

## ЗМІСТ

Вступ.....	5
1 Аналіз предметної області.....	7
2 Розроблення бази даних засобами скбд MySQL.....	10
2.1 Створення таблиць .....	10
2.2 Створення тригерів .....	24
2.3 Створення запитів .....	30
2.4 Створення представлень.....	36
2.5 Створення процедур та функцій.....	40
2.6 Додавання користувачів та надання їм прав .....	46
3 Розроблення бази даних засобами скбд PostgreSQL .....	48
3.1 Створення таблиць .....	48
3.2 Створення тригерів .....	62
3.3 Створення запитів .....	68
3.4 Створення представлень.....	76
3.5 Створення процедур та функцій.....	81
3.6 Додавання користувачів та надання їм прав .....	88
Висновки .....	89
Список використаних джерел .....	91
Додаток А ER-діаграма .....	92
Додаток Б Вигляд таблиць .....	93
Додаток В SQL-код бази даних у MySQL.....	105
Додаток Г SQL-код бази даних у PostgreSQL .....	135

					<b>КП.ПІ-18-01.05.02.00.000 ПЗ</b>			
<b>Змн.</b>	<b>Арк.</b>	<b>№ докум.</b>	<b>Підпис</b>	<b>Дата</b>				
Розроб.		Гринюк В. Р.			Розроблення бази даних з предметної області "Резервування квитків"		Лім.	Арк.
Перевір.		Шевчук О. В.						Аркушів
Н. контр.								
					Пояснювальна записка		ФКЕП ІФНТУНГ, ПІ-18-01	
							4	91

## ВСТУП

База даних (англ. database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти.

У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД). Система керування базами даних — це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних (за стандартом ISO/IEC 2382:2015). Найпопулярнішими СКБД є MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird та IBM DB2. [4]

Дослідження та розроблення бази даних необхідно починати з узагальнення та систематизації основних теоретичних понять та питань, пов'язаних із застосуванням різних технологій створення, проектування та реалізації баз даних для застосування в різних інформаційних системах.

Для виконання курсового проєкту було обрано системи керування базами даних MySQL та PostgreSQL.

MySQL — вільна система керування реляційними базами даних, яка була розроблена компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

PostgreSQL (вимовляється «Пост-грес-К'ю-ель», або «постгрес») — об'єктно-реляційна система керування базами даних (СКБД). Порівняно з

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

іншими проєктами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення. [4]

DataGrip — комерційна крос-платформна IDE для роботи з MySQL, PostgreSQL, Oracle, SQL Server, Sybase, DB2, SQLite, HyperSQL, Apache Derby і H2. Розробляється компанією JetBrains. DataGrip надає інструменти для роботи з об'єктами бази даних. При створенні або зміні таблиці, додаванні або зміні колонки, індексу, ключа генерується відповідний скрипт, який можна відразу виконати в базі, а можна скопіювати згенерований DDL-запит в редактор і працювати вже безпосередньо з кодом. DataGrip підтримує автодоповнення коду, шаблони для однотипного коду, пошук за кодом і перейменування, фільтр даних та навігація за даними, текстовий редактор, інтеграція з системами контролю версій (Git, Subversion та ін.). [5]

Виконання курсового проєкту з розроблення бази даних формує фахові компетентності, які визначають здатність володіти знаннями про інформаційні моделі даних, створювати програмне забезпечення для зберігання, видобування та опрацювання даних; забезпечувати захищеність програм і даних від несанкціонованих дій; аналізувати, вибирати і застосовувати методи і засоби для забезпечення інформаційної безпеки.

Метою курсового проєктування є закріплення, поглиблення та узагальнення знань, а також практичних навиків із розроблення та створення бази даних, структур таблиць і зв'язків між ними, роботи із системами керування базами даних MySQL та PostgreSQL, а саме: написання SQL-запитів на виведення, групування, сортування, об'єднання та оновлення даних, створення тригерів, представлень, процедур та функцій користувача, користувачів та надання їм відповідних прав.

Темою курсового проєкту є розроблення та дослідження методів і засобів проєктування бази даних для предметної області «Резервування квитків».

					<i>КП.ПІ-18-01.05.02.00.000 ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Для створення бази даних було опрацьовано предметну область "Резервування квитків". Розроблення логічної моделі методом «сутність-зв'язок» (ER-методом) виконано методом низхідного проектування. Спроековано ER-діаграму для предметної області «Резервування квитків». Виділено наступні сутності: авіаквитки, авіаквитки пасажирів, автобусні квитки, автобусні квитки пасажирів, міста, клієнти, фірми, рейси, міста рейсів, маршрути, корабельні квитки, корабельні квитки пасажирів, залізничні квитки, залізничні квитки пасажирів, транспортні засоби, квитки на транспорт, типи квитків, типи перевезень. Для усіх таблиця даної моделі даних було створено відповідні первинні ключі та прокладено усі потрібні зв'язки між таблицями.

База даних Tickets\_booking складається з вісімнадцяти таблиць:

- airplane\_tickets – список авіаквитків;
- airplane\_tickets\_has\_customers – список квитків пасажирів;
- bus\_tickets – список автобусних квитків;
- bus\_tickets\_has\_customers – список автобусних квитків пасажирів;
- cities – список міст;
- customers – список пасажирів;
- firms – список фірм;
- flights – список рейсів;
- flights\_has\_cities – список міст рейсів;
- routes – список маршрутів;
- ship\_tickets – список корабельних квитків;
- ship\_tickets\_has\_customers – список корабельних квитків пасажирів;
- train\_tickets – список залізничних квитків;
- train\_tickets\_has\_customers – список залізничних квитків пасажирів;

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- transport – список транспортних засобів;
- transport\_has\_type\_tickets – список кількості типів квитків
- type\_tickets – список типів квитків;
- type\_transportations – список типів перевезень.

Результатом аналізу предметної області є виділення наступних атрибутів:

- airplane\_tickets: a\_ticket\_ID, at\_price, Flights\_has\_Cities\_primaryID;
- airplane\_tickets\_has\_customers: a\_booking\_time, a\_seat, Airplane\_tickets\_a\_ticket\_ID, at\_class, comments, Customers\_customer\_ID;
- bus\_tickets: b\_ticket\_ID, bt\_price, idRoute, idType\_tickets;
- bus\_tickets\_has\_customers:
- cities: city\_ID, city\_name, country;
- customers: c\_birthday, c\_name, customer\_ID
- firms: email, firm\_ID, idTransportation, name, regin\_activity, tel;
- flights: endtime, f\_distance, f\_from, f\_to, flight\_ID, idFirm, idTransport, starttime, status;
- flights\_has\_cities: idCity1, idCity2, idFlight, primaryID;
- routes: Flights\_has\_Cities\_primaryID, idCity, route\_ID;
- ship\_tickets: Flights\_has\_Cities\_primaryID, idType\_tickets, s\_ticket\_ID, st\_price;
- ship\_tickets\_has\_customers: comments, Customers\_customer\_ID, s\_booking\_time, s\_class, s\_seat, Ship\_tickets\_s\_ticket\_ID;
- train\_tickets: idRoute, idType\_tickets, t\_ticket\_ID, tt\_price;
- train\_tickets\_has\_customers: comments, Customers\_customer\_ID, t\_booking\_time, t\_numwagon, t\_seat, Train\_tickets\_t\_ticket\_ID;
- transport: countplace, name, number, transport\_ID;
- transport\_has\_type\_tickets: count\_seats\_type, Transport\_idTransport, Type\_ticket\_a\_idType\_tickets;
- type\_tickets: percent\_afterpay, type\_afterpay, typeticket\_ID;

					КП.ІІІ-18-01.05.02.00.000 ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

— type\_transportations: transportation\_ID, type\_name.

При аналізі предметної області і розробці ER-діаграми було використано тип зв'язку "один-до-багатьох" між таблицями: customers (ключовим полем) - train\_tickets\_has\_customers (поле Customers\_customer\_ID), customers (ключовим полем) - ship\_tickets\_has\_customers (поле Customers\_customer\_ID), customers (ключовим полем) — bus\_tickets\_has\_customers (поле Customers\_customer\_ID), customers (ключовим полем) — airplane\_tickets\_has\_customers (поле Customers\_customer\_ID), train\_tickets (ключовим полем) — train\_tickets\_has\_customers (поле Train\_tickets\_t\_ticket\_ID), ship\_tickets (ключовим полем) — ship\_tickets\_has\_customers (поле Ship\_tickets\_s\_ticket\_ID), bus\_tickets (ключовим полем) - bus\_tickets\_has\_customers (поле Bus\_tickets\_b\_ticket\_ID), airplane\_tickets (ключовим полем) — airplane\_tickets\_has\_customers (поле Airplane\_tickets\_a\_ticket\_ID), train\_tickets (поле idType\_tickets) – Type\_tickets (ключовим полем), train\_tickets (поле idType\_tickets) – Type\_tickets (ключовим полем), bus\_tickets (поле idType\_tickets) – Type\_tickets (ключовим полем), ship\_tickets (поле idType\_tickets) – Type\_tickets (ключовим полем), airplane\_tickets (поле idType\_tickets) – Type\_tickets (ключовим полем), Type\_tickets (ключовим полем) – Transport\_has\_type\_tickets (поле Type\_tickets\_a\_idType\_tickets), Transport\_has\_type\_tickets (поле Transport\_idTransport) – Transport (ключовим полем), Transport (поле idFlight) – Flights (ключовим полем), Flights (поле idFirm) – Firms (ключовим полем), Firms (поле idTransportation) – Type\_transportation (ключовим полем), Flights (ключовим полем) – Flights\_has\_Cities (поле idFlight), Flights\_has\_Cities (поле idCity) – Cities (ключовим полем), Flights\_has\_Cities (ключовим полем) – train\_tickets (поле Flights\_has\_Cities\_primary\_ID), Flights\_has\_Cities (ключовим полем) – airplane\_tickets (поле Flights\_has\_Cities\_primary\_ID), Flights\_has\_Cities (ключовим полем) – bus\_tickets (поле Flights\_has\_Cities\_primary\_ID), Flights\_has\_Cities (ключовим полем) – ship\_tickets (поле Flights\_has\_Cities\_primary\_ID).

					КП.ІІІ-18-01.05.02.00.000 ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

## 2 РОЗРОБЛЕННЯ БАЗИ ДАНИХ ЗАСОБАМИ СКБД MYSQL

MySQL — це система управління реляційними базами даних, яка має більше шести тисяч об'єктів. Програма працює як сервер забезпечення “багатокористувацького доступу” до об'єктів баз даних. [5]

### 2.1 Створення таблиць

Для створення таблиць програмним способом використовують оператор CREATE TABLE. Для цього потрібно вказати наступні дані:

- ім'я таблиці, яке вказується після ключового слова CREATE TABLE;
- імена та визначення стовпців таблиці, що відділені комами;
- в деяких СУБД також вимагається, щоби було вказано місце розташування таблиці.

Щоб додати новий рядок в таблицю, потрібно вказати назву таблиці, перелічити назви стовпців та вказати значення для кожного стовпця за допомогою конструкції INSERT INTO назва\_таблиці (поле1, поле2 ...) VALUES (значення1, значення2 ...). [5]

У відповідності до розробленої схеми даних було створено SQL таблиці бази даних.

Створено таблицю "Type\_transportations". Структура таблиці наведена на рисунку 2.1.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Type_transportations (  
    transportation_ID INT AUTO_INCREMENT NOT NULL,  
    type_name VARCHAR(45) NOT NULL,  
    PRIMARY KEY (transportation_ID));
```

Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Type_transportations (type_name)  
VALUES ('Повітряне'), ('Колійне'), ('Водне'), ('Наземне');  
COMMIT ;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

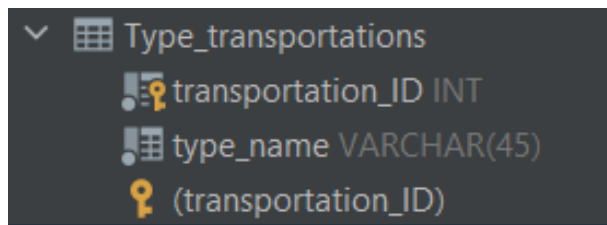


Рисунок 2.1 – Структура таблиці "Type\_transportations"

Створено таблицю " Firms". Структура таблиці наведена на рисунку 2.2.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Firms (
    firm_ID INT NOT NULL AUTO_INCREMENT,
    idTransportation INT NOT NULL,
    name VARCHAR(45) NOT NULL,
    email VARCHAR(45) NOT NULL,
    tel VARCHAR(20) NOT NULL,
    regin_activity VARCHAR(20) NULL,
    PRIMARY KEY (firm_ID),
    FOREIGN KEY (idTransportation) REFERENCES Type_transportations
    (transportation_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Firms(idTransportation, name, email, tel, regin_activity)
VALUES (1,'МАУ','uia@flyuia.com','+38 (044) 581-50-50','Україна'),
    (1,'KLM','KLM.Ukraine@klm.nl','+31 (0) 20 649 91
23','Нідерланди'),
    (1,'Emirates Airlines','pr@emirates.com','600 555 555','OAE'),
    ...
    (4,'Ecolines','help@ecolines.ua','+38 044 594 90 10',''),
    (4,'EAST WEST EUROLINES','support@ewe.ua','+380988154444',''),
    (4,'TransTempo','transtempo@ukr.net','+38 (067) 467-44-77','');
COMMIT ;
```

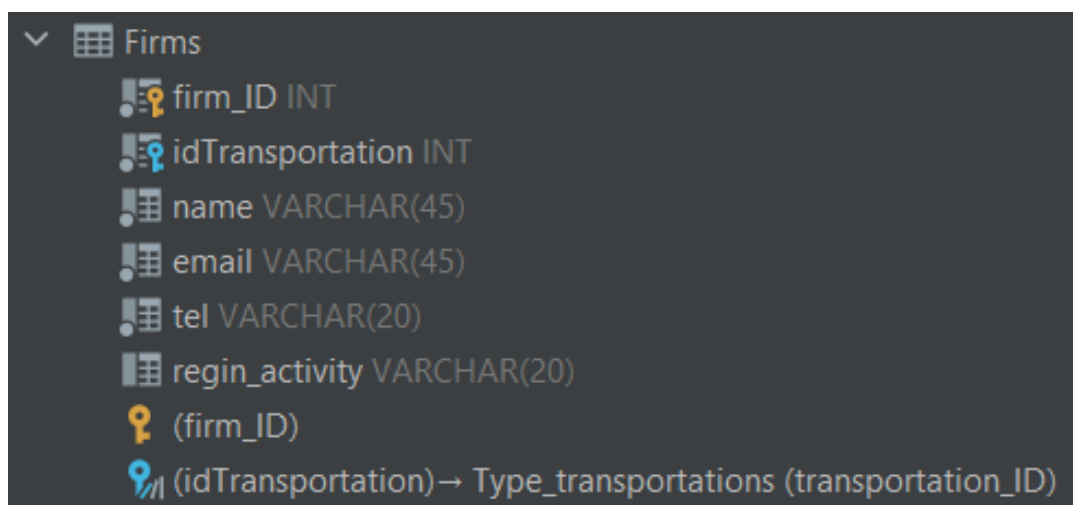


Рисунок 2.2 – Структура таблиці "Firms"



Створено таблицю "Flights". Структура таблиці наведена на рисунку 2.3.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Flights (  
    flight_ID INT NOT NULL AUTO_INCREMENT,  
    idFirm INT NOT NULL,  
    idTransport INT NOT NULL,  
    f_from VARCHAR(45) NULL,  
    f_to VARCHAR(45) NULL,  
    f_distance FLOAT NULL,  
    starttime DATETIME NOT NULL,  
    endtime DATETIME NULL,  
    status ENUM('Скасований', 'Відбувся', 'Очікується') DEFAULT  
'Очікується',  
    PRIMARY KEY (flight_ID),  
    FOREIGN KEY (idFirm) REFERENCES Firms(firm_ID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    FOREIGN KEY (idTransport) REFERENCES Transport(transport_ID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport,  
starttime)  
VALUES (8, 'Київ', 'Дніпро', 488, 51, '2021-10-24 21:22:46'),  
      (10, 'Херсон', 'Миколаїв', 80, 57, '2021-05-08 14:01:34'),  
      (11, 'Київ', 'Луганськ', 922, 6, '2021-08-09 20:18:56'),  
      ...  
      (10, 'Кропивницький', 'Миколаїв', 182, 53, '2021-03-10  
02:42:48'),  
      (3, 'Хмельницький', 'Берн', 1444, 4, '2021-03-19 11:52:21'),  
      (1, 'Київ', 'Осло', 1630, 3, '2021-01-02 18:36:57');  
COMMIT ;
```

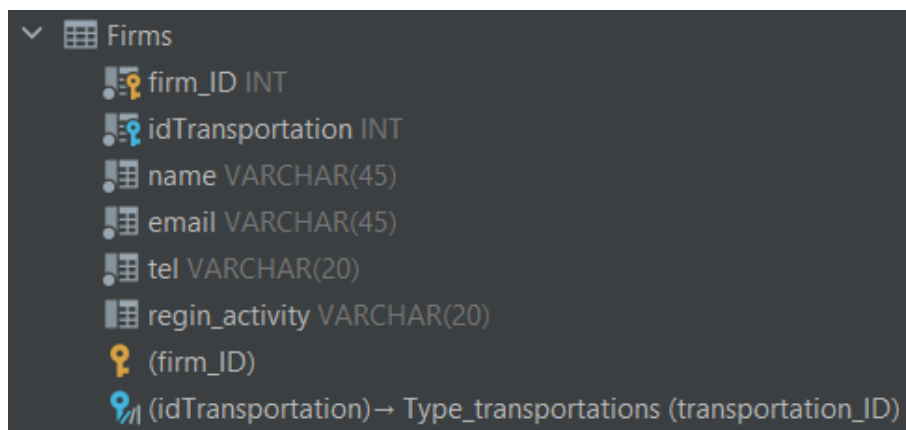


Рисунок 2.3 – Структура таблиці "Flights"

Створено таблицю "Customers". Структура таблиці наведена на рисунку 2.4.

Код створення таблиці:

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
CREATE TABLE IF NOT EXISTS Customers (
    customer_ID INT NOT NULL AUTO_INCREMENT,
    c_name VARCHAR(45) NOT NULL,
    c_birthday DATE NOT NULL,
    PRIMARY KEY (customer_ID));
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Customers(c_name, c_birthday)
VALUES ('Тарасович Ратимир Вадимович', '1952-08-10'),
       ('Ніколенко Ромашка Зорянівна', '1963-05-17'),
       ('Кириленко Муховіст Полянович', '1987-03-19'),
       ...
       ('Мухопад Царук Адріанович', '1982-04-20'),
       ('Бурбан Верніслав Артемович', '1993-03-05'),
       ('Штинь Атрей Русланович', '1956-11-19'),
       ('Деркач Лютобор Романович', '1966-07-25');
COMMIT ;
```

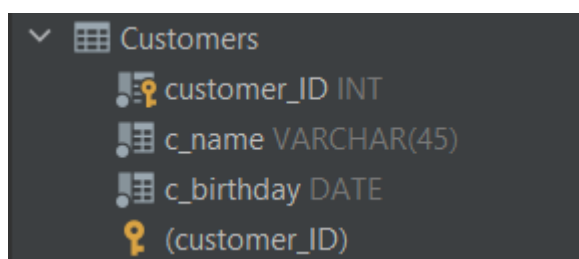


Рисунок 2.4 – Структура таблиці "Customers"

Створено таблицю "Type\_tickets". Структура таблиці наведена на рисунку 2.5.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Type_tickets (
    typeticket_ID INT NOT NULL AUTO_INCREMENT,
    type_afterpay ENUM('Купе дорослий', 'Купе дитячий', 'Купе студентський', 'Плацкарт дорослий',
        'Плацкарт дитячий', 'Плацкарт студентський', 'Бізнес клас',
        'Економ клас', 'Перший клас', 'Автобусний',
        'Автобусний студентський', 'Морський', 'Морський VIP', 'Морський економ') NOT NULL,
    percent_afterpay INT NOT NULL,
    PRIMARY KEY (typeticket_ID));
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Type_tickets (type_afterpay, percent_afterpay)
VALUES ('Купе дорослий', 50),
       ('Купе студентський', 30),
       ('Купе дитячий', 10),
       ('Плацкарт дорослий', 0),
       ('Плацкарт дитячий', -30),
       ('Плацкарт студентський', -50),
       ('Бізнес клас', 150),
       ('Економ клас', -10),
       ('Перший клас', 50),
       ('Автобусний', 0),
       ('Автобусний студентський', -20),
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

```

( 'Морський', 0),
( 'Морський VIP', 75),
( 'Морський економ', -15);
COMMIT ;

```

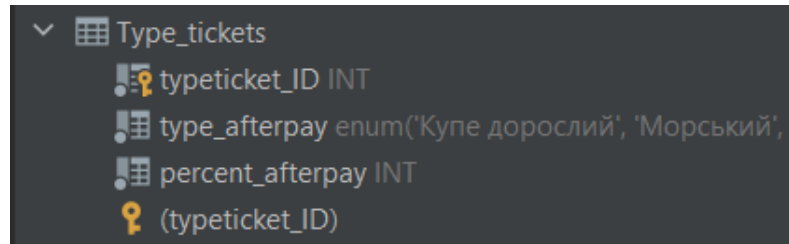


Рисунок 2.5 – Структура таблиці "Type\_tickets"

Створено таблицю "Cities". Структура таблиці наведена на рисунку 2.6.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Cities (
    city_ID INT NOT NULL AUTO_INCREMENT,
    city_name VARCHAR(45) NOT NULL,
    country VARCHAR(45) NOT NULL,
    PRIMARY KEY (city_ID));

```

Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Cities (country, city_name)
VALUES ('Молдова', 'Кишинів'),
('Греція', 'Афіни'),
('Хорватія', 'Загреб'),
...
('Україна', 'Хмельницький'),
('Україна', 'Черкаси'),
('Україна', 'Чернівці'),
('Україна', 'Чернігів');
COMMIT ;

```

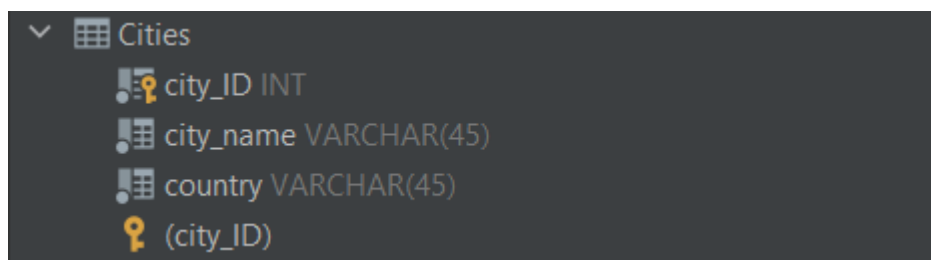


Рисунок 2.6 – Структура таблиці "Cities"

Створено таблицю "Flights\_has\_Cities". Структура таблиці наведена на рисунку 2.7.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Flights_has_Cities (
    idFlight INT NOT NULL,
    idCity1 INT NOT NULL,
    idCity2 INT NULL,

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

```

primaryID INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY (primaryID),
FOREIGN KEY (idFlight) REFERENCES Flights (flight_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (idCity1) REFERENCES Cities (city_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (idCity2) REFERENCES Cities (city_ID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Flights_has_Cities(idFlight, idCity1, idCity2)
(SELECT flight_ID, city_ID, (SELECT city_ID FROM Cities WHERE f_to =
city_name) FROM Flights, Cities
WHERE f_from = city_name);
COMMIT ;

```



Рисунок 2.7 – Структура таблиці "Flights\_has\_Cities"

Створено таблицю "Airplane\_tickets". Структура таблиці наведена на рисунку 2.8.

#### Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Airplane_tickets (
a_ticket_ID INT NOT NULL AUTO_INCREMENT,
at_price DECIMAL NULL,
idType_tickets INT NOT NULL,
Flights_has_Cities_primaryID INT NOT NULL,
PRIMARY KEY (a_ticket_ID),
FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES
Flights_has_Cities (primaryID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

BEGIN ;

```

					КП.ІІІ-18-01.05.02.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

```

INSERT INTO Airplane_tickets(idType_tickets,
Flights_has_Cities_primaryID)
VALUES (7, 10),
      (9, 50),
      (9, 13),
      (8, 25),
      ...
      (8, 37),
      (9, 7),
      (8, 6);
COMMIT ;

```

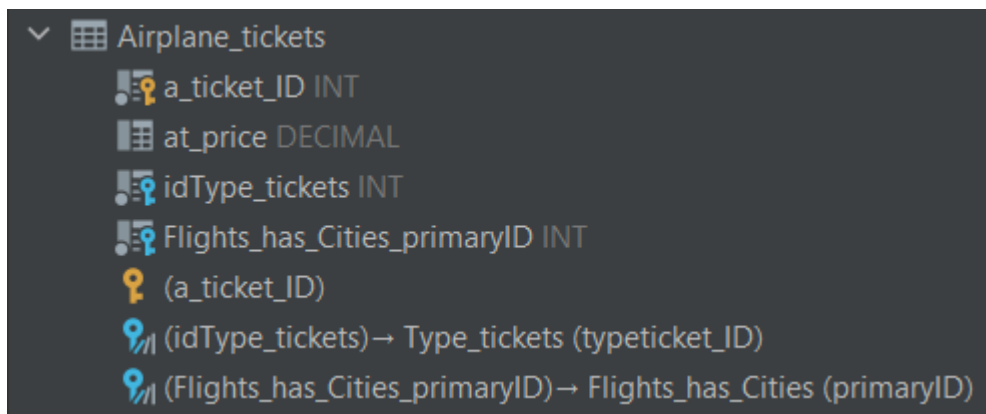


Рисунок 2.8 – Структура таблиці "Airplane\_tickets"

Створено таблицю "Routes". Структура таблиці наведена на рисунку 2.9.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Routes (
  route_ID INT NOT NULL AUTO_INCREMENT,
  Flights_has_Cities_primaryID INT NOT NULL,
  idCity INT NOT NULL,
  PRIMARY KEY (route_ID),
  FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES
Flights_has_Cities (primaryID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idCity) REFERENCES Cities (city_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Routes(Flights_has_Cities_primaryID, idCity)
VALUES ( 8, 60),
      ( 8, 63),
      ( 8, 50),
      ( 29, 43),
      ...
      ( 47, 64),
      ( 47, 55),
      ( 9, 50),
      ( 9, 60);
COMMIT ;

```

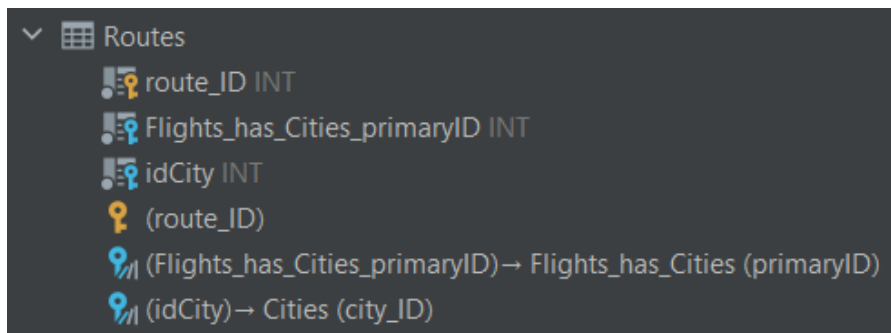


Рисунок 2.9 – Структура таблиці "Routes"

Створено таблицю "Train\_tickets". Структура таблиці наведена на рисунку 2.10.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Train_tickets (
  t_ticket_ID INT NOT NULL AUTO_INCREMENT,
  tt_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  idRoute INT NOT NULL,
  PRIMARY KEY (t_ticket_ID),
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idRoute) REFERENCES Routes (route_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Train_tickets( tt_price, idType_tickets, idRoute)
VALUES ( 394, 1, 31),
       ( 554, 1, 32),
       ( 221, 2, 33),
       ...
       ( 185, 1, 53),
       ( 394, 4, 54),
       ( 472, 5, 55);

COMMIT ;
```

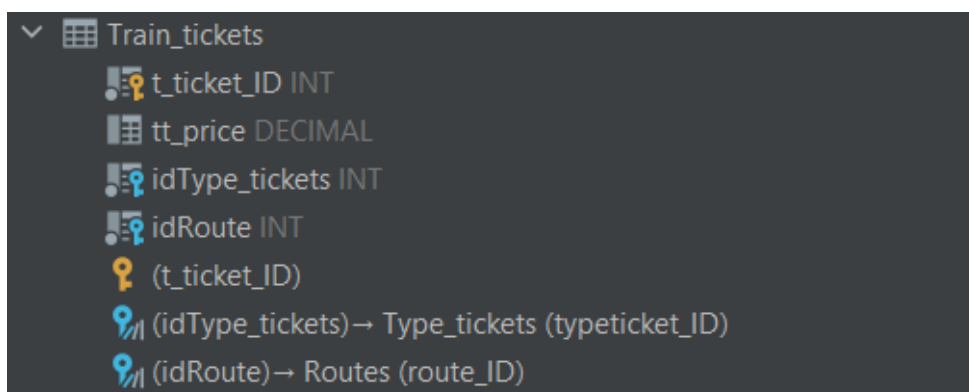


Рисунок 2.10 – Структура таблиці "Train\_tickets"

Створено таблицю "Bus\_tickets". Структура таблиці наведена на рисунку

2.11.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Bus_tickets (  
    b_ticket_ID INT NOT NULL AUTO_INCREMENT,  
    bt_price DECIMAL NULL,  
    idType_tickets INT NOT NULL,  
    idRoute INT NOT NULL,  
    PRIMARY KEY (b_ticket_ID),  
    FOREIGN KEY (idRoute) REFERENCES Routes (route_ID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Bus_tickets(bt_price, idType_tickets, idRoute)  
VALUES ( 318, 10, 1),  
        ( 212, 10, 2),  
        ( 448, 10, 3),  
        ...  
        ( 638, 11, 8),  
        ( 953, 10, 9),  
        ( 3754, 10, 10);  
COMMIT ;
```

