejsc_autokaru_trigger ON wycieczki
CASCADE;
CREATE TRIGGER sprawdz_liczbe_miejsc_autokaru_trigger BEFORE INSERT OR
UPDATE ON wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_liczbe_miejsc_autokaru();
```

- czy w danym terminie wycieczki dostępny jest wybrany pilot

```
CREATE OR REPLACE FUNCTION sprawdz_terminy_pilota() RETURNS TRIGGER AS
$$
BEGIN
    IF(SELECT EXISTS(SELECT 1 FROM wycieczki WHERE id_pilota = NEW.id_pilota
            AND ((NEW.termin_od BETWEEN termin_od AND termin_do) OR
(NEW.termin_do BETWEEN termin_od AND termin_do)))) THEN
```

```
        RAISE EXCEPTION 'Wybrany pilot jest w tym terminie niedostępny!';
    END IF;
    IF(SELECT EXISTS(SELECT 1 FROM wycieczki WHERE id_pilota = NEW.id_pilota
        AND ((termin_od BETWEEN NEW.termin_od AND NEW.termin_do) OR
(termin_do BETWEEN NEW.termin_od AND NEW.termin_do)))) THEN

        RAISE EXCEPTION 'Wybrany pilot jest w tym terminie niedostępny!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

DROP TRIGGER IF EXISTS sprawdz_terminy_pilota_trigger ON wycieczki
CASCADE;
CREATE TRIGGER sprawdz_terminy_pilota_trigger BEFORE INSERT OR UPDATE
ON wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_terminy_pilota();
```

- czy w danym terminie wycieczki dostępny jest wybrany kierowca

```
CREATE OR REPLACE FUNCTION sprawdz_terminy_kierowcy() RETURNS TRIGGER
AS $$
BEGIN
    IF(SELECT EXISTS(SELECT 1 FROM wycieczki WHERE id_kierowcy =
NEW.id_kierowcy
        AND ((NEW.termin_od BETWEEN termin_od AND termin_do) OR
(NEW.termin_do BETWEEN termin_od AND termin_do)))) THEN

        RAISE EXCEPTION 'Wybrany kierowca jest w tym terminie
niedostępny!';
    END IF;
    IF(SELECT EXISTS(SELECT 1 FROM wycieczki WHERE id_kierowcy =
NEW.id_kierowcy
        AND ((termin_od BETWEEN NEW.termin_od AND NEW.termin_do) OR
(termin_do BETWEEN NEW.termin_od AND NEW.termin_do)))) THEN

        RAISE EXCEPTION 'Wybrany kierowca jest w tym terminie
niedostępny!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
```

```
DROP TRIGGER IF EXISTS sprawdz_terminy_kierowcy_trigger ON wycieczki
CASCADE;
CREATE TRIGGER sprawdz_terminy_kierowcy_trigger BEFORE INSERT OR UPDATE
ON wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_terminy_kierowcy();
```

- czy w danym terminie wycieczki dostępny jest wybrany autokar

```
CREATE OR REPLACE FUNCTION sprawdz_terminy_autokaru() RETURNS TRIGGER
AS $$
BEGIN
    IF(SELECT EXISTS(SELECT 1 FROM wycieczki WHERE id_autokaru =
NEW.id_autokaru
            AND ((NEW.termin_od BETWEEN termin_od AND termin_do) OR
(NEW.termin_do BETWEEN termin_od AND termin_do)))) THEN

            RAISE EXCEPTION 'Wybrany autokar jest w tym terminie niedostępny!';
    END IF;
    IF(SELECT EXISTS(SELECT 1 FROM wycieczki WHERE id_autokaru =
NEW.id_autokaru
            AND ((termin_od BETWEEN NEW.termin_od AND NEW.termin_do) OR
(termin_do BETWEEN NEW.termin_od AND NEW.termin_do)))) THEN

            RAISE EXCEPTION 'Wybrany autokar jest w tym terminie niedostępny!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

DROP TRIGGER IF EXISTS sprawdz_terminy_autokaru_trigger ON wycieczki
CASCADE;
CREATE TRIGGER sprawdz_terminy_autokaru_trigger BEFORE INSERT OR UPDATE
ON wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_terminy_autokaru();
```

- czy wybrany pilot ma uprawnienia do poprowadzenia danej wycieczki

```
    CREATE OR REPLACE FUNCTION sprawdz_uprawnienia() RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.id_pilota NOT IN(
    SELECT p.id_pilota FROM piloci p
```

