Baza danych dla biura turystycznego

Autorzy: Czernecki Paweł, Dziarkowski Michał, Matuszyński Wojciech, Szkarłat Szymon

1. Wymagania i funkcje systemu

1.1. Role:

W bazie danych mogą istnieć podmioty, które mają więcej niż jedną rolę. Jeżeli podmiot jest zarówno uczestnikiem jak i klientem wycieczki, zyskuje uprawnienia związane z obiema rolami. W przypadku, gdy podmiot jest jednocześnie klientem biura i jego administratorem, przewidziane są dwa osobne konta (jedno na stronie biura, drugie w panelu administracyjnym).

1.1.1. Właściciel biura

- Posiada minimalne uprawnienia w systemie.
- Jego uprawnienia nie przekraczają uprawnienia zwykłego użytkownika systemu

1.1.2. Admin

- Posiada konto w panelu administracyjnym.
- Może tworzyć/usuwać nowe wycieczki i zmieniać ich właściwości, w wypadku uzyskania odpowiedniego komunikatu od sekretarza/sekretarki.
- Ma dostęp do listy uczestników i listy klientów dla każdej wycieczki/fakultetu. Może usunąć klienta z listy klientów wycieczki, jeżeli nie dopełni swych obowiązków.

1.1.3. Sekretarz/Sekretarka

- Odpowiada za płatności przebiegające między klientami a systemem.
- W przypadku rezygnacji z wycieczki/fakultetu bądź niespełnienia wymogów, zwraca pieniądze klientom na ich konto.
- Kontaktuje się z firmami przewoźniczymi i hotelarskimi w celu uzgodnienia zakwaterowania uczestników, a także dat i miejsc zbiórek.
- Tydzień przed rozpoczęciem wycieczki wysyła administratorom listę klientów, którzy dopełnili obowiązków.
- Informuje administratorów na temat zmiany/dodania szczegółów wycieczki (np. zmiana daty rozpoczęcia, udostępnienie miejscu i czasu zbiórki na wycieczkę).

1.1.4. Klient

- Może być podmiotem indywidualnym lub firmą.
- Posiada konto na stronie.
- Może rezerwować wycieczki/fakultety.
- Może rezygnować z wycieczki/fakultetu.
- Może podawać dane uczestników.

- Ma dostęp do listy zarezerwowanych wycieczek oraz ma dostęp do listy uczestników, uczestniczących w zarezerwowanej wycieczce.

1.1.5. Uczestnik

- Posiada konto na stronie.
- Może zobaczyć wycieczki i fakultety, na które jest zapisany. Dla każdej wycieczki/fakultetu może sprawdzić miejsce i czas zbiórki(np. czas wylotu i lokalizacja lotniska na wycieczkę zagranicą), a dla wycieczek może też sprawdzić miejsce i czas przybycia na miejsce, oraz datę powrotu.

1.2. Działanie biura turystycznego

1.2.1. Wycieczki

- Główna część oferty biura podróży
- Składa się z jedno- i wielodniowych wypraw, zakup wycieczki wiąże się z przygotowaniem dojazdu i zakwaterowania dla uczestników takowej oferty.
- Zakup atrakcji/fakultetu jest możliwy tylko w przypadku uprzednio zakupionej wycieczki.
- Każda wycieczka ma własny dla siebie limit miejsc oraz cenę za osobę.

1.2.2. Fakultety (Atrakcje dodatkowe)

- Dodatkowa/opcjonalna część oferty biura.
- Klient może urozmacić wycieczkę uczestników poprzez zakup usług dodatkowych, które odbędą się w czasie wycieczki.
- Aby uczestnik mógł wziąć udział w fakultecie, musi być uczestnikiem związanej wycieczki.
- Każdy fakultet ma własny dla siebie limit miejsc oraz dodatkową cenę za osobę.