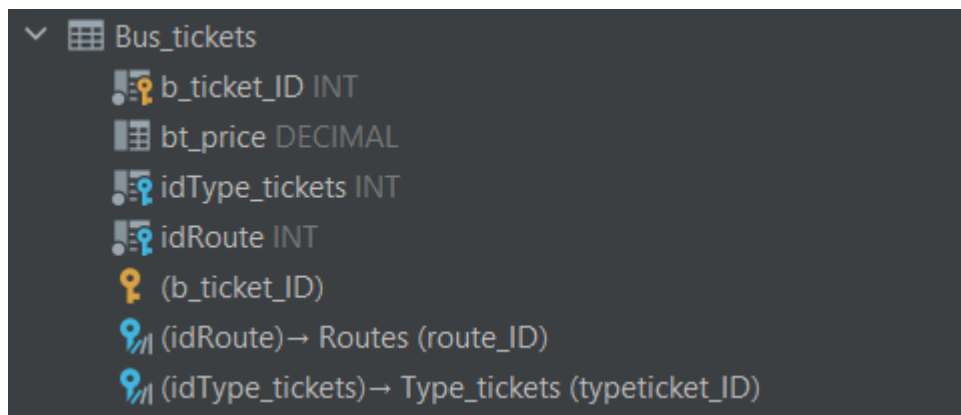


Рисунок 2.11 – Структура таблиці "Bus\_tickets"

Створено таблицю "Ship\_tickets". Структура таблиці наведена на рисунку 2.12.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Ship_tickets (  
    s_ticket_ID INT NOT NULL AUTO_INCREMENT,  
    st_price DECIMAL NULL,  
    idType_tickets INT NOT NULL,  
    Flights_has_Cities_primaryID INT NOT NULL,  
    PRIMARY KEY (s_ticket_ID),  
    FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

```

ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES
Flights_has_Cities (primaryID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Ship_tickets(idType_tickets, Flights_has_Cities_primaryID)
VALUES ( 12, 32),
      ( 13, 32),
      ( 14, 32),
      ...
      ( 13, 32),
      ( 13, 32),
      ( 13, 32);
COMMIT ;

```

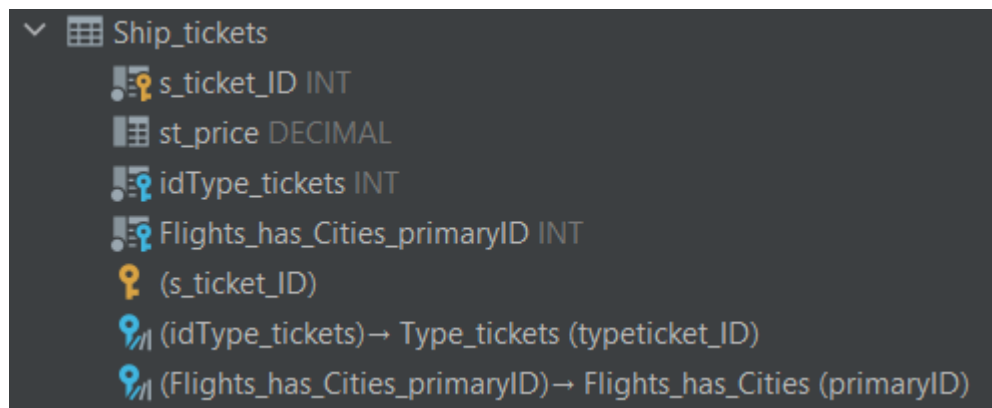


Рисунок 2.12 – Структура таблиці "Ship\_tickets"

Створено таблицю "Train\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 2.13.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Train_tickets_has_Customers (
  Train_tickets_t_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  t_booking_time DATETIME NOT NULL,
  t_numwagon INT NOT NULL,
  t_seat INT NOT NULL,
  comments VARCHAR(45) NULL,
  PRIMARY KEY (Train_tickets_t_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Train_tickets_t_ticket_ID) REFERENCES Train_tickets
(t_ticket_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

BEGIN ;

```



```

INSERT INTO Train_tickets_has_Customers(train_tickets_t_ticket_id,
customers_customer_id, t_booking_time, t_numwagon, t_seat)
VALUES ( 1, 2, '2020-03-15 09:59:44', 1, 21),
      ( 2, 4, '2020-09-01 05:27:15', 6, 13),
      ( 3, 6, '2020-11-11 16:03:38', 4, 17),
      ...
      ( 23, 46, '2020-05-06 04:59:54', 7, 42),
      ( 24, 48, '2020-12-04 22:03:56', 1, 45),
      ( 25, 50, '2020-05-01 14:58:58', 2, 21);
COMMIT ;

```

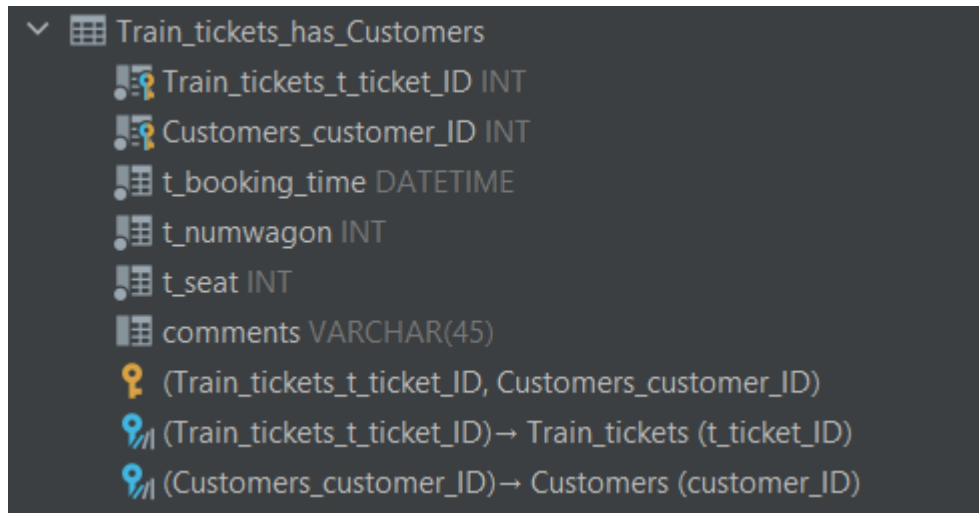


Рисунок 2.13 – Структура таблиці "Train\_tickets\_has\_Customers"

Створено таблицю "Airplane\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 2.14.

#### Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Airplane_tickets_has_Customers (
  Airplane_tickets_a_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  a_booking_time DATETIME NOT NULL,
  at_class VARCHAR(15) NULL,
  a_seat INT NOT NULL,
  comments VARCHAR(45) NULL,
  PRIMARY KEY (Airplane_tickets_a_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Airplane_tickets_a_ticket_ID) REFERENCES
Airplane_tickets (a_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

COMMIT ;
INSERT INTO Airplane_tickets_has_Customers
  (Airplane_tickets_a_ticket_ID, Customers_customer_ID,
  a_booking_time, a_seat, comments)
VALUES ( 1, 1, '2020-10-09 20:21:36', 1, NULL),
      ( 2, 2, '2020-01-14 09:55:01', 2, NULL),

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

```

( 3, 3, '2020-06-23 14:27:41', 3, NULL),
...
( 23, 23, '2020-04-16 16:54:07', 23, NULL),
( 24, 24, '2020-06-25 04:22:31', 24, NULL),
( 25, 25, '2020-10-28 08:31:29', 25, NULL);
COMMIT ;

```

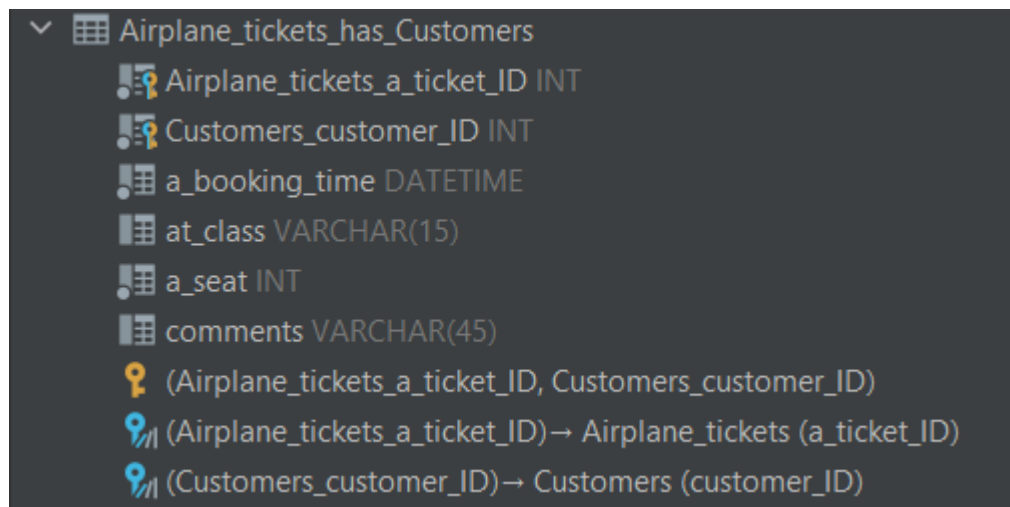


Рисунок 2.14 – Структура таблиці "Airplane\_tickets\_has\_Customers"

Створено таблицю "Bus\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 2.15.

#### Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Bus_tickets_has_Customers (
    Bus_tickets_b_ticket_ID INT NOT NULL,
    Customers_customer_ID INT NOT NULL,
    b_booking_time DATETIME NOT NULL,
    b_seat INT NOT NULL,
    comments VARCHAR(45) NULL,
    PRIMARY KEY (Bus_tickets_b_ticket_ID, Customers_customer_ID),
    FOREIGN KEY (Bus_tickets_b_ticket_ID) REFERENCES Bus_tickets
(b_ticket_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Bus_tickets_has_Customers(bus_tickets_b_ticket_id,
customers_customer_id, b_booking_time, b_seat)
VALUES ( 1, 26, '2020-05-08 02:11:23', 1),
( 2, 27, '2020-10-06 21:24:28', 4),
( 3, 28, '2020-05-14 23:59:25', 7),
...
( 33, 29, '2020-08-12 08:28:02', 19),
( 34, 26, '2020-10-04 00:22:49', 1),
( 35, 29, '2020-01-25 05:00:16', 4);
COMMIT ;

```

Bus_tickets_has_Customers
Bus_tickets_b_ticket_ID INT
Customers_customer_ID INT
b_booking_time DATETIME
b_seat INT
comments VARCHAR(45)
(Bus_tickets_b_ticket_ID, Customers_customer_ID)
(Bus_tickets_b_ticket_ID)→ Bus_tickets (b_ticket_ID)
(Customers_customer_ID)→ Customers (customer_ID)

Рисунок 2.15 – Структура таблиці "Bus\_tickets\_has\_Customers"

Створено таблицю "Ship\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 2.16.

#### Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Ship_tickets_has_Customers (
  Ship_tickets_s_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  s_booking_time DATETIME NOT NULL,
  s_class VARCHAR(15) NULL,
  s_seat INT NOT NULL,
  comments VARCHAR(45) NULL,
  PRIMARY KEY (Ship_tickets_s_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Ship_tickets_s_ticket_ID) REFERENCES Ship_tickets
(s_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);
```

#### Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Ship_tickets_has_Customers(ship_tickets_s_ticket_id,
customers_customer_id, s_booking_time, s_seat, comments)
VALUES ( 1, 26, '2020-03-24 05:43:44', 23, NULL),
( 2, 27, '2020-02-25 13:56:49', 97, NULL),
( 3, 28, '2020-09-11 10:54:59', 45, NULL),
( 4, 29, '2020-11-19 02:10:09', 12, NULL),
( 5, 30, '2020-02-14 01:36:50', 88, NULL);
COMMIT ;
```

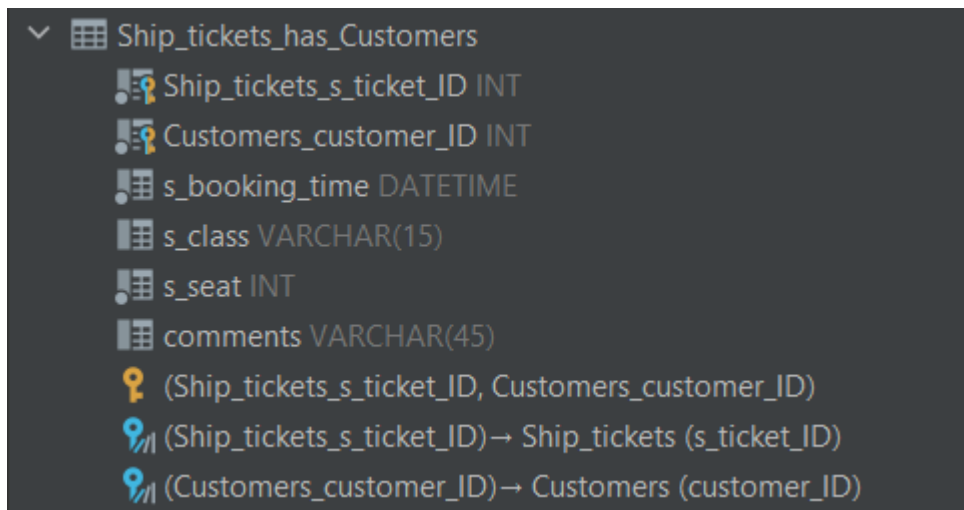


Рисунок 2.16 – Структура таблиці "Ship\_tickets\_has\_Customers"

Створено таблицю "Transport\_has\_Type\_tickets". Структура таблиці наведена на рисунку 2.17.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Transport_has_Type_tickets (
  Transport_idTransport INT NOT NULL,
  Type_ticketeta_idType_tickets INT NOT NULL,
  count_seats_type INT NOT NULL,
  PRIMARY KEY (Transport_idTransport, Type_ticketeta_idType_tickets),
  FOREIGN KEY (Transport_idTransport) REFERENCES Transport
  (transport_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Type_ticketeta_idType_tickets) REFERENCES Type_tickets
  (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Transport_has_Type_tickets(transport_idtransport,
type_ticketeta_idtype_tickets, count_seats_type)
VALUES (1, 7, 50),
      (1, 8, 75),
      ...
      (66,13, 50),
      (66,14, 50);
COMMIT ;
```

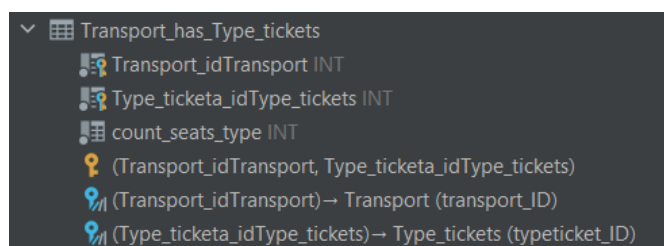


Рисунок 2.17 – Структура таблиці "Transport\_has\_Type\_tickets"

Створено таблицю "Transport". Структура таблиці наведена на рисунку

2.18.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Transport (  
    transport_ID INT NOT NULL AUTO_INCREMENT,  
    name VARCHAR(45) NOT NULL,  
    number VARCHAR(10) NOT NULL,  
    countplace INT NOT NULL,  
    PRIMARY KEY (transport_ID));
```

Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Transport(name, number, countplace)  
VALUES ( 'Boeing', '777', 250),  
       ( 'Airbus', 'A340', 170),  
       ( 'Airbus', 'A330', 125),  
       ...  
       ( 'Celebrity Xpedition', '', 5530),  
       ( 'Forse le Carib', '', 570),  
       ( 'Porco Fugo', '', 240);  
COMMIT ;
```

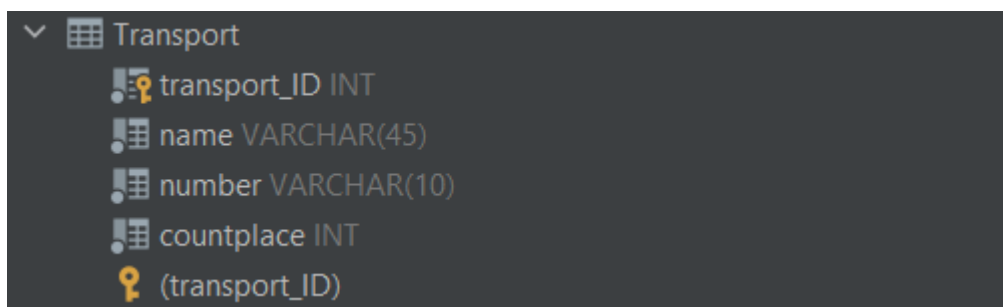


Рисунок 2.18 – Структура таблиці "Transport"

У ER-діаграмі бази даних "Резервування квитків" відображено всі таблиці, їх поля та зв'язки між таблицями і подано в додатку А. Вигляд таблиць подано у додатку Б.

## 2.2 Створення тригерів

Тригер – це збережена процедура, яка не викликається безпосередньо, а виконується при настанні певної події (вставка, видалення, оновлення рядка). [5]

У результаті аналізу завдання на курсове проєктування було створено наступні тригери.

Створено тригер, який записує видалені дані з таблиці “Transport” в таблицю “Del\_transport” (рисунок 2.19):

```
delimiter //
CREATE TRIGGER saveDelTransport BEFORE DELETE ON Transport
FOR EACH ROW
BEGIN
    INSERT INTO Del_transport(transport_ID, name, number,
countplace, time)
VALUES(OLD.transport_ID, OLD.name, OLD.number, OLD.countplace,
NOW());
END;//
delimiter ;
```

Код створення допоміжної таблиці:

```
CREATE TABLE Del_transport LIKE Transport;
ALTER TABLE Del_transport ADD COLUMN time DATETIME NULL;
```

Виконано тестове видалення для перевірки результату:

```
DELETE FROM Transport WHERE transport_ID < 5;
```

	transport_ID	name	number	countplace	time
1	1	Boeing	777	250	2021-05-18 19:53:06
2	2	Airbus	A340	170	2021-05-18 19:53:06
3	3	Airbus	A330	125	2021-05-18 19:53:06
4	4	Boeing	747	245	2021-05-18 19:53:06

Рисунок 2.19 – Результат виконання тригера

Створено тригер, який викликає збережену процедуру, що обчислює приблизний час прибуття рейсу (рисунок 2.20):

```
delimiter //
CREATE TRIGGER setEndTimeFlight BEFORE INSERT ON Flights
FOR EACH ROW
BEGIN
    CASE
        WHEN NEW.idFirm <= 7
            THEN CALL calc_end_time(NEW.endtime, NEW.f_distance,
NEW.starttime, 800);
        WHEN NEW.idFirm > 7 AND NEW.idFirm <= 11
            THEN CALL calc_end_time(NEW.endtime, NEW.f_distance,
NEW.starttime, 50);
        WHEN NEW.idFirm > 11 AND NEW.idFirm <= 15
            THEN CALL calc_end_time(NEW.endtime, NEW.f_distance,
NEW.starttime, 22);
        WHEN NEW.idFirm > 15
            THEN CALL calc_end_time(NEW.endtime, NEW.f_distance,
NEW.starttime, 65);
    END CASE;
END;//
delimiter ;
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport,
starttime)
VALUES (8, 'Київ', 'Дніпро', 488, 51, '2021-10-24 21:22:46');
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

```
(10, 'Херсон', 'Миколаїв', 80, 57, '2021-05-08 14:01:34');
```

	flight_ID	f_from	f_to	starttime	endtime
1	1	Київ	Дніпро	2021-10-24 21:22:46	2021-10-25 07:38:46
2	2	Херсон	Миколаїв	2021-05-08 14:01:34	2021-05-08 16:01:34

Рисунок 2.20 – Результат виконання тригера

Створено тригер, який виводить користувачке повідомлення про помилку, якщо введено від’ємне значення відстані (рисунок 2.21):

```
delimiter //
CREATE TRIGGER checkDistance BEFORE INSERT ON Flights
FOR EACH ROW
BEGIN
CASE
WHEN NEW.f_distance <= 0
THEN SIGNAL sqlstate '45001' SET message_text = 'Invalid
distance!';
WHEN NEW.f_from REGEXP '[0-9]' OR NEW.f_to REGEXP '[0-9]'
THEN SIGNAL sqlstate '45001' SET message_text = 'This
string can't include digits!';
ELSE BEGIN END;
END CASE;
END;//
delimiter ;
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport,
starttime)
VALUES (8, 'Київ', 'Дніпро', -488, 51, '2021-10-24 21:22:46');
```

```
tickets_booking> INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport, starttime)
VALUES (8, 'Київ', 'Дніпро', -488, 51, '2021-10-24 21:22:46')
[2021-05-18 20:16:35] [45001][1644] Invalid distance!
[2021-05-18 20:16:35] [HY000][1644] Invalid distance!
```

Рисунок 2.21 – Результат виконання тригера

Створено тригер, який встановлює статус рейсу відносно поточної дати та часу (рисунок 2.22):

```
delimiter //
CREATE TRIGGER checkStatus BEFORE UPDATE ON Flights
FOR EACH ROW
BEGIN
CASE
WHEN CURRENT_TIMESTAMP < NEW.endtime AND OLD.status =
'Відбувся'
THEN SET NEW.status = 'Очікується';
WHEN NEW.endtime < CURRENT_TIMESTAMP AND OLD.status =
'Очікується'
THEN SET NEW.status = 'Відбувся';
END CASE;
END;//
delimiter ;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

Введено тестові дані для перевірки результату:

```
UPDATE Flights SET endtime = '2021-01-08 22:00:00' WHERE flight_ID = 6;
```

	flight_ID	f_from	f_to	starttime	endtime	status
1	6	Київ	Тирана	2021-01-08 21:21:34	2021-01-08 22:00:00	Відбувся

Рисунок 2.22 – Результат виконання тригера

Створено тригер, який викликає процедуру, що встановлює вартість авіаквитка (рисунок 2.23):

```
delimiter //
CREATE TRIGGER setPriceAirTicket BEFORE INSERT ON Airplane_tickets
FOR EACH ROW
BEGIN
    CALL calc_price_at( NEW.at_price,
NEW.Flights_has_Cities_primaryID, NEW.idType_tickets);
END; //
delimiter ;
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Airplane_tickets(idType_tickets,
Flights_has_Cities_primaryID)
VALUES (7, 10),
(9, 50),
(9, 13);
```

	a_ticket_ID	at_price	idType_tickets	Flights_has_Cities_primaryID
1	1	7127	7	10
2	2	2374	9	50
3	3	4804	9	13

Рисунок 2.23 – Результат виконання тригера

Створено тригер, який визначає в якому класі знаходиться місце пасажира за його квитком (рисунок 2.24):

```
delimiter //
CREATE TRIGGER setClassAirHasCus BEFORE INSERT ON
Airplane_tickets_has_Customers
FOR EACH ROW
BEGIN
    CASE
        WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 7
            THEN SET NEW.at_class = 'Бізнес';
        WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 8
            THEN SET NEW.at_class = 'Економ';
        WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 9
            THEN SET NEW.at_class = 'Перший';
    END CASE;
END; //
delimiter ;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		



Введено тестові дані для перевірки результату:

```
INSERT INTO Airplane_tickets_has_Customers(Airplane_tickets_a_ticket_ID,
Customers_customer_ID, a_booking_time, a_seat, comments)
VALUES ( 1, 1, '2020-10-09 20:21:36', 1, NULL),
( 2, 2, '2020-01-14 09:55:01', 2, NULL);
```

	Airplane_...	Customers_cust...	a_booking_time	at_class	a_seat	comments
1		1	2020-10-09 20:21:36	Бізнес	1	<null>
2		2	2020-01-14 09:55:01	Перший	2	<null>

Рисунок 2.24 – Результат виконання тригера

Створено тригер, який викликає процедуру, що встановлює вартість квитка на водні види транспорту (рисунок 2.25):

```
delimiter //
CREATE TRIGGER setPriceShipTicket BEFORE INSERT ON Ship_tickets
FOR EACH ROW
BEGIN
CALL calc_price_st( NEW.st_price,
NEW.Flights_has_Cities_primaryID, NEW.idType_tickets);
END;//
delimiter ;
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Ship_tickets(idType_tickets, Flights_has_Cities_primaryID)
VALUES ( 12, 32),
( 13, 32);
```

	s_ticket_ID	st_price	idType_tickets	Flights_has_Cities_prima...
1	1	8930	12	32
2	2	11064	13	32

Рисунок 2.25 – Результат виконання тригера

Створено тригер, який визначає в якому класі знаходиться місце пасажира за його квитком (рисунок 2.26):

```
delimiter //
CREATE TRIGGER setClassShipHasCus BEFORE INSERT ON
Ship_tickets_has_Customers
FOR EACH ROW
BEGIN
CASE
WHEN (SELECT idType_tickets
FROM Ship_tickets
WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 12
THEN SET NEW.s_class = 'Стандарт';
WHEN (SELECT idType_tickets
FROM Ship_tickets
WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 13
THEN SET NEW.s_class = 'Преміум';
WHEN (SELECT idType_tickets
FROM Ship_tickets
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

```

        WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 14
        THEN SET NEW.s_class = 'Економ';
    END CASE;
END; //
delimiter ;

```

**Введено тестові дані для перевірки результату:**

```

INSERT INTO Ship_tickets_has_Customers(ship_tickets_s_ticket_id,
customers_customer_id, s_booking_time, s_seat, comments)
VALUES ( 1, 26, '2020-03-24 05:43:44', 23, NULL),
( 2, 27, '2020-02-25 13:56:49', 97, NULL);

```

	Ship_tickets_s_ticket_ID	Customers_customer_ID	s_booking_time	s_class	s_seat	comments
1	1	26	2020-03-24 05:43:44	Стандарт	23	<null>
2	2	27	2020-02-25 13:56:49	Преміум	97	<null>

**Рисунок 2.26 – Результат виконання тригера**

Створено тригер, який викликає процедуру, що встановлює вартість автобусного квитка (рисунок 2.27):

```

delimiter //
CREATE TRIGGER setPriceBusTicket BEFORE INSERT ON Bus_tickets
FOR EACH ROW
BEGIN
    CALL calc_price_bt(NEW.bt_price,NEW.idType_tickets);
END; //
delimiter ;

```

**Введено тестові дані для перевірки результату:**

```

INSERT INTO Bus_tickets(bt_price, idType_tickets, idRoute)
VALUES ( 318, 10, 1),
( 212, 10, 2);

```

	b_ticket_ID	bt_price	idType_tickets	idRoute
1	1	318	10	1
2	2	212	10	2

**Рисунок 2.27 – Результат виконання тригера**

Створено тригер, який викликає процедуру, що встановлює вартість квитка на поїзд (рисунок 2.28):

```

delimiter //
CREATE TRIGGER setPriceTrainTicket BEFORE INSERT ON Train_tickets
FOR EACH ROW
BEGIN
    CALL calc_price_tt(NEW.tt_price,NEW.idType_tickets);
END; //
delimiter ;

```

**Введено тестові дані для перевірки результату:**

```

INSERT INTO Train_tickets( tt_price, idType_tickets, idRoute)
VALUES ( 394, 1, 31),

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

( 554, 1, 32);

	t_ticket_ID	tt_price	idType_tickets	idRoute
1	1	296	1	31
2	2	416	1	32

Рисунок 2.28 – Результат виконання тригера

## 2.3 Створення запитів

SELECT — оператор мови SQL, котрий повертає рядки з однієї чи багатьох таблиць. Повний синтаксис оператора SELECT є складним, проте його можна описати так:

SELECT список\_вибірки

[ INTO нова\_таблиця ]

FROM таблиця

[ WHERE умови\_пошуку ]

[ GROUP BY умова\_групування ]

[ HAVING умови\_пошуку ]

[ ORDER BY умова\_сортування [ ASC | DESC ] ]

Повертає нуль або більше рядків з однієї або більше таблиць, тимчасових таблиць, або ж представлень бази даних. У більшості застосунків, SELECT — найчастіша команда Data Manipulation Language (DML).[6]

У результаті аналізу завдання на курсове проектування було створено наступні запити, які відображають усі можливі дії з даними, що містяться у таблицях.