```
    JOIN uprawnienia_pilota up USING (id_pilota)
    WHERE NEW.rodzaj_wycieczki = up.id_rodzaju_wycieczki))
    THEN
            RAISE EXCEPTION 'Podany pilot NIE ma uprawnien do danej
wycieczki!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';


DROP TRIGGER sprawdz_uprawnienia_trigger ON wycieczki CASCADE;
CREATE TRIGGER sprawdz_uprawnienia_trigger BEFORE INSERT ON wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_uprawnienia();
```

-czy dodanie nowego uczestnika wycieczki nie przekroczy maksymalnej liczby uczestników

```
CREATE OR REPLACE FUNCTION sprawdz_max_liczbe_uczestnikow() RETURNS
TRIGGER AS $$
DECLARE
 liczba INTEGER DEFAULT 0;
 BEGIN
  SELECT count(uw.id_uczestnika) INTO liczba
  FROM wycieczki w
  JOIN uczestnicy_wycieczki uw USING(id_wycieczki)
  GROUP BY w.id_wycieczki
  HAVING w.id_wycieczki = NEW.id_wycieczki;
  IF (liczba >= (SELECT max_liczba_uczestnikow
  FROM wycieczki
  WHERE id_wycieczki = NEW.id_wycieczki))
    THEN
            RAISE EXCEPTION 'Nie ma wolnych miejsc na te wycieczke!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';


DROP TRIGGER sprawdz_max_liczbe_uczestnikow_trigger ON
uczestnicy_wycieczki CASCADE;
CREATE TRIGGER sprawdz_max_liczbe_uczestnikow_trigger BEFORE INSERT ON
uczestnicy_wycieczki
```

```
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_max_liczbe_uczestnikow();
```

- czy atrybut ,,pozostało do zapłacenia” >= 0 i <= cena wycieczki

```
    CREATE OR REPLACE FUNCTION sprawdz_czy_za_malo() RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.pozostalo_do_zaplacenia < 0)
    THEN
            RAISE EXCEPTION 'Zla wartosc pozostalo do zaplacenia!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';



DROP TRIGGER sprawdz_czy_za_malo_trigger ON uczestnicy_wycieczki
CASCADE;
CREATE TRIGGER sprawdz_czy_za_malo_trigger BEFORE INSERT OR UPDATE ON
uczestnicy_wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_czy_za_malo();


    CREATE OR REPLACE FUNCTION sprawdz_czy_za_duzo() RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.pozostalo_do_zaplacenia >
    (SELECT rw.cena
    FROM wycieczki w
    JOIN rodzaje_wycieczek rw ON(rw.id_rodzaju_wycieczki = w.rodzaj_wycieczki)
    WHERE w.id_wycieczki = NEW.id_wycieczki))
    THEN
            RAISE EXCEPTION 'Zla wartosc pozostalo do zaplacenia!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';



DROP TRIGGER sprawdz_czy_za_duzo_trigger ON uczestnicy_wycieczki
CASCADE;
```

```
CREATE TRIGGER sprawdz_czy_za_duzo_trigger BEFORE INSERT OR UPDATE ON
uczestnicy_wycieczki
    FOR EACH ROW EXECUTE PROCEDURE sprawdz_czy_za_duzo();
```

- początkowa kwota = domyślnie cenie wycieczki

```
 CREATE OR REPLACE FUNCTION poczatkowa_kwota() RETURNS TRIGGER AS $$
BEGIN
 IF (NEW.pozostalo_do_zaplacenia IS NULL)
 THEN
  SELECT rw.cena INTO NEW.pozostalo_do_zaplacenia
  FROM wycieczki w
  JOIN rodzaje_wycieczek rw ON(rw.id_rodzaju_wycieczki = w.rodzaj_wycieczki)
  WHERE w.id_wycieczki = NEW.id_wycieczki;
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';


DROP TRIGGER poczatkowa_kwota_trigger ON uczestnicy_wycieczki CASCADE;
CREATE TRIGGER poczatkowa_kwota_trigger BEFORE INSERT ON
uczestnicy_wycieczki
    FOR EACH ROW EXECUTE PROCEDURE poczatkowa_kwota();
```

2.4 Inserty

```
INSERT INTO autokary(liczba_miejsc) VALUES
            (40), (50), (60), (16), (45), (55),
            (45), (55), (60), (30), (50), (60);


INSERT INTO kierowcy(imie_kierowcy, nazwisko_kierowcy) VALUES
            ('Jan', 'Kowalski'), ('Adam', 'Nowak'),
            ('Janina', 'Kowalski'), ('Krzysztof', 'Nowakowski'),
            ('Robert', 'Kowal'), ('Piotr', 'Szymański'),
            ('Anna', 'Malinowska'), ('Zygmunt', 'Opania'),
            ('Weronika', 'Adamska'), ('Szczepan', 'Nowak'),
            ('Jan', 'Bartoszewski'), ('Adam', 'Marczewski');
```

