НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії
(повна назва кафедри)

КУРСОВА РОБОТА

з дисципліни «Бази даних»
(назва дисципліни)

на тему: База даних для бронювання та пошуку авіаквитків

Студента __2__ курсу _ІП-15_ групи
спеціальності 121 «Інженерія програмного забезпечення»
_Плугатирьова Дмитра Валерійовича__
(прізвище та ініціали)

Керівник _Марченко Олена Іванівна_
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2023 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс 2 Група ІП-15 Семестр 3

З А В Д А Н Н Я
НА КУРСОВУ РОБОТУ СТУДЕНТУ

Плугатирьову Дмитру Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи База даних для бронювання та пошуку авіаквитків

керівник роботи Марченко Олена Іванівна

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 14.01.2023

3. Вихідні дані до роботи база даних, котра містить інформацію про систему авіарейсів. Тобто, рейси, літаки котрі беруть участь у них, білети на ці рейси, користувачі та інші дані, які є супутніми до вище наведених

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз предметного середовища
2) Побудова ER-моделі
3) Побудова реляційної схеми з ER-моделі
4) Створення бази даних, у форматі обраної системи управління базою даних
5) Створення користувачів бази даних
6) Імпорт даних з використанням засобів СУБД в створену базу даних
7) Створення мовою SQL запитів
8) Оптимізація роботи запитів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 08.11.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	24.12.2022	
2	Побудова ER-моделі	24.12.2022	
3	Побудова реляційної схеми з ER-моделі	25.12.2022	
4	Створення бази даних, у форматі обраної системи управління базою даних	03.01.2023	
5	Створення користувачів бази даних	13.01.2023	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	04.01.2023	
7	Створення мовою SQL запитів	04.01.2023	
8	Оптимізація роботи запитів	13.01.2023	
9	Оформлення пояснювальної записки	13.01.2023	
10	Захист курсової роботи	14.01.2023	

Студент _____ _____Плугатирьов Д.В._____
(підпис) (прізвище та ініціали)

Керівник роботи _____ ___Марченко О.І._____
(підпис) (прізвище та ініціали)

Оглавление

ВСТУП

Дана курсова робота присвячена розробці бази даних та запитів до неї, у ході чого буде проаналізовано створені можливості.

Ціллю роботи є розробка бази даних «Бронювання та пошук авіаквитків» та забезпечення можливістю пошуку авіаквитків у базі даних з використанням запитів.

Загальною ціллю курсової роботи є покращення навичок проектування, реалізації баз даних під поставлену задачу, уміння її використовувати та писати скрипти.

1 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА

Необхідно розробити систему бронювання та пошуку квитків. Система має кілька видів квитків та, відповідно, місць у літаку.

Кожен «квиток» характеризується наступними параметрами:

- ідентифікатор;
- статус;
- клас;
- ціна;
- політ;
- дата польоту.

«Статус» квитка:

- ідентифікатор;
- назва.

«Клас квитка» має наступні характеристики:

- ідентифікатор;
- назва.

«Рейс» має наступні характеристики:

- назва;
- відправна точка;
- призначення;
- час відправлення;
- час прибуття.

«Літак» має наступні характеристики:

- ідентифікатор;
- дата виготовлення;

- к-сть vip сидінь;
- к-сть звичайних сидінь.

Для кожного рейсу квитки створюються працівниками за 3 місяці до дати відправлення та можуть видалятися/додаватися, якщо можливо та необхідно.

Рейс може мати багато літаків, а до літака може бути призначено багато рейсів.

Один і той самий літак не може записатися на один і той самий рейс кілька разів. Коли літак досягає 20-ти річного віку, то він не може бути задіяний у рейсах.

«Літак до рейсу» має наступні характеристики:

- літак;
- рейс.

База даних містить зареєстрованих користувачів.

Про кожного «користувача» в базу заносяться наступні дані:

- особа;
- електронна адреса;
- номер телефону;
- к-сть доларів;
- номер кредитної картки;
- місто;
- країна;
- номер паспорту;
- країна видачі паспорту.

«Особа» складається із:

- ідентифікатор;

- ім'я;
- прізвище;
- рік народження.

«Кредитна картка» складається із:

- номер;
- дата сплину строку придатності;
- дата створення.

«Паспорт» складається із:

- номер;
- країна видачі;
- дата закінчення дії;
- національність.

Користувачі системи бронювання та пошуку квитків мають здатність шукати наявні рейси для певного призначення на певну дату і переглядати не куплені квитки до них. Після знаходження і вибору бажаного польоту, користувач може бронювати квиток.

Після затвердження факту «бронювання» користувачем квитку, буде занесено відповідний запис до бази даних, який має наступні складові:

- користувач;
- квиток;
- поточна дата.

За відміни користувачем бронювання, цей запис видаляється.

Передбачити наступні вимоги щодо інформації в системі:

1. Квиток може бути не куплений. В такому разі, після старту польоту, його видаляє із бази даних робітник.
2. Вік користувача не може бути меншим за 18 років.

3. Кожен користувач за реєстрації має вказати або телефонний номер, або адресу електронної пошти для підтримки зв’язку.
4. Кількість квитків на літак обмежена кількістю наявних місць.
5. Користувач може заплатити за квиток лише за наявності необхідної кількості грошей на рахунку.

Із даною інформаційною системою мають працювати наступні групи користувачів:

1. Адміністратор.
2. Працівник.
3. Користувач.

Під час роботи із системою користувач повинен мати змогу виконати наступні дії:

1. Дізнатися інформацію про існуючі квитки із можливістю фільтрації результатів запиту за датою польоту та точками відльоту/прильоту.
2. Дізнатися про кількість наявних звичайних, vip-квитків на літак та його дату виготовлення.
3. Отримати історію куплених квитків користувача, надавши номер його паспорту.
4. Забронювати квиток, якщо має достатньо доларів на рахунку.
5. Поповнити рахунок.
6. Відмінити бронювання (якщо до польоту залишилось менше 7 годин, то треба платити відшкодування у 30%).
7. Отримати інформацію про місця на вільний літак.

Під час роботи із системою працівник повинен мати змогу робити те, що користувач та вирішити наступні задачі:

1. Отримати дані користувача з можливістю фільтрації результату за будь-яким із полів.

2. Відмінити бронювання, замовити квиток, виконати операції CUD за вимогою користувача над його даними.
3. Видалити незабронювані квитки за необхідності.
4. Видалити користувача за виявлення певних порушень з його боку.
5. Виконати операції CD над рейсами до літаків та CRUD над даними самих літаків та рейсів.
6. Видалити не куплені квитки після сплину дати відправлення літака.
7. Подивитися, чи прострочені літаки та які саме.
8. Дізнатися про користувачів, у яких закінчився термін придатності паспорта.