Створено запит, який формує список пасажирів що придбали плацкартний білет (рисунок 2.29):

```
SELECT c_name, concat(f_from, ' - ', f_to) AS Рейс, CONCAT(f_from, ' - ', city_name) AS Маршрут, f_distance, tt_price
FROM Customers INNER JOIN Train_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID INNER JOIN
Train_tickets Tt
ON TthC.Train_tickets_t_ticket_ID = Tt.t_ticket_ID INNER JOIN Routes R
ON Tt.idRoute = R.route_ID INNER JOIN Flights_has_Cities FhC
ON R.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID INNER JOIN Cities C
ON R.idCity = C.city_ID INNER JOIN Type_tickets T on Tt.idType_tickets =
T.typeticket_ID
WHERE type_afterpay = 'Плацкарт студентський'
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		30

ORDER BY tt\_price;

	c_name	Рейс	Маршрут	f_distance	tt_price
1	Токарчук Родіон Найденович	Херсон - Київ	Херсон - Київ	612	136
2	Сімович Цвітан Августинів	Львів - Харків	Львів - Полтава	1109	233
3	Чубатий Івантослав Давидович	Львів - Харків	Львів - Харків	1109	252

Рисунок 2.29 – Результат виконання запиту

Створено запит, який формує список пасажирів що придбали плацкартний білет (рисунок 2.30):

```
SELECT DISTINCT concat(f_from, ' - ', f_to) AS Рейс, starttime
FROM Customers INNER JOIN Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID INNER JOIN
Bus_tickets Bt
ON BthC.Bus_tickets_b_ticket_ID = Bt.b_ticket_ID INNER JOIN Routes R
ON Bt.idRoute = R.route_ID INNER JOIN Flights_has_Cities FhC
ON R.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID INNER JOIN Cities C
ON R.idCity = C.city_ID
WHERE DAYOFMONTH(starttime) BETWEEN 0 AND 15 AND MONTH(starttime) IN
(6,7,8);
```

	Рейс	starttime
1	Тернопіль - Рівне	2021-07-10 19:58:31
2	Чернігів - Кропивницький	2021-07-08 10:02:28

Рисунок 2.30 – Результат виконання запиту

Створено запит, який виводить кількість проданих та зарезервованих квитків кожного виду (рисунок 2.31):

```
SELECT type_afterpay, LPAD(CONCAT(COUNT(typeticket_ID), ' шт. '),10,' ')
AS Sold
FROM Type_tickets INNER JOIN Airplane_tickets A
ON Type_tickets.typeticket_ID = A.idType_tickets INNER JOIN
Airplane_tickets_has_Customers AthC
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID GROUP BY 1
UNION
SELECT type_afterpay, LPAD(CONCAT(COUNT(typeticket_ID), ' шт. '),10,' ')
FROM Type_tickets INNER JOIN Train_tickets T
ON Type_tickets.typeticket_ID = T.idType_tickets INNER JOIN
Train_tickets_has_Customers TthC
ON T.t_ticket_ID = TthC.Train_tickets_t_ticket_ID GROUP BY 1
UNION
SELECT type_afterpay, LPAD(CONCAT(COUNT(typeticket_ID), ' шт. '),10,' ')
FROM Type_tickets INNER JOIN Bus_tickets B
ON Type_tickets.typeticket_ID = B.idType_tickets INNER JOIN
Bus_tickets_has_Customers BthC
ON B.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID GROUP BY 1
UNION
SELECT type_afterpay, LPAD(CONCAT(COUNT(typeticket_ID), ' шт. '),10,' ')
FROM Type_tickets INNER JOIN Ship_tickets S
ON Type_tickets.typeticket_ID = S.idType_tickets INNER JOIN
Ship_tickets_has_Customers SthC
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

ON S.s\_ticket\_ID = SthC.Ship\_tickets\_s\_ticket\_ID GROUP BY 1;

	type_afterpay	Sold
3	Перший клас	7 шт.
4	Купе дорослий	4 шт.
5	Купе студентський	5 шт.
6	Купе дитячий	3 шт.
7	Плацкарт дорослий	7 шт.
8	Плацкарт дитячий	3 шт.
9	Плацкарт студентський	3 шт.
10	Автобусний	27 шт.
11	Автобусний студентський	8 шт.
12	Морський	2 шт.
13	Морський VIP	2 шт.
14	Морський економ	1 шт.

Рисунок 2.31 – Результат виконання запиту

Створено запит, який виводить інформацію про рейси, що відправляються за межі України (рисунок 2.32):

```
SELECT T.name AS Transport, CONCAT(f_from, ' - ', f_to) AS Flight,
F.name AS Firm, type_name AS Type_trip
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
ON Flights.idTransport = T.transport_ID INNER JOIN Flights_has_Cities
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON FhC.idCity2 = C.city_ID
WHERE country <> 'Україна';
```

	Transport	Flight	Firm	Type_trip
1	Enchantment of the Seas	Одеса - Афіни	Royal Caribbean	Водне
2	Boeing Next Generation	Київ - Тирана	KLM	Повітряне
3	Dassault Falcon	Рівне - Монако	Emirates Airlines	Повітряне
4	AN	Житомир - Люксембург	MAY	Повітряне
5	Farman Goliath	Чернівці - Бухарест	MAY	Повітряне
6	Boeing	Донецьк - Мінськ	Lufthansa	Повітряне
7	Dassault Falcon	Івано-Франківськ - Валлетта	Turkish Airlines	Повітряне
8	AN	Київ - Вільнюс	Wizz Air	Повітряне
9	Neoplan	Одеса - Мадрид	EuroClub	Наземне
10	Farman Goliath	Кропивницький - Амстердам	Turkish Airlines	Повітряне
11	Dassault Falcon	Луцьк - Любляна	MAY	Повітряне
12	Boeing Classic	Черкаси - Брюссель	Emirates Airlines	Повітряне
13	Airbus	Донецьк - Андорра-ла-Велья	Lufthansa	Повітряне
14	Boeing Next Generation	Хмельницький - Париж	KLM	Повітряне
15	Dassault Falcon	Суми - Гельсінкі	Turkish Airlines	Повітряне
16	AN	Харків - Бухарест	KLM	Повітряне
17	Douglas	Ужгород - Кишинів	MAY	Повітряне
18	Douglas	Київ - Лісабон	KLM	Повітряне
19	AN	Полтава - Софія	MAY	Повітряне
20	Boeing Classic	Миколаїв - Сараєво	MAY	Повітряне
21	Boeing	Хмельницький - Берн	Emirates Airlines	Повітряне
22	Airbus	Київ - Осло	MAY	Повітряне

Рисунок 2.32 – Результат виконання запиту

Створено запит, що виводить співвідношення наявної та загальної кількостей квитків на автобусні рейси (рисунки 2.33):

```
SELECT CONCAT(f_from, ' - ', f_to) AS Flight, CONCAT(T.name, '\t',
T.number) AS Bus_and_number, type_name,
CONCAT(T.countplace - (SELECT COUNT(Bus_tickets_b_ticket_ID)
FROM Bus_tickets_has_Customers INNER JOIN Bus_tickets Bt
ON Bus_tickets_has_Customers.Bus_tickets_b_ticket_ID =
Bt.b_ticket_ID
INNER JOIN Routes R ON Bt.idRoute = R.route_ID INNER JOIN
Flights_has_Cities FhC
ON R.Flights_has_Cities_primaryID = FhC.primaryID
WHERE flight_ID = idFlight), '/', T.countplace) AS Tickets
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
ON Flights.idTransport = T.transport_ID
WHERE type_name = 'Наземне';
```

	Flight	Bus_and_number	type_name	Tickets
1	Житомир - Івано-Франківськ	ЕТАЛОН BI7837EA	Наземне	22/28
2	Миколаїв - Хмельницький	ISUZU BC6097MK	Наземне	24/28
3	Тернопіль - Харків	GULERYUZ BK3137HH	Наземне	16/24
4	Одеса - Мадрид	Neoplan AT6426EP	Наземне	26/30
5	Тернопіль - Рівне	Neoplan BC5474MK	Наземне	31/32
6	Чернігів - Кропивницький	Богдан BC3458MK	Наземне	22/25
7	Миколаїв - Київ	Богдан AE5501AA	Наземне	22/25
8	Вінниця - Чернігів	Neoplan BI7798EX	Наземне	28/30
9	Херсон - Київ	ISUZU BM1049CM	Наземне	26/30
10	Дніпро - Харків	Neoplan AX5109HX	Наземне	30/30
11	Дніпро - Луцьк	ISUZU AB2775HT	Наземне	33/33

Рисунок 2.33 – Результат виконання запиту

Створено запит, який вибирає назву міста в яке відправляється найбільша кількість залізничних рейсів (рисунки 2.34):

```
SELECT city_name, COUNT(idCity2) AS count
FROM Routes INNER JOIN Flights_has_Cities FhC
ON Routes.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Cities
C
ON FhC.idCity2 = C.city_ID INNER JOIN Flights
ON FhC.idFlight = flight_ID INNER JOIN Firms F
ON Flights.idFirm = firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
ON Flights.idTransport = T.transport_ID
WHERE type_name = 'Колійне'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

	city_name	count
1	Луганськ	10

Рисунок 2.34 – Результат виконання запиту

Створено запит, який виводить рейси з середньою вартістю квитків більше 5000 грн., які прямують в будь-які країни світу окрім України, Франції і Болгарії (рисунок 2.35):

```
SELECT CONCAT(f_from, ' - ', f_to) AS Flight, AVG(at_price) AS
avg_ticket_price
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Airplane_tickets A
ON FhC.primaryID = A.Flights_has_Cities_primaryID INNER JOIN Cities C
ON FhC.idCity2 = C.city_ID
WHERE country NOT IN ('Україна', 'Франція', 'Болгарія')
GROUP BY 1
HAVING 5000 < avg_ticket_price
ORDER BY 2;
```

	Flight	avg_ticket_price
1	Київ - Тирана	5138.0000
2	Хмельницький - Берн	6016.0000
3	Київ - Осло	6353.0000
4	Рівне - Монако	6403.0000
5	Житомир - Люксембург	7127.0000
6	Кропивницький - Амстердам	7547.0000
7	Черкаси - Брюссель	7768.0000
8	Івано-Франківськ - Валлетта	8567.0000
9	Донецьк - Андорра-ла-Велья	11250.0000
10	Київ - Лісабон	15141.0000

Рисунок 2.35 – Результат виконання запиту

Створено запит, який виводить ініціали пасажирів, що придбали більше одного квитка на автобусні рейси (рисунок 2.36):

```
SELECT c_name, COUNT(b_ticket_ID) AS tickets, SUM(bt_price) AS total_sum
FROM Customers LEFT JOIN Bus_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID RIGHT JOIN
Bus_tickets Tt
ON Tt.b_ticket_ID = TthC.Bus_tickets_b_ticket_ID
GROUP BY 1
HAVING tickets <> 1
ORDER BY 3 DESC;
```

	c_name	tickets	total_sum
1	Лещинська Жозефіна Августинівна	4	4952
2	Німченко Йосифата Борисівна	5	2355
3	Смішко Іларіон Ігорович	3	1473
4	Ганущак Фауст Антонович	2	302

Рисунок 2.36 – Результат виконання запиту

Створено запит, який виводить найдовший та найкоротший за тривалістю рейси (рисунок 2.37):

```
(SELECT 'Найтриваліший рейс' AS Descrip, CONCAT(f_from, ' - ', f_to) AS Flight,
SEC_TO_TIME(TIMESTAMPDIFF(SECOND, starttime, endtime)) AS time_interval
FROM Flights
ORDER BY 3 DESC
LIMIT 1)
UNION
(SELECT 'Найкоротший рейс', CONCAT(f_from, ' - ', f_to) AS Flight,
SEC_TO_TIME(TIMESTAMPDIFF(SECOND, starttime, endtime))
FROM Flights
ORDER BY 3
LIMIT 1);
```

	Descrip	Flight	time_interval
1	Найтриваліший рейс	Одеса - Афіни	81:18:00
2	Найкоротший рейс	Луганськ - Суми	00:33:00

Рисунок 2.37 – Результат виконання запиту

Створено запит, який виводить автобусні рейси до Харкова та Києва з вартістю менше середньої (рисунок 2.38):

```
SELECT b_ticket_ID, CONCAT(f_from, ' - ', city_name) AS Flight,
CONCAT(bt_price, ' грн.')
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID INNER JOIN Bus_tickets
Bt
ON R.route_ID = Bt.idRoute INNER JOIN Bus_tickets_has_Customers BthC
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE bt_price < (SELECT AVG(bt_price)
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID INNER JOIN
Bus_tickets Bt
ON R.route_ID = Bt.idRoute INNER JOIN Bus_tickets_has_Customers BthC
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID INNER JOIN Cities C
ON FhC.idCity2 = C.city_ID)
HAVING Flight LIKE '%Київ' OR Flight LIKE '%Харків'
ORDER BY bt_price DESC;
```



	b_ticket_ID	Flight	price
1	7	Тернопіль - Харків	577 грн.
2	32	Тернопіль - Харків	577 грн.
3	22	Херсон - Київ	545 грн.
4	8	Тернопіль - Харків	510 грн.
5	33	Тернопіль - Харків	510 грн.
6	19	Миколаїв - Київ	384 грн.
7	17	Миколаїв - Київ	310 грн.
8	25	Херсон - Київ	294 грн.
9	24	Херсон - Київ	243 грн.
10	18	Миколаїв - Київ	184 грн.
11	6	Тернопіль - Харків	90 грн.
12	31	Тернопіль - Харків	90 грн.
13	23	Херсон - Київ	66 грн.

Рисунок 2.38 – Результат виконання запиту

## 2.4 Створення представлень

Представлення – об’єкт бази даних, що є результатом виконання запиту до бази даних, визначеного за допомогою оператора SELECT, в момент звернення до представлення.

Представлення іноді називають «віртуальними таблицями». Така назва пов’язана з тим, що представлення доступне для користувача як таблиця, але саме воно не містить даних, а витягує їх з таблиць в момент звернення до нього. Якщо дані змінені в базовій таблиці, то користувач отримає актуальні дані при зверненні до представлення, котрі використовують цю таблицю.

Представлення можуть ґрунтуватися як на таблицях, так і на інших представленнях, тобто можуть бути вкладеними (до 32 рівнів вкладеності). [6]

У відповідності до поставленого завдання на курсове проектування було створено наступні представлення.

Створено представлення, яке містить інформацію про всі продані залізничні квитки (рисунок 2.39):

```
CREATE VIEW sold_train_tickets AS SELECT c_name, concat(f_from, ' - ',
f_to) AS Рейс,
CONCAT(f_from, ' - ', city_name) AS Маршрут, f_distance, tt_price
FROM Customers INNER JOIN Train_tickets_has_Customers TthC
```

```

ON Customers.customer_ID = TthC.Customers_customer_ID INNER JOIN
Train_tickets Tt
ON TthC.Train_tickets_t_ticket_ID = Tt.t_ticket_ID INNER JOIN Routes R
ON Tt.idRoute = R.route_ID INNER JOIN Flights_has_Cities FhC
ON R.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID INNER JOIN Cities C
ON R.idCity = C.city_ID INNER JOIN Type_tickets T on Tt.idType_tickets =
T.typeticket_ID
ORDER BY tt_price;

```

№	c_name	Рейс	Маршрут	f_distance	tt_price
1	Луційчук Дан Жданович	Херсон - Київ	Херсон - Миколаїв	612	36
2	Могиленко Йосип Сарматович	Херсон - Миколаїв	Херсон - Миколаїв	80	43
3	Довгань Гаїна Тимурівна	Херсон - Київ	Херсон - Кропивницький	612	85
4	Боечко Лев Антонович	Рівне - Луганськ	Рівне - Житомир	1252	94
5	Дурдинець Триріг Любомирович	Івано-Франківськ - Суми	Івано-Франківськ - Хмельницький	1007	133
6	Токарчук Родіон Найденович	Херсон - Київ	Херсон - Київ	612	136
7	Вайда Турбід Драганович	Кропивницький - Миколаїв	Кропивницький - Миколаїв	182	139
8	Вітренко Власт Денисович	Полтава - Миколаїв	Полтава - Кропивницький	463	144
9	Барвінський Антон Охримович	Рівне - Луганськ	Рівне - Київ	1252	164
10	Деркач Лятобор Романович	Київ - Луганськ	Київ - Харків	922	165
11	Бурбан Верніслав Артемович	Київ - Луганськ	Київ - Полтава	922	197
12	Нечитайло Данко Мстиславович	Ужгород - Чернівці	Ужгород - Чернівці	432	200
13	Міняйло Йосифата Ярославівна	Полтава - Миколаїв	Полтава - Миколаїв	463	222
14	Сімович Цвітін Августинів	Львів - Харків	Львів - Полтава	1109	233
15	Пастушенко Ніна Ростиславівна	Херсон - Київ	Херсон - Черкаси	612	239
16	Чубатий Івантослав Давидович	Львів - Харків	Львів - Харків	1109	252
17	Медяник Йозеф Богуславович	Івано-Франківськ - Вінниця	Івано-Франківськ - Вінниця	381	273
18	Ніколенко Ромашка Зорянівна	Київ - Дніпро	Київ - Полтава	488	296
19	Павлович Гладко Пилипович	Львів - Харків	Львів - Київ	1109	350
20	Замора Єгор Жданович	Рівне - Луганськ	Рівне - Полтава	1252	360
21	Мулярчук Хвалімир Остапович	Івано-Франківськ - Суми	Івано-Франківськ - Київ	1007	371
22	Німченко Йосифата Борисівна	Рівне - Луганськ	Рівне - Харків	1252	399
23	Смішко Іларіон Ігорович	Рівне - Луганськ	Рівне - Луганськ	1252	402
24	Аліськевич Мар'ян Устимович	Київ - Дніпро	Київ - Дніпро	488	416
25	Вінтоняк Альберт Пилипович	Івано-Франківськ - Суми	Івано-Франківськ - Суми	1007	453

Рисунок 2.39 – Результат виконання представлення

Створено представлення, яке виводить рейси, що відправляються влітку (рисунок 2.40):

```

CREATE VIEW summer_flights AS SELECT DISTINCT concat(f_from, ' - ',
f_to) AS Рейс, starttime
FROM Customers INNER JOIN Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID INNER JOIN
Bus_tickets Bt
ON BthC.Bus_tickets_b_ticket_ID = Bt.b_ticket_ID INNER JOIN Routes R
ON Bt.idRoute = R.route_ID INNER JOIN Flights_has_Cities FhC
ON R.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID INNER JOIN Cities C
ON R.idCity = C.city_ID
WHERE MONTH(starttime) IN (6,7,8);

```

№	Рейс	starttime
1	Миколаїв - Хмельницький	2021-06-23 10:10:45
2	Одеса - Мадрид	2021-08-20 16:54:35
3	Тернопіль - Рівне	2021-07-10 19:58:31
4	Чернігів - Кропивницький	2021-07-08 10:02:28

Рисунок 2.40 – Результат виконання представлення

Створено представлення, яке виводить інформацію про транспортні засоби, що не мають рейсів (рисунок 2.41):

```
CREATE VIEW unused_transport AS SELECT *
FROM Transport
WHERE transport_ID NOT IN (
    SELECT DISTINCT transport_ID
    FROM Transport INNER JOIN Flights F
    ON Transport.transport_ID = F.idTransport);
```

	transport_ID	name	number	countplace
1	13	AN	178	110
2	19	Neoplan	AE5740KP	30
3	20	Neoplan	AX3116BM	40
4	21	ЕТАЛОН	BM6175CM	30
5	24	Богдан	AB1994IA	25
6	25	ISUZU	AE1095AA	32
7	26	Богдан	AE1095AA	25
8	27	ЕТАЛОН	BH4091AA	28
9	28	ЕТАЛОН	BC4780MB	28
10	29	ЕТАЛОН	B05913CP	28
11	30	Богдан	AE48580X	25
12	32	Neoplan	KA2376BT	30
13	33	ISUZU	AT7929EP	26
14	35	Богдан	AC1501E0	25
15	36	ISUZU	A02753EP	29
16	37	ISUZU	KA10020P	34
17	38	Богдан	BT1552CC	25
18	41	GULERYUZ	BC9664MB	22
19	42	ЕТАЛОН	AM9254E0	28
20	44	GULERYUZ	BX5339EM	36
21	45	GULERYUZ	BI8403EE	28
22	46	Neoplan	BH21370H	32
23	47	ЕТАЛОН	AA4243PT	30
24	62	Symphony of the Seas		1300
25	63	Celebrity Xploration		845
26	64	Celebrity Xpedition		5530
27	65	Forse le Carib		570
28	66	Porco Fugo		240

Рисунок 2.41 – Результат виконання представлення

Створено представлення, яке виводить рейси, що відправляються за межі України (рисунок 2.42):

```
CREATE VIEW foreign_flights AS SELECT T.name AS Transport,
CONCAT(f_from, ' - ', f_to) AS Flight, F.name AS Firm, type_name AS
Type_trip
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

ON Flights.idTransport = T.transport_ID INNER JOIN Flights_has_Cities
FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON FhC.idCity2 = C.city_ID
WHERE country <> 'Україна';

```

	Transport	Flight	Firm	Type_trip
1	Enchantment of the Seas	Одеса - Афіни	Royal Caribbean	Водне
2	Boeing Next Generation	Київ - Тирана	KLM	Повітряне
3	Dassault Falcon	Рівне - Монако	Emirates Airlines	Повітряне
4	AN	Житомир - Люксембург	MAY	Повітряне
5	Farman Goliath	Чернівці - Бухарест	MAY	Повітряне
6	Boeing	Донецьк - Мінськ	Lufthansa	Повітряне
7	Dassault Falcon	Івано-Франківськ - Валлетта	Turkish Airlines	Повітряне
8	AN	Київ - Вільнюс	Wizz Air	Повітряне
9	Neoplan	Одеса - Мадрид	EuroClub	Наземне
10	Farman Goliath	Кропивницький - Амстердам	Turkish Airlines	Повітряне
11	Dassault Falcon	Луцьк - Любляна	MAY	Повітряне
12	Boeing Classic	Черкаси - Брюссель	Emirates Airlines	Повітряне
13	Airbus	Донецьк - Андорра-ла-Велья	Lufthansa	Повітряне
14	Boeing Next Generation	Хмельницький - Париж	KLM	Повітряне
15	Dassault Falcon	Суми - Гельсінкі	Turkish Airlines	Повітряне
16	AN	Харків - Бухарест	KLM	Повітряне
17	Douglas	Ужгород - Кишинів	MAY	Повітряне
18	Douglas	Київ - Лісабон	KLM	Повітряне
19	AN	Полтава - Софія	MAY	Повітряне
20	Boeing Classic	Миколаїв - Сараєво	MAY	Повітряне
21	Boeing	Хмельницький - Берн	Emirates Airlines	Повітряне
22	Airbus	Київ - Осло	MAY	Повітряне

Рисунок 2.42 – Результат виконання представлення

Створено представлення, яке виводить п'ять найпопулярніших автобусних та залізничних рейсів. (рисунок 2.43):

```

CREATE VIEW the_most_popular_travel_city AS SELECT city_name,
COUNT(idCity2)
FROM Routes INNER JOIN Flights_has_Cities FhC
ON Routes.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Cities
C
ON FhC.idCity2 = C.city_ID INNER JOIN Flights
ON FhC.idFlight = flight_ID INNER JOIN Firms F
ON Flights.idFirm = firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
ON Flights.idTransport = T.transport_ID
WHERE type_name IN ('Наземне', 'Колійне')
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;

```

	city_name	'COUNT(idCity2)'
1	Київ	11
2	Луганськ	10
3	Харків	8
4	Івано-Франківськ	5
5	Миколаїв	4

Рисунок 2.43 – Результат виконання представлення

## 2.5 Створення процедур та функцій

В MySQL 5 є багато нових можливостей, однією з найважливіших із яких є створення збережених процедур та функцій. Збережені процедури і функції є підпрограмами, що створюються за допомогою операторів CREATE PROCEDURE і CREATE FUNCTION.

Процедура викликається за допомогою оператора call і може тільки передавати значення назад, використовуючи вихідні змінні.

Збережена процедура - це спосіб інкапсуляції повторюваних дій. В збережених процедурах можна оголошувати змінні, управляти потоками даних, а також застосовувати інші техніки програмування. Збережені процедури можуть викликати інші збережені процедури.

Функція - фрагмент програми, який повертає значення, а її виклик може використовуватися в програмі як вираз. Можна також розглядати функцію як окрему систему - чорну скриньку, на вхід якої надходять деякі управляючі дії у вигляді значень аргументів, а на виході системи одержуємо результат виконання.

Функції можуть повертати скалярне значення і викликаються з оператора точно так само, як і будь-які інші функції (тобто, через вказівку імені функції)[6].

У відповідності до поставленого завдання на курсове проектування було створено наступні процедури та функції.

Створено функцію, для переведення чисел з плаваючою крапкою у формат "hh:mm:ss" (рисунок 2.44):

```
delimiter //
CREATE FUNCTION getHour (inHour FLOAT)
RETURNS TIME
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE result VARCHAR(15);
    IF ~(ROUND(inHour,2)*100)%100 >= 60
        THEN SET result = CONCAT(FLOOR(inHour) + 1, ':',
        (~(ROUND(inHour,2)*100)%100) - 60, ':00');
        ELSE SET result = CONCAT(FLOOR(inHour), ':',
        ~(ROUND(inHour,2)*100)%100, ':00');
    END IF;
    RETURN result;
END;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```
delimiter ;
SELECT getHour(1.3);
```

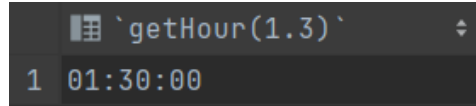


Рисунок 2.44 – Результат виконання функції

Створено функцію, яка генерує випадкову дату з проміжку (рисунок 2.45):

```
delimiter //
CREATE FUNCTION randDATETIME (from_date DATETIME, to_date DATETIME)
RETURNS DATETIME
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE result DATETIME;
    SET result = FROM_UNIXTIME(RAND()*(UNIX_TIMESTAMP(to_date)-
        UNIX_TIMESTAMP(from_date))+UNIX_TIMESTAMP(from_date));
    RETURN result;
END;//
delimiter ;
SELECT randDATETIME('2000:01:01 00:00:00', CURRENT_TIMESTAMP) AS
rand_datetime, randDATETIME('2000:01:01 00:00:00', CURRENT_TIMESTAMP) AS
rand_datetime, randDATETIME('2000:01:01 00:00:00', CURRENT_TIMESTAMP) AS
rand_datetime, randDATETIME('2000:01:01 00:00:00', CURRENT_TIMESTAMP) AS
rand_datetime;
```

	rand_datetime	rand_datetime	rand_datetime	rand_datetime
1	2015-12-16 04:54:08	2013-11-25 21:44:51	2000-04-05 15:35:08	2002-05-12 23:35:32

Рисунок 2.45 – Результат виконання функції

Створено функцію, яка повертає всі числа, що знаходяться у вхідному рядку (рисунок 2.46):

```
DELIMITER $$
CREATE FUNCTION ExtractNumber(in_string VARCHAR(50))
RETURNS INT
NO SQL
BEGIN
    DECLARE ctrNumber VARCHAR(50);
    DECLARE finNumber VARCHAR(50) DEFAULT '';
    DECLARE sChar VARCHAR(1);
    DECLARE inti INTEGER DEFAULT 1;
    IF LENGTH(in_string) > 0
    THEN WHILE(inti <= LENGTH(in_string)) DO
        SET sChar = SUBSTRING(in_string, inti, 1);
        SET ctrNumber = FIND_IN_SET(sChar, '0,1,2,3,4,5,6,7,8,9');
        IF ctrNumber > 0
        THEN SET finNumber = CONCAT(finNumber, sChar);
        END IF;
        SET inti = inti + 1;
    END WHILE;
    RETURN CAST(finNumber AS UNSIGNED);
END;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

ELSE
RETURN 0;
END IF;
END; $$
DELIMITER ;
SELECT ExtractNumber(number) AS "Номери автобусів без регіонального
кодування і з ним", number
FROM Transport INNER JOIN Flights F
ON Transport.transport_ID = F.idTransport INNER JOIN Firms F2
ON F.idFirm = F2.firm_ID INNER JOIN Type_transportations Tt
ON F2.idTransportation = Tt.transportation_ID
WHERE type_name = 'Наземне';

```

	Номери автобусів без регіонального кодування і з ним	number
1	7837	BI7837EA
2	6097	BC6097MK
3	3137	BK3137HH
4	6426	AT6426EP
5	5474	BC5474MK
6	3458	BC3458MK
7	5501	AE5501AA
8	7798	BI7798EX
9	1049	BM1049CM
10	5109	AX5109HX
11	2775	AB2775HT

Рисунок 2.46 – Результат виконання функції

Створено процедуру, яка обчислює та встановлює вартість авіаквитків  
(рисунок 2.47):