```
INSERT INTO piloci(imie_pilota, nazwisko_pilota) VALUES
        ('Adam', 'Ciekawski'), ('Robert', 'Kompas'),
        ('Adrianna', 'Ciesielska'), ('Roman', 'Kamiński'),
        ('Dariusz', 'Kowalczyk'), ('Adam', 'Mazur'),
        ('Anastazja', 'Lewandowska'), ('Rozalia', 'Wojciechowska'),
        ('Amelia', 'Jankowska'), ('Roma', 'Krawczyk'),
        ('Krzysztof', 'Jankowski'), ('Ida', 'Jablońska');


INSERT INTO grupy_docelowe(opis_grupy) VALUES
        ('szkolna'),('dla rodzin'),('dla seniorow'),
        ('dla par'),('uniwersalna');


INSERT INTO rodzaje_wycieczek(opis_wycieczki, cena, grupa_docelowa) VALUES
        ('Kraków w 3 dni', 300, 1),
        ('Wroclaw w 3 dni', 300, 1),
        ('Warszawa w 3 dni', 350, 1),
        ('Sopot na weekend', 350, 4),
        ('Wroclaw na weekend', 250, 4),
        ('Kolobrzeg na weekend', 300, 4),
        ('Kraków na weekend', 250, 4),
        ('Kraków w 5 dni', 600, 5),
        ('Wroclaw w 5 dni',600, 5),
        ('Warszawa w 5 dni', 650, 5),
        ('Zakopane w 7 dni', 1000 , 2),
        ('Kolobrzeg w 7 dni', 1100 , 2),
        ('Krynica Zdrój w 14 dni', 2800 , 3),
        ('Lądek Zdrój w 14 dni', 2500 , 3);

INSERT INTO uprawnienia_pilota(id_pilota, id_rodzaju_wycieczki) VALUES
        (1, 1),(1, 12),(1, 3),(1, 4),(1, 5),
        (2, 1),(2, 2),(2, 3),(3, 3),(4, 2),
        (4, 13),(5, 8),(6, 4),(6, 5),(7, 1),
        (8, 6),(8, 1),(8, 3),(8, 14),(8, 10),
        (9, 11),(9, 2),(10, 10),(10, 3),(10, 5),
        (11, 4),(11, 9),(12, 7),(12, 4),(12, 5);


INSERT INTO wycieczki(id_pilota, id_autokaru, id_kierowcy, rodzaj_wycieczki,
termin_od, termin_do, max_liczba_uczestnikow) VALUES
        (1, 1, 1, 5, '2023-07-01', '2023-07-02', 40),
        (2, 2, 2, 2, '2023-05-04', '2023-05-07', 50),
```

```
(3, 3, 3, 3, '2023-05-04', '2023-05-07', 60),
(4, 4, 4, 13, '2023-07-01', '2023-07-14', 16),
(5, 5, 5, 8, '2023-08-01', '2023-08-05', 45),
(6, 6, 6, 4, '2023-07-08', '2023-07-09', 55),
(7, 7, 7, 1, '2023-06-01', '2023-06-03', 45),
(8, 8, 8, 6, '2023-07-01', '2023-07-02', 55),
(9, 9, 9, 11, '2023-08-01', '2023-08-07', 60),
(10, 10, 10, 10, '2023-07-01', '2023-07-07', 30),
(11, 11, 11, 9, '2023-04-29', '2023-05-03', 50),
(12, 12, 12, 7, '2023-06-03', '2023-06-04', 60),
(1, 12, 12, 12, '2023-08-15', '2023-08-29', 60),
(8, 11, 11, 1, '2023-03-01', '2023-03-03', 50),
(8, 10, 10, 14, '2023-06-17', '2023-06-18', 30);
```