Під час роботи із системою адміністратор повинен мати змогу робити те, що працівник та вирішити наступні задачі:

1. Дізнатися інформацію про популярність рейсів.
2. Отримати дані про літаки, яким не призначені рейси.
3. Дізнатися інформацію про к-сть куплених vip-квитків у літаках на певних рейсах.
4. Дізнатися кількість громадян відповідних країн, котрі зареєстровані у системі авіакомпанії.
5. Отримати дані про цінову політику рейсів.
6. Отримати дані про користувачів, які ніколи не купували vip-квитки.
7. Дізнатися про аеропорти, куди літаки літають найчастіше.
8. Дізнатися кількість проданих квитків відносно дат відправлення літаків.

2 ПОБУДОВА ER-МОДЕЛІ

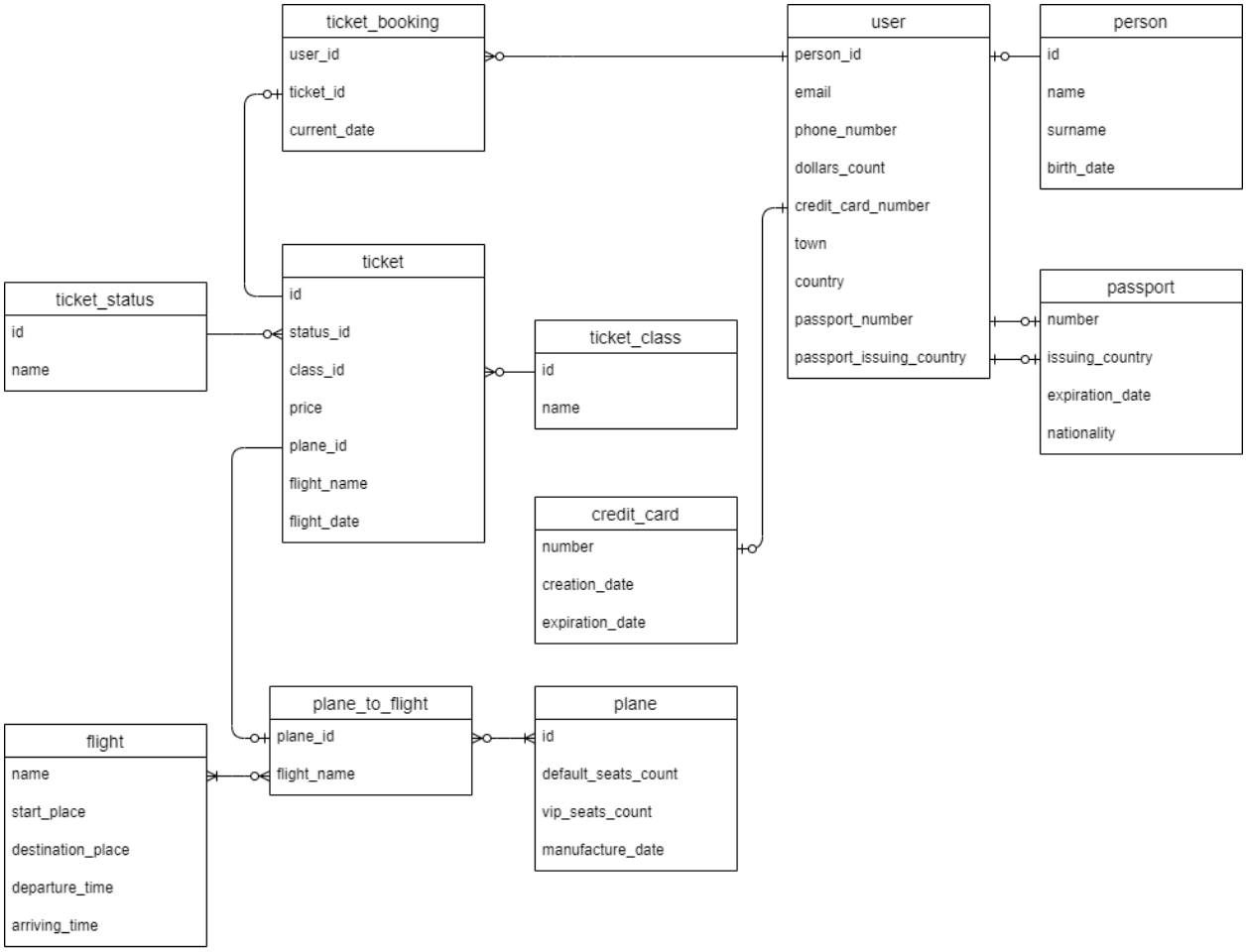


Рисунок 2.1 – ER-діаграма

3 ПОБУДОВА РЕЛЯЦІЙНОЇ СХЕМИ З ER-МОДЕЛІ

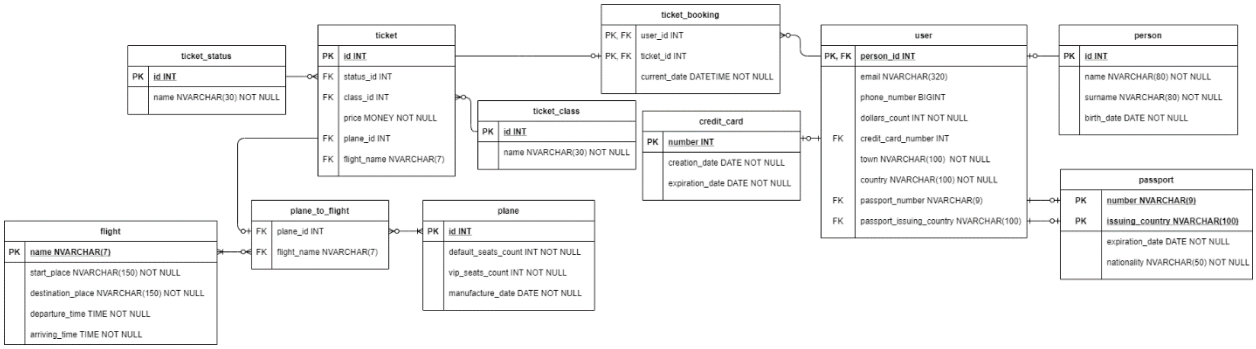


Рисунок 3.1 - реляційна схема

4 СТВОРЕННЯ БАЗИ ДАНИХ, У ФОРМАТІ ОБРАНОЇ СИСТЕМИ УПРАВЛІННЯ БАЗОЮ ДАНИХ

Створення БД

```
USE master;

CREATE DATABASE air_tickets_search_and_booking;

SELECT name, size, size*1.0/128 AS [Size in MBs]
FROM sys.master_files
     WHERE name = N'air_tickets_search_and_booking';
```