```

DELIMITER $$
CREATE PROCEDURE `calc_price_at`(OUT new_price DECIMAL, IN new_FhCpID
INTEGER, IN new_idTt INTEGER)
BEGIN
SET new_price = 4 * (SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC on Flights.flight_ID = FhC.idFlight WHERE
primaryID = new_FhCpID) + ((SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
on Flights.flight_ID = FhC.idFlight WHERE idFlight = new_FhCpID) *
(SELECT percent_afterpay/100 FROM Type_tickets WHERE new_idTt =
Type_tickets.typeticket_ID));
END; $$
CALL calc_price_at(@test,10,7);
SELECT ROUND(@test,2) AS "Ціна", f_distance AS "Відстань",
CONCAT(f_from, ' - ', f_to) AS "Рейс"
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight
WHERE idFlight = 10;

```

	Ціна	Відстань	Рейс
1	7127.00	463	Полтава - Миколаїв

Рисунок 2.47 – Результат виконання процедури

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

Створено процедуру, яка обчислює та встановлює вартість квитків на морські рейси (рисунок 2.48):

```
DELIMITER $$
CREATE PROCEDURE `calc_price_st`(OUT new_price DECIMAL, IN new_FhCpID
INTEGER, IN new_idTt INTEGER)
BEGIN
    SET new_price = 5 * (SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
on Flights.flight_ID = FhC.idFlight WHERE primaryID =
new_FhCpID) +
((SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
on Flights.flight_ID = FhC.idFlight WHERE idFlight =
new_FhCpID) *
(SELECT percent_afterpay/100 FROM Type_tickets
WHERE new_idTt = Type_tickets.typeticket_ID));
END; $$
DELIMITER ;
CALL calc_price_st(@test,32,12);
SELECT ROUND(@test,2) AS "Ціна", f_distance AS "Відстань",
CONCAT(f_from, ' - ', f_to) AS "Рейс"
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight
WHERE idFlight = 32;
```

	Ціна	Відстань	Рейс
1	8930.00	2845	Одеса - Андорра-ла-Велья

Рисунок 2.48 – Результат виконання процедури

Створено процедуру, яка обчислює та встановлює вартість залізничних квитків (рисунок 2.49):

```
DELIMITER $$
CREATE PROCEDURE `calc_price_tt`(INOUT new_price DECIMAL, IN new_idTt
INTEGER)
BEGIN
    SET new_price = new_price/2 + (new_price/2)*(SELECT
percent_afterpay/100
FROM Type_tickets WHERE typeticket_ID = new_idTt);
END; $$
DELIMITER ;
SET @test := 394;
CALL calc_price_tt(@test,1);
SELECT ROUND(@test,2) AS "Ціна", f_distance AS "Відстань",
CONCAT('Одеса', ' - ', f_to) AS "Рейс"
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight
WHERE idFlight = 31;
```

	Ціна	Відстань	Рейс
1	296.00	466	Одеса - Івано-Франківськ

Рисунок 2.49 – Результат виконання процедури



Створено процедуру, яка виконує пошук маршрутів за точкою відправлення і точкою прибуття (рисунки 2.50):

```
DELIMITER $$
CREATE PROCEDURE `find_routes`(IN from_city VARCHAR(45), IN to_city
VARCHAR(45))
BEGIN
    DECLARE flight_from VARCHAR(45);
    DECLARE flight_to VARCHAR(45);
    DECLARE route_from VARCHAR(45);
    DECLARE route_to VARCHAR(45);
    DECLARE flight_time DATETIME;
    DECLARE myCondition BOOLEAN;
    DECLARE success BOOLEAN DEFAULT FALSE;
    DECLARE myCursor CURSOR FOR SELECT f_from, f_to, f_from, city_name,
starttime
FROM Flights INNER JOIN
Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight
INNER JOIN Routes R
ON FhC.primaryID =
R.Flights_has_Cities_primaryID INNER JOIN Cities C
ON C.city_ID = R.idCity INNER JOIN Firms
F
ON Flights.idFirm = F.firm_ID INNER JOIN
Type_transportations Tt
ON F.idTransportation =
Tt.transportation_ID;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET myCondition = FALSE;
    OPEN myCursor;
    SET myCondition = TRUE;
    WHILE myCondition DO
        FETCH myCursor INTO flight_from, flight_to,
route_from, route_to, flight_time;
        IF(route_from = from_city AND route_to = to_city) THEN
            SELECT flight_time AS "Час відправлення",
CONCAT(flight_from, ' - ', flight_to) AS "Рейс",
CONCAT(route_from, ' - ', route_to) AS
"Маршрут";
            SET success = TRUE;
        END IF;
    END WHILE;
    CLOSE myCursor;
    IF NOT success
        THEN SELECT 'Маршрутів не знайдено!' AS "Повідомлення";
    END IF;
END; $$
DELIMITER ;
CALL find_routes('Львів', 'Київ');
```

	Час відправлення	Рейс	Маршрут
1	2021-09-05 01:47:01	Львів - Харків	Львів - Київ

Рисунок 2.50 – Результат виконання процедури

Створено процедуру, яка виконує пошук рейсів за датою відправлення (рисунки 2.51, рисунок 2.52):

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

DELIMITER $$
CREATE PROCEDURE `find_flight`(IN flight_datetime DATETIME)
BEGIN
    DECLARE flight_from VARCHAR(45);
    DECLARE flight_to VARCHAR(45);
    DECLARE flight_distance FLOAT;
    DECLARE flight_startTime DATETIME;
    DECLARE flight_endTime DATETIME;
    DECLARE myCondition BOOLEAN;
    DECLARE success BOOLEAN DEFAULT FALSE;
    DECLARE myCursor CURSOR FOR SELECT f_from, f_to, f_distance,
starttime, endtime FROM Flights;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET myCondition = FALSE;
    OPEN myCursor;
    SET myCondition = TRUE;
    WHILE myCondition DO
        FETCH myCursor INTO flight_from, flight_to,
flight_distance, flight_startTime, flight_endTime;
        IF(flight_startTime = flight_datetime) THEN
            SELECT 'Точний збір' AS ' ', flight_from, flight_to,
flight_distance, flight_startTime, flight_endTime;
            SET success = TRUE;
        ELSEIF(DATE(flight_startTime) = DATE(flight_datetime)) THEN
            SELECT 'Схожий рейс' AS 'Точних зборів не знайдено!',
flight_from, flight_to, flight_distance, flight_startTime,
flight_endTime;
            SET success = TRUE;
        END IF;
    END WHILE;
    CLOSE myCursor;
    IF NOT success
        THEN SELECT 'Рейсів не знайдено!' AS "Повідомлення";
    END IF;
END; $$
DELIMITER ;
CALL find_flight('2021-08-09 20:18:56');
CALL find_flight('2021-08-09 00:00:00');

```

<anonymous>	flight_from	flight_to	flight_distance	flight_startTime	flight_endTime
1 Точний збір	Київ	Луганськ	922	2021-08-09 20:18:56	2021-08-10 15:02:56

Рисунок 2.51 – Рейс з ідентичною датою та часом відправлення

Точних зборів не знайдено!	flight_from	flight_to	flight_distance	flight_startTime	flight_endTime
1 Схожий рейс	Київ	Луганськ	922	2021-08-09 20:18:56	2021-08-10 15:02:56

Рисунок 2.52 – Рейс з ідентичною датою відправлення

Створено процедуру, яка обчислює дату та час прибуття рейсу (рисунок 2.53):

```

DELIMITER $$
CREATE PROCEDURE `calc_end_time`(OUT flight_endTime DATETIME, IN
flight_distance FLOAT, IN flight_startTime DATETIME, IN avg_speed FLOAT)
BEGIN SET flight_endTime = ADDTIME(flight_startTime,
getHour(flight_distance/avg_speed));
END; $$
DELIMITER ;
SELECT 50 AS avg_speed, 488 AS distance, '2021-10-24 21:22:46' AS
startTime, @test AS endTime;SELECT @test;

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

	avg_speed	distance	startTime	endTime
1	50	488	2021-10-24 21:22:46	2021-10-25 07:38:46

Рисунок 2.53 – Результат виконання процедури

## 2.6 Додавання користувачів та надання їм прав

СКБД MySQL є багатокористувацьким середовищем, тому для доступу до таблиць БД можуть бути створені різні облікові записи з різним рівнем привілеїв. Обліковому запису редактора можна надати привілеї на перегляд таблиці, додавання нових записів і оновлення вже існуючих. Адміністратору БД можна надати більш широкі повноваження (можливість створення таблиць, редагування та видалення вже існуючих). Для користувача БД достатньо лише перегляду таблиць. [6]

У результаті аналізу завдання на курсове проєктування було створено наступних користувачів.

Створено користувача, який має всі права для бази Tickets\_booking (рисунок 2.54):

```
CREATE USER 'Developer'@'localhost:3306' IDENTIFIED BY 'helloWorld';
GRANT ALL PRIVILEGES ON Tickets_booking.* TO
'Developer'@'localhost:3306';
FLUSH PRIVILEGES;
```

	Grants for Developer@localhost:3306
1	GRANT USAGE ON *.* TO `Developer`@`localhost:3306`
2	GRANT ALL PRIVILEGES ON `tickets_booking`.* TO `Developer`@`localhost:3306`

Рисунок 2.54 – Результат створення користувача

Створено користувача, який має доступ до редагування таблиць та створення представлень для бази Tickets\_booking (рисунок 2.55):

```
CREATE USER 'Admin'@'localhost:3306' IDENTIFIED BY 'myAdmin';
GRANT SELECT, INSERT, DELETE, UPDATE, CREATE VIEW, EXECUTE ON
Tickets_booking.* TO 'Admin'@'localhost:3306';
FLUSH PRIVILEGES;
```

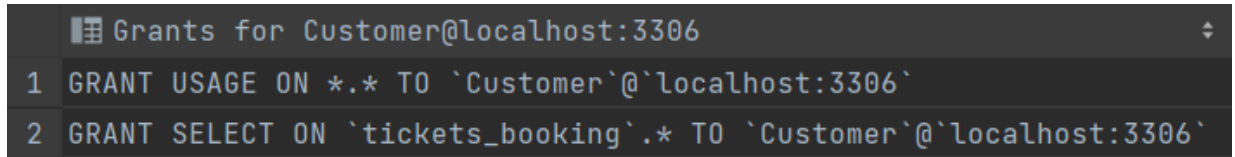
	Grants for Admin@localhost:3306
1	GRANT USAGE ON *.* TO `Admin`@`localhost:3306`
2	GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE, CREATE VIEW ON `tickets_booking`.* TO `Admin`@`localhost:3306`

Рисунок 2.55 – Результат створення користувача

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

Створено користувача, який має доступ до вибірки даних з всіх таблиць бази Tickets\_booking (рисунок 2.56):

```
CREATE USER 'Customer'@'localhost:3306' IDENTIFIED BY 'qwerty123';  
GRANT SELECT ON Tickets_booking.* TO 'Customer'@'localhost:3306';  
FLUSH PRIVILEGES;
```



```
mysql> GRANTS for Customer@localhost:3306  
1 GRANT USAGE ON *.* TO `Customer`@`localhost:3306`  
2 GRANT SELECT ON `tickets_booking`.* TO `Customer`@`localhost:3306`
```

Рисунок 2.56 – Результат створення користувача

Повний код створеної бази даних у СУБД MySQL наведено у додатку В.

### 3 РОЗРОБЛЕННЯ БАЗИ ДАНИХ ЗАСОБАМИ СКБД POSTGRESQL

PostgreSQL – це об'єктно-реляційна система управління базами даних (ОРСУБД, ORDBMS), заснована на POSTGRES, Version 4.2 – програмою, розробленою на факультеті комп'ютерних наук Каліфорнійського університету в Берклі. У POSTGRES з'явилося безліч нововведень, які були реалізовані в деяких комерційних СУБД набагато пізніше. [7]

#### 3.1 Створення таблиць

Вся інформація таблиць PostgreSQL знаходиться в оперативній пам'яті. Не можливо створити таблицю, яка буде не в пам'яті. Записи таблиці упорядковано відповідно до індексу, це забезпечує можливість швидкої вибірки даних.

В цілому PostgreSQL працює швидше ніж MySQL, за виключенням використання первинних ключів. [7]

У відповідності до розробленої схеми даних було створено SQL таблиці бази даних.

Першим кроком створено типи перерахування:

```
CREATE TYPE FLIGHTSTATUS AS ENUM('Скасований', 'Відбувся',  
'Очікується');  
CREATE TYPE TYPETICKET AS ENUM('Купе дорослий', 'Купе дитячий', 'Купе  
студентський', 'Плацкарт дорослий', 'Плацкарт дитячий', 'Плацкарт  
студентський', 'Бізнес клас', 'Економ клас', 'Перший клас',  
'Автобусний', 'Автобусний студентський', 'Морський', 'Морський VIP',  
'Морський економ');
```

Створено таблицю "Type\_transportations". Структура таблиці наведена на рисунку 3.1.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Type_tickets(  
    typeticket_ID SERIAL NOT NULL,  
    type_afterpay TYPETICKET NOT NULL,  
    percent_afterpay INT NOT NULL,  
    PRIMARY KEY (typeticket_ID));
```

Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Type_transportations(type_name)  
VALUES ('Повітряне'), ('Колійне'), ('Водне'), ('Наземне');  
COMMIT ;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

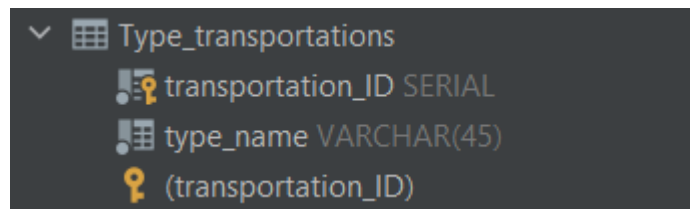


Рисунок 3.1 – Структура таблиці "Type\_transportations"

Створено таблицю " Firms". Структура таблиці наведена на рисунку 3.2.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Firms (
    firm_ID SERIAL NOT NULL,
    idTransportation INT NOT NULL,
    name VARCHAR(45) NOT NULL,
    email VARCHAR(45) NOT NULL,
    tel VARCHAR(20) NOT NULL,
    region_activity VARCHAR(20) NULL,
    PRIMARY KEY (firm_ID),
    FOREIGN KEY (idTransportation) REFERENCES Type_transportations
    (transportation_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Firms(idTransportation, name, email, tel, regin_activity)
VALUES (1,'MAV','uia@flyuia.com','+38 (044) 581-50-50','Україна'),
    (1,'KLM','KLM.Ukraine@klm.nl','+31 (0) 20 649 91
23','Нідерланди'),
    (1,'Emirates Airlines','pr@emirates.com','600 555 555','OAE'),
    ...
    (4,'Ecolines','help@ecolines.ua','+38 044 594 90 10',''),
    (4,'EAST WEST EUROLINES','support@ewe.ua','+380988154444',''),
    (4,'TransTempo','transtempo@ukr.net','+38 (067) 467-44-77','');
COMMIT ;
```

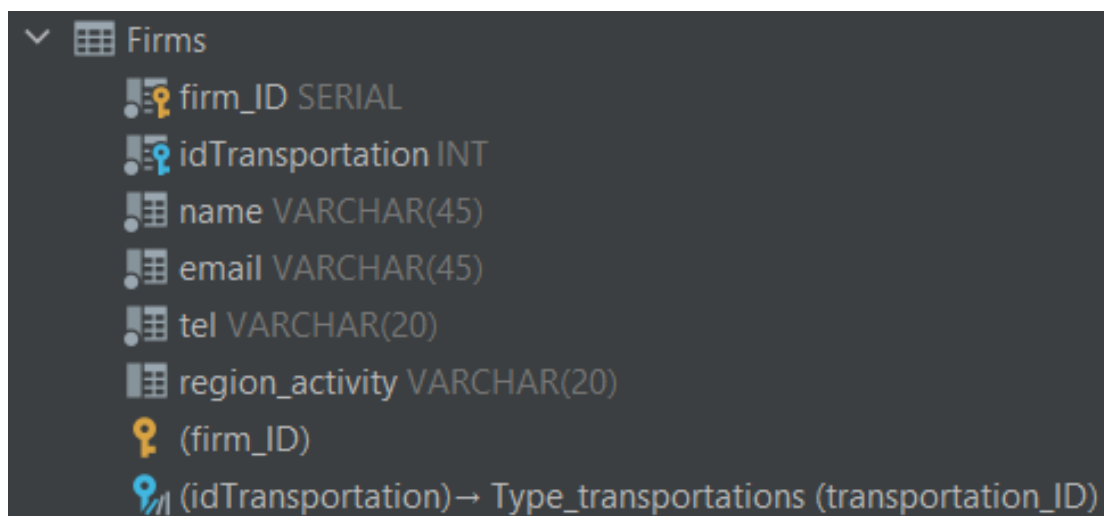


Рисунок 3.2 – Структура таблиці "Firms"

Створено таблицю "Flights". Структура таблиці наведена на рисунку 3.3.

#### Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Flights (  
    flight_ID SERIAL NOT NULL,  
    idFirm INT NOT NULL,  
    idTransport INT NOT NULL,  
    f_from VARCHAR(45) NULL,  
    f_to VARCHAR(45) NULL,  
    f_distance FLOAT NULL,  
    starttime TIMESTAMP NOT NULL,  
    endtime TIMESTAMP NULL,  
    status FLIGHTSTATUS DEFAULT 'Очікується',  
    PRIMARY KEY (flight_ID),  
    FOREIGN KEY (idFirm) REFERENCES Firms(firm_ID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    FOREIGN KEY (idTransport) REFERENCES Transport(transport_ID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE);
```

#### Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport,  
starttime)  
VALUES (8, 'Київ', 'Дніпро', 488, 51, '2021-10-24 21:22:46'),  
        (10, 'Херсон', 'Миколаїв', 80, 57, '2021-05-08 14:01:34'),  
        (11, 'Київ', 'Луганськ', 922, 6, '2021-08-09 20:18:56'),  
        ...  
        (10, 'Кропивницький', 'Миколаїв', 182, 53, '2021-03-10  
02:42:48'),  
        (3, 'Хмельницький', 'Берн', 1444, 4, '2021-03-19 11:52:21'),  
        (1, 'Київ', 'Осло', 1630, 3, '2021-01-02 18:36:57');  
COMMIT ;
```

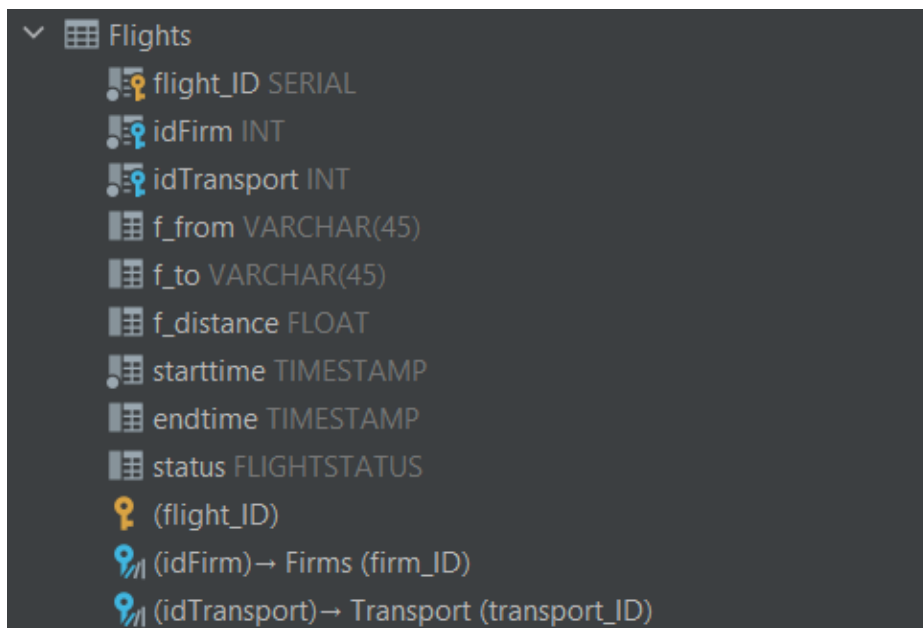


Рисунок 3.3 – Структура таблиці "Flights"

3.4. Створено таблицю "Customers". Структура таблиці наведена на рисунку

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Customers (
  customer_ID SERIAL NOT NULL,
  c_name VARCHAR(45) NOT NULL,
  c_birthday DATE NOT NULL,
  PRIMARY KEY (customer_ID));
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Customers(c_name, c_birthday)
VALUES ('Тарасович Ратимир Вадимович', '1952-08-10'),
       ('Ніколенко Ольга Вікторівна', '1963-05-17'),
       ('Кириленко Муховіст Полянович', '1987-03-19'),
       ...
       ('Мухопад Царук Адріанович', '1982-04-20'),
       ('Бурбан Верніслав Артемович', '1993-03-05'),
       ('Штинь Атрей Русланович', '1956-11-19'),
       ('Деркач Лютобор Романович', '1966-07-25');
COMMIT ;
```

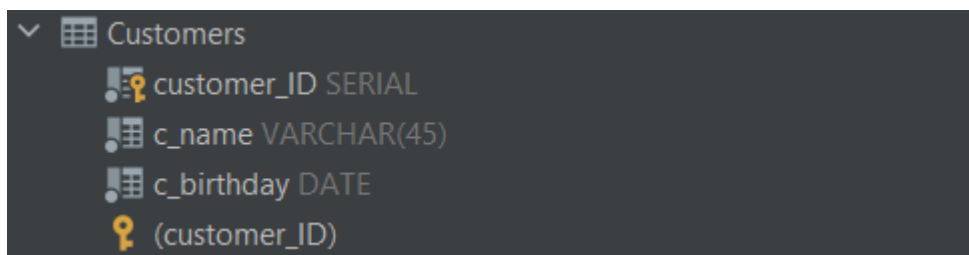


Рисунок 3.4 – Структура таблиці "Customers"

Створено таблицю "Type\_tickets". Структура таблиці наведена на рисунку 3.5.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Type_tickets(
  typeticket_ID SERIAL NOT NULL,
  type_afterpay TYPETICKET NOT NULL,
  percent_afterpay INT NOT NULL,
  PRIMARY KEY (typeticket_ID));
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Type_tickets (type_afterpay, percent_afterpay)
VALUES ( 'Купе дорослий', 50),
       ( 'Купе студентський', 30),
       ( 'Купе дитячий', 10),
       ( 'Плацкарт дорослий', 0),
       ( 'Плацкарт дитячий', -30),
       ( 'Плацкарт студентський', -50),
       ( 'Бізнес клас', 150),
       ( 'Економ клас', -10),
       ( 'Перший клас', 50),
       ( 'Автобусний', 0),
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		



```

( 'Автобусний студентський', -20),
( 'Морський', 0),
( 'Морський VIP', 75),
( 'Морський економ', -15);
COMMIT ;

```

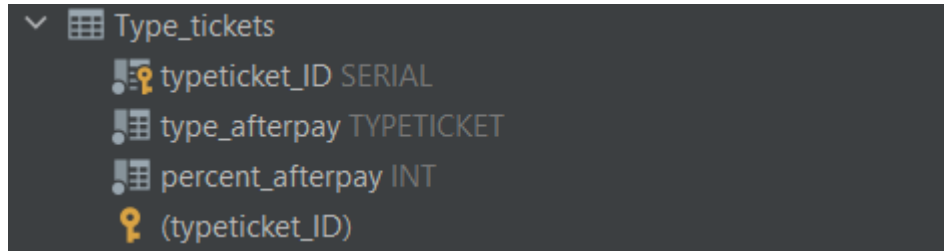


Рисунок 3.5 – Структура таблиці "Type\_tickets"

Створено таблицю "Cities". Структура таблиці наведена на рисунку 3.6.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Cities (
  city_ID SERIAL NOT NULL,
  city_name VARCHAR(45) NOT NULL,
  country VARCHAR(45) NOT NULL,
  PRIMARY KEY (city_ID));

```

Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Cities (country, city_name)
VALUES ('Молдова', 'Кишинів'),
       ('Греція', 'Афіни'),
       ('Хорватія', 'Загреб'),
       ...
       ('Україна', 'Хмельницький'),
       ('Україна', 'Черкаси'),
       ('Україна', 'Чернівці'),
       ('Україна', 'Чернігів');
COMMIT ;

```

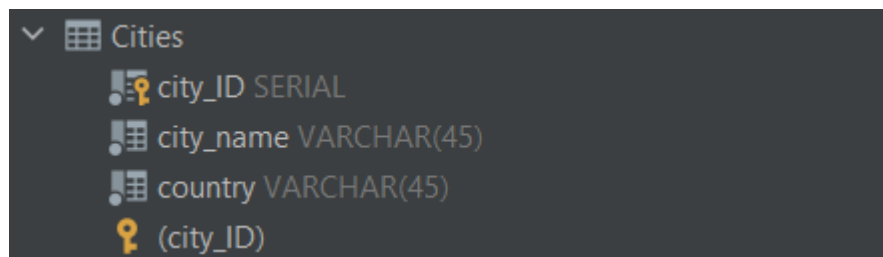


Рисунок 3.6 – Структура таблиці "Cities"

Створено таблицю "Flights\_has\_Cities". Структура таблиці наведена на рисунку 3.7.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Flights_has_Cities (
  primaryID SERIAL NOT NULL,
  idFlight INT NOT NULL,

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

idCity1 INT NOT NULL,
idCity2 INT NULL,
PRIMARY KEY (primaryID),
FOREIGN KEY (idFlight) REFERENCES Flights (flight_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (idCity1) REFERENCES Cities (city_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (idCity2) REFERENCES Cities (city_ID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Flights_has_Cities(idFlight, idCity1, idCity2)
(SELECT flight_ID, city_ID, (SELECT city_ID FROM Cities WHERE f_to =
city_name) FROM Flights, Cities
WHERE f_from = city_name);
COMMIT ;

```



Рисунок 3.7 – Структура таблиці "Flights\_has\_Cities"

Створено таблицю "Airplane\_tickets". Структура таблиці наведена на рисунку 3.8.

#### Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Airplane_tickets (
a_ticket_ID SERIAL NOT NULL,
at_price DECIMAL NULL,
idType_tickets INT NOT NULL,
Flights_has_Cities_primaryID INT NOT NULL,
PRIMARY KEY (a_ticket_ID),
FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES
Flights_has_Cities (primaryID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

BEGIN ;

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

INSERT INTO Airplane_tickets(idType_tickets,
Flights_has_Cities_primaryID)
VALUES (7, 10),
      (9, 50),
      (9, 13),
      (8, 25),
      ...
      (8, 37),
      (9, 7),
      (8, 6);
COMMIT ;

```

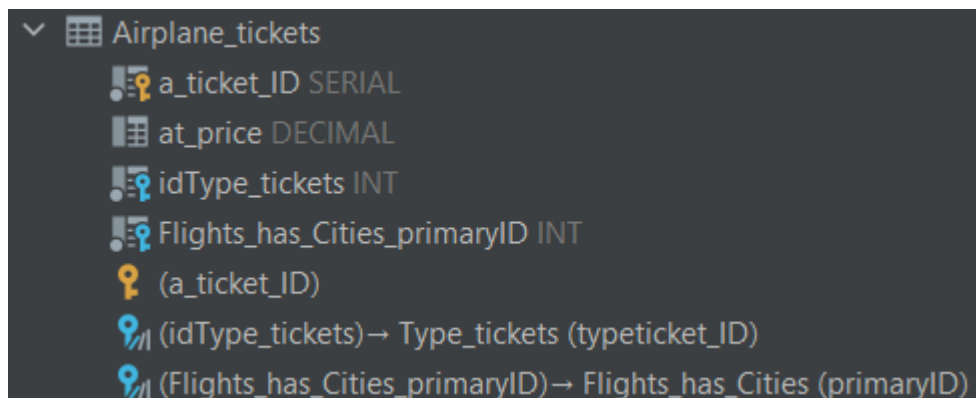


Рисунок 3.8 – Структура таблиці "Airplane\_tickets"

Створено таблицю "Routes". Структура таблиці наведена на рисунку 3.9.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Routes (
  route_ID SERIAL NOT NULL,
  Flights_has_Cities_primaryID INT NOT NULL,
  idCity INT NOT NULL,
  PRIMARY KEY (route_ID),
  FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES
Flights_has_Cities (primaryID)
ON DELETE CASCADE
ON UPDATE CASCADE,
  FOREIGN KEY (idCity) REFERENCES Cities (city_ID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Routes(Flights_has_Cities_primaryID, idCity)
VALUES ( 8, 60),
      ( 8, 63),
      ( 8, 50),
      ( 29, 43),
      ...
      ( 47, 64),
      ( 47, 55),
      ( 9, 50),
      ( 9, 60);
COMMIT ;

```

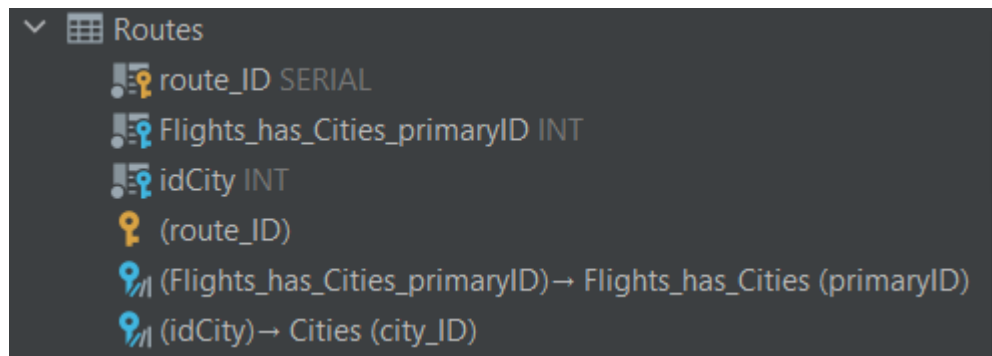


Рисунок 3.9 – Структура таблиці "Routes"

Створено таблицю "Train\_tickets". Структура таблиці наведена на рисунку 3.10.

#### Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Train_tickets (
  t_ticket_ID SERIAL NOT NULL,
  tt_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  idRoute INT NOT NULL,
  PRIMARY KEY (t_ticket_ID),
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idRoute) REFERENCES Routes (route_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);
```

#### Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Train_tickets( tt_price, idType_tickets, idRoute)
VALUES ( 394, 1, 31),
       ( 554, 1, 32),
       ( 221, 2, 33),
       ...
       ( 185, 1, 53),
       ( 394, 4, 54),
       ( 472, 5, 55);

COMMIT ;
```

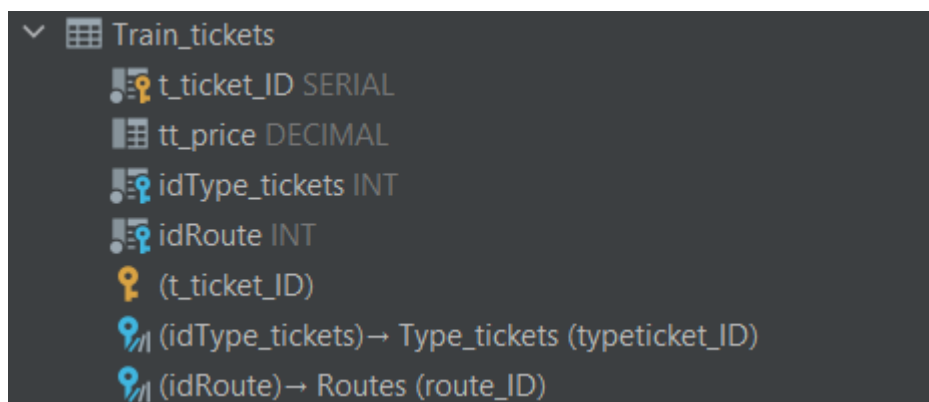


Рисунок 3.10 – Структура таблиці "Train\_tickets"

Створено таблицю "Bus\_tickets". Структура таблиці наведена на рисунку

3.11.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Bus_tickets (
  b_ticket_ID SERIAL NOT NULL,
  bt_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  idRoute INT NOT NULL,
  PRIMARY KEY (b_ticket_ID),
  FOREIGN KEY (idRoute) REFERENCES Routes (route_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Bus_tickets(bt_price, idType_tickets, idRoute)
VALUES ( 318, 10, 1),
      ( 212, 10, 2),
      ( 448, 10, 3),
      ...
      ( 638, 11, 8),
      ( 953, 10, 9),
      ( 3754, 10, 10);
COMMIT ;
```

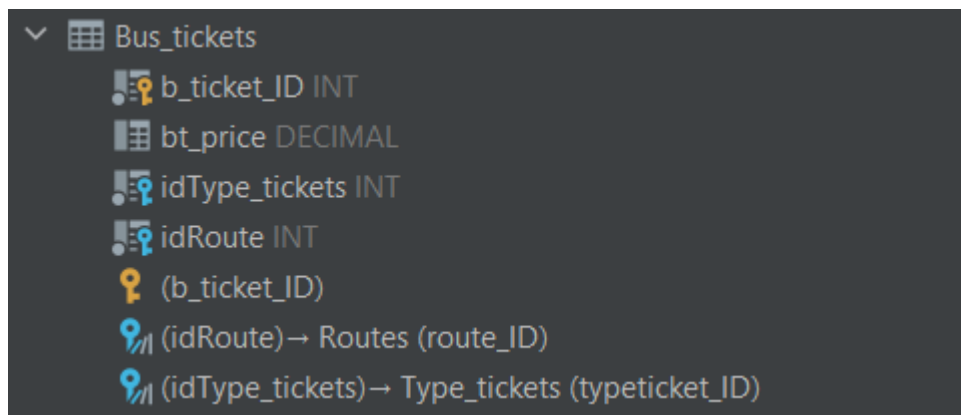


Рисунок 3.11 – Структура таблиці "Bus\_tickets"

Створено таблицю "Ship\_tickets". Структура таблиці наведена на рисунку 3.12.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Ship_tickets (
  s_ticket_ID SERIAL NOT NULL,
  st_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  Flights_has_Cities_primaryID INT NOT NULL,
  PRIMARY KEY (s_ticket_ID),
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
```

```

ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES
Flights_has_Cities (primaryID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Ship_tickets(idType_tickets, Flights_has_Cities_primaryID)
VALUES ( 12, 32),
      ( 13, 32),
      ( 14, 32),
      ...
      ( 13, 32),
      ( 13, 32),
      ( 13, 32);
COMMIT ;

```

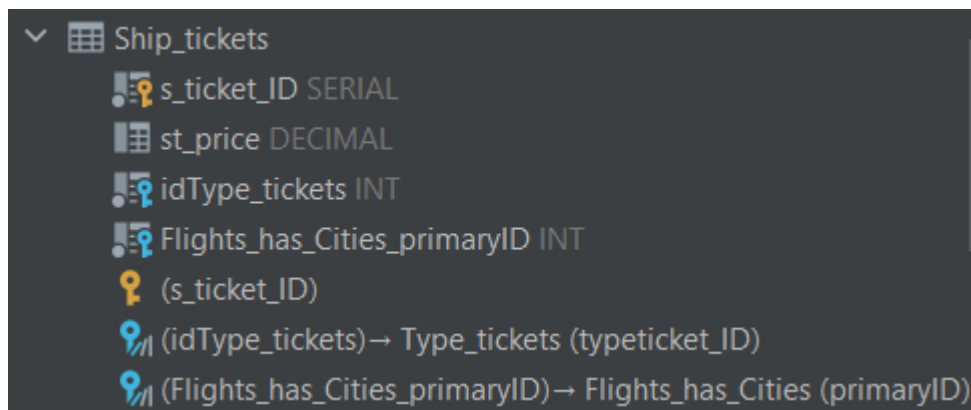


Рисунок 3.12 – Структура таблиці "Ship\_tickets"

Створено таблицю "Train\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 3.13.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Train_tickets_has_Customers (
  Train_tickets_t_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  t_booking_time TIMESTAMP NOT NULL,
  t_numwagon INT NOT NULL,
  t_seat INT NOT NULL,
  comments VARCHAR(45) NULL,
  PRIMARY KEY (Train_tickets_t_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Train_tickets_t_ticket_ID) REFERENCES Train_tickets
(t_ticket_ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
ON DELETE CASCADE
ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

BEGIN ;

```

```

INSERT INTO Train_tickets_has_Customers(train_tickets_t_ticket_id,
customers_customer_id, t_booking_time, t_numwagon, t_seat)
VALUES ( 1, 2, '2020-03-15 09:59:44', 1, 21),
      ( 2, 4, '2020-09-01 05:27:15', 6, 13),
      ( 3, 6, '2020-11-11 16:03:38', 4, 17),
      ...
      ( 23, 46, '2020-05-06 04:59:54', 7, 42),
      ( 24, 48, '2020-12-04 22:03:56', 1, 45),
      ( 25, 50, '2020-05-01 14:58:58', 2, 21);
COMMIT ;

```

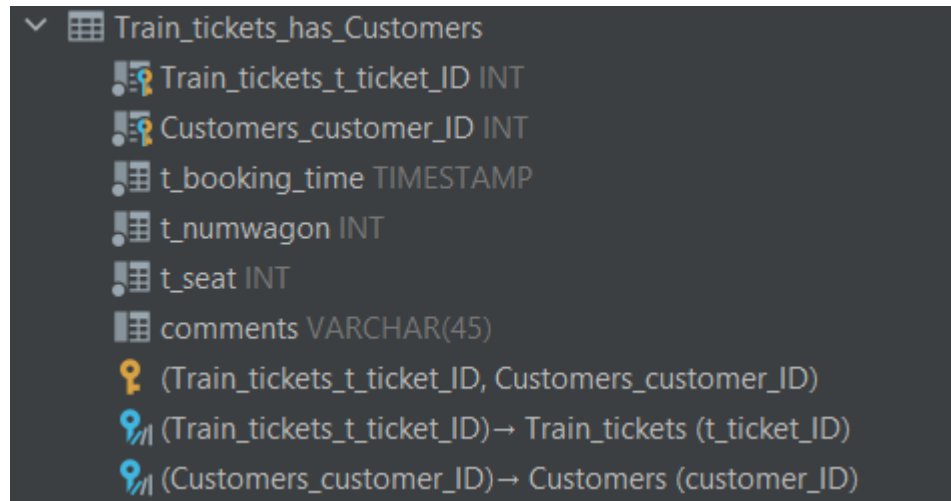


Рисунок 3.13 – Структура таблиці "Train\_tickets\_has\_Customers"

Створено таблицю "Airplane\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 3.14.

Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Airplane_tickets_has_Customers (
  Airplane_tickets_a_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  a_booking_time TIMESTAMP NOT NULL,
  at_class VARCHAR(15) NULL,
  a_seat INT NOT NULL,
  comments VARCHAR(45) NULL,
  PRIMARY KEY (Airplane_tickets_a_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Airplane_tickets_a_ticket_ID) REFERENCES
Airplane_tickets (a_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

```

Код заповнення таблиці:

```

COMMIT ;
INSERT INTO Airplane_tickets_has_Customers
  (Airplane_tickets_a_ticket_ID, Customers_customer_ID,
a_booking_time, a_seat, comments)
VALUES ( 1, 1, '2020-10-09 20:21:36', 1, NULL),
      ( 2, 2, '2020-01-14 09:55:01', 2, NULL),
      ( 3, 3, '2020-06-23 14:27:41', 3, NULL),

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

...
( 23, 23, '2020-04-16 16:54:07', 23, NULL),
( 24, 24, '2020-06-25 04:22:31', 24, NULL),
( 25, 25, '2020-10-28 08:31:29', 25, NULL);
COMMIT ;

```

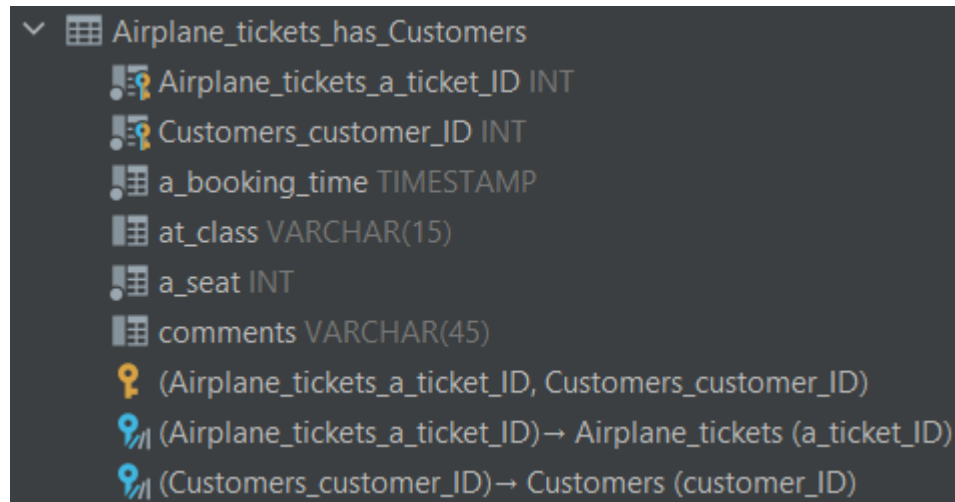


Рисунок 3.14 – Структура таблиці "Airplane\_tickets\_has\_Customers"

Створено таблицю "Bus\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 3.15.

#### Код створення таблиці:

```

CREATE TABLE IF NOT EXISTS Bus_tickets_has_Customers (
    Bus_tickets_b_ticket_ID INT NOT NULL,
    Customers_customer_ID INT NOT NULL,
    b_booking_time TIMESTAMP NOT NULL,
    b_seat INT NOT NULL,
    comments VARCHAR(45) NULL,
    PRIMARY KEY (Bus_tickets_b_ticket_ID, Customers_customer_ID),
    FOREIGN KEY (Bus_tickets_b_ticket_ID) REFERENCES Bus_tickets
(b_ticket_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);

```

#### Код заповнення таблиці:

```

BEGIN ;
INSERT INTO Bus_tickets_has_Customers(bus_tickets_b_ticket_id,
customers_customer_id, b_booking_time, b_seat)
VALUES ( 1, 26, '2020-05-08 02:11:23', 1),
( 2, 27, '2020-10-06 21:24:28', 4),
( 3, 28, '2020-05-14 23:59:25', 7),
...
( 33, 29, '2020-08-12 08:28:02', 19),
( 34, 26, '2020-10-04 00:22:49', 1),
( 35, 29, '2020-01-25 05:00:16', 4);
COMMIT ;

```



Bus_tickets_has_Customers
Bus_tickets_b_ticket_ID INT
Customers_customer_ID INT
b_booking_time TIMESTAMP
b_seat INT
comments VARCHAR(45)
(Bus_tickets_b_ticket_ID, Customers_customer_ID)
(Bus_tickets_b_ticket_ID) → Bus_tickets (b_ticket_ID)
(Customers_customer_ID) → Customers (customer_ID)

Рисунок 3.15 – Структура таблиці "Bus\_tickets\_has\_Customers"

Створено таблицю "Ship\_tickets\_has\_Customers". Структура таблиці наведена на рисунку 3.16.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Ship_tickets_has_Customers (
  Ship_tickets_s_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  s_booking_time TIMESTAMP NOT NULL,
  s_class VARCHAR(15) NULL,
  s_seat INT NOT NULL,
  comments VARCHAR(45) NULL,
  PRIMARY KEY (Ship_tickets_s_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Ship_tickets_s_ticket_ID) REFERENCES Ship_tickets
(s_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers
(customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Ship_tickets_has_Customers(ship_tickets_s_ticket_id,
customers_customer_id, s_booking_time, s_seat, comments)
VALUES ( 1, 26, '2020-03-24 05:43:44', 23, NULL),
( 2, 27, '2020-02-25 13:56:49', 97, NULL),
( 3, 28, '2020-09-11 10:54:59', 45, NULL),
( 4, 29, '2020-11-19 02:10:09', 12, NULL),
( 5, 30, '2020-02-14 01:36:50', 88, NULL);
COMMIT ;
```

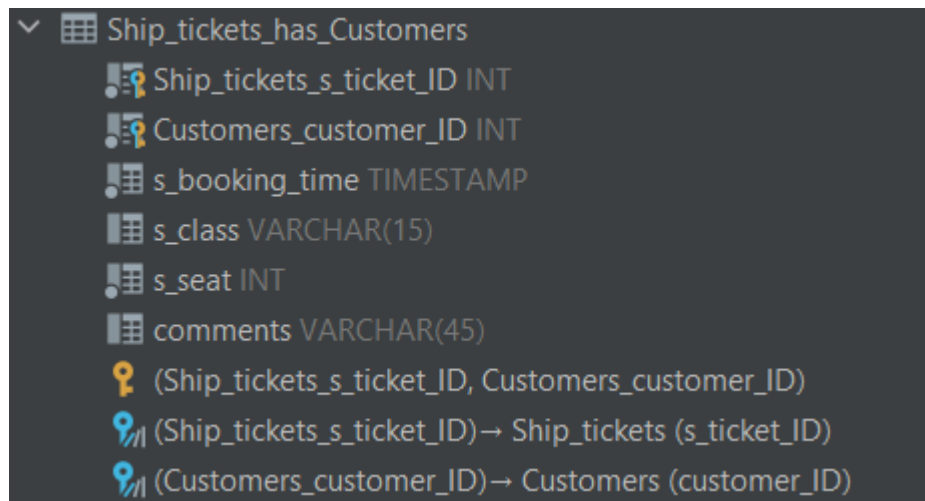


Рисунок 3.16 – Структура таблиці "Ship\_tickets\_has\_Customers"

Створено таблицю "Transport\_has\_Type\_tickets". Структура таблиці наведена на рисунку 3.17.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Transport_has_Type_tickets (
    Transport_idTransport INT NOT NULL,
    Type_ticketa_idType_tickets INT NOT NULL,
    count_seats_type INT NOT NULL,
    PRIMARY KEY (Transport_idTransport, Type_ticketa_idType_tickets),
    FOREIGN KEY (Transport_idTransport) REFERENCES Transport
    (transport_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (Type_ticketa_idType_tickets) REFERENCES Type_tickets
    (typeticket_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Код заповнення таблиці:

```
BEGIN ;
INSERT INTO Transport_has_Type_tickets(transport_idtransport,
type_ticketa_idtype_tickets, count_seats_type)
VALUES (1, 7, 50),
       (1, 8, 75),
       ...
       (66,13, 50),
       (66,14, 50);
COMMIT ;
```

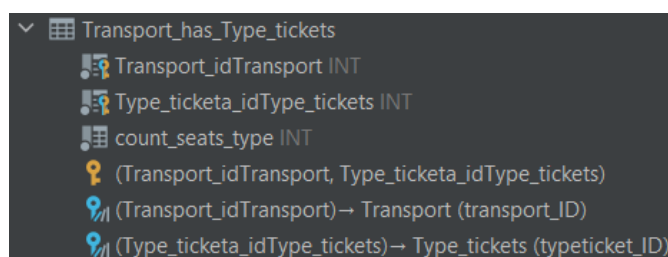


Рисунок 3.17 – Структура таблиці "Transport\_has\_Type\_tickets"

Створено таблицю "Transport". Структура таблиці наведена на рисунку 3.18.

Код створення таблиці:

```
CREATE TABLE IF NOT EXISTS Transport (  
    transport_ID SERIAL NOT NULL,  
    name VARCHAR(45) NOT NULL,  
    number VARCHAR(10) NOT NULL,  
    countplace INT NOT NULL,  
    PRIMARY KEY (transport_ID));
```

Код заповнення таблиці:

```
BEGIN ;  
INSERT INTO Transport(name, number, countplace)  
VALUES ( 'Boeing', '777', 250),  
       ( 'Airbus', 'A340', 170),  
       ( 'Airbus', 'A330', 125),  
       ...  
       ( 'Celebrity Xpedition', '', 5530),  
       ( 'Forse le Carib', '', 570),  
       ( 'Porco Fugo', '', 240);  
COMMIT ;
```

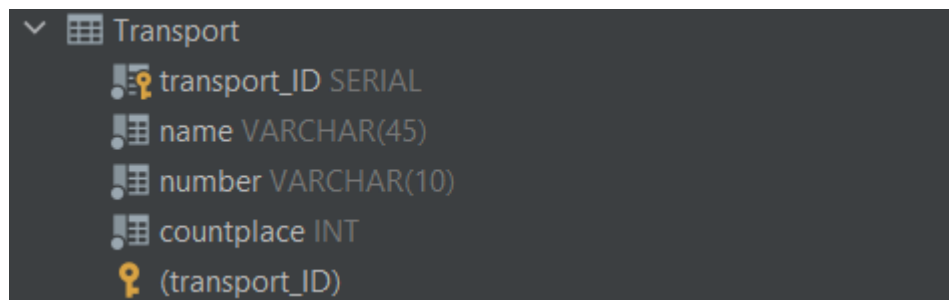


Рисунок 3.18 – Структура таблиці "Transport"

Вигляд таблиць подано у додатку Б.

### 3.2 Створення тригерів

CREATE TRIGGER створює новий тригер. Тригер буде пов'язаний з вказаною таблицею, представленням або сторонньою таблицею і буде виконувати задану функцію при певних подіях. [7]

У результаті аналізу завдання на курсове проектування було створено наступні тригери.

Створено тригер, який записує видалені дані з таблиці "Transpot" в таблицю "Del\_transport" (рисунок 3.19):

```
CREATE OR REPLACE FUNCTION insertInDel_Transport() RETURNS trigger AS  
$saveDelTransport$
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

```

BEGIN
    INSERT INTO Del_transport(transport_ID, name, number,
countplace, time)
    VALUES(OLD.transport_ID, OLD.name, OLD.number, OLD.countplace,
NOW());
    RETURN NULL;
END;
$saveDelTransport$ LANGUAGE plpgsql;

CREATE TRIGGER saveDelTransport BEFORE DELETE ON Transport
    FOR EACH ROW EXECUTE PROCEDURE insertInDel_Transport();

```

### Код створення допоміжної таблиці:

```

CREATE TABLE Del_transport (LIKE Transport INCLUDING ALL);
ALTER TABLE Del_transport ADD COLUMN time TIMESTAMP NULL;

```

### Виконано тестове видалення для перевірки результату:

```
DELETE FROM Transport WHERE transport_ID < 5;
```

	transport_ID	name	number	countplace	time
1	1	Boeing	777	250	2021-05-18 19:53:06
2	2	Airbus	A340	170	2021-05-18 19:53:06
3	3	Airbus	A330	125	2021-05-18 19:53:06
4	4	Boeing	747	245	2021-05-18 19:53:06

Рисунок 3.19 – Результат виконання тригера

Створено тригер, який обчислює приблизний час прибуття рейсу (рисунок 3.20):

```

CREATE OR REPLACE FUNCTION calc_end_time() RETURNS trigger AS
$setEndTimeFlight$
BEGIN
    CASE
        WHEN NEW.idFirm <= 7
            THEN NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/800.0) * 60.0), 'MI:SS')::INTERVAL;
        WHEN NEW.idFirm > 7 AND NEW.idFirm <= 11
            THEN NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/50.0) * 60.0), 'MI:SS')::INTERVAL;
        WHEN NEW.idFirm > 11 AND NEW.idFirm <= 15
            THEN NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/22.0) * 60.0), 'MI:SS')::INTERVAL;
        ELSE NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/65.0) * 60.0), 'MI:SS')::INTERVAL;
    END CASE;
    RETURN NEW;
END;
$setEndTimeFlight$ LANGUAGE plpgsql;

CREATE TRIGGER setEndTimeFlight BEFORE INSERT ON Flights
    FOR EACH ROW EXECUTE PROCEDURE calc_end_time();

```

### Введено тестові дані для перевірки результату:

```

INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport,
starttime)
VALUES (8, 'Київ', 'Дніпро', 488, 51, '2021-10-24 21:22:46'),
(10, 'Херсон', 'Миколаїв', 80, 57, '2021-05-08 14:01:34');

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

	flight_ID	f_from	f_to	starttime	endtime
1	1	Київ	Дніпро	2021-10-24 21:22:46	2021-10-25 07:38:46
2	2	Херсон	Миколаїв	2021-05-08 14:01:34	2021-05-08 16:01:34

Рисунок 3.20 – Результат виконання тригера

Створено тригер, який виводить користувачке повідомлення про помилку, якщо введено від’ємне значення відстані або цифри на місці міста відправлення чи прибуття (рисунок 3.21):

```
CREATE OR REPLACE FUNCTION validFlight() RETURNS trigger AS
$checkDistance$
BEGIN
    CASE
        WHEN NEW.f_distance <= 0
            THEN RAISE 'Invalid distance!';
        WHEN NEW.f_from ~ '[0-9]' OR NEW.f_to ~ '[0-9]'
            THEN RAISE 'This string can't include digits!';
        ELSE RETURN NEW;
    END CASE;
END;
$checkDistance$ LANGUAGE plpgsql;
CREATE TRIGGER checkDistance BEFORE INSERT ON Flights
FOR EACH ROW EXECUTE PROCEDURE validFlight();
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport,
starttime)
VALUES (8, 'Київ', 'Дніпро', -488, 51, '2021-10-24 21:22:46');
```

```
tickets_booking> INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport, starttime)
VALUES (8, 'Київ', 'Дніпро', -488, 51, '2021-10-24 21:22:46')
[2021-05-18 20:16:35] [45001][1644] Invalid distance!
[2021-05-18 20:16:35] [HY000][1644] Invalid distance!
```

Рисунок 3.21 – Результат виконання тригера

Створено тригер, який встановлює статус рейсу відносно поточної дати та часу (рисунок 3.22):

```
CREATE OR REPLACE FUNCTION setStatus() RETURNS trigger AS $checkStatus$
BEGIN
    CASE
        WHEN CURRENT_TIMESTAMP < NEW.endtime AND OLD.status =
'Відбувся'
            THEN NEW.status = 'Очікується';
        WHEN NEW.endtime < CURRENT_TIMESTAMP AND OLD.status =
'Очікується'
            THEN NEW.status = 'Відбувся';
        ELSE RETURN NEW;
    END CASE;
    RETURN NEW;
END;
$checkStatus$ LANGUAGE plpgsql;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		64

```
CREATE TRIGGER checkStatus BEFORE UPDATE OR INSERT ON Flights
FOR EACH ROW EXECUTE PROCEDURE setStatus();
```

Введено тестові дані для перевірки результату:

```
UPDATE Flights SET endtime = '2021-01-08 22:00:00' WHERE flight_ID = 6;
```

flight_ID	f_from	f_to	starttime	endtime	status
1	6 Київ	Тирана	2021-01-08 21:21:34	2021-01-08 22:00:00	Відбувся

Рисунок 3.22 – Результат виконання тригера

Створено тригер, який встановлює вартість авіаквитка (рисунок 3.23):

```
CREATE FUNCTION generate_air_ticket_price() RETURNS trigger AS
$setPriceAirTicket$
BEGIN
    NEW.at_price = 4.0 * (SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
on Flights.flight_ID = FhC.idFlight WHERE primaryID =
NEW.Flights_has_Cities_primaryID) +
((SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
on Flights.flight_ID = FhC.idFlight WHERE idFlight =
NEW.Flights_has_Cities_primaryID) *
(SELECT percent_afterpay/100.0 FROM Type_tickets
WHERE NEW.idType_tickets = Type_tickets.typpeticket_ID));
RETURN NEW;
END;
$setPriceAirTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceAirTicket BEFORE INSERT ON Airplane_tickets
FOR EACH ROW EXECUTE PROCEDURE generate_air_ticket_price();
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Airplane_tickets(idType_tickets,
Flights_has_Cities_primaryID)
VALUES (7, 10),
(9, 50),
(9, 13);
```

a_ticket_ID	at_price	idType_tickets	Flights_has_Cities_primaryID
1	7127	7	10
2	2374	9	50
3	4804	9	13

Рисунок 3.23 – Результат виконання тригера

Створено тригер, який визначає в якому класі знаходиться місце пасажира за його квитком (рисунок 3.24):

```
CREATE FUNCTION set_air_class() RETURNS trigger AS $setClassAirHasCus$
BEGIN
CASE
WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 7
THEN NEW.at_class = 'Бізнес';
WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 8
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

        THEN NEW.at_class = 'Економ';
    WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 9
        THEN NEW.at_class = 'Перший';
    END CASE;
    RETURN NEW;
END;
$setClassAirHasCus$ LANGUAGE plpgsql;
CREATE TRIGGER setClassAirHasCus BEFORE INSERT ON
Airplane_tickets_has_Customers
    FOR EACH ROW EXECUTE PROCEDURE set_air_class();

```

**Введено тестові дані для перевірки результату:**

```

INSERT INTO Airplane_tickets_has_Customers(Airplane_tickets_a_ticket_ID,
Customers_customer_ID, a_booking_time, a_seat, comments)
VALUES ( 1, 1, '2020-10-09 20:21:36', 1, NULL),
( 2, 2, '2020-01-14 09:55:01', 2, NULL);

```

	Airplane_...	Customers_cust...	a_booking_time	at_class	a_seat	comments
1	1	1	2020-10-09 20:21:36	Бізнес	1	<null>
2	2	2	2020-01-14 09:55:01	Перший	2	<null>

**Рисунок 3.24 – Результат виконання тригера**

**Створено тригер, який встановлює вартість квитка на водні види транспорту (рисунок 3.25):**

```

CREATE FUNCTION generate_ship_ticket_price() RETURNS trigger AS
$setPriceShipTicket$
BEGIN
    NEW.st_price = 5.0 * (SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
        on Flights.flight_ID = FhC.idFlight WHERE primaryID =
NEW.Flights_has_Cities_primaryID) +
        ((SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
        on Flights.flight_ID = FhC.idFlight WHERE idFlight =
NEW.Flights_has_Cities_primaryID) *
        (SELECT percent_afterpay/100.0 FROM Type_tickets
        WHERE NEW.idType_tickets = Type_tickets.typeticket_ID));
    RETURN NEW;
END;
$setPriceShipTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceShipTicket BEFORE INSERT ON Ship_tickets
    FOR EACH ROW EXECUTE PROCEDURE generate_ship_ticket_price();

```

**Введено тестові дані для перевірки результату:**