1.2.3. Rezerwacje Ofert

- Klienci rezerwują wycieczki i fakultety w imieniu jednego lub wielu uczestników.
- Przy rezerwacji wycieczki podają liczbę uczestników.
- W terminie do tygodnia przed rozpoczęciem wycieczki klienci są zobowiązani do podania listy uczestników (mowa tutaj o podaniu danych kontaktowych uczestników: imię, nazwisko, email oraz nr telefonu), oraz wpłacenia pełnej kwoty za wycieczkę.
- W przypadku niedopełnienia obowiązku, rezerwacja jest usuwana przez administratora, a wpłacone dotychczas pieniądze zwracane są przez sekretarkę.
- Fakultet można zamówić w dowolnym momencie po zamówieniu wycieczki, do tygodnia przed datą rozpoczęcia wycieczki. Musi przypisać do niego uczestnika imiennie do tygodnia przed rozpoczęciem wycieczki.
- W przypadku, kiedy klient zmniejszy liczbę uczestników biorących udział w wycieczce/fakultecie, przysługuje mu zwrot wpłaconych pieniędzy pod warunkiem, że poinformował o tym fakcie biuro na tydzień przed rozpoczęciem wycieczki.

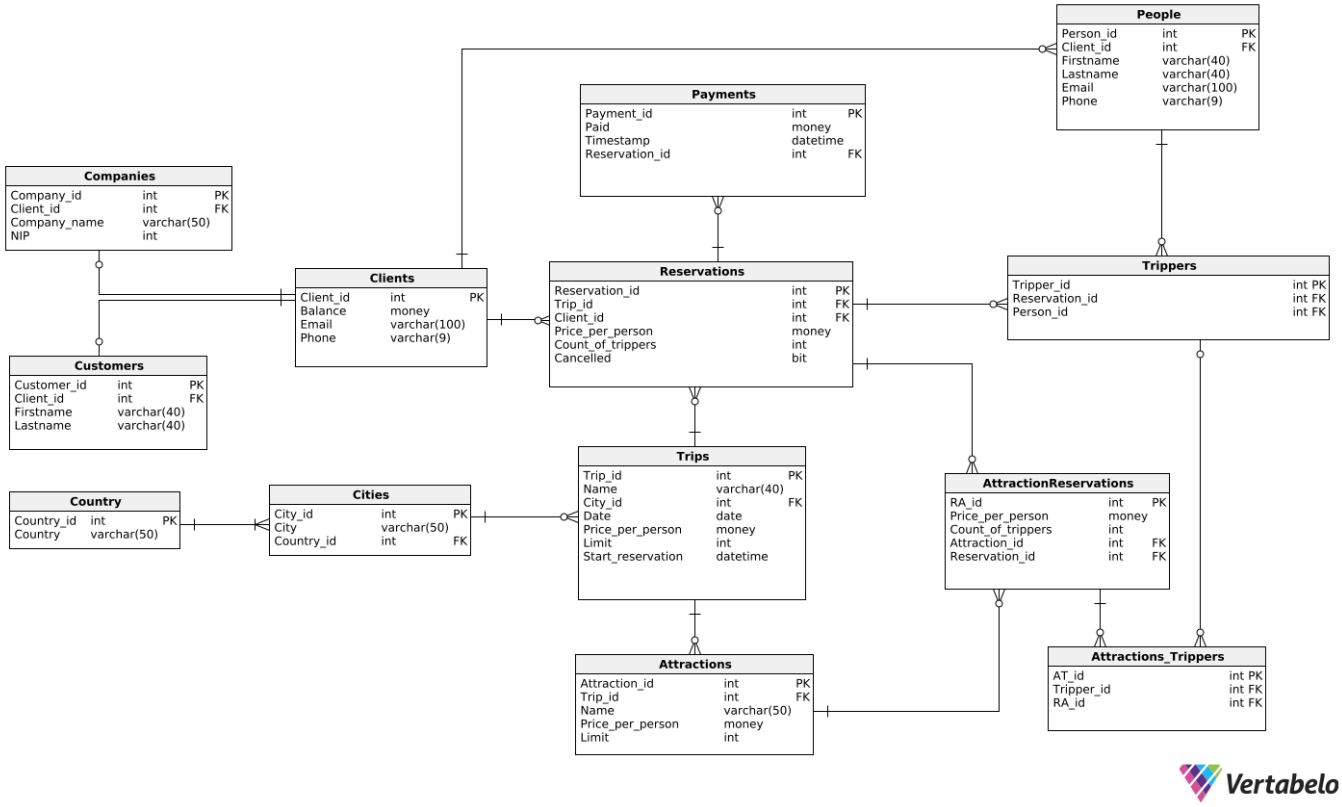
1.2.4. Zbiórki

- Informacje na temat lokalizacji i daty, gdzie mają zebrać się uczestnicy w celu rozpoczęcia wycieczki dostarczane są nie później niż tydzień przed rozpoczęciem wycieczki.