Створення таблиць

```
USE air_tickets_search_and_booking;

CREATE TABLE person
(
     id INT IDENTITY(1,1) PRIMARY KEY,
     [name] NVARCHAR(80) NOT NULL,
     surname NVARCHAR(80) NOT NULL,
     birth_date DATE NOT NULL,
     CONSTRAINT CK_person_is_adult
          CHECK (DATEDIFF(YEAR, birth_date, CONVERT(DATE, GETDATE()))
>= 18)
);

CREATE TABLE passport
(
     number NVARCHAR(9),
     issuing_country NVARCHAR(100),
     expiration_date DATE NOT NULL,
     nationality NVARCHAR(50) NOT NULL,
     PRIMARY KEY (number, issuing_country),
     CONSTRAINT CK_expiration_date_is_bigger_than_current
          CHECK (expiration_date > CONVERT(DATE, GETDATE()))
);

CREATE TABLE credit_card
(
     number BIGINT PRIMARY KEY,
     creation_date DATE NOT NULL,
     expiration_date DATE NOT NULL,
     CONSTRAINT CK_creation_date_is_not_bigger_than_current
          CHECK (creation_date <= CONVERT(DATE, GETDATE())),
     CONSTRAINT CK_card_has_at_least_one_year_before_expiration
          CHECK (expiration_date >= DATEADD(YEAR, 1,
CAST(creation_date AS SMALLDATETIME))),
```

```
	CONSTRAINT
CK_number_has_bigger_than_seven_digits_and_less_than_twenty
		CHECK (9999999 < number AND number < 10000000000000000000)
);

CREATE TABLE [user]
(
	person_id INT PRIMARY KEY,
	email NVARCHAR(320),
	phone_number BIGINT,
	dollars_count MONEY NOT NULL
		CONSTRAINT DF_money_count DEFAULT 0,
	credit_card_number BIGINT,
	town NVARCHAR(100) NOT NULL,
	country NVARCHAR(100) NOT NULL,
	passport_number NVARCHAR(9),
	passport_issuing_country NVARCHAR(100),
	CONSTRAINT CK_phone_number_length_is_correct
		CHECK (999 < phone_number AND phone_number < 1000000000000),
	CONSTRAINT FK_user_person
		FOREIGN KEY (person_id)
		REFERENCES person(id),
	CONSTRAINT CK_AtLeastOneContact CHECK
	(
		email IS NOT NULL
		OR
		phone_number IS NOT NULL
	),
	CONSTRAINT FK_user_credit_card
		FOREIGN KEY (credit_card_number)
		REFERENCES credit_card(number)
		ON DELETE SET NULL,
	CONSTRAINT FK_user_passport
		FOREIGN KEY (passport_number, passport_issuing_country)
		REFERENCES passport(number, issuing_country)
		ON DELETE SET NULL
);

CREATE TABLE plane
(
	id INT IDENTITY(1,1) PRIMARY KEY,
	default_seats_count INT NOT NULL
		CONSTRAINT DF_default_seats_count DEFAULT 1,
	vip_seats_count INT NOT NULL
		CONSTRAINT DF_vip_seats_count DEFAULT 0,
	manufacture_date DATE NOT NULL,
	CONSTRAINT CK_manufacture_date_is_less_than_current
		CHECK (manufacture_date <= CONVERT(DATE, GETDATE()))
);

CREATE TABLE flight
```

```
(
	[name] NVARCHAR(7) PRIMARY KEY,
	start_place NVARCHAR(150) NOT NULL,
	destination_place NVARCHAR(150) NOT NULL,
	departure_time TIME NOT NULL,
	arriving_time TIME NOT NULL
);

CREATE TABLE plane_to_flight
(
	plane_id INT,
	flight_name NVARCHAR(7),
	PRIMARY KEY (plane_id, flight_name),
	CONSTRAINT FK_plane_to_flight_plane
		FOREIGN KEY (plane_id)
		REFERENCES plane(id),
	CONSTRAINT FK_plane_to_flight_flight
		FOREIGN KEY (flight_name)
		REFERENCES flight([name]),
	CONSTRAINT CK_plane_is_not_outdated
		CHECK (dbo.IsPlaneOutdated(plane_id) = 0)
);

CREATE TABLE ticket_status
(
	id INT IDENTITY(1,1) PRIMARY KEY,
	[name] NVARCHAR(30) NOT NULL UNIQUE
);

CREATE TABLE ticket_class
(
	id INT IDENTITY(1,1) PRIMARY KEY,
	name NVARCHAR(30) NOT NULL UNIQUE
);

CREATE TABLE ticket
(
	id INT IDENTITY(1,1) PRIMARY KEY,
	status_id INT
		CONSTRAINT DF_available
		DEFAULT dbo.GetTicketStatusIdByName('Available'),
	ticket_class_id INT,
	price MONEY NOT NULL,
	plane_id INT,
	flight_name NVARCHAR(7),
	flight_date DATETIME
		CONSTRAINT DF_flight_date_in_three_months
		DEFAULT DATEADD(MONTH, 3, GETDATE()),
	CONSTRAINT FK_ticket_ticket_status
		FOREIGN KEY (status_id)
		REFERENCES ticket_status(id),
```

```
	CONSTRAINT FK_ticket_ticket_class
		FOREIGN KEY (ticket_class_id)
		REFERENCES ticket_class(id),
	CONSTRAINT FK_ticket_plane_to_flight
		FOREIGN KEY (plane_id, flight_name)
		REFERENCES plane_to_flight(plane_id, flight_name)
		ON DELETE SET NULL,
	CONSTRAINT CK_plane_has_not_attached_flight_on_the_date
		CHECK (dbo.PlaneHasAttachedFlightOnTheDate(plane_id,
flight_date) = 1),
	CONSTRAINT CK_new_ticket_can_be_created
		CHECK (dbo.SeatForNewTicketIsAvailable(ticket_class_id,
plane_id) = 1)
);

CREATE TABLE ticket_booking
(
	[user_id] INT,
	ticket_id INT UNIQUE,
	[current_date] DATETIME
		CONSTRAINT DF_current_date_is_today
			DEFAULT GETDATE(),
	PRIMARY KEY([user_id], ticket_id),
	CONSTRAINT FK_ticket_booking_user
		FOREIGN KEY ([user_id])
		REFERENCES [user](person_id),
	CONSTRAINT FK_ticket_booking_ticket
		FOREIGN KEY (ticket_id)
		REFERENCES ticket(id),
	CONSTRAINT CK_user_has_enough_money_to_buy_ticket
		CHECK (dbo.UserHasEnoughMoneyToBuyTicket([user_id],
ticket_id) = 1)
);
```