```

INSERT INTO Ship_tickets(idType_tickets, Flights_has_Cities_primaryID)
VALUES ( 12, 32),
( 13, 32);

```

	s_ticket_ID	st_price	idType_tickets	Flights_has_Cities_prima...
1	1	8930	12	32
2	2	11064	13	32

**Рисунок 3.25 – Результат виконання тригера**

Створено тригер, який визначає в якому класі знаходиться місце пасажира за його квитком (рисунок 3.26):

```
CREATE FUNCTION set_ship_class() RETURNS trigger AS $setClassShipHasCus$
BEGIN
    CASE
        WHEN (SELECT idType_tickets
              FROM Ship_tickets
              WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 12
            THEN NEW.s_class = 'Стандарт';
        WHEN (SELECT idType_tickets
              FROM Ship_tickets
              WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 13
            THEN NEW.s_class = 'Преміум';
        WHEN (SELECT idType_tickets
              FROM Ship_tickets
              WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 14
            THEN NEW.s_class = 'Економ';
    END CASE;
    RETURN NEW;
END;
$setClassShipHasCus$ LANGUAGE plpgsql;

CREATE TRIGGER setClassShipHasCus BEFORE INSERT ON
Ship_tickets_has_Customers
FOR EACH ROW EXECUTE PROCEDURE set_ship_class();
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Ship_tickets_has_Customers(ship_tickets_s_ticket_id,
customers_customer_id, s_booking_time, s_seat, comments)
VALUES ( 1, 26, '2020-03-24 05:43:44', 23, NULL),
( 2, 27, '2020-02-25 13:56:49', 97, NULL);
```

	Ship_tickets_s_ticket_ID	Customers_customer_ID	s_booking_time	s_class	s_seat	comments
1	1	26	2020-03-24 05:43:44	Стандарт	23	<null>
2	2	27	2020-02-25 13:56:49	Преміум	97	<null>

Рисунок 3.26 – Результат виконання тригера

Створено тригер, який викликає процедуру, що встановлює вартість автобусного квитка (рисунок 3.27):

```
CREATE FUNCTION generate_bus_ticket_price() RETURNS trigger AS
$setPriceBusTicket$
BEGIN
    NEW.bt_price = NEW.bt_price + NEW.bt_price*(SELECT
percent_afterpay/100.0
    FROM Type_tickets WHERE typeticket_ID = NEW.idType_tickets);
    RETURN NEW;
END;
$setPriceBusTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceBusTicket BEFORE INSERT ON Bus_tickets
FOR EACH ROW EXECUTE PROCEDURE generate_bus_ticket_price();
```

Введено тестові дані для перевірки результату:

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		67



```
INSERT INTO Bus_tickets(bt_price, idType_tickets, idRoute)
VALUES ( 318, 10, 1),
      ( 212, 10, 2);
```

	b_ticket_ID	bt_price	idType_tickets	idRoute
1	1	318	10	1
2	2	212	10	2

Рисунок 3.27 – Результат виконання тригера

Створено тригер, який встановлює вартість квитка на поїзд (рисунок 3.28):

```
CREATE FUNCTION generate_train_ticket_price() RETURNS trigger AS
$setPriceTrainTicket$
BEGIN
    NEW.tt_price = NEW.tt_price/2.0 + (NEW.tt_price/2.0)*(SELECT
percent_afterpay/100.0
    FROM Type_tickets WHERE typeticket_ID = NEW.idType_tickets);
    RETURN NEW;
END;
$setPriceTrainTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceTrainTicket BEFORE INSERT ON Train_tickets
FOR EACH ROW EXECUTE PROCEDURE generate_train_ticket_price();
```

Введено тестові дані для перевірки результату:

```
INSERT INTO Train_tickets( tt_price, idType_tickets, idRoute)
VALUES ( 394, 1, 31),
      ( 554, 1, 32);
```

	t_ticket_ID	tt_price	idType_tickets	idRoute
1	1	296	1	31
2	2	416	1	32

Рисунок 3.28 – Результат виконання тригера

### 3.3 Створення запитів

PostgreSQL надає можливість використовувати оператор WITH у запитах. WITH надає спосіб записувати додаткові оператори для застосування у великих запитах. Ці оператори, які також називають загальними табличними виразами (Common Table Expressions, CTE), можна уявити як визначення тимчасових таблиць, які існують лише для одного запиту. Основне призначення SELECT в реченні WITH полягає в розбитті складних запитів на прості частини. [7]

У результаті аналізу завдання на курсове проектування було створено наступні запити, які відображають усі можливі дії з даними, що містяться у таблицях.

Створено запит, який формує список фірми, що проводять тільки закордонні рейси та обчислює їхню кількість (рисунок 3.29):

```
SELECT name, COUNT(flight_ID) AS Foreign_flights
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country <> 'Україна' AND NOT EXISTS( SELECT COUNT(flight_ID)
FROM Flights INNER JOIN Firms Fi
ON Flights.idFirm = Fi.firm_ID INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country = 'Україна' AND F.name = Fi.name
HAVING COUNT(flight_ID) <> 0)
GROUP BY 1;
```

	name	foreign_flights
1	Emirates Airlines	3
2	EuroClub	1
3	KLM	4
4	Lufthansa	2
5	Royal Caribbean	1
6	Turkish Airlines	3
7	Wizz Air	1

Рисунок 3.29 – Результат виконання запиту

Створено запит, який виводить ініціали всіх пасажирів і вартості їхніх квитків на авіа, автобуні, залізничні та морські рейси (рисунок 3.30):

```
SELECT customer_ID, c_name, at_price, Bt.bt_price, Tt.tt_price,
st_price
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID LEFT JOIN
Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID LEFT JOIN
Bus_tickets Bt
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID LEFT JOIN
Train_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID LEFT JOIN
Train_tickets Tt
ON Tt.t_ticket_ID = TthC.Train_tickets_t_ticket_ID LEFT JOIN
Ship_tickets_has_Customers SthC
ON Customers.customer_ID = SthC.Customers_customer_ID LEFT JOIN
Ship_tickets S
ON S.s_ticket_ID = SthC.Ship_tickets_s_ticket_ID LEFT JOIN
Flights has_Cities FhC
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

ON A.Flights_has_Cities_primaryID = FhC.primaryID LEFT JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets
B
ON R.route_ID = B.idRoute LEFT JOIN Train_tickets T
ON R.route_ID = T.idRoute;

```

	customer_id :	C_name :	st_price :	st_price :	tt_price :	st_price :
1	2	Ніколенко Ромашка Зорянівна	2374	<null>	295.5	<null>
2	4	Алиськевич Мар'ян Устимович	2436.4	<null>	415.5	<null>
3	6	Вітренко Власт Денисович	3928	<null>	143.65	<null>
4	8	Міняйло Йосифата Ярославівна	5958	<null>	222.2	<null>
5	10	Мечитайло Данко Мстиславович	7086	<null>	208	<null>
6	12	Токарчук Родіон Найденович	6352.6	<null>	136.25	<null>
7	14	Пупійчук Дан Хданович	7361.5	<null>	36.3	<null>
8	16	Догань Гаїна Тимурівна	15148.5	<null>	85.05	<null>
9	18	Пастушенко Ніна Ростиславівна	7767.9	<null>	239.2	<null>
10	20	Босчко Лев Антонович	2277.6	<null>	94	<null>
11	22	Барвінський Антон Охримович	7546.8	<null>	163.5	<null>
12	24	Замора Егор Хданович	4595.5	<null>	368	<null>
13	26	Німченко Йосифата Борисівна	<null>	318	398.5	8930
14	28	Смішко Іларіон Ігорович	<null>	448	401.8	8503.25
15	30	Медяник Йозеф Богуславович	<null>	554	273	11063.75
16	32	Мулярчук Хвалимир Остапович	<null>	577	378.5	<null>
17	34	Дурдинець Триріг Лебомирович	<null>	953	132.55	<null>
18	36	Вінтоняк Альберт Пилипович	<null>	496	453	<null>
19	38	Могиленко Йосип Сарматович	<null>	153	42.9	<null>
20	40	Павлович Гладко Пилипович	<null>	297	358.35	<null>
21	42	Сімович Цейтан Августинович	<null>	318	232.75	<null>
22	44	Чубатий Івантослав Давидович	<null>	384	252.25	<null>
23	46	Вайда Турбрід Драганович	<null>	409	138.75	<null>
24	48	Бурбан Верніслав Артемович	<null>	66	197	<null>
25	50	Деркач Летобор Романович	<null>	294.4	165.2	<null>
26	25	Німчук Злотан Найденович	11250	<null>	<null>	<null>
27	27	Ганушак Фауст Антонович	<null>	212	<null>	11063.75
28	11	Сомко Яромира Остапівна	4454.9	<null>	<null>	<null>
29	39	Коструба Шастислав Федорович	<null>	112	<null>	<null>
30	17	Іденко Олесь Денисович	6402.6	<null>	<null>	<null>
31	33	Мисько Живосил Чеславович	<null>	518.4	<null>	<null>
32	31	Тихий Тихон Ёхимович	<null>	89.6	<null>	<null>
33	47	Мухопад Царук Адріанович	<null>	545	<null>	<null>
34	15	Пазорський Никифор Олександрович	4210	<null>	<null>	<null>
35	13	Соломченко Хотимир Златович	5137.7	<null>	<null>	<null>
36	21	Гриневецький Хотян Сарматович	8566.5	<null>	<null>	<null>
37	5	Сєвка Щедра Фролівна	4167.3	<null>	<null>	<null>
38	19	Пустовойт Колодар Богданович	6815.5	<null>	<null>	<null>
39	37	Чалий Йозеф Костянтинівич	<null>	894	<null>	<null>
40	29	Ведимська Мозефіна Августинівна	<null>	344	<null>	8930
41	41	Петрицький Зборислав Давидович	<null>	423	<null>	<null>
42	23	Куц Югіна Арсенівна	4758.4	<null>	<null>	<null>
43	1	Тарасович Ратимир Вадимович	7126.5	<null>	<null>	<null>
44	49	Штинь Атрей Русланович	<null>	243	<null>	<null>
45	45	Сорока Фрії Гордиславович	<null>	268	<null>	<null>
46	43	Щацелєх Євстафій Августинович	<null>	184	<null>	<null>
47	3	Кириленко Муховіст Полянович	4804	<null>	<null>	<null>
48	35	Магура Ізяслав Азарович	<null>	3754	<null>	<null>
49	9	Семків Йоган Артемович	2581.1	<null>	<null>	<null>
50	7	Смолій Ярослав Зорянович	871.6	<null>	<null>	<null>

Рисунок 3.30 – Результат виконання запиту

Створено запит, який виводить всі можливі автобусні та залізничні маршрути до Києва з різних міст України (рисунок 3.31):

```

SELECT flight_ID, route_ID, f_from, city_name,idCity
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID INNER JOIN Cities C
ON C.city_ID = R.idCity
WHERE city_name = 'Київ' AND country = 'Україна';

```

	flight_id	route_id	f_from	city_name	idcity
1	26	14	Чернігів	Київ	51
2	45	19	Миколаїв	Київ	51
3	27	20	Вінниця	Київ	51
4	39	22	Херсон	Київ	51
5	41	36	Херсон	Київ	51
6	5	41	Рівне	Київ	51
7	38	46	Івано-Франківськ	Київ	51
8	11	50	Львів	Київ	51

Рисунок 3.31 – Результат виконання запиту

Створено запит, який виводить фірми, що займаються повітряним або наземним перевезеннями та кількість рейсів яких більше трьох і водночас менше тринадцяти (рисунок 3.32):

```
SELECT F.name AS firm_name, type_name AS transition, COUNT(flight_ID) AS
number_of_flights
FROM Flights INNER JOIN Firms F
ON F.firm_ID = Flights.idFirm INNER JOIN Type_transportations Tt
ON Tt.transportation_ID = F.idTransportation
WHERE type_name IN ('Наземне', 'Повітряне')
GROUP BY 1, 2
HAVING COUNT(flight_ID) BETWEEN 3 AND 13;
```

	firm_name	transition	number_of_flights
1	Emirates Airlines	Повітряне	3
2	KLM	Повітряне	4
3	Orionbus	Наземне	3
4	Turkish Airlines	Повітряне	3
5	MAY	Повітряне	12

Рисунок 3.32 – Результат виконання запиту

Створено запит, що виводить список авіаквитків зарезервованих протягом останніх шести місяців (рисунок 3.33):

```
SELECT a_booking_time, at_price, at_class, a_seat
FROM Airplane_tickets INNER JOIN Airplane_tickets_has_Customers AthC
ON Airplane_tickets.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID
WHERE CURRENT_TIMESTAMP - INTERVAL '6 MONTH' < a_booking_time;
```

	a_booking_time	at_price	at_class	a_seat
1	2020-12-20 21:12:27.000000	5137.7	Економ	13
2	2020-12-23 04:59:29.000000	6402.6	Економ	17
3	2020-12-17 13:17:31.000000	7546.8	Економ	22

Рисунок 3.33 – Результат виконання запиту

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Створено запит, що визначає тривалість рейсів, які виходять за межі України (рисунок 3.34):

```
SELECT CONCAT(f_from, ' - ', f_to) AS Flight,
       CONCAT(ROUND((EXTRACT(EPOCH FROM (endtime - starttime)) /
3600.00)::NUMERIC, 1), ' год.') AS Hours,
       CASE
         WHEN EXTRACT(EPOCH FROM (endtime - starttime)) / 3600.00 <= 3.0
           THEN 'Швидкий'
         WHEN EXTRACT(EPOCH FROM (endtime - starttime)) / 3600.00 <= 9.0
           THEN 'Середній'
         ELSE 'Тривалий'
       END AS Тривалість_рейсу
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country <> 'Україна'
ORDER BY EXTRACT(EPOCH FROM (endtime - starttime)) / 3600.00 DESC;
```

	flight	hours	Тривалість_рейсу
1	Одеса - Афіни	21.2 год.	Тривалий
2	Київ - Лісабон	4.2 год.	Середній
3	Донецьк - Андорра-ла-Велья	3.6 год.	Середній
4	Кропивницький - Амстердам	2.5 год.	Швидкий
5	Черкаси - Брюссель	2.5 год.	Швидкий
6	Хмельницький - Париж	2.2 год.	Швидкий
7	Івано-Франківськ - Валлетта	2.1 год.	Швидкий
8	Київ - Осло	2.0 год.	Швидкий
9	Рівне - Монако	2.0 год.	Швидкий
10	Житомир - Люксембург	2.0 год.	Швидкий
11	Хмельницький - Берн	1.8 год.	Швидкий
12	Київ - Тирана	1.6 год.	Швидкий
13	Суми - Гельсінкі	1.5 год.	Швидкий
14	Миколаїв - Сараєво	1.4 год.	Швидкий
15	Полтава - Софія	1.4 год.	Швидкий
16	Харків - Бухарест	1.2 год.	Швидкий
17	Луцьк - Любляна	1.2 год.	Швидкий
18	Донецьк - Мінськ	1.2 год.	Швидкий
19	Ужгород - Кишинів	0.8 год.	Швидкий
20	Київ - Вільнюс	0.7 год.	Швидкий
21	Чернівці - Бухарест	0.5 год.	Швидкий
22	Одеса - Мадрид	0.5 год.	Швидкий

Рисунок 3.34 – Результат виконання запиту

Створено запит, який виводить список транспортних засобів та загальну кількість кожного виду квитків (рисунок 3.35):

```
SELECT CONCAT(name, ' ', number), STRING_AGG(CONCAT(type_afterpay, ': ',
count_seats_type, ' шт. '), ' ')
FROM Transport INNER JOIN Transport_has_Type_tickets ThTt
```

```

ON Transport.transport_ID = ThTt.Transport_idTransport INNER JOIN
Type_tickets Tt
ON Tt.typeticket_ID = ThTt.Type_ticketeta_idType_tickets
GROUP BY 1;

```

concat	string_agg
1 Призов`я 989/988	Купе дорослий: 398 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50
2 GULERYUZ BX5339EM	Автобусний: 30 шт. Автобусний студентський: 6 шт.
3 GULERYUZ BK3137HH	Автобусний: 20 шт. Автобусний студентський: 4 шт.
4 GULERYUZ BC9664MB	Автобусний: 20 шт. Автобусний студентський: 2 шт.
5 ISUZU KA10020P	Автобусний: 30 шт. Автобусний студентський: 4 шт.
6 Celebrity Xploration	Морський: 500 шт. Морський VIP: 145 шт. Морський економ: 200 шт.
7 Neoplan BI7798EX	Автобусний: 25 шт. Автобусний студентський: 5 шт.
8 AN 178	Бізнес клас: 10 шт. Економ клас: 75 шт. Перший клас: 25 шт.
9 Airbus A340	Бізнес клас: 30 шт. Економ клас: 100 шт. Перший клас: 40 шт.
10 Південний експрес 686/685	Купе дорослий: 200 шт. Купе студентський: 36 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 300 шт. Плацкарт дитячий: 50
11 ЕТАЛОН BC4780MB	Автобусний: 25 шт. Автобусний студентський: 3 шт.
12 ЕТАЛОН AA4243PT	Автобусний: 30 шт. Автобусний студентський: 0 шт.
13 Богдан AE5501AA	Автобусний: 20 шт. Автобусний студентський: 5 шт.
14 Подільський експрес 103/104	Купе дорослий: 270 шт. Купе студентський: 25 шт. Купе дитячий: 25 шт. Плацкарт дорослий: 250 шт. Плацкарт дитячий: 50
15 Airbus A330	Бізнес клас: 25 шт. Економ клас: 75 шт. Перший клас: 25 шт.
16 Douglas DC-1	Бізнес клас: 35 шт. Економ клас: 200 шт. Перший клас: 50 шт.
17 Boeing Classic 737	Бізнес клас: 30 шт. Економ клас: 150 шт. Перший клас: 50 шт.
18 Богдан AC1501EO	Автобусний: 20 шт. Автобусний студентський: 5 шт.
19 ISUZU AB2775HT	Автобусний: 30 шт. Автобусний студентський: 3 шт.
20 Neoplan BH21370H	Автобусний: 30 шт. Автобусний студентський: 2 шт.
21 Neoplan AX3116BM	Автобусний: 30 шт. Автобусний студентський: 10 шт.
22 AN 13	Бізнес клас: 14 шт. Економ клас: 100 шт. Перший клас: 20 шт.
23 Dassault Falcon 900	Бізнес клас: 30 шт. Економ клас: 200 шт. Перший клас: 50 шт.
24 Boeing Next Generation 737	Бізнес клас: 40 шт. Економ клас: 200 шт. Перший клас: 50 шт.
25 ISUZU A02753EP	Автобусний: 20 шт. Автобусний студентський: 9 шт.
26 Богдан AE1095AA	Автобусний: 20 шт. Автобусний студентський: 5 шт.
27 ЕТАЛОН BI7837EA	Автобусний: 25 шт. Автобусний студентський: 3 шт.
28 AN 74	Бізнес клас: 10 шт. Економ клас: 50 шт. Перший клас: 40 шт.
29 Богдан BT1552CC	Автобусний: 20 шт. Автобусний студентський: 5 шт.
30 Neoplan KA2376BT	Автобусний: 25 шт. Автобусний студентський: 5 шт.
31 AN 275	Бізнес клас: 20 шт. Економ клас: 60 шт. Перший клас: 30 шт.
32 AN 225	Бізнес клас: 15 шт. Економ клас: 80 шт. Перший клас: 20 шт.
33 Neoplan AT6426EP	Автобусний: 25 шт. Автобусний студентський: 5 шт.
34 AN 1326	Бізнес клас: 25 шт. Економ клас: 200 шт. Перший клас: 40 шт.
35 Промінь 421/420	Купе дорослий: 9 шт. Купе студентський: 5 шт. Купе дитячий: 5 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50 шт.
36 Farman Goliath F.60	Бізнес клас: 30 шт. Економ клас: 200 шт. Перший клас: 40 шт.
37 Богдан AE48580X	Автобусний: 20 шт. Автобусний студентський: 5 шт.
38 ISUZU BM1049CM	Автобусний: 25 шт. Автобусний студентський: 5 шт.
39 Enchantment of the Seas	Морський: 1500 шт. Морський VIP: 405 шт. Морський економ: 500 шт.
40 Dassault Falcon 7X	Бізнес клас: 10 шт. Економ клас: 150 шт. Перший клас: 100 шт.
41 Boeing 747	Бізнес клас: 45 шт. Економ клас: 150 шт. Перший клас: 50 шт.
42 ISUZU AE1095AA	Автобусний: 30 шт. Автобусний студентський: 2 шт.
43 ЕТАЛОН AM9254EO	Автобусний: 20 шт. Автобусний студентський: 8 шт.
44 Північний експрес 784/785	Купе дорослий: 241 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 350 шт. Плацкарт дитячий: 75
45 AN 28	Бізнес клас: 80 шт. Економ клас: 150 шт. Перший клас: 100 шт.
46 Богдан AB1994IA	Автобусний: 20 шт. Автобусний студентський: 5 шт.
47 Celebrity Xpedition	Морський: 4530 шт. Морський VIP: 300 шт. Морський економ: 700 шт.
48 ЕТАЛОН BH4091AA	Автобусний: 25 шт. Автобусний студентський: 3 шт.
49 Pogo Fugo	Морський: 140 шт. Морський VIP: 50 шт. Морський економ: 50 шт.
50 GULERYUZ BI8403EE	Автобусний: 25 шт. Автобусний студентський: 3 шт.
51 ЕТАЛОН B05913CP	Автобусний: 25 шт. Автобусний студентський: 3 шт.
52 Севастополь 542/543	Купе дорослий: 34 шт. Купе студентський: 95 шт. Купе дитячий: 55 шт. Плацкарт дорослий: 300 шт. Плацкарт дитячий: 75
53 Хаджибей 683/682	Купе дорослий: 143 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 300 шт. Плацкарт дитячий: 50
54 Богдан BC3458MK	Автобусний: 20 шт. Автобусний студентський: 5 шт.
55 Таврія 835/834	Купе дорослий: 55 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50
56 Forse le Carib	Морський: 370 шт. Морський VIP: 100 шт. Морський економ: 100 шт.
57 ISUZU AT7929EP	Автобусний: 20 шт. Автобусний студентський: 6 шт.
58 Neoplan BC5474MK	Автобусний: 30 шт. Автобусний студентський: 2 шт.
59 ISUZU BC6097MK	Автобусний: 20 шт. Автобусний студентський: 8 шт.
60 Neoplan AX5109HX	Автобусний: 25 шт. Автобусний студентський: 5 шт.
61 Чайка 093/094	Купе дорослий: 142 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50
62 ЕТАЛОН BM6175CM	Автобусний: 25 шт. Автобусний студентський: 5 шт.
63 Symphony of the Seas	Морський: 750 шт. Морський VIP: 250 шт. Морський економ: 300 шт.
64 Neoplan AE5740KP	Автобусний: 25 шт. Автобусний студентський: 5 шт.
65 Boeing 777	Бізнес клас: 50 шт. Економ клас: 75 шт. Перший клас: 125 шт.
66 Рось 117/118	Купе дорослий: 102 шт. Купе студентський: 75 шт. Купе дитячий: 65 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 80

Рисунок 3.35 – Результат виконання запиту

					Арк.
Змн.	Арк.	№ докум.	Підпис	Дата	73

КП.ПІ-18-01.05.02.00.000 ПЗ



Створено запит, який виводить три найзбитковіші автобусні рейси (рисуюнок 3.36):

```
SELECT flight_ID, CONCAT(f_from, ' - ', f_to) AS Flight,
starttime::TIMESTAMP(0), endtime::TIMESTAMP(0),
CONCAT(T.name, ' ', T.number) AS Bus_and_number,
(SELECT COUNT(Bus_tickets_b_ticket_ID)
FROM Bus_tickets_has_Customers INNER JOIN Bus_tickets Bt
ON Bus_tickets_has_Customers.Bus_tickets_b_ticket_ID =
Bt.b_ticket_ID
INNER JOIN Routes R ON Bt.idRoute = R.route_ID INNER JOIN
Flights_has_Cities FhC
ON R.Flights_has_Cities_primaryID = FhC.primaryID
WHERE flight_ID = idFlight) AS Sold_tickets
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
ON Flights.idTransport = T.transport_ID
WHERE type_name = 'Наземне'
ORDER BY 6
LIMIT 3;
```

	flight_id	flight	starttime	endtime	bus_and_number	sold_tickets
1	22	Дніпро - Харків	2021-10-24 19:04:23	2021-10-24 22:25:23	Neoplan AX5109HX	0
2	48	Дніпро - Луцьк	2021-10-21 07:47:48	2021-10-21 21:20:48	ISUZU AB2775HT	0
3	23	Тернопіль - Рівне	2021-07-10 19:58:31	2021-07-10 22:19:31	Neoplan BC5474MK	1

Рисуюнок 3.36 – Результат виконання запиту

Запит, який виводить пасажирів, що витратили найбільше грошей для покупку квитків на будь-який вид перевезень (рисуюнок 3.37):

```
SELECT customer_ID, c_name, LPAD(CONCAT(ROUND(COALESCE(at_price, 0) +
COALESCE(Bt.bt_price, 0) +
COALESCE(Tt.tt_price, 0) +
COALESCE(st_price, 0), 2), ' грн.'), 18, ' ') AS total
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID LEFT JOIN
Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID LEFT JOIN
Bus_tickets Bt
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID LEFT JOIN
Train_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID LEFT JOIN
Train_tickets Tt
ON Tt.t_ticket_ID = TthC.Train_tickets_t_ticket_ID LEFT JOIN
Ship_tickets_has_Customers SthC
ON Customers.customer_ID = SthC.Customers_customer_ID LEFT JOIN
Ship_tickets S
ON S.s_ticket_ID = SthC.Ship_tickets_s_ticket_ID LEFT JOIN
Flights_has_Cities FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID LEFT JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets
B
ON R.route_ID = B.idRoute LEFT JOIN Train_tickets T
ON R.route_ID = T.idRoute
ORDER BY 3 DESC
LIMIT 10;
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

	customer_id	c_name	total
1	16	Довгань Гаїна Тимурівна	15225.55 грн.
2	30	Медяник Йозеф Богуславович	11890.75 грн.
3	27	Ганущак Фауст Антонович	11275.75 грн.
4	25	Німчук Злотан Найденович	11250.00 грн.
5	26	Німченко Йосифата Борисівна	9646.50 грн.
6	28	Смішко Іларіон Ігорович	9353.05 грн.
7	29	Лещинська Жозефіна Августинівна	9274.00 грн.
8	21	Гриневецький Хотян Сарматович	8566.50 грн.
9	18	Пастушенко Ніна Ростиславівна	8007.10 грн.
10	22	Барвінський Антон Охримович	7710.30 грн.

Рисунок 3.37 – Результат виконання запиту

Створено запит, який виводить кількість маршрутів на кожному рейсі (рисунок 3.38):

```
SELECT type_name AS Transportation, CONCAT(f_from, ' - ', f_to) AS
Flight, COUNT(route_ID) AS number_of_routes,
      STRING_AGG(CONCAT(f_from, ' - ', city_name), ', ')
FROM Flights INNER JOIN Flights_has_Cities
FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN
Bus_tickets Bt
ON R.route_ID = Bt.idRoute LEFT JOIN Train_tickets Tt
ON R.route_ID = Tt.idRoute INNER JOIN Firms F
ON F.firm_ID = Flights.idFirm INNER JOIN Type_transportations T
ON T.transportation_ID = F.idTransportation
INNER JOIN Cities C
ON C.city_ID = R.idCity
GROUP BY 1, 2
HAVING 1 < COUNT(route_ID)
ORDER BY 3 DESC;
```

transpor...	flight	number_of_routes	string_agg
1 Колійне	Рівне - Луганськ	5	Рівне - Полтава, Рівне - Луганськ, Рівне - Житомир, Рівне - Київ, Рівне - Харків
2 Наземне	Тернопіль - Харків	4	Тернопіль - Полтава, Тернопіль - Хмельницький, Тернопіль - Черкаси, Тернопіль -
3 Колійне	Херсон - Київ	4	Херсон - Кропивницький, Херсон - Миколаїв, Херсон - Київ, Херсон - Черкаси
4 Наземне	Дніпро - Луцьк	4	Дніпро - Луцьк, Дніпро - Рівне, Дніпро - Чернігів, Дніпро - Житомир
5 Наземне	Херсон - Київ	4	Херсон - Черкаси, Херсон - Кропивницький, Херсон - Київ, Херсон - Миколаїв
6 Наземне	Чернігів - Кропивницький	3	Чернігів - Черкаси, Чернігів - Київ, Чернігів - Кропивницький
7 Колійне	Івано-Франківськ - Суми	3	Івано-Франківськ - Суми, Івано-Франківськ - Хмельницький, Івано-Франківськ - Київ
8 Колійне	Київ - Луганськ	3	Київ - Полтава, Київ - Харків, Київ - Луганськ
9 Колійне	Львів - Харків	3	Львів - Київ, Львів - Полтава, Львів - Харків
10 Колійне	Черкаси - Львів	3	Черкаси - Хмельницький, Черкаси - Львів, Черкаси - Тернопіль
11 Наземне	Житомир - Івано-Франківськ	3	Житомир - Хмельницький, Житомир - Тернопіль, Житомир - Івано-Франківськ
12 Наземне	Миколаїв - Київ	3	Миколаїв - Черкаси, Миколаїв - Кропивницький, Миколаїв - Київ
13 Наземне	Одеса - Мадрид	3	Одеса - Мадрид, Одеса - Чернівці, Одеса - Ужгород
14 Колійне	Київ - Дніпро	2	Київ - Дніпро, Київ - Полтава
15 Наземне	Миколаїв - Хмельницький	2	Миколаїв - Хмельницький, Миколаїв - Вінниця
16 Колійне	Житомир - Івано-Франківськ	2	Житомир - Івано-Франківськ, Житомир - Тернопіль
17 Наземне	Вінниця - Чернігів	2	Вінниця - Чернігів, Вінниця - Київ
18 Колійне	Хмельницький - Луганськ	2	Хмельницький - Луганськ, Хмельницький - Черкаси
19 Колійне	Полтава - Миколаїв	2	Полтава - Миколаїв, Полтава - Кропивницький

Рисунок 3.38 – Результат виконання запиту



### 3.4 Створення представлень

Представлення в PostgreSQL реалізовані на основі системи правил. Фактично по суті немає ніякої відмінності

```
CREATE VIEW myview AS SELECT * FROM mytab;
```

від наступних двох команд:

```
CREATE TABLE myview ( same column list as mytab);
```

```
CREATE RULE "_RETURN" AS ON SELECT TO myview DO INSTEAD  
SELECT * FROM mytab;
```

так як саме ці дії CREATE VIEW виконує всередині. Це має деякі побічні ефекти. Зокрема, інформація про подання в системних каталогах PostgreSQL нічим не відрізняється від інформації про таблиці. Тому при аналізі запиту немає абсолютно ніякої різниці між таблицями і представленнями. Вони являють собою одне й те ж – відношення. [7]

У відповідності до поставленого завдання на курсове проєктування було створено наступні представлення.