```
INSERT INTO uczestnicy(imie_uczestnika, nazwisko_uczestnika, telefon, email)
    VALUES ('Jan', 'Wycieczkowicz', 123456789, 'jan@wp.pl'),
    ('Adam', 'Wycieczkowicz', 123456789, 'jan@wp.pl'),
    ('Janina', 'Wycieczkowicz', 123456789, 'jan@wp.pl'),
    ('Hanna', 'Wycieczkowicz', 123456789, 'jan@wp.pl'),
    ('Anna', 'Wycieczkowicz', 123456789, 'jan@wp.pl'),
    ('Bogdan', 'Adamczyk', 123956387, 'bogdan@wp.pl'),
    ('Janina', 'Adamczyk', 980765287, 'janina@wp.pl'),
    ('Hanna', 'Kowalska', 392647842, 'hanna@wp.pl'),
    ('Hanna', 'Michalska', 392647841, 'hanna1@wp.pl'),
    ('Janina', 'Nowicka', 392237842, 'nowica@wp.pl'),
    ('Zuzanna', 'Dudek', 102647842, 'zuza@wp.pl'),
    ('Jan', 'Szewczyk', 980165287, 'janek@wp.pl'),
    ('Henryk', 'Kowalski', 302647842, 'henio@wp.pl'),
    ('Harold', 'Zalewski', 392090841, 'h1@wp.pl'),
    ('Jan', 'Nowicki', 222237842, 'now@wp.pl'),
    ('Zygmunt', 'Dudekowski', 111147842, 'zyg@wp.pl'),
    ('Zuzanna', 'Dudkowska', 112647842, 'zuza1@wp.pl'),
    ('Jan', 'Walczak', 980156287, 'janek234@wp.pl'),
    ('Henryk', 'Kowalczyk', 309847842, 'henio211@wp.pl'),
    ('Harold', 'Zalewski', 392090841, 'h1@wp.pl'),
    ('Jan', 'Pawlak', 222227842, 'pawlak@wp.pl'),
    ('Zygmunt', 'michalak', 111111842, 'zyg12@wp.pl'),
    ('Adam', 'Sikora', 223456789, 'sikora@wp.pl'),
    ('Janina', 'Sikora', 223456789, 'sikora@wp.pl'),
    ('Hanna', 'Sikora', 223456789, 'sikora@wp.pl'),
    ('Anna', 'Sikora', 223456789, 'sikora@wp.pl');
```

```
INSERT INTO uczestnicy_wycieczki(id_uczestnika, id_wycieczki,
pozostalo_do_zaplacenia)
    VALUES (1, 9, 100),(2, 9, 100),(3, 9, 100),(4, 9, 100),
        (5, 9, 100),(6, 6, 0),(7, 6, 0),(6, 1, 249.99),
        (7, 1, 249.99),(1, 6, 100),(2, 6, 100),(7, 4 ,1000),
        (8, 4, 250),(9, 4, 0),(10, 4, 1100),(11, 4 ,2000),
        (12, 4, 250),(13, 4, 0),(14, 4, 1100),(15, 4 ,2000),
        (16, 4, 250),(17, 4, 0),(18, 4, 1100),(19, 4 ,2000),
        (20, 4, 250),(21, 4, 0),(22, 4, 1100),
        (23, 10, 500),(24, 10, 500),(25, 10, 500),(26, 10, 500),
        (25, 14, 100),(26, 2, 100),(23, 8, 0), (24, 8, 0),
        (3, 2, 100),(12, 11, 0), (1, 2, NULL);
```

2.5 Przykładowe inserty, przy których pojawia się błąd

```
INSERT INTO wycieczki(id_pilota, id_autokaru, id_kierowcy, rodzaj_wycieczki,
termin_od, termin_do, max_liczba_uczestnikow) VALUES
        (10, 1, 1, 5, '2023-07-01', '2023-07-02', 40);
```

```
INSERT INTO wycieczki(id_pilota, id_autokaru, id_kierowcy, rodzaj_wycieczki,
termin_od, termin_do, max_liczba_uczestnikow) VALUES
        (2, 1, 1, 2, '2023-07-01', '2023-07-02', 40);
```

```
INSERT INTO wycieczki(id_pilota, id_autokaru, id_kierowcy, rodzaj_wycieczki,
termin_od, termin_do, max_liczba_uczestnikow) VALUES
        (10, 1, 1, 2, '2023-09-01', '2023-09-02', 40);
```

```
INSERT INTO wycieczki(id_pilota, id_autokaru, id_kierowcy, rodzaj_wycieczki,
termin_od, termin_do, max_liczba_uczestnikow) VALUES
        (2, 1, 1, 2, '2023-07-01', '2023-07-02', 47);
```

```
INSERT INTO uczestnicy_wycieczki(id_uczestnika, id_wycieczki,
pozostalo_do_zaplacenia)
    VALUES (3, 4, 1100);
```

```
INSERT INTO uczestnicy_wycieczki(id_uczestnika, id_wycieczki,
pozostalo_do_zaplacenia)
    VALUES (18, 9, -1);


INSERT INTO uczestnicy_wycieczki(id_uczestnika, id_wycieczki,
pozostalo_do_zaplacenia)
    VALUES (20, 8, 90909090);
```