Створення тригерів

```
USE air_tickets_search_and_booking;
GO

CREATE TRIGGER specifying_ticket_status_as_booked
ON ticket_booking
AFTER INSERT
AS
BEGIN
	UPDATE ticket
	SET status_id = dbo.GetTicketStatusIdByName('Booked')
	WHERE id IN (SELECT ins.ticket_id FROM inserted AS ins)
END;
GO

CREATE TRIGGER specifying_ticket_status_as_available
```

```
ON ticket_booking
AFTER DELETE
AS
BEGIN
	UPDATE ticket
	SET status_id = dbo.GetTicketStatusIdByName('Available')
	WHERE id IN (SELECT del.ticket_id FROM deleted AS del)
END;
GO

CREATE TRIGGER taking_money_for_booking_ticket
ON ticket_booking
AFTER INSERT
AS
BEGIN
	DECLARE @user_id INT
	DECLARE @ticket_price MONEY

	DECLARE users_payment_cursor CURSOR FOR
	SELECT ins.[user_id], t.price
	FROM inserted AS ins
	INNER JOIN ticket AS t
		ON t.id = ins.ticket_id

	OPEN users_payment_cursor

	FETCH NEXT FROM users_payment_cursor
		INTO @user_id, @ticket_price

	WHILE @@FETCH_STATUS = 0
		BEGIN
			UPDATE [user]
			SET dollars_count = dollars_count - @ticket_price
			WHERE person_id = @user_id

			FETCH NEXT FROM users_payment_cursor
				INTO @user_id, @ticket_price
		END

	CLOSE users_payment_cursor
	DEALLOCATE users_payment_cursor
END;
GO

CREATE TRIGGER returning_money_due_to_unbooking_ticket
ON ticket_booking
AFTER DELETE
AS
BEGIN
	DECLARE @deleted_borrowing_records TABLE
	(
```

```
            id INT IDENTITY(1,1) PRIMARY KEY,
            [user_id] INT NOT NULL,
            flight_date DATETIME NOT NULL,
            booking_date DATETIME NOT NULL,
            ticket_price MONEY NOT NULL
      )

      INSERT INTO @deleted_borrowing_records([user_id], flight_date,
booking_date, ticket_price)
      SELECT del.[user_id], t.flight_date, del.[current_date], t.price
      FROM ticket AS t
      INNER JOIN deleted AS del
            ON del.ticket_id = t.id

      DECLARE @user_id INT
      DECLARE @flight_date DATETIME
      DECLARE @booking_date DATETIME
      DECLARE @ticket_price MONEY

      DECLARE deleted_borrowing_records_cursor CURSOR FOR
      SELECT dbr.[user_id], dbr.flight_date, dbr.booking_date,
dbr.ticket_price
      FROM @deleted_borrowing_records AS dbr

      OPEN deleted_borrowing_records_cursor

      FETCH NEXT FROM deleted_borrowing_records_cursor
            INTO @user_id, @flight_date, @booking_date, @ticket_price

      WHILE @@FETCH_STATUS = 0
            BEGIN
                  UPDATE [user]
                  SET dollars_count = dollars_count +
                  (
                        CASE
                              WHEN
                                    DATEDIFF(HOUR, @booking_date,
@flight_date) <= 7
                              THEN @ticket_price * 0.7
                              ELSE @ticket_price
                        END
                  )
                  WHERE person_id = @user_id

                  FETCH NEXT FROM deleted_borrowing_records_cursor
                        INTO @user_id, @flight_date, @booking_date,
@ticket_price
            END

      CLOSE deleted_borrowing_records_cursor
      DEALLOCATE deleted_borrowing_records_cursor
```

```
END;
GO
```

Створення представлень

```
USE air_tickets_search_and_booking;

GO
CREATE VIEW tickets_details_view
AS
	SELECT DISTINCT t.id, f.[name] AS flight_name,
f.destination_place, f.start_place, t.flight_date,
		f.arriving_time, f.departure_time, t.price, tc.[name] AS
ticket_class_name
	FROM ticket AS t
	INNER JOIN plane_to_flight AS ptf
		ON t.flight_name = ptf.flight_name
			AND t.plane_id = ptf.plane_id
	INNER JOIN flight AS f
		ON PTF.flight_name = f.[name]
	INNER JOIN ticket_class AS tc
		ON tc.id = t.ticket_class_id;

GO
CREATE VIEW users_details_view
AS
	SELECT u.person_id, p.[name], p.surname, pa.nationality,
cd.number AS credit_card_number,
		u.country, u.town, u.email, u.phone_number
	FROM [user] AS u
	INNER JOIN person AS p
		ON u.person_id = p.id
	INNER JOIN passport AS pa
		ON pa.number = u.passport_number
	INNER JOIN credit_card AS cd
		ON cd.number = u.credit_card_number;

GO
CREATE VIEW ticket_booking_details_view
AS
	SELECT udv.[name], udv.surname, udv.country,
udv.credit_card_number, udv.email, udv.phone_number,
		tdv.start_place, tdv.destination_place, tdv.price,
tdv.flight_date, tdv.departure_time
	FROM ticket_booking AS tb
	INNER JOIN dbo.users_details_view AS udv
		ON tb.[user_id] = udv.person_id
	INNER JOIN dbo.tickets_details_view AS tdv
		ON tdv.id = tb.ticket_id;
```

Створення процедур

```
USE air_tickets_search_and_booking;
GO

CREATE PROCEDURE dbo.spdelete_user(@user_id INT)
AS
	SET NOCOUNT ON
		DELETE FROM [user]
		WHERE person_id = @user_id
	GO

CREATE PROCEDURE dbo.spdelete_ticket(@ticket_id INT)
AS
	SET NOCOUNT ON
	IF
	(
		NOT EXISTS(
			SELECT *
			FROM ticket_booking AS tb
			WHERE tb.ticket_id = @ticket_id
		)
	)
		DELETE FROM ticket
		WHERE id = @ticket_id
GO

CREATE PROCEDURE dbo.spdelete_expired_tickets
AS
	SET NOCOUNT ON
	DELETE FROM ticket
	WHERE status_id = 'Available'
		AND flight_date < GETDATE();
GO

CREATE PROCEDURE dbo.spunbook_ticket(@user_id INT, @ticket_id INT)
AS
	SET NOCOUNT ON
	DELETE FROM ticket_booking
	WHERE [user_id] = @user_id
		AND ticket_id = @ticket_id;
GO

CREATE PROCEDURE dbo.spbook_ticket(@user_id INT, @ticket_id INT)
AS
	SET NOCOUNT ON
	INSERT INTO ticket_booking([user_id], ticket_id)
	VALUES
		(@user_id, @ticket_id);
GO
```

```
CREATE PROCEDURE dbo.sptransfer_money_to_account(@user_id INT,
@money_count MONEY)
AS
      SET NOCOUNT ON
      UPDATE [user]
      SET dollars_count = dollars_count + @money_count
      WHERE person_id = @user_id;
GO

CREATE PROCEDURE dbo.spquery_ticket_booking_history(@passport_number
NVARCHAR(9))
AS
      SET NOCOUNT ON
            SELECT t.flight_date, t.price, tb.[current_date] AS
booking_date
            FROM ticket_booking AS tb
            INNER JOIN ticket AS t
                  ON tb.ticket_id = t.id
            INNER JOIN ticket_status AS ts
                  ON ts.id = t.status_id
            INNER JOIN [user] AS u
                  ON tb.[user_id] = u.person_id
            INNER JOIN passport AS pa
                  ON pa.number = u.passport_number
            WHERE pa.number = @passport_number
GO
```