- Data zbiórki wycieczki może być jednorazowo przesunięta do 24 godzin w przyszłość, nie później niż dobę przed oryginalną datą zbiórki.
- Informacje na temat zbiórki na usługi fakultatywne pojawiają się nie później niż dobę przed datą rozpoczęcia usługi fakultatywnej.

2. Baza danych

2.1. Schemat bazy danych



2.2. Opis poszczególnych tabel

Clients

Tabela zawiera klientów, którzy złożyli zamówienia na dane wycieczki, dokonali rezerwacji dla określonej liczby osób. Mogą to być zarówno osoby prywatne, jak i firmy. Dodatkowo mamy sprawdzenie czy stan konta jest dodatni.

Nazwa atrybutu	Typ	Opis/Uwagi
Client_id	int	Primary Key (PK)
Balance	money	Stan konta
Email	varchar(100)	Adres email
Phone	varchar(9)	Nr telefonu

```
CREATE TABLE Clients (
    Client_id int  NOT NULL IDENTITY(1, 1),
    Balance money  NOT NULL,
    Email varchar(100)  NOT NULL,
    Phone varchar(9)  NOT NULL,
    CONSTRAINT check_balance CHECK (Balance > 0),
    CONSTRAINT Clients_pk PRIMARY KEY  (Client_id)
);
```

Companies

Tabela zawiera firmy, które dokonały rezerwacji wycieczki w systemie oraz dane je identyfikujące, tj. nazwa firmy oraz nr NIP.

Nazwa atrybutu	Typ	Opis/Uwagi
Company_id	int	PK
Client_id	int	FK
Company_name	varchar(50)	Nazwa firmy
NIP	int	Nr NIP

```
CREATE TABLE Companies (
    Company_id int  NOT NULL IDENTITY(1, 1),
    Client_id int  NOT NULL,
    Company_name varchar(50)  NOT NULL,
    NIP int  NOT NULL,
    CONSTRAINT Companies_pk PRIMARY KEY  (Company_id)
);
```

Definiowanie relacji pomiędzy tabelą Companies a Clients.

```
ALTER TABLE Companies ADD CONSTRAINT Companies_Clients
    FOREIGN KEY (Client_id)
    REFERENCES Clients (Client_id);
```

Customers

Tabela zawiera informacje o klienta (osobach prywatnych), którzy dokonali rezerwacji wycieczki w systemie.

Nazwa atrybutu	Typ	Opis/Uwagi
Customer_id	int	PK
Client_id	int	FK

Nazwa atrybutu	Typ	Opis/Uwagi
Firstname	varchar(40)	Imię
Lastname	varchar(40)	Nazwisko

```
CREATE TABLE Customers (
    Customer_id int  NOT NULL IDENTITY(1, 1),
    Client_id int  NOT NULL,
    Firstname varchar(40)  NOT NULL,
    Lastname varchar(40)  NOT NULL,
    CONSTRAINT Customers_pk PRIMARY KEY  (Customer_id)
);
```

Definiowanie relacji pomiędzy tabelą Customers a Clients.

```
ALTER TABLE Customers ADD CONSTRAINT Customers_Clients
    FOREIGN KEY (Client_id)
    REFERENCES Clients (Client_id);
```

Trips

Tabela zawiera wycieczki, które oferuje biuro turystyczne. Dodatkowo mamy sprawdzenie czy limit oraz cena za osobę są wartościami dodatnimi.

Nazwa atrybutu	Typ	Opis/Uwagi
Trip_id	int	PK
Name	int	Nazwa wycieczki
City_id	int	FK
Date	date	Data odbycia się wycieczki
Price_per_person	money	Cena za osobę
Limit	int	Limit osób na wycieczce
Start_reservation	datetime	Data rozpoczęcia rezerwacji

```
CREATE TABLE Trips (
    Trip_id int  NOT NULL IDENTITY(1, 1),
    Name varchar(40)  NOT NULL,
    City_id int  NOT NULL,
    Date date  NOT NULL,
    Price_per_person money  NOT NULL,
    Limit int  NOT NULL,
    Start_reservation datetime  NOT NULL,
```

```
    CONSTRAINT check_limit CHECK (Limit > 0),
    CONSTRAINT check_price_per_person CHECK (Price_per_person > 0),
    CONSTRAINT Trips_pk PRIMARY KEY  (Trip_id)
);
```

Definiowanie relacji pomiędzy tabelą Trips a Cities.

```
ALTER TABLE Trips ADD CONSTRAINT Trips_Cities
    FOREIGN KEY (City_id)
    REFERENCES Cities (City_id);
```

Cities

Tabela zawiera nazwy miast, w których odbywają się wycieczki.