Створено представлення, яке містить інформацію про фірми, що проводять тільки закордонні рейси та кількість цих рейсів (рисунок 3.39):

```
CREATE VIEW foreign_firms AS SELECT firm_ID, name, COUNT(flight_ID) AS  
amount_of_flights  
FROM Flights INNER JOIN Firms F  
ON Flights.idFirm = F.firm_ID INNER JOIN Flights_has_Cities FhC  
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C  
ON C.city_ID = FhC.idCity2  
WHERE country <> 'Україна' AND NOT EXISTS( SELECT COUNT(flight_ID)  
FROM Flights INNER JOIN Firms Fi  
ON Flights.idFirm = Fi.firm_ID INNER JOIN Flights_has_Cities FhC  
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C  
ON C.city_ID = FhC.idCity2  
WHERE country = 'Україна' AND F.name = Fi.name  
HAVING COUNT(flight_ID) <> 0)  
GROUP BY 1;
```

	firm_id	name	amount_of_flights
1	2	KLM	4
2	3	Emirates Airlines	3
3	5	Wizz Air	1
4	6	Turkish Airlines	3
5	7	Lufthansa	2
6	14	Royal Caribbean	1
7	18	EuroClub	1

Рисунок 3.39 – Результат виконання представлення

Створено представлення, яке об'єднує найважливішу інформацію про авіаквитки (рисунок 3.40):

```
CREATE VIEW tickets_between_date AS SELECT c_name AS Name,
CONCAT(f_from, ' - ', f_to) AS Flight,
CONCAT(starttime, ' - ', endtime) AS Time, at_class AS Class,
a_seat AS Seat,
at_price AS Price, a_booking_time::TIMESTAMP(0) AS Booking
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID INNER JOIN
Flights_has_Cities FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID INNER JOIN Firms F2
ON F.idFirm = F2.firm_ID INNER JOIN Type_transportations Tt
ON F2.idTransportation = Tt.transportation_ID
WHERE type_name = 'Повітряне';
```

	name	flight	time	class	seat	price	booking
1	Тарасович Ратимир Вадимович	Житомир - Люксембург	2021-10-20 17:46:29 - 2021-10-20 19:46:29	Бізнес	1	7126.5	2020-10-09 20:21:36
2	Ніколенко Ромашка Зорянівна	Чернівці - Бухарест	2021-09-15 10:20:25 - 2021-09-15 10:50:25	Перший	2	2374	2020-01-14 09:55:01
3	Кириленко Муховіст Полянович	Запоріжжя - Івано-Франківськ	2021-08-08 13:54:20 - 2021-08-08 15:20:20	Перший	3	4804	2020-06-23 14:27:41
4	Алиськевич Мар'ян Устимович	Львів - Одеса	2021-09-23 10:00:51 - 2021-09-23 10:46:51	Економ	4	2436.4	2020-05-26 03:05:58
5	Савка Щедра Фролівна	Суми - Івано-Франківськ	2021-09-12 11:39:37 - 2021-09-12 12:59:37	Економ	5	4167.3	2020-04-08 16:28:10
6	Вітренко Власт Денисович	Луцьк - Любляна	2021-02-23 19:18:07 - 2021-02-23 20:29:07	Бізнес	6	3920	2020-04-14 02:14:58
7	Смолій Ярослав Зорянович	Луганськ - Суми	2021-02-27 13:29:13 - 2021-02-27 13:49:13	Економ	7	871.6	2020-08-27 05:01:43
8	Міняйло Йосифата Ярославівна	Донецьк - Ужгород	2021-04-17 00:03:44 - 2021-04-17 01:42:44	Перший	8	5958	2020-09-17 00:14:03
9	Семків Йоган Артемович	Ужгород - Кишинів	2021-03-16 12:28:59 - 2021-03-16 13:18:59	Економ	9	2581.1	2020-04-08 06:09:20
10	Нечитайло Данко Мстиславович	Полтава - Софія	2021-01-14 01:33:36 - 2021-01-14 02:55:36	Бізнес	10	7086	2020-04-07 05:52:12
11	Сомко Яромира Остапівна	Миколаїв - Сараєво	2021-10-06 18:55:15 - 2021-10-06 20:19:15	Економ	11	4454.9	2020-06-02 16:57:06
12	Токарчук Родіон Найденович	Київ - Осло	2021-01-02 18:36:57 - 2021-01-02 20:38:57	Економ	12	6352.6	2020-06-01 14:13:55
13	Соломченко Хотимир Златович	Київ - Тирана	2021-01-08 21:21:34 - 2021-01-08 22:58:34	Економ	13	5137.7	2020-12-20 21:12:27
14	Лупійчук Дан Жданович	Хмельницький - Париж	2021-01-02 08:07:23 - 2021-01-02 10:21:23	Перший	14	7361.5	2020-04-30 03:00:09
15	Лазорський Никифор Олександр	Харків - Бухарест	2021-03-03 03:42:24 - 2021-03-03 04:55:24	Перший	15	4210	2020-01-10 06:32:32
16	Довгань Гаїна Тимурівна	Київ - Лісабон	2021-09-26 09:34:54 - 2021-09-26 13:45:54	Бізнес	16	15140.5	2020-10-19 16:27:48
17	Юденко Олесь Денисович	Рівне - Монако	2021-04-21 00:32:33 - 2021-04-21 02:34:33	Економ	17	6402.6	2020-12-23 04:59:29
18	Пастушенко Ніна Ростиславівна	Черкаси - Брюссель	2021-07-30 09:44:29 - 2021-07-30 12:11:29	Економ	18	7767.9	2020-03-08 17:26:36
19	Пустовойт Колодар Богданович	Хмельницький - Берн	2021-03-19 11:52:21 - 2021-03-19 13:40:21	Перший	19	6015.5	2020-06-22 02:36:05
20	Бочко Лев Антонович	Київ - Вільнюс	2021-11-05 17:59:33 - 2021-11-05 18:42:33	Економ	20	2277.6	2020-02-10 15:28:49
21	Гриневецкий Хотян Сарматов...	Івано-Франківськ - Вал...	2021-09-19 12:36:24 - 2021-09-19 14:41:24	Бізнес	21	8566.5	2020-06-22 08:47:22
22	Барвінський Антон Охримович	Кропивницький - Амстер...	2021-03-12 08:58:18 - 2021-03-12 11:26:18	Економ	22	7546.8	2020-12-17 13:17:31
23	Куц Югіна Арсенівна	Суми - Гельсінкі	2021-03-15 02:15:15 - 2021-03-15 03:46:15	Економ	23	4758.4	2020-04-16 16:54:07
24	Замора Єгор Жданович	Донецьк - Мінськ	2021-11-04 00:17:22 - 2021-11-04 01:27:22	Перший	24	4595.5	2020-06-25 04:22:31

Рисунок 3.40 – Результат виконання представлення

Створено представлення, яке зберігає загальну суму витрачених коштів кожним пасажиром (рисунок 3.41):

```
CREATE VIEW customer_money AS SELECT customer_ID, c_name,
LPAD(CONCAT(ROUND(COALESCE(at_price, 0) +
COALESCE(Bt.bt_price, 0) +
COALESCE(Tt.tt_price, 0) +
COALESCE(st_price, 0), 2), ' грн.'), 18, '
') AS total
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID LEFT JOIN
Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID LEFT JOIN
Bus_tickets Bt
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID LEFT JOIN
Train_tickets_has_Customers TthC
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

ON Customers.customer_ID = TthC.Customers_customer_ID LEFT JOIN
Train_tickets Tt
ON Tt.t_ticket_ID = TthC.Train_tickets_t_ticket_ID LEFT JOIN
Ship_tickets_has_Customers SthC
ON Customers.customer_ID = SthC.Customers_customer_ID LEFT JOIN
Ship_tickets S
ON S.s_ticket_ID = SthC.Ship_tickets_s_ticket_ID LEFT JOIN
Flights_has_Cities FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID LEFT JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets
B
ON R.route_ID = B.idRoute LEFT JOIN Train_tickets T
ON R.route_ID = T.idRoute
ORDER BY 3 DESC;

```

	customer_id	c_name	total
1	16	Довгань Гаїна Тимурівна	15225.55 грн.
2	30	Медяник Йозеф Богуславович	11890.75 грн.
3	27	Ганущак Фауст Антонович	11275.75 грн.
4	25	Німчук Злотан Найденович	11250.00 грн.
5	26	Німченко Йосифата Борисівна	9646.50 грн.
6	28	Смішко Іларіон Ігорович	9353.05 грн.
7	29	Лещинська Жозефіна Августинівна	9274.00 грн.
8	21	Гриневецький Хотян Сарматович	8566.50 грн.
9	18	Пастушенко Ніна Ростиславівна	8007.10 грн.
10	22	Барвінський Антон Охримович	7710.30 грн.
11	14	Лупійчук Дан Жданович	7397.80 грн.
12	10	Нечитайло Данко Мстиславович	7286.00 грн.
13	1	Тарасович Ратимир Вадимович	7126.50 грн.
14	12	Токарчук Родіон Найденович	6488.85 грн.
15	17	Юденко Олесь Денисович	6402.60 грн.
16	8	Міняйло Йосифата Ярославівна	6180.20 грн.
17	19	Пустовойт Колодар Богданович	6015.50 грн.
18	13	Соломченко Хотимир Златович	5137.70 грн.
19	24	Замора Єгор Жданович	4955.50 грн.
20	3	Кириленко Муховіст Полянович	4804.00 грн.
21	23	Куш Югіна Арсенівна	4758.40 грн.
22	11	Сомко Яромира Остапівна	4454.90 грн.
23	15	Лазорський Никифор Олександрович	4210.00 грн.
24	5	Савка Щедра Фролівна	4167.30 грн.
25	6	Вітренко Власт Денисович	4063.65 грн.
26	35	Магура Ізяслав Азарович	3754.00 грн.
27	4	Алиськевич Мар'ян Устимович	2851.90 грн.
28	2	Ніколенко Ромашка Зорянівна	2669.50 грн.
29	9	Семків Йоган Артемович	2581.10 грн.
30	20	Боечко Лев Антонович	2371.60 грн.
31	34	Дурдинець Триріг Любомирович	1085.55 грн.
32	36	Вінтоняк Альберт Пилипович	949.00 грн.
33	32	Мулярчук Хвалимир Остапович	947.50 грн.
34	37	Чалий Йозеф Костянтинович	894.00 грн.
35	7	Смолій Ярослав Зорянович	871.60 грн.
36	40	Павлович Гладко Пилипович	647.35 грн.
37	44	Чубатий Івантослав Давидович	636.25 грн.
38	46	Вайда Турбрід Драганович	547.75 грн.
39	47	Мухопад Царук Адріанович	545.00 грн.
40	42	Сімович Цвітан Августинович	542.75 грн.
41	33	Мисько Живосил Чеславович	510.40 грн.
42	50	Деркач Лютобор Романович	459.60 грн.
43	41	Петрицький Зборислав Давидович	423.00 грн.
44	45	Сорока Юрій Гордиславович	268.00 грн.
45	48	Бурбан Вернислав Артемович	263.00 грн.
46	49	Штинь Атрей Русланович	243.00 грн.
47	38	Могиленко Йосип Сарматович	195.90 грн.
48	43	Мацелюх Євстафій Августинович	184.00 грн.
49	39	Коструба Щастислав Федорович	112.00 грн.
50	31	Тихий Тихон Юхимович	89.60 грн.

Рисунок 3.41 – Результат виконання представлення

Створено представлення, яке зберігає список маршрутів на кожному рейсі (рисунок 3.42):

```
CREATE VIEW routes_on_flight AS SELECT CONCAT(f_from, ' - ', f_to) AS
Flight, COUNT(route_ID) AS number_of_routes,
      STRING_AGG(CONCAT(f_from, ' - ', city_name), ', ')
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets
Bt
ON R.route_ID = Bt.idRoute LEFT JOIN Train_tickets Tt
ON R.route_ID = Tt.idRoute INNER JOIN Firms F
ON F.firm_ID = Flights.idFirm INNER JOIN Type_transportations T
ON T.transportation_ID = F.idTransportation INNER JOIN Cities C
ON C.city_ID = R.idCity
GROUP BY 1
ORDER BY 2 DESC;
```

flight	number_of_routes	string_agg
1 Херсон - Київ	8	Херсон - Київ, Херсон - Миколаїв, Херсон - Кропивницький, Херсон - Черкаси, Херсон - Черкаси, Х...
2 Житомир - Івано-Франківськ	5	Житомир - Івано-Франківськ, Житомир - Тернопіль, Житомир - Хмельницький, Житомир - Тернопіль, Ж...
3 Рівне - Луганськ	5	Рівне - Житомир, Рівне - Київ, Рівне - Полтава, Рівне - Харків, Рівне - Луганськ
4 Дніпро - Луцьк	4	Дніпро - Житомир, Дніпро - Луцьк, Дніпро - Рівне, Дніпро - Чернігів
5 Тернопіль - Харків	4	Тернопіль - Полтава, Тернопіль - Хмельницький, Тернопіль - Черкаси, Тернопіль - Харків
6 Черкаси - Львів	3	Черкаси - Тернопіль, Черкаси - Львів, Черкаси - Хмельницький
7 Івано-Франківськ - Суми	3	Івано-Франківськ - Київ, Івано-Франківськ - Хмельницький, Івано-Франківськ - Суми
8 Одеса - Мадрид	3	Одеса - Мадрид, Одеса - Чернівці, Одеса - Ужгород
9 Миколаїв - Київ	3	Миколаїв - Черкаси, Миколаїв - Кропивницький, Миколаїв - Київ
10 Львів - Харків	3	Львів - Київ, Львів - Полтава, Львів - Харків
11 Київ - Луганськ	3	Київ - Полтава, Київ - Харків, Київ - Луганськ
12 Чернігів - Кропивницький	3	Чернігів - Черкаси, Чернігів - Кропивницький, Чернігів - Київ
13 Київ - Дніпро	2	Київ - Полтава, Київ - Дніпро
14 Полтава - Миколаїв	2	Полтава - Кропивницький, Полтава - Миколаїв
15 Хмельницький - Луганськ	2	Хмельницький - Черкаси, Хмельницький - Луганськ
16 Вінниця - Чернігів	2	Вінниця - Чернігів, Вінниця - Київ
17 Миколаїв - Хмельницький	2	Миколаїв - Хмельницький, Миколаїв - Вінниця
18 Ужгород - Чернівці	1	Ужгород - Чернівці
19 Кропивницький - Миколаїв	1	Кропивницький - Миколаїв
20 Тернопіль - Рівне	1	Тернопіль - Рівне
21 Дніпро - Харків	1	Дніпро - Харків
22 Херсон - Миколаїв	1	Херсон - Миколаїв
23 Івано-Франківськ - Вінниця	1	Івано-Франківськ - Вінниця

Рисунок 3.42 – Результат виконання представлення

Створено представлення, яке зберігає інформацію по кількості квитків кожного типу на будь-який транспортний засіб (рисунок 3.43):

```
CREATE VIEW transport_tickets AS SELECT CONCAT(name, ' ', number),
STRING_AGG(CONCAT(type_afterpay, ': ', count_seats_type, ' шт.'), ' ')
FROM Transport INNER JOIN Transport_has_Type_tickets ThTt
ON Transport.transport_ID = ThTt.Transport_idTransport INNER JOIN
Type_tickets Tt
ON Tt.typpeticket_ID = ThTt.Type_ticketeta_idType_tickets
GROUP BY 1;
```

№ concat	№ string_agg
1 Призов'я 989/988	Купе дорослий: 398 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50
2 GULERYUZ BX5339EM	Автобусний: 30 шт. Автобусний студентський: 6 шт.
3 GULERYUZ BK3137NH	Автобусний: 20 шт. Автобусний студентський: 4 шт.
4 GULERYUZ BC9664MB	Автобусний: 20 шт. Автобусний студентський: 2 шт.
5 ISUZU KA10020P	Автобусний: 30 шт. Автобусний студентський: 4 шт.
6 Celebrity Xploration	Морський: 500 шт. Морський VIP: 145 шт. Морський економ: 200 шт.
7 Neoplan BI7798EX	Автобусний: 25 шт. Автобусний студентський: 5 шт.
8 AN 178	Бізнес клас: 10 шт. Економ клас: 75 шт. Перший клас: 25 шт.
9 Airbus A340	Бізнес клас: 30 шт. Економ клас: 100 шт. Перший клас: 40 шт.
10 Південний експрес 686/685	Купе дорослий: 200 шт. Купе студентський: 36 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 300 шт. Плацкарт дитячий: 50
11 ЕТАЛОН BC4780MB	Автобусний: 25 шт. Автобусний студентський: 3 шт.
12 ЕТАЛОН AA4243PT	Автобусний: 30 шт. Автобусний студентський: 0 шт.
13 Богдан AE5501AA	Автобусний: 20 шт. Автобусний студентський: 5 шт.
14 Подільський експрес 103/104	Купе дорослий: 270 шт. Купе студентський: 25 шт. Купе дитячий: 25 шт. Плацкарт дорослий: 250 шт. Плацкарт дитячий: 50
15 Airbus A330	Бізнес клас: 25 шт. Економ клас: 75 шт. Перший клас: 25 шт.
16 Douglas DC-1	Бізнес клас: 35 шт. Економ клас: 200 шт. Перший клас: 50 шт.
17 Boeing Classic 737	Бізнес клас: 30 шт. Економ клас: 150 шт. Перший клас: 50 шт.
18 Богдан AC1501E0	Автобусний: 20 шт. Автобусний студентський: 5 шт.
19 ISUZU AB2775HT	Автобусний: 30 шт. Автобусний студентський: 3 шт.
20 Neoplan BH21370H	Автобусний: 30 шт. Автобусний студентський: 2 шт.
21 Neoplan AX3116BM	Автобусний: 30 шт. Автобусний студентський: 10 шт.
22 AN 13	Бізнес клас: 14 шт. Економ клас: 100 шт. Перший клас: 20 шт.
23 Dassault Falcon 900	Бізнес клас: 30 шт. Економ клас: 200 шт. Перший клас: 50 шт.
24 Boeing Next Generation 737	Бізнес клас: 40 шт. Економ клас: 200 шт. Перший клас: 50 шт.
25 ISUZU A02753EP	Автобусний: 20 шт. Автобусний студентський: 9 шт.
26 Богдан AE1095AA	Автобусний: 20 шт. Автобусний студентський: 5 шт.
27 ЕТАЛОН BI7837EA	Автобусний: 25 шт. Автобусний студентський: 3 шт.
28 AN 74	Бізнес клас: 10 шт. Економ клас: 50 шт. Перший клас: 40 шт.
29 Богдан BT1552CC	Автобусний: 20 шт. Автобусний студентський: 5 шт.
30 Neoplan KA2376BT	Автобусний: 25 шт. Автобусний студентський: 5 шт.
31 AN 275	Бізнес клас: 20 шт. Економ клас: 60 шт. Перший клас: 30 шт.
32 AN 225	Бізнес клас: 15 шт. Економ клас: 80 шт. Перший клас: 20 шт.
33 Neoplan AT6426EP	Автобусний: 25 шт. Автобусний студентський: 5 шт.
34 AN 1326	Бізнес клас: 25 шт. Економ клас: 200 шт. Перший клас: 40 шт.
35 Промінь 421/420	Купе дорослий: 9 шт. Купе студентський: 5 шт. Купе дитячий: 5 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50 шт.
36 Farman Goliath F.60	Бізнес клас: 30 шт. Економ клас: 200 шт. Перший клас: 40 шт.
37 Богдан AE48580X	Автобусний: 20 шт. Автобусний студентський: 5 шт.
38 ISUZU BM1049CM	Автобусний: 25 шт. Автобусний студентський: 5 шт.
39 Enchantment of the Seas	Морський: 1500 шт. Морський VIP: 405 шт. Морський економ: 500 шт.
40 Dassault Falcon 7X	Бізнес клас: 10 шт. Економ клас: 150 шт. Перший клас: 100 шт.
41 Boeing 747	Бізнес клас: 45 шт. Економ клас: 150 шт. Перший клас: 50 шт.
42 ISUZU AE1095AA	Автобусний: 30 шт. Автобусний студентський: 2 шт.
43 ЕТАЛОН AM9254E0	Автобусний: 20 шт. Автобусний студентський: 8 шт.
44 Північний експрес 784/785	Купе дорослий: 241 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 350 шт. Плацкарт дитячий: 75
45 AN 28	Бізнес клас: 80 шт. Економ клас: 150 шт. Перший клас: 100 шт.
46 Богдан AB1994IA	Автобусний: 20 шт. Автобусний студентський: 5 шт.
47 Celebrity Xpedition	Морський: 4530 шт. Морський VIP: 300 шт. Морський економ: 700 шт.
48 ЕТАЛОН BH4091AA	Автобусний: 25 шт. Автобусний студентський: 3 шт.
49 Rogso Fugo	Морський: 140 шт. Морський VIP: 50 шт. Морський економ: 50 шт.
50 GULERYUZ BI8403EE	Автобусний: 25 шт. Автобусний студентський: 3 шт.
51 ЕТАЛОН B05913CP	Автобусний: 25 шт. Автобусний студентський: 3 шт.
52 Севастополь 542/543	Купе дорослий: 34 шт. Купе студентський: 95 шт. Купе дитячий: 55 шт. Плацкарт дорослий: 300 шт. Плацкарт дитячий: 75
53 Хаджибей 683/682	Купе дорослий: 143 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 300 шт. Плацкарт дитячий: 50
54 Богдан BC3458MK	Автобусний: 20 шт. Автобусний студентський: 5 шт.
55 Таврія 835/834	Купе дорослий: 55 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50
56 Forse le Carib	Морський: 370 шт. Морський VIP: 100 шт. Морський економ: 100 шт.
57 ISUZU AT7929EP	Автобусний: 20 шт. Автобусний студентський: 6 шт.
58 Neoplan BC5474MK	Автобусний: 30 шт. Автобусний студентський: 2 шт.
59 ISUZU BC6097MK	Автобусний: 20 шт. Автобусний студентський: 8 шт.
60 Neoplan AX5109HX	Автобусний: 25 шт. Автобусний студентський: 5 шт.
61 Чайка 093/094	Купе дорослий: 142 шт. Купе студентський: 50 шт. Купе дитячий: 50 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 50
62 ЕТАЛОН BM6175CM	Автобусний: 25 шт. Автобусний студентський: 5 шт.
63 Symphony of the Seas	Морський: 750 шт. Морський VIP: 250 шт. Морський економ: 300 шт.
64 Neoplan AE5740KP	Автобусний: 25 шт. Автобусний студентський: 5 шт.
65 Boeing 777	Бізнес клас: 50 шт. Економ клас: 75 шт. Перший клас: 125 шт.
66 Рось 117/118	Купе дорослий: 102 шт. Купе студентський: 75 шт. Купе дитячий: 65 шт. Плацкарт дорослий: 200 шт. Плацкарт дитячий: 80

Рисунок 3.43 – Результат виконання представлення

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

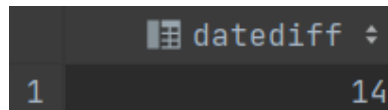
### 3.5 Створення процедур та функцій

PostgreSQL допускає перевантаження функцій; тобто, дозволяє використовувати одне ім'я для кількох різних функцій, якщо у них різняться типи вхідних аргументів. [7]

У відповідності до поставленого завдання на курсове проектування було створено наступні процедури та функції.