Створення функцій

```
USE air_tickets_search_and_booking;

GO
CREATE FUNCTION dbo.GetTicketStatusIdByName(@ticket_status_name
NVARCHAR(30))
RETURNS INT
WITH EXECUTE AS CALLER
AS
BEGIN
      RETURN (
            SELECT ts.id
            FROM ticket_status AS ts
            WHERE ts.[name] = @ticket_status_name
      )
END;

GO
CREATE FUNCTION dbo.GetTicketClassIdByName(@ticket_class_name
NVARCHAR(30))
RETURNS INT
WITH EXECUTE AS CALLER
AS
```

```
BEGIN
	RETURN (
		SELECT tc.id
		FROM ticket_class AS tc
		WHERE tc.[name] = @ticket_class_name
	)
END;

GO
CREATE FUNCTION dbo.IsPlaneOutdated(@plane_id INT)
RETURNS BIT
WITH EXECUTE AS CALLER
AS
BEGIN
	DECLARE @manufacture_date DATE =
		(
			SELECT p.manufacture_date
			FROM plane AS p
			WHERE @plane_id = p.id
		)

	IF (DATEDIFF(YEAR, @manufacture_date, CONVERT(DATE, GETDATE()))
>= 20)
		RETURN 1

	RETURN 0
END;

GO
CREATE FUNCTION dbo.PlaneHasAttachedFlightOnTheDate(@plane_id INT,
@flight_date DATETIME)
RETURNS BIT
WITH EXECUTE AS CALLER
AS
BEGIN
	IF
	(
		(
			SELECT COUNT(*)
			FROM plane AS p
			INNER JOIN ticket AS t
			ON t.plane_id = p.id
			WHERE p.id = @plane_id
				AND t.flight_date = @flight_date
		) > 0
	)
		RETURN 1

	RETURN 0
END;
```

```
GO
CREATE FUNCTION dbo.SeatForNewTicketIsAvailable(@ticket_class_id INT,
@plane_id INT)
RETURNS BIT
WITH EXECUTE AS CALLER
AS
BEGIN
      DECLARE @ticket_class_name NVARCHAR(30) =
      (
            SELECT tc.[name]
            FROM ticket_class AS tc
            WHERE tc.id = @ticket_class_id
      )
      DECLARE @unbooked_seats_count INT

      IF (@ticket_class_name = 'VIP')
            SET @unbooked_seats_count =
            (
                  SELECT p.vip_seats_count
                  FROM plane AS p
                  WHERE p.id = @plane_id
            ) - dbo.GetTicketsCount(@ticket_class_name, @plane_id)

      IF (@ticket_class_name = 'Default')
            SET @unbooked_seats_count =
            (
                  SELECT p.default_seats_count
                  FROM plane AS p
                  WHERE p.id = @plane_id
            ) - dbo.GetTicketsCount(@ticket_class_name, @plane_id)

      IF (@unbooked_seats_count + 1 > 0)
            RETURN 1

      RETURN 0
END;

GO
CREATE FUNCTION dbo.GetTicketsCount(@ticket_class_name NVARCHAR(30),
@plane_id INT)
RETURNS INT
WITH EXECUTE AS CALLER
AS
BEGIN
      RETURN (
            SELECT COUNT(*) AS existing_tickets_count
            FROM ticket AS t
            INNER JOIN ticket_class AS tc
            ON t.ticket_class_id = tc.id
            INNER JOIN plane AS p
            ON p.id = t.plane_id
```

```
            WHERE p.id = @plane_id
                  AND tc.[name] = @ticket_class_name
      )
END;

GO
CREATE FUNCTION dbo.UserHasEnoughMoneyToPay(@user_id INT,
@money_to_pay MONEY)
RETURNS BIT
WITH EXECUTE AS CALLER
AS
BEGIN
      IF EXISTS (
            SELECT * FROM [user] AS u
            WHERE u.person_id = @user_id
                  AND u.dollars_count >= @money_to_pay
      )
            RETURN 1

      RETURN 0
END;

GO
CREATE FUNCTION dbo.UserHasEnoughMoneyToBuyTicket(@user_id INT,
@ticket_id INT)
RETURNS BIT
WITH EXECUTE AS CALLER
AS
BEGIN
      DECLARE @ticket_price MONEY = (
            SELECT t.price
            FROM ticket AS t
            WHERE t.id = @ticket_id
      )

      RETURN dbo.UserHasEnoughMoneyToPay(@user_id, @ticket_price)
END;
```

5 СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ

Створення логінів

```
USE air_tickets_search_and_booking;

CREATE LOGIN [user] WITH PASSWORD = 'sa';

CREATE LOGIN worker WITH PASSWORD = 'sa';

CREATE LOGIN administrator WITH PASSWORD = 'sa';
```

Створення ролей

```
USE air_tickets_search_and_booking;

CREATE ROLE db_user;

GRANT SELECT ON ticket TO db_user;

GRANT SELECT ON plane TO db_user;

GRANT EXECUTE ON OBJECT::dbo.spunbook_ticket TO db_user;

GRANT EXECUTE ON OBJECT::dbo.spbook_ticket TO db_user;

GRANT EXECUTE ON OBJECT::dbo.sptransfer_money_to_account TO db_user;

GRANT EXECUTE ON OBJECT::dbo.spquery_ticket_booking_history TO
db_user;

GRANT SELECT ON plane_to_flight TO db_user;


CREATE ROLE db_worker;

GRANT EXECUTE ON OBJECT::dbo.spdelete_user TO db_worker;

GRANT EXECUTE ON OBJECT::dbo.spdelete_ticket TO db_worker;

GRANT EXECUTE ON OBJECT::dbo.spdelete_expired_tickets TO db_worker;

GRANT SELECT ON SCHEMA :: [dbo] TO db_worker;

GRANT ALTER ON SCHEMA :: [dbo] TO db_worker;

GRANT CONTROL ON SCHEMA :: [dbo] TO db_worker;

GRANT CREATE SEQUENCE ON SCHEMA :: [dbo] TO db_worker;
```

```
GRANT DELETE ON SCHEMA :: [dbo] TO db_worker;

GRANT EXECUTE ON SCHEMA :: [dbo] TO db_worker;

GRANT INSERT ON SCHEMA :: [dbo] TO db_worker;

GRANT REFERENCES ON SCHEMA :: [dbo] TO db_worker;

GRANT UPDATE ON SCHEMA :: [dbo] TO db_worker;

GRANT VIEW CHANGE TRACKING ON SCHEMA :: [dbo] TO db_worker;

GRANT VIEW DEFINITION ON SCHEMA :: [dbo] TO db_worker;


CREATE ROLE db_administrator;

GRANT SELECT ON SCHEMA :: [dbo] TO db_administrator;

GRANT ALTER ON SCHEMA :: [dbo] TO db_administrator;

GRANT CONTROL ON SCHEMA :: [dbo] TO db_administrator;

GRANT CREATE SEQUENCE ON SCHEMA :: [dbo] TO db_administrator;

GRANT DELETE ON SCHEMA :: [dbo] TO db_administrator;

GRANT EXECUTE ON SCHEMA :: [dbo] TO db_administrator;

GRANT INSERT ON SCHEMA :: [dbo] TO db_administrator;

GRANT REFERENCES ON SCHEMA :: [dbo] TO db_administrator;

GRANT TAKE OWNERSHIP ON SCHEMA :: [dbo] TO db_administrator;

GRANT UPDATE ON SCHEMA :: [dbo] TO db_administrator;

GRANT VIEW CHANGE TRACKING ON SCHEMA :: [dbo] TO db_administrator;

GRANT VIEW DEFINITION ON SCHEMA :: [dbo] TO db_administrator;
```