Nazwa atrybutu	Typ	Opis/Uwagi
City_id	int	PK
City	varchar(50)	Miasto
Country_id	int	FK

```
CREATE TABLE Cities (
    City_id int  NOT NULL IDENTITY(1, 1),
    City varchar(50)  NOT NULL,
    Country_id int  NOT NULL,
    CONSTRAINT Cities_pk PRIMARY KEY  (City_id)
);
```

Definiowanie relacji pomiędzy tabelą Cities a Country

```
ALTER TABLE Cities ADD CONSTRAINT Cities_Country
    FOREIGN KEY (Country_id)
    REFERENCES Country (Country_id);
```

Country

Tabela słownikowa zawierająca dopuszczalne nazwy państw.

Nazwa atrybutu	Typ	Opis/Uwagi
Customer_id	int	PK
Country	varchar(50)	Nazwa kraju

```
CREATE TABLE Country (
    Country_id int  NOT NULL IDENTITY(1, 1),
    Country varchar(50)  NOT NULL,
    CONSTRAINT Country_pk PRIMARY KEY  (Country_id)
);
```

Attractions

Atrakcje oferowane przez biuro podróży. Aby można było dokonać zakupu atrakcji dodatkowych należy najpierw zakupić wycieczkę. Dodatkowo mamy sprawdzenie czy limit miejsc oraz cena za osobę są dodatnie.

Nazwa atrybutu	Typ	Opis/Uwagi
Attraction_id	int	PK
Trip_id	int	FK
Name	varchar(50)	Nazwa atrakcji
Price_per_person	varchar(50)	Cena za osobę
Limit	int	Limit osób na wycieczce

```
CREATE TABLE Attractions (
    Attraction_id int  NOT NULL IDENTITY(1, 1),
    Trip_id int  NOT NULL,
    Name varchar(50)  NOT NULL,
    Price_per_person money  NOT NULL,
    Limit int  NOT NULL,
    CONSTRAINT check_limit CHECK (Limit > 0),
    CONSTRAINT check_price_per_person CHECK (Price_per_person > 0),
    CONSTRAINT Attractions_pk PRIMARY KEY  (Attraction_id)
);
```

Definiowanie relacji pomiędzy tabelą Attractions a Trips_Attractions

```
ALTER TABLE Attractions ADD CONSTRAINT Trips_Attractions
    FOREIGN KEY (Trip_id)
    REFERENCES Trips (Trip_id);
```

AttractionReservations

W tej tabeli umieszczane są zarezerwowane atrakcje. Dodatkow mamy sprawdzenie czy liczba zarezerwowanych miejsc jest dodatnia oraz czy cena za osobę również jest dodatnia.

Nazwa atrybutu	Typ	Opis/Uwagi

Nazwa atrybutu	Typ	Opis/Uwagi
RA_id	int	Rezerwacja atrakcji (PK)
Price_per_person	money	Cena za osobę
Count_of_trippers	int	Liczba zarezerwowanych miejsc
Attraction_id	varchar(50)	FK
Reservation_id	int	FK

```
CREATE TABLE AttractionReservations (
    RA_id int  NOT NULL IDENTITY(1, 1),
    Price_per_person money  NOT NULL,
    Count_of_trippers int  NOT NULL,
    Attraction_id int  NOT NULL,
    Reservation_id int  NOT NULL,
    CONSTRAINT check_count_of_trippers CHECK (Count_of_trippers > 0),
    CONSTRAINT check_price_per_person CHECK (Price_per_person > 0),
    CONSTRAINT AttractionReservations_pk PRIMARY KEY  (RA_id)
);
```

Definiowanie relacji pomiędzy tabelą AttractionReservations a Attractions.

```
ALTER TABLE AttractionReservations ADD CONSTRAINT
AttractionReservations_Attractions
    FOREIGN KEY (Attraction_id)
    REFERENCES Attractions (Attraction_id);
```

Definiowanie relacji pomiędzy tabelą AttractionReservations a Reservations.

```
ALTER TABLE AttractionReservations ADD CONSTRAINT
AttractionReservations_Reservations
    FOREIGN KEY (Reservation_id)
    REFERENCES Reservations (Reservation_id);
```

Attractions_Trippers

Tabela zawiera wylistowanych uczestników poszczególnych atrakcji.