Створено функцію, яка повертає різницю між датами в днях (рисунок 3.44):

```
CREATE OR REPLACE FUNCTION DATEDIFF(timestamp, timestamp)
RETURNS INTEGER
AS $$
    SELECT ABS(CAST($1 AS date) - CAST($2 AS date)) as DateDifference
$$ LANGUAGE SQL IMMUTABLE;
SELECT DATEDIFF('2000-01-01 00:00:00','2000-01-15 00:00:00');
```



datediff
14

Рисунок 3.44 – Результат виконання функції

Створено функцію, яка виконує пошук авіаквитків за ПІБ пасажирів (рисунок 3.45):

```
CREATE OR REPLACE FUNCTION search_customer_tickets(search_name TEXT)
RETURNS SETOF text AS $$
DECLARE
    result TEXT DEFAULT '';
    records RECORD;
    myCursor CURSOR(search_name TEXT)
    FOR SELECT c_name AS Name, CONCAT(f_from, ' - ', f_to) AS Flight,
        CONCAT(starttime, ' - ', endtime) AS Time,
        at_class AS Class, a_seat AS Seat,
        at_price AS Price, a_booking_time::TIMESTAMP(0) AS Booking
    FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
    ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
    Airplane_tickets A ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID
    INNER JOIN Flights_has_Cities FhC
    ON A.Flights_has_Cities_primaryID = FhC.primaryID INNER
    JOIN Flights F
    ON FhC.idFlight = F.flight_ID;

BEGIN
    OPEN myCursor(search_name);-- відкриваємо курсор
    LOOP
        FETCH myCursor INTO records;
        EXIT WHEN NOT FOUND;
        IF records.Name = search_name
            THEN result := 'Пасажир: ' || records.Name || ' Рейс: ' ||
            records.Flight || ' Час: (' || records.Time || ') Клас: ' ||
            records.Class
```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		81



```

|| ' Місце: ' || records.Seat || ' Ціна: '
|| records.Price || ' Час бронювання: ' || records.Booking;
RETURN NEXT result;
END IF;
END LOOP;
CLOSE myCursor;-- закриваєм курсор
END;
$$ LANGUAGE plpgsql;

SELECT search_customer_tickets('Сомко Яромира Остапівна');

```

search_customer_tickets	1
1 Пасажир: Сомко Яромира Остапівна Рейс: Миколаїв - Сараєво Час: (2021-10-06 18:55:15 - 2021-10-06 20:19:15) Клас: Економ Місце: 11 Ціна: 4454.9 Час бронювання: 2020-06--	

Рисунок 3.45 – Результат виконання функції

Створено функцію, яка змінює статус рейсу, якщо дата прибуття менша за поточну дату та повертає оновлені дані (рисунок 3.46):

```

CREATE OR REPLACE FUNCTION update_status ()
RETURNS TABLE(flight_ID INT, idFirm INT, idTransport INT, from_city
VARCHAR(45), to_city VARCHAR(45), distance FLOAT, timeStart
TIMESTAMP(0), timeEnd TIMESTAMP(0), flight_status FLIGHTSTATUS)
AS $$
BEGIN
RETURN QUERY
UPDATE Flights
SET status = 'Відбувся'
WHERE endtime < CURRENT_TIMESTAMP AND status = 'Очікується'
RETURNING *;
END;
$$ LANGUAGE plpgsql;

```

	flight_id	idfirm	idtransport	from_city	to_city	distance	timestart	timeend	flight_status
1	2	10	57	Херсон	Миколаїв	80	2021-05-08 14:01:34.000000	2021-05-08 15:37:34.000000	Відбувся
2	6	2	5	Київ	Тирана	1300	2021-01-08 21:21:34.000000	2021-01-08 22:58:34.000000	Відбувся
3	7	3	15	Рівне	Монако	1631	2021-04-21 00:32:33.000000	2021-04-21 02:34:33.000000	Відбувся
4	13	8	60	Ужгород	Чернівці	432	2021-03-02 15:16:33.000000	2021-03-02 23:54:33.000000	Відбувся
5	27	20	40	Вінниця	Чернівці	411	2021-01-15 00:13:26.000000	2021-01-15 06:32:26.000000	Відбувся
6	28	6	16	Кропивницький	Амстердам	1985	2021-03-12 08:58:18.000000	2021-03-12 11:26:18.000000	Відбувся
7	29	1	17	Луцьк	Лобляна	950	2021-02-23 19:18:07.000000	2021-02-23 20:29:07.000000	Відбувся
8	33	2	5	Хмельницький	Париж	1788	2021-01-02 08:07:23.000000	2021-01-02 10:21:23.000000	Відбувся
9	34	1	11	Луганськ	Суми	267	2021-02-27 13:29:13.000000	2021-02-27 13:49:13.000000	Відбувся
10	35	6	15	Суми	Гельсінкі	1214	2021-03-15 02:15:15.000000	2021-03-15 03:46:15.000000	Відбувся
11	36	1	1	Донецьк	Ужгород	1333	2021-04-17 00:03:44.000000	2021-04-17 01:42:44.000000	Відбувся
12	37	2	8	Харків	Бухарест	976	2021-03-03 03:42:24.000000	2021-03-03 04:55:24.000000	Відбувся
13	38	9	54	Івано-Франківськ	Суми	1007	2021-03-16 08:17:25.000000	2021-03-17 04:25:25.000000	Відбувся
14	39	21	49	Херсон	Київ	545	2021-01-27 19:51:46.000000	2021-01-28 04:14:46.000000	Відбувся
15	40	1	14	Ужгород	Кишинів	673	2021-03-16 12:28:59.000000	2021-03-16 13:18:59.000000	Відбувся
16	41	8	58	Херсон	Київ	612	2021-01-12 18:07:00.000000	2021-01-13 06:21:00.000000	Відбувся
17	43	17	34	Тернопіль	Харків	897	2021-02-22 01:45:39.000000	2021-02-22 15:33:39.000000	Відбувся
18	44	1	7	Полтава	Софія	1101	2021-01-14 01:33:36.000000	2021-01-14 02:55:36.000000	Відбувся
19	45	19	18	Миколаїв	Київ	479	2021-02-04 15:28:19.000000	2021-02-04 22:50:19.000000	Відбувся
20	46	16	23	Житомир	Івано-Франківськ	419	2021-03-09 18:59:38.000000	2021-03-10 01:25:38.000000	Відбувся
21	49	10	53	Кропивницький	Миколаїв	182	2021-03-10 02:42:48.000000	2021-03-10 06:20:48.000000	Відбувся
22	50	3	4	Хмельницький	Берн	1444	2021-03-19 11:52:21.000000	2021-03-19 13:40:21.000000	Відбувся
23	51	1	3	Київ	Осло	1630	2021-01-02 18:36:57.000000	2021-01-02 20:38:57.000000	Відбувся

Рисунок 3.46 – Результат виконання функції

Створено перевантажену функцію, яка повертає дату або масив дат відправлення введеного рейсу з заданого часового проміжку, або без нього. (рисунок 3.47, рисунок 3.48):

```

CREATE OR REPLACE FUNCTION getStartTime(TEXT, TEXT)

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

RETURNS TEXT
AS $$
    SELECT string_agg(starttime::TEXT, ', ')
    FROM Flights
    WHERE f_from = $1 AND f_to = $2
    ORDER BY 1
    LIMIT 1;
$$ LANGUAGE SQL IMMUTABLE;

CREATE OR REPLACE FUNCTION getStartTime(TEXT, TEXT, DATE, DATE)
RETURNS TEXT
AS $$
    SELECT string_agg(starttime::TEXT, ', ')
    FROM Flights
    WHERE f_from = $1 AND f_to = $2 AND (starttime BETWEEN $3::TIMESTAMP
AND $4::TIMESTAMP)
    ORDER BY 1
    LIMIT 1;
$$ LANGUAGE SQL IMMUTABLE;

SELECT f_from, f_to, starttime AS from_table, getStartTime(f_from, f_to,
'2021-01-01', '2021-09-01') AS from_function
FROM Flights
ORDER BY 1;

SELECT f_from, f_to, starttime AS from_table, getStartTime(f_from, f_to,
'2021-01-01', '2021-09-01') AS from_function
FROM Flights

LIMIT 3;

```

	f_from	f_to	from_table	from_function
1	Вінниця	Чернірів	2021-01-15 00:13:26.000000	2021-01-15 00:13:26
2	Дніпро	Луцьк	2021-10-21 07:47:48.000000	<null>
3	Дніпро	Харків	2021-10-24 19:04:23.000000	<null>
4	Донецьк	Мінськ	2021-11-04 00:17:22.000000	<null>
5	Донецьк	Андорра-ла-Велья	2021-07-03 11:36:39.000000	2021-07-03 11:36:39
6	Донецьк	Ужгород	2021-04-17 00:03:44.000000	2021-04-17 00:03:44
7	Житомир	Люксембург	2021-10-20 17:46:29.000000	<null>
8	Житомир	Івано-Франківськ	2021-07-21 12:52:50.000000	2021-07-21 12:52:50, 2021-03-09 18:59:38
9	Житомир	Івано-Франківськ	2021-03-09 18:59:38.000000	2021-07-21 12:52:50, 2021-03-09 18:59:38

Рисунок 3.47 – Результат виконання функції з двома аргументами

	f_from	f_to	from_table	from_function
1	Київ	Дніпро	2021-10-24 21:22:46.000000	<null>
2	Херсон	Миколаїв	2021-05-08 14:01:34.000000	2021-05-08
3	Київ	Луганськ	2021-08-09 20:18:56.000000	2021-08-09

Рисунок 3.48 – Результат виконання функції з чотирма аргументами

Створено перевантажену функцію, яка формує графік відправлення рейсів між обласними центрами України (рисунок 3.49, рисунок 3.50, рисунок 3.51, рисунок 3.52):

```

CREATE OR REPLACE FUNCTION timeTable()
RETURNS TABLE(Cities VARCHAR(45), "Івано-Франківськ" TEXT, Вінниця TEXT,
Дніпро TEXT, Донецьк TEXT, Житомир TEXT, Запоріжжя TEXT,

```



```

        Київ TEXT, Кропивницький TEXT, Луганськ
TEXT, Луцьк TEXT, Львів TEXT,
        Миколаїв TEXT, Одеса TEXT, Полтава TEXT,
Рівне TEXT, Суми TEXT,
        Тернопіль TEXT, Ужгород TEXT, Харків TEXT,
Херсон TEXT, Хмельницький TEXT,
        Черкаси TEXT, Чернівці TEXT, Чернігів TEXT)
AS $$
BEGIN
    RETURN QUERY
    SELECT DISTINCT city_name, getStartTime(city_name,'Івано-
Франківськ') AS "Івано-Франківськ",
        getStartTime(city_name,'Вінниця') AS "Вінниця",
getStartTime(city_name,'Дніпро') AS "Дніпро",
        getStartTime(city_name,'Донецьк') AS "Донецьк",
        getStartTime(city_name,'Житомир') AS "Житомир",
getStartTime(city_name,'Запоріжжя') AS "Запоріжжя",
        getStartTime(city_name,'Київ') AS "Київ",
getStartTime(city_name,'Кропивницький') AS "Кропивницький",
        getStartTime(city_name,'Луганськ') AS "Луганськ",
getStartTime(city_name,'Луцьк') AS "Луцьк",
        getStartTime(city_name,'Львів') AS "Львів",
getStartTime(city_name,'Миколаїв') AS "Миколаїв",
        getStartTime(city_name,'Одеса') AS "Одеса",
getStartTime(city_name,'Полтава') AS "Полтава",
        getStartTime(city_name,'Рівне') AS "Рівне",
getStartTime(city_name,'Суми') AS "Суми",
        getStartTime(city_name,'Тернопіль') AS "Тернопіль",
getStartTime(city_name,'Ужгород') AS "Ужгород",
        getStartTime(city_name,'Харків') AS "Харків",
getStartTime(city_name,'Херсон') AS "Херсон",
        getStartTime(city_name,'Хмельницький') AS
"Хмельницький", getStartTime(city_name,'Черкаси') AS "Черкаси",
        getStartTime(city_name,'Чернівці') AS "Чернівці",
getStartTime(city_name,'Чернігів') AS "Чернігів"
    FROM Flights LEFT JOIN Flights_has_Cities FhC on Flights.flight_ID =
FhC.idFlight
    INNER JOIN Cities C on C.city_ID = FhC.idCity1 OR C.city_ID =
FhC.idCity2
    WHERE country = 'Україна'
    ORDER BY 1;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION timeTable(DATE, DATE)
RETURNS TABLE(Cities VARCHAR(45), "Івано-Франківськ" TEXT, Вінниця TEXT,
Дніпро TEXT , Донецьк TEXT, Житомир TEXT, Запоріжжя TEXT,
        Київ TEXT, Кропивницький TEXT, Луганськ
TEXT, Луцьк TEXT, Львів TEXT,
        Миколаїв TEXT, Одеса TEXT, Полтава TEXT,
Рівне TEXT, Суми TEXT,
        Тернопіль TEXT, Ужгород TEXT, Харків TEXT,
Херсон TEXT, Хмельницький TEXT,
        Черкаси TEXT, Чернівці TEXT, Чернігів TEXT)
AS $$
BEGIN
    IF $1 > $2
        THEN RAISE 'ERROR: timeTable(N DATE,M DATE); N must be <= M';
    ELSE RETURN QUERY
        SELECT DISTINCT city_name, getStartTime(city_name,'Івано-
Франківськ', $1, $2) AS "Івано-Франківськ",
            getStartTime(city_name,'Вінниця', $1, $2) AS
"Вінниця", getStartTime(city_name,'Дніпро', $1, $2) AS "Дніпро",

```

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

        getStartTime(city_name,'Донецьк', $1, $2) AS
"Донецьк",
        getStartTime(city_name,'Житомир', $1, $2) AS
"Житомир", getStartTime(city_name,'Запоріжжя', $1, $2) AS "Запоріжжя",
        getStartTime(city_name,'Київ', $1, $2) AS "Київ",
getStartTime(city_name,'Кропивницький', $1, $2) AS "Кропивницький",
        getStartTime(city_name,'Луганськ', $1, $2) AS
"Луганськ", getStartTime(city_name,'Луцьк', $1, $2) AS "Луцьк",
        getStartTime(city_name,'Львів', $1, $2) AS "Львів",
getStartTime(city_name,'Миколаїв', $1, $2) AS "Миколаїв",
        getStartTime(city_name,'Одеса', $1, $2) AS "Одеса",
getStartTime(city_name,'Полтава', $1, $2) AS "Полтава",
        getStartTime(city_name,'Рівне', $1, $2) AS "Рівне",
getStartTime(city_name,'Суми', $1, $2) AS "Суми",
        getStartTime(city_name,'Тернопіль', $1, $2) AS
"Тернопіль", getStartTime(city_name,'Ужгород', $1, $2) AS "Ужгород",
        getStartTime(city_name,'Харків', $1, $2) AS
"Харків", getStartTime(city_name,'Херсон', $1, $2) AS "Херсон",
        getStartTime(city_name,'Хмельницький', $1, $2) AS
"Хмельницький", getStartTime(city_name,'Черкаси', $1, $2) AS "Черкаси",
        getStartTime(city_name,'Чернівці', $1, $2) AS
"Чернівці", getStartTime(city_name,'Чернігів', $1, $2) AS "Чернігів"
FROM Flights LEFT JOIN Flights_has_Cities FhC on
Flights.flight_ID = FhC.idFlight
INNER JOIN Cities C on C.city_ID = FhC.idCity1 OR C.city_ID =
FhC.idCity2
WHERE country = 'Україна'
ORDER BY 1;
END IF;
END;
$$ LANGUAGE plpgsql;

```

cities	"Івано-Франківськ"	"Вінниця"	"Дніпро"	"Донецьк"	"Житомир"	"Запоріжжя"
1 Вінниця	<null>	<null>	<null>	<null>	<null>	<null>
2 Дніпро	<null>	<null>	<null>	<null>	<null>	<null>
3 Донецьк	<null>	<null>	<null>	<null>	<null>	<null>
4 Житомир	2021-07-21 12:52:50, 2021-03-09 18:59:38	<null>	<null>	<null>	<null>	<null>
5 Запоріжжя	2021-08-08 13:54:20	<null>	<null>	<null>	<null>	<null>
6 Івано-Франківськ	<null>	2021-09-12 05:25:12	<null>	<null>	<null>	<null>
7 Київ	<null>	<null>	2021-10-24 21:22:46	<null>	<null>	<null>
8 Кропивницький	<null>	<null>	<null>	<null>	<null>	<null>
9 Луганськ	<null>	<null>	<null>	<null>	<null>	<null>
10 Луцьк	<null>	<null>	<null>	<null>	<null>	<null>
11 Львів	<null>	<null>	<null>	<null>	<null>	<null>
12 Миколаїв	<null>	<null>	<null>	<null>	<null>	<null>
13 Одеса	<null>	<null>	<null>	<null>	<null>	<null>
14 Полтава	<null>	<null>	<null>	<null>	<null>	<null>
15 Рівне	<null>	<null>	<null>	<null>	<null>	<null>
16 Суми	2021-09-12 11:39:37	<null>	<null>	<null>	<null>	<null>
17 Тернопіль	<null>	<null>	<null>	<null>	<null>	<null>
18 Ужгород	<null>	<null>	<null>	<null>	<null>	<null>
19 Харків	<null>	<null>	<null>	<null>	<null>	<null>
20 Херсон	<null>	<null>	<null>	<null>	<null>	<null>
21 Хмельницький	<null>	<null>	<null>	<null>	<null>	<null>
22 Черкаси	<null>	<null>	<null>	<null>	<null>	<null>
23 Чернівці	<null>	<null>	<null>	<null>	<null>	<null>
24 Чернігів	<null>	<null>	<null>	<null>	<null>	<null>

Рисунок 3.49 – Результат виконання функції

cities	"Київ"	"Кропивницький"	"Луганськ"	"Луцьк"	"Львів"
1 Вінниця	<null>	<null>	<null>	<null>	<null>
2 Дніпро	<null>	<null>	<null>	2021-10-21 07:47:48	<null>
3 Донецьк	<null>	<null>	<null>	<null>	<null>
4 Житомир	<null>	<null>	<null>	<null>	<null>
5 Запоріжжя	<null>	<null>	<null>	<null>	<null>
6 Івано-Франківськ	<null>	<null>	<null>	<null>	<null>
7 Київ	<null>	<null>	2021-08-09 20:18:56	<null>	<null>
8 Кропивницький	<null>	<null>	<null>	<null>	<null>
9 Луганськ	<null>	<null>	<null>	<null>	<null>
10 Луцьк	<null>	<null>	<null>	<null>	<null>
11 Львів	<null>	<null>	<null>	<null>	<null>
12 Миколаїв	2021-02-04 15:28:19	<null>	<null>	<null>	<null>
13 Одеса	<null>	<null>	<null>	<null>	<null>
14 Полтава	<null>	<null>	<null>	<null>	<null>
15 Рівне	<null>	<null>	2021-09-13 14:36:39	<null>	<null>
16 Суми	<null>	<null>	<null>	<null>	<null>
17 Тернопіль	<null>	<null>	<null>	<null>	<null>
18 Ужгород	<null>	<null>	<null>	<null>	<null>
19 Харків	<null>	<null>	<null>	<null>	<null>
20 Херсон	2021-01-27 19:51:46, 2021-01-12 18:07:00	<null>	<null>	<null>	<null>
21 Хмельницький	<null>	<null>	2021-09-15 14:12:39	<null>	<null>
22 Черкаси	<null>	<null>	<null>	<null>	2021-12-04 13:10:36
23 Чернівці	<null>	<null>	<null>	<null>	<null>
24 Чернігів	<null>	2021-07-08 10:02:28	<null>	<null>	<null>

Рисунок 3.50 – Результат виконання функції

cities	"Миколаїв"	"Одеса"	"Полт..."	"Рівне"	"Суми"	"Терн..."	"Ужгород"
1 Вінниця	<null>	<null>	<null>	<null>	<null>	<null>	<null>
2 Дніпро	<null>	<null>	<null>	<null>	<null>	<null>	<null>
3 Донецьк	<null>	<null>	<null>	<null>	<null>	<null>	2021-04-17 00:03:44
4 Житомир	<null>	<null>	<null>	<null>	<null>	<null>	<null>
5 Запоріжжя	<null>	<null>	<null>	<null>	<null>	<null>	<null>
6 Івано-Франківськ	<null>	<null>	<null>	<null>	2021-03-16 08:17:25	<null>	<null>
7 Київ	<null>	<null>	<null>	<null>	<null>	<null>	<null>
8 Кропивницький	2021-03-10 02:42:48	<null>	<null>	<null>	<null>	<null>	<null>
9 Луганськ	<null>	<null>	<null>	<null>	2021-02-27 13:29:13	<null>	<null>
10 Луцьк	<null>	<null>	<null>	<null>	<null>	<null>	<null>
11 Львів	<null>	2021-09-23 10:00:51	<null>	<null>	<null>	<null>	<null>
12 Миколаїв	<null>	<null>	<null>	<null>	<null>	<null>	<null>
13 Одеса	<null>	<null>	<null>	<null>	<null>	<null>	<null>
14 Полтава	2021-09-05 16:22:17	<null>	<null>	<null>	<null>	<null>	<null>
15 Рівне	<null>	<null>	<null>	<null>	<null>	<null>	<null>
16 Суми	<null>	<null>	<null>	<null>	<null>	<null>	<null>
17 Тернопіль	<null>	<null>	<null>	2021-07-10 19:58:31	<null>	<null>	<null>
18 Ужгород	<null>	<null>	<null>	<null>	<null>	<null>	<null>
19 Харків	<null>	<null>	<null>	<null>	<null>	<null>	<null>
20 Херсон	2021-05-08 14:01:34	<null>	<null>	<null>	<null>	<null>	<null>
21 Хмельницький	<null>	<null>	<null>	<null>	<null>	<null>	<null>
22 Черкаси	<null>	<null>	<null>	<null>	<null>	<null>	<null>
23 Чернівці	<null>	<null>	<null>	<null>	<null>	<null>	<null>
24 Чернігів	<null>	<null>	<null>	<null>	<null>	<null>	<null>

Рисунок 3.51 – Результат виконання функції

	cities	"Харків"	"Херсон"	"Хмельницький"	"Черкаси"	"Чернівці"	"Чернігів"
1	Вінниця	<null>	<null>	<null>	<null>	<null>	2021-01-15 00:13:26
2	Дніпро	2021-10-24 19:04:23	<null>	<null>	<null>	<null>	<null>
3	Донецьк	<null>	<null>	<null>	<null>	<null>	<null>
4	Житомир	<null>	<null>	<null>	<null>	<null>	<null>
5	Запоріжжя	<null>	<null>	<null>	<null>	<null>	<null>
6	Івано-Франківськ	<null>	<null>	<null>	<null>	<null>	<null>
7	Київ	<null>	<null>	<null>	<null>	<null>	<null>
8	Кропивницький	<null>	<null>	<null>	<null>	<null>	<null>
9	Луганськ	<null>	<null>	<null>	<null>	<null>	<null>
10	Луцьк	<null>	<null>	<null>	<null>	<null>	<null>
11	Львів	2021-09-05 01:47:01	<null>	<null>	<null>	<null>	<null>
12	Миколаїв	<null>	<null>	2021-06-23 10:10:45	<null>	<null>	<null>
13	Одеса	<null>	<null>	<null>	<null>	<null>	<null>
14	Полтава	<null>	<null>	<null>	<null>	<null>	<null>
15	Рівне	<null>	<null>	<null>	<null>	<null>	<null>
16	Суми	<null>	<null>	<null>	<null>	<null>	<null>
17	Тернопіль	2021-02-22 01:45:39	<null>	<null>	<null>	<null>	<null>
18	Ужгород	<null>	<null>	<null>	<null>	2021-03-02 15:16:33	<null>
19	Харків	<null>	<null>	<null>	<null>	<null>	<null>
20	Херсон	<null>	<null>	<null>	<null>	<null>	<null>
21	Хмельницький	<null>	<null>	<null>	<null>	<null>	<null>
22	Черкаси	<null>	<null>	<null>	<null>	<null>	<null>
23	Чернівці	<null>	<null>	<null>	<null>	<null>	<null>
24	Чернігів	<null>	<null>	<null>	<null>	<null>	<null>

Рисунок 3.52 – Результат виконання функції

Результат виконання функції з двома аргументами наведено на рисунку 3.53. Для зручності перегляду було приховано стовпці, які містили тільки null значення.

	cities	"Київ"	"Суми"	"Харків"	"Чернігів"
1	Вінниця	<null>	<null>	<null>	2021-01-15 00:13:26
2	Дніпро	<null>	<null>	<null>	<null>
3	Донецьк	<null>	<null>	<null>	<null>
4	Житомир	<null>	<null>	<null>	<null>
5	Запоріжжя	<null>	<null>	<null>	<null>
6	Івано-Франківськ	<null>	<null>	<null>	<null>
7	Київ	<null>	<null>	<null>	<null>
8	Кропивницький	<null>	<null>	<null>	<null>
9	Луганськ	<null>	2021-02-27 13:29:13	<null>	<null>
10	Луцьк	<null>	<null>	<null>	<null>
11	Львів	<null>	<null>	<null>	<null>
12	Миколаїв	2021-02-04 15:28:19	<null>	<null>	<null>
13	Одеса	<null>	<null>	<null>	<null>
14	Полтава	<null>	<null>	<null>	<null>
15	Рівне	<null>	<null>	<null>	<null>
16	Суми	<null>	<null>	<null>	<null>
17	Тернопіль	<null>	<null>	2021-02-22 01:45:39	<null>
18	Ужгород	<null>	<null>	<null>	<null>
19	Харків	<null>	<null>	<null>	<null>
20	Херсон	2021-01-27 19:51:46, 2021-01-12 18:07:00	<null>	<null>	<null>
21	Хмельницький	<null>	<null>	<null>	<null>
22	Черкаси	<null>	<null>	<null>	<null>
23	Чернівці	<null>	<null>	<null>	<null>
24	Чернігів	<null>	<null>	<null>	<null>

Рисунок 3.53 – Результат виконання функції з двома аргументами

### 3.6 Додавання користувачів та надання їм прав

Команда CREATE USER є просто синонімом CREATE ROLE . Єдина відмінність в тому, що для команди, записаної у вигляді CREATE USER, за замовчуванням мається на увазі LOGIN, а у вигляді CREATE ROLE мається на увазі NOLOGIN.

Оператор CREATE USER є розширенням PostgreSQL . У стандарті SQL визначення користувачів вважається залежним від реалізації. [7]

У результаті аналізу завдання на курсове проектування було створено наступних користувачів.

Створено користувач, який має всі права для бази Tickets\_booking (рисуюнок 3.54):

```
CREATE USER Developer WITH PASSWORD 'Developer123';  
GRANT ALL PRIVILEGES ON DATABASE Tickets_booking TO Developer;
```

	username	usesysid	usecreatedb	usesuper	userepl	usebypassrls	passwd	valuntil	useconfig
1	developer	16839	false	false	false	false	md5c95109178c0...	<null>	<null>

Рисуюнок 3.54 – Результат створення користувача

Створено користувач, який має доступ до редагування таблиць та створення представлень для бази Tickets\_booking (рисуюнок 3.55):

```
CREATE USER Administrator WITH PASSWORD 'Administrator123';  
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO Administrator;
```

	username	usesysid	usecreatedb	usesuper	userepl	usebypassrls	passwd	valuntil	useconfig
1	administrator	16838	false	false	false	false	md535922745...	<null>	<null>

Рисуюнок 3.55 – Результат створення користувача

Створено користувач, який має доступ до вибірки даних з всіх таблиць бази Tickets\_booking (рисуюнок 3.56):

```
CREATE USER Customer WITH PASSWORD 'Customer123';  
GRANT SELECT ON * TO Customer;
```

	username	usesysid	usecreatedb	usesuper	userepl	usebypassrls	passwd	valuntil	useconfig
1	customer	16835	false	false	false	false	md5631b7a76...	<null>	<null>

Рисуюнок 3.56 – Результат створення користувача

Повний код створеної бази даних у СУБД PostgreSQL наведено у додатку Г.

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

## ВИСНОВКИ

Під час виконання курсового проєкту було засвоєно основні тенденції та перспективи розвитку систем керування базами даних. Створено базу даних на основі предметної області “Резервування квитків” у СКБД MySQL та PostgreSQL, з урахуванням особливостей і переваг кожної з них. Бази даних складатися з вісімнадцяти таблиць, що містять різноманітні обмеження і зв’язані за допомогою первинних і зовнішніх ключів та чітко описують дану структуру. Для роботи з базою даних головні таблиці заповнено більш ніж 50 записами.

Для маніпуляцій даними використовувались SQL запити категорії DML – INSERT, UPDATE, DELETE.

Для даної предметної області створено 10 запитів, 10 тригерів, 5 представлень, 3 користувача у обох СКБД, 7 процедур, 3 функції в MySQL та 10 функцій в PostgreSQL. Набуто навички щодо використання різноманітних функцій, операторів та нових можливостей обох СКБД.

Робота над даним проєктом дала змогу познайомитися із способами створення баз даних і таблиць, їх заповненням, виведенням і обробкою записів. Розглянуто вбудовані функції, транзакції, збережені процедури, тригери, курсори, цикли, динамічні запити. Опрацьовано способи забезпечення цілісності та безпеки даних.

Заповнення здійснювалося за допомогою транзакцій. Працюючи із транзакціями було прослідковано функціональність та корисність даної опції. Заповнення чи то видалення даних у базі даних значно полегшело використання саме транзакцій.

При розробці запитів опрацьовано усі функції для полегшення роботи із БД, а також продемонстровано різноманітні можливості використання тої чи іншої операції. Було створено запити на об’єднання SELF, LEFT, RIGHT, INNER JOIN та відстежено користь застосування того чи іншого випадку об’єднання. Також було співставлено можливості обох середовищ і зроблено

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		89

висновок щодо суттєвих відмінностей між ними. Для забезпечення роботи із значенням року в даті у СУБД PostgreSQL було використано оператор EXTRACT, що полегшив доступ саме року із дати. У MySQL натомість служить оператор YEAR, чого немає у PostgreSQL.

Використання збережених процедур забезпечило можливість опрацювання та модифікації даних. Дана опція дала можливість об'єднати послідовність однакових запитів і зберегти їх на сервері.

Розглянуто функції у PostgreSQL, за допомогою яких значно легше отримати доступ до потрібних даних. Також простежено різні варіанти щодо повернення даних функцією, що говорить про функціональність даного інструменту.

Для того щоб передбачити певні умови щодо валідації даних, що будуть вводитися, застосовано тригери. Передбачено усі можливості неправильного або не можливого способу модифікації даних за допомогою реалізації тригерів для заданої предметної області.

Для збереження тимчасових та наочно потрібних даних використано представлення, які навідміну від простих запитів зберігають дані. У СКБД PostgreSQL було створено матеріалізовані представлення, основною перевагою і відмінністю який є те, що дані які вони містять можна оновлювати.

Для створення бази даних обрано метод низхідного проєктування та відображено її у вигляді ER-діаграми, яка надалі була приведена до третьої нормальної форми.

Отже, при розробці курсового проєкту було засвоєно велику кількість навиків щодо роботи із базами даних, опрацьовано нові можливості DataGrip при створенні та керуванні базою даних. Під час розроблення курсового проєкту закріплено і поглиблено знання, пов'язані з принципами побудови та технологією проєктування баз даних у СУБД MySQL і PostgreSQL.

					КП.ПІ-18-01.05.02.00.000 ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Освітньо-професійна програма «Інженерія програмного забезпечення» від 31 серпня 2020 року. – URL: <http://ifkepnung.if.ua/>
2. Пітчук Л.В. Курсове та дипломне проєктування. Методичні вказівки. – URL: <http://ifkepnung.if.ua/>
3. Левицький І.В. , Шевчук О.В. Методичні вказівки для виконання курсових проєктів з дисципліни "Бази даних (Частина II)" для студентів денної та заочної форм навчання за напрямом підготовки 12 – Інформаційні технології, спеціальністю 121 – Інженерія програмного забезпечення, 2020.- 35 с.
4. Бази даних – Wikipedia. [Електронний ресурс]. – URL: <https://uk.wikipedia.org/wiki/>
5. Habr - collaborative blog about IT. [Електронний ресурс]. – URL: <https://habr.com/en/>
6. Довідник про MySQL. [Електронний ресурс]. – URL: <http://www.e-helper.com.ua/>
7. Документація до PostgreSQL 11.6. The PostgreSQL Global Development Group: Компанія «Постгрес Професійний». [Електронний ресурс]. – URL: <https://postgrespro.en/docs/postgresql>
8. Гайна Г. А. Основи проєктування баз даних : навч. посіб. для вищ. навч. закладів / Г. А. Гайна. – Київ : Кондор, 2018. – 202 с.
9. MySQL: науковий посібник / Люк Веллінг, Лора То, 2016. – 304 с.
10. Євгеній Моргунов, PostgreSQL. Основи мови SQL: навчальний посібник, 2019. – 336с.