Створення користувачів

```
USE air_tickets_search_and_booking;

IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name =
'user')
BEGIN
    CREATE USER [user] FOR LOGIN [user]
    EXEC sp_addrolemember 'db_user', 'user'
```

```
END;

IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name =
'worker')
BEGIN
    CREATE USER worker FOR LOGIN worker
    EXEC sp_addrolemember 'db_worker', 'worker'
END;

IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name =
'administrator')
BEGIN
    CREATE USER administrator FOR LOGIN administrator
    EXEC sp_addrolemember 'db_administrator', 'administrator'
END;
```

6 ІМПОРТ ДАНИХ З ВИКОРИСТАННЯМ ЗАСОБІВ СУБД У СТВОРЕНУ БАЗУ ДАНИХ

Вставка записів до таблиці із квитками

```
USE air_tickets_search_and_booking;

INSERT INTO ticket(flight_name, plane_id, price, ticket_class_id)
VALUES
      ('ALW2218', 1, 213, 1),
      ('ALW2218', 1, 213, 1),
      ('ALW2218', 1, 213, 1),
      ('ALW2218', 1, 213, 1),
      ('ALW2218', 1, 213, 1),
      ('ALW2218', 1, 213, 1),
      ('ALW2218', 1, 213, 1),
      ('AMO5728', 2, 343, 2),
      ('AYF6886', 3, 212, 1),
      ('BEH8327', 4, 543, 2),
      ('BIZ4896', 5, 222, 1),
      ('BXD8029', 6, 111, 2),
      ('DKP1394', 7, 345, 1),
      ('EVX9840', 8, 642, 2),
      ('EXI4377', 9, 113, 2),
      ('GAY1868', 10, 232, 1),
      ('GAY4438', 11, 543, 2),
      ('GBM5598', 12, 632, 1),
      ('GCT2386', 13, 533, 1),
      ('GDL0413', 14, 264, 2),
      ('HUF7184', 15, 278, 1),
      ('HXG3718', 16, 856, 2),
      ('JLH1268', 17, 456, 1),
      ('JWO1903', 18, 345, 2),
      ('JYS9357', 19, 367, 1),
      ('KOL4082', 20, 675, 2),
      ('LQP8414', 21, 436, 1),
      ('LZA5300', 22, 546, 2),
      ('MGA8488', 23, 345, 1),
      ('NTQ4658', 24, 654, 2),
      ('NUR2617', 25, 374, 1),
      ('NZF6822', 26, 564, 2),
      ('OBW6873', 27, 432, 1),
      ('OIN6170', 28, 654, 2),
      ('PWV2994', 29, 345, 1),
      ('QGU9981', 30, 436, 2),
      ('RHY3509', 31, 534, 1),
      ('SRD2265', 32, 367, 2),
      ('SRL2683', 33, 456, 1),
      ('TUG6663', 34, 364, 2),
```

```
    ('TXQ6585', 35, 574, 1),
    ('UCF5496', 36, 547, 2),
    ('UJL3666', 37, 433, 1),
    ('UKA3808', 38, 876, 2),
    ('VPK9010', 39, 235, 1),
    ('VQU5334', 40, 222, 2),
    ('WAB2792', 41, 132, 1),
    ('WAZ2594', 42, 634, 2),
    ('WDU5437', 43, 345, 1),
    ('WOA2036', 44, 234, 2),
    ('WSR1982', 45, 564, 1),
    ('WVJ8320', 46, 876, 2),
    ('YAN3742', 47, 324, 1),
    ('YXS3148', 48, 534, 2),
    ('ZJR3836', 49, 235, 1),
    ('ZWB8224', 50, 435, 2);
```

Вставка записів до таблиці із бронюванням квитків

```
USE air_tickets_search_and_booking;

INSERT INTO ticket_booking(ticket_id, [user_id], [current_date])
VALUES
    (1, 1, '2023-13-04 15:00'),
    (2, 1, '2023-13-04 15:00'),
    (3, 1, '2023-13-04 15:00'),
    (4, 1, '2023-13-04 15:00'),
    (5, 1, '2023-13-04 15:00'),
    (6, 1, '2023-13-04 15:00'),
    (7, 1, '2023-13-04 15:00'),
    (15, 1, '2023-13-04 15:00'),
    (22, 2, '2023-04-03'),
    (32, 3, '2023-04-03'),
    (42, 4, '2023-04-03'),
    (35, 5, '2023-04-03'),
    (47, 7, '2023-04-03'),
    (18, 8, '2023-04-03'),
    (29, 9, '2023-04-03')
```