Nazwa atrybutu	Typ	Opis/Uwagi
AT_id	int	Atracja dla danej wycieczki (PK)
Tripper_id	int	FK
RA_id	int	FK

```
CREATE TABLE Attractions_Trippers (
    AT_id int  NOT NULL IDENTITY(1, 1),
    Tripper_id int  NOT NULL,
    RA_id int  NOT NULL,
    CONSTRAINT Attractions_Trippers_pk PRIMARY KEY  (AT_id)
);
```

Definiowanie relacji pomiędzy tabelą Attractions_Trippers a AttractionReservations.

```
ALTER TABLE Attractions_Trippers ADD CONSTRAINT
Attractions_Trippers_AttractionReservations
    FOREIGN KEY (RA_id)
    REFERENCES AttractionReservations (RA_id);
```

Definiowanie relacji pomiędzy tabelą Attractions_Trippers a Trippers.

```
ALTER TABLE Attractions_Trippers ADD CONSTRAINT Attractions_Trippers_Trippers
    FOREIGN KEY (Tripper_id)
    REFERENCES Trippers (Tripper_id);
```

Reservations

W tabeli tej mamy umieszczone informacje o wszystkich dokonanych rezerwacjach. Dodano również sprawdzenie czy cena za osobę oraz liczba zarezerwowanych miejsc są dodatnie.

Nazwa atrybutu	Typ	Opis/Uwagi
Reservation_id	int	PK
Trip_id	int	FK
Client_id	int	FK
Count_of_trippers	int	Liczba zarezerwowanych miejsc
Cancelled	bit	Informacja czy rezerwacja została anulowana

```
CREATE TABLE Reservations (
    Reservation_id int  NOT NULL IDENTITY(1, 1),
    Trip_id int  NOT NULL,
    Client_id int  NOT NULL,
    Price_per_person money  NOT NULL,
    Count_of_trippers int  NOT NULL,
    Cancelled bit  NOT NULL,
    CONSTRAINT check_price_per_person CHECK (Price_per_person > 0),
    CONSTRAINT Reservations_pk PRIMARY KEY  (Reservation_id)
);
```

Definiowanie relacji pomiędzy tabelą Reservations a Clients.

```
ALTER TABLE Reservations ADD CONSTRAINT Reservations_Clients
    FOREIGN KEY (Client_id)
    REFERENCES Clients (Client_id);
```

Definiowanie relacji pomiędzy tabelą Reservations a Trips.

```
-- Reference: Reservations_Trips (table: Reservations)
ALTER TABLE Reservations ADD CONSTRAINT Reservations_Trips
    FOREIGN KEY (Trip_id)
    REFERENCES Trips (Trip_id);
```

Trippers

Tabela zawiera wylistowanych uczestników poszczególnych wycieczek, które znajdują się w ofercie biura turystycznego.

Nazwa atrybutu	Typ	Opis/Uwagi
Tripper_id	int	PK
Reservation_id	int	FK
Person_id	int	FK

```
CREATE TABLE Trippers (
    Tripper_id int  NOT NULL IDENTITY(1, 1),
    Reservation_id int  NOT NULL,
    Person_id int  NOT NULL,
    CONSTRAINT Trippers_pk PRIMARY KEY  (Tripper_id)
);
```

Definiowanie relacji pomiędzy tabelą Trippers a People.

```
ALTER TABLE Trippers ADD CONSTRAINT Trippers_People
    FOREIGN KEY (Person_id)
    REFERENCES People (Person_id);
```

Definiowanie relacji pomiędzy tabelą Trippers a Reservations.

```
ALTER TABLE Trippers ADD CONSTRAINT Trippers_Reservations
    FOREIGN KEY (Reservation_id)
    REFERENCES Reservations (Reservation_id);
```

People

Tabela zawiera dane uczestników poszczególnych wycieczek.