Вставка рейсів

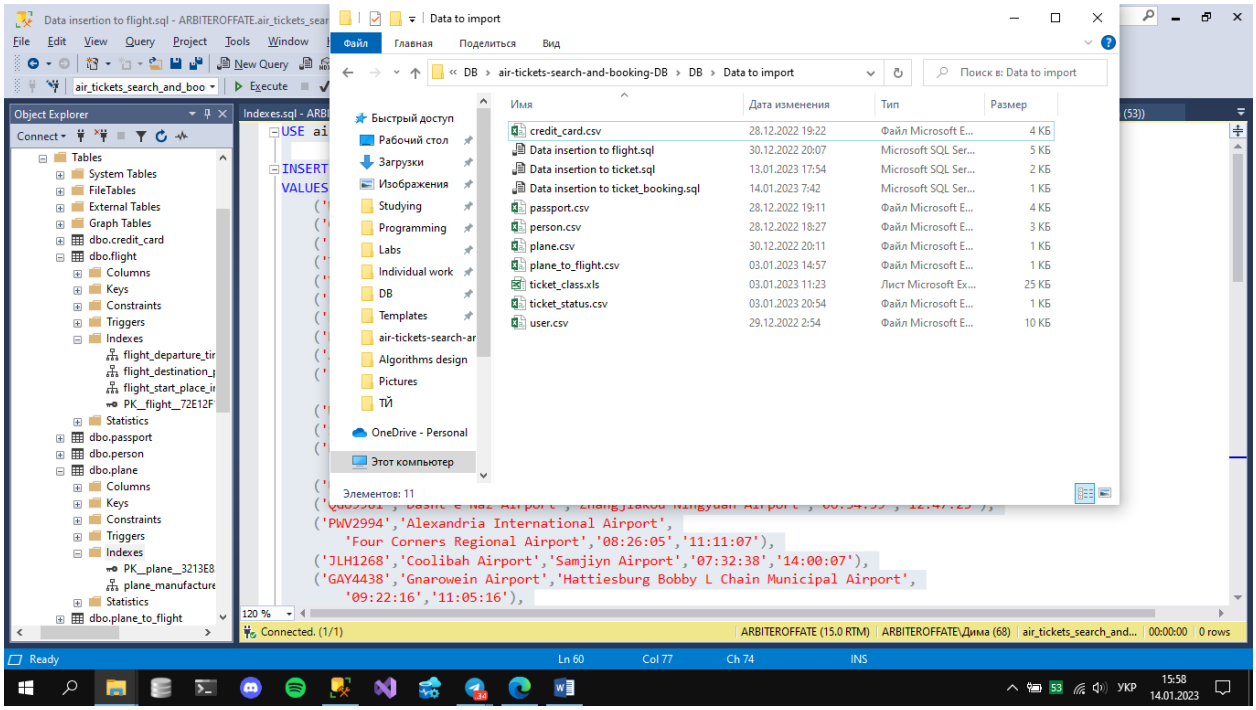
```
USE air_tickets_search_and_booking;

INSERT INTO flight
VALUES
    ('WVJ8320','Easton State Airport','Queenstown
Airport','09:22:42','16:45:16'),
    ('OBW6873','Yuendumu Airport','Kaya
Airport','08:40:09','13:23:00'),
```

```
      ('EVX9840','Siargao Airport','Juan Casiano
Airport','09:03:41','16:37:06'),
      ('YAN3742','Inhambane Airport','La Plata
Airport','06:19:35','11:41:40'),
      ('VPK9010','Dewitt Field Old Town Municipal Airport','Merdei
Airport','07:01:39','10:55:19'),
      ('EXI4377','Aseki Airport','Andi Jemma
Airport','06:34:56','13:43:20'),
      ('BEH8327','Aomori Airport','Shah Mokhdum
Airport','08:38:19','16:16:57'),
      ('DKP1394','Chungribu Airport','Vunisea
Airport','07:35:29','10:56:57'),
      ('ALW2218','Tabiteuea North Airport','Los Alamos
Airport','09:55:44','13:48:35'),
      ('LZA5300','Netaji Subhash Chandra Bose International Airport',
            'Adi Sutjipto International Airport','09:07:28','16:27:24'),
      ('WDU5437','Tungsten (Cantung) Airport','Kagi
Airport','07:10:53','14:25:08'),
      ('SRL2683','Don Mueang International Airport','Castlebar
Airport','10:01:54','13:41:20'),
      ('NUR2617','Pontoise - Cormeilles-en-Vexin Airport',
            'Robin Hood Doncaster Sheffield
Airport','09:26:54','16:23:12'),
      ('UJL3666','Luclia Airport','Tambacounda
Airport','09:42:05','12:46:21'),
      ('QGU9981','Dasht-e Naz Airport','Zhangjiakou Ningyuan
Airport','06:34:39','12:47:23'),
      ('PWV2994','Alexandria International Airport',
            'Four Corners Regional Airport','08:26:05','11:11:07'),
      ('JLH1268','Coolibah Airport','Samjiyn
Airport','07:32:38','14:00:07'),
      ('GAY4438','Gnarowein Airport','Hattiesburg Bobby L Chain
Municipal Airport',
            '09:22:16','11:05:16'),
      ('TUG6663','Ed Carlson Memorial Field South Lewis County
Airport',
            'Owen Sound / Billy Bishop Regional
Airport','08:20:09','12:15:29'),
      ('ZWB8224','Iliamna Airport','Montgomery-Gibbs Executive
Airport','09:10:57','10:15:42'),
      ('GBM5598','El Trompillo Airport','Borlange
Airport','09:24:17','11:01:44'),
      ('RHY3509','Semnan Municipal Airport','Walgett
Airport','08:21:16','11:48:29'),
      ('WAZ2594','Simmons Army Air Field','Cape Gloucester
Airport','09:30:18','11:44:10'),
      ('WOA2036','German Olano Airport','Haifa International
Airport','07:20:39','10:51:36'),
      ('HXG3718','La Tontouta International Airport','Mosquera
Airport','08:05:45','11:13:51'),
```

```
      ('HUF7184','El Eden Airport','Aktau
Airport','08:35:40','16:31:02'),
      ('WSR1982','Salt Lake City International Airport','Alenquer
Airport','07:12:45','16:23:35'),
      ('ZJR3836','Vanrook Station Airport','Mandinga
Airport','08:45:17','15:39:45'),
      ('TXQ6585','Nma Airport','Cheadle
Airport','09:12:56','14:28:27'),
      ('UKA3808','Islita Airport','Jnkping
Airport','08:28:21','10:45:16'),
      ('KOL4082','Arkalyk North Airport','Whangarei
Airport','06:52:02','11:21:37'),
      ('JWO1903','Lien Khuong Airport','Malolo Lailai Island
Airport','08:16:45','15:52:05'),
      ('AYF6886','Bost Airport','Balgo Hill
Airport','10:04:16','11:24:33'),
      ('BXD8029','Zephyrhills Municipal Airport','General Manuel Carlos
Piar International Airport',
            '07:40:40','13:35:40'),
      ('GAY1868','Avaratra Airport','Tucupita
Airport','07:29:43','10:16:02'),
      ('YXS3148','Coolah Airport','Andavadoaka
Airport','07:49:56','14:53:21'),
      ('NZF6822','Barksdale Air Force Base','Colac
Airport','08:55:08','10:24:35'),
      ('BIZ4896','Ellington Airport','Jacksonville Executive at Craig
Airport','09:47:52','15:05:06'),
      ('GDL0413','Bauru - Arealva Airport','Turin
Airport','09:11:05','12:48:02'),
      ('JYS9357','Hopedale Airport','Sanga Sanga
Airport','09:16:29','12:57:54'),
      ('VQU5334','Lese Airport','Guasopa
Airport','09:09:57','12:04:01'),
      ('SRD2265','Bamiyan Airport','Basankusu
Airport','06:54:48','10:33:05'),
      ('WAB2792','Reyes Murillo Airport','Herendeen Bay
Airport','07:00:30','12:35:03'),
      ('UCF5496','Sfax Thyna International Airport','Ghinnir
Airport','07:53:03','16:38:38'),
      ('OIN6170','Tillamook Airport','Blue Lagoon Seaplane
Base','07:45:37','14:11:35'),
      ('MGA8488','Tekadu Airport','Russell Municipal
Airport','07:17:44','11:57:39'),
      ('NTQ4658','So Loureno do Sul Airport','Kambalda
Airport','08:18:16','10:34:54'),
      ('GCT2386','Ouahigouya Airport','Rouses Point Seaplane
Base','08:34:14','16:55:39'),
      ('LQP8414','Greenbrier Valley Airport','Mururoa Atoll
Airport','08:46:05','16:35:37'),
```

```
('AMO5728','Rabah Bitat Airport','Tchien Airport','08:13:09','16:55:49')
```

Знімок екрану з файлами, котрі містять дані для інших таблиць та імпортуються до таблиць за допомогою інструментів СУБД