Nazwa atrybutu	Typ	Opis/Uwagi
Person_id	int	PK
Client_id	int	FK
Firstname	varchar(40)	Imię
Lastname	varchar(40)	Nazwisko
Email	varchar(100)	Adres email
Phone	varchar(9)	Nr telefonu

```
CREATE TABLE People (
    Person_id int  NOT NULL IDENTITY(1, 1),
    Client_id int  NOT NULL,
    Firstname varchar(40)  NOT NULL,
    Lastname varchar(40)  NOT NULL,
    Email varchar(100)  NOT NULL,
    Phone varchar(9)  NOT NULL,
    CONSTRAINT People_pk PRIMARY KEY  (Person_id)
);
```

Definiowanie relacji pomiędzy uczestnikami, a klientami zamawiającymi wycieczki.

```
ALTER TABLE People ADD CONSTRAINT People_Clients
    FOREIGN KEY (Client_id)
    REFERENCES Clients (Client_id);
```

Payments

Tabela zawiera wszystkie operacje (wpłaty i zwrot na konto) jakich dokonano w biurze turystycznym. Dodano sprawdzenie czy kwota transakcji jest dodatnia.

Nazwa atrybutu	Typ	Opis/Uwagi
Payment_id	int	PK

Nazwa atrybutu	Typ	Opis/Uwagi
Paid	money	Kwota zapłacona/wypłacona
Timestamp	datetime	Czas dokonania transakcji
Reservation_id	int	FK

```
CREATE TABLE Payments (
    Payment_id int  NOT NULL IDENTITY(1, 1),
    Paid money  NOT NULL,
    Timestamp datetime  NOT NULL,
    Reservation_id int  NOT NULL,
    CONSTRAINT check_paid CHECK (Paid > 0),
    CONSTRAINT Payments_pk PRIMARY KEY  (Payment_id)
);
```

Definiowanie relacji pomiędzy tabelą Payments a Reservations.

```
ALTER TABLE Payments ADD CONSTRAINT Payments_Reservations
    FOREIGN KEY (Reservation_id)
    REFERENCES Reservations (Reservation_id);
```

3. Widoki, procedury/funkcje, triggery

Widoki

Oferty wycieczek w biurze turystycznym

```
create view Trip_Offers as
select t.Name, ci.City, co.Country, t.Price_per_person, t.Limit, t.Date
from Trips t
join Cities ci on ci.City_id=t.City_id
join Country co on co.Country_id=ci.Country_id
join Attractions a on a.Trip_id=t.Trip_id
```

Stan konta dla firm.

```
create view Company_Balance as
select cl.Client_id, co.Company_name, cl.Balance
from Clients cl
join Companies co on co.Client_id=cl.Client_id
```

Stan konta dla osób prywatnych.

```
create view Customer_Balance as
select cu.Firstname, cu.Lastname, cl.Balance
from Clients cl
join Customers cu on cu.Client_id=cl.Client_id
```

Widok, który dla każdego klienta pokazuje ile będą go kosztować zarezerwowane przez niego wycieczki i atrakcje.

```
create view Trips_value_for_clients as
select c.Client_id,
       COALESCE((r.Count_of_trippers * r.Price_per_person) +
       (at.Count_of_trippers * at.Price_per_person),0)
       Trips_and_attractions_value
from Reservations r
left join Clients c on c.Client_id=r.Client_id
left join AttractionReservations at on at.Reservation_id=r.Reservation_id
where Cancelled = 'false'
```

Lista osób na poszczególne wycieczki, kóre oferuje biuro turystyczne.

```
create view Trippers_lists as
select Name as TripName, Firstname, Lastname
from People p
right join Trippers on p.Person_id=Trippers.Person_id
right join Reservations r on r.Reservation_id=Trippers.Reservation_id
left join Trips on Trips.Trip_id=r.Trip_id
```

Widok ten przedstawia stan konta dla klienta po uwzględnieniu dokonanych opłat oraz kosztów związanych z organizacją wycieczek i atrkacji dla określonej liczby osób.

```
select
    c.Client_id,
    COALESCE(pa.paid - COALESCE((r.Count_of_trippers * r.Price_per_person) +
    (at.Count_of_trippers * at.Price_per_person),0), 0)
    balance
from Reservations r
right join Clients c on c.Client_id=r.Client_id
left join AttractionReservations at on at.Reservation_id=r.Reservation_id
left join Payments pa on pa.Reservation_id=r.Reservation_id
```