7 СТВОРЕННЯ МОВОЮ SQL ЗАПИТІВ

Запити користувача

```
USE air_tickets_search_and_booking;

-- пошук наявних польотів для певного призначення на певну дату
SELECT f.start_place, f.destination_place, t.flight_date,
ptf.flight_name
FROM flight AS f
INNER JOIN plane_to_flight AS ptf
     ON f.[name] = ptf.flight_name
INNER JOIN ticket AS t
     ON ptf.plane_id = t.plane_id
     AND ptf.flight_name = t.flight_name
WHERE f.destination_place like 'На%'
     AND t.flight_date < '2023-01-06';

-- визначення кількості відповідних місць у літаку та його дату
виготовлення
SELECT
     (
          SELECT COUNT(*)
          FROM ticket AS t
          INNER JOIN ticket_status AS ts
               ON ts.id = t.status_id
          WHERE t.plane_id = p.id
               AND ts.[name] = 'Available'
               AND t.status_id =
dbo.GetTicketClassIdByName('Default')
     ) AS default_tickets_count,
     (
          SELECT COUNT(*)
          FROM ticket AS t
          INNER JOIN ticket_status AS ts
               ON ts.id = t.status_id
          WHERE t.plane_id = p.id
               AND ts.[name] = 'Available'
               AND t.status_id = dbo.GetTicketClassIdByName('VIP')
     ) AS vip_tickets_count, p.manufacture_date
FROM plane AS p
WHERE p.id = 3;

-- отримання історії куплених квитків
SELECT t.flight_date, t.price, tb.[current_date] AS booking_date
FROM ticket_booking AS tb
INNER JOIN ticket AS t
     ON tb.ticket_id = t.id
INNER JOIN ticket_status AS ts
     ON ts.id = t.status_id
INNER JOIN [user] AS u
```

```
     ON tb.[user_id] = u.person_id
WHERE u.passport_number = '117484885';

-- отримати інформацію про місця на вільний літак
SELECT DISTINCT ptf.flight_name, p.id AS plane_id,
p.default_seats_count, p.vip_seats_count
FROM plane AS p
INNER JOIN plane_to_flight AS ptf
     ON ptf.plane_id = p.id
INNER JOIN ticket AS t
     ON t.plane_id = p.id;
```

Запити робітника

```
USE air_tickets_search_and_booking;

-- здобуття інформації про користувача
SELECT p.[name], p.surname, p.birth_date, u.country,
u.credit_card_number
FROM [user] AS u
INNER JOIN person AS p
     ON u.person_id = p.id
WHERE u.person_id = 4;

-- перевірка літаків на прострочені
SELECT p.manufacture_date, (
          SELECT COUNT(*)
          FROM plane_to_flight AS ptf
          WHERE ptf.plane_id = p.id
     ) AS flights_count, p.default_seats_count, p.vip_seats_count
FROM plane AS p
WHERE dbo.IsPlaneOutdated(p.id) = 1;

-- дізнатися про користувачів, у яких закінчився термін придатності
паспорта
SELECT pa.expiration_date, p.[name], p.surname, pa.nationality,
pa.issuing_country
FROM person AS p
INNER JOIN [user] AS u
     ON u.person_id = p.id
INNER JOIN passport AS pa
     ON pa.number = u.passport_number
WHERE pa.expiration_date < GETDATE();

-- дізнатися про користувачів, у яких закінчився термін придатності
кредитної картки
SELECT p.[name], p.surname, cc.expiration_date, u.email,
u.phone_number
FROM [user] AS u
INNER JOIN credit_card AS cc
```

```
      ON u.credit_card_number = cc.number
INNER JOIN person As p
      ON u.person_id = p.id
WHERE GETDATE() > cc.expiration_date;
```

Запити адміністратора

```
USE air_tickets_search_and_booking;

-- інформація про популярність рейсів
SELECT f.[name], COUNT(*) AS sold_tickets_count
FROM ticket_booking AS tb
INNER JOIN ticket AS t
      ON tb.ticket_id = t.id
INNER JOIN flight AS f
      ON f.[name] = t.flight_name
GROUP BY f.[name]
ORDER BY sold_tickets_count;

-- отримання даних про літаки, яким не призначені рейси
SELECT p.manufacture_date, p.default_seats_count, p.vip_seats_count
FROM plane AS p
WHERE p.id NOT IN (
      SELECT ptf.plane_id
      FROM plane_to_flight AS ptf
);

-- отримання даних про кількість куплених vip-квитків на певних рейсах
SELECT t.flight_name, COUNT(*) AS bought_tickets_count
FROM ticket AS t
INNER JOIN plane_to_flight AS ptf
      ON t.flight_name = ptf.flight_name
INNER JOIN ticket_class AS tc
      ON t.ticket_class_id = tc.id
INNER JOIN ticket_status AS ts
      ON ts.id = t.status_id
WHERE tc.[name] = 'VIP'
      AND ts.[name] = 'Booked'
GROUP BY t.flight_name
ORDER BY bought_tickets_count DESC;

-- дізнатися кількість громадян відповідних країн, котрі зареєстровані
у системі авіакомпанії
SELECT pa.issuing_country, COUNT(*) AS people_count,
      AVG(DATEDIFF(YEAR, p.birth_date, GETDATE())) AS
average_people_age
FROM passport AS pa
INNER JOIN [user] AS u
      ON pa.number = u.passport_number
            AND u.passport_issuing_country = pa.issuing_country
```

```
INNER JOIN person AS p
      ON p.id = u.person_id
GROUP BY pa.issuing_country
ORDER BY people_count DESC;

-- отримати дані про цінову політику рейсів
SELECT DISTINCT ptf.flight_name, t.price
FROM ticket AS t
INNER JOIN plane_to_flight AS ptf
      ON t.flight_name = ptf.flight_name
ORDER BY t.price DESC;

-- отримати дані про користувачів, які ніколи не купували vip-квитки
SELECT p.[name], p.surname, u.dollars_count
FROM [user] AS u
INNER JOIN person AS p
      ON u.person_id = p.id
INNER JOIN ticket_booking AS tb
      ON tb.[user_id] = u.person_id
INNER JOIN ticket AS t
      ON t.id = tb.ticket_id
INNER JOIN ticket_class AS tc
      ON t.ticket_class_id = tc.id
WHERE tc.[name] = 'Default';

-- дізнатися про аеропорти, куди літаки літають найчастіше
SELECT f.destination_place, (
      SELECT COUNT(*)
      FROM plane_to_flight AS ptf
      WHERE ptf.flight_name = f.[name]
      ) AS planes_count
FROM flight AS f
GROUP BY f.destination_place, f.[name]
ORDER BY f.destination_place DESC;

-- дізнатися кількість проданих квитків відносно дат відправлення
літаків
SELECT t.flight_date, COUNT(*) AS booked_tickets_count
FROM ticket AS t
INNER JOIN plane_to_flight AS ptf
      ON t.flight_name = ptf.flight_name
            AND t.plane_id = ptf.plane_id
INNER JOIN ticket_status AS ts
      ON ts.id = t.status_id
WHERE ts.[name] = 'Booked'
GROUP BY t.flight_date
ORDER BY t.flight_date DESC
```

8 ОПТИМІЗАЦІЯ РОБОТИ ЗАПИТІВ

```
USE air_tickets_search_and_booking;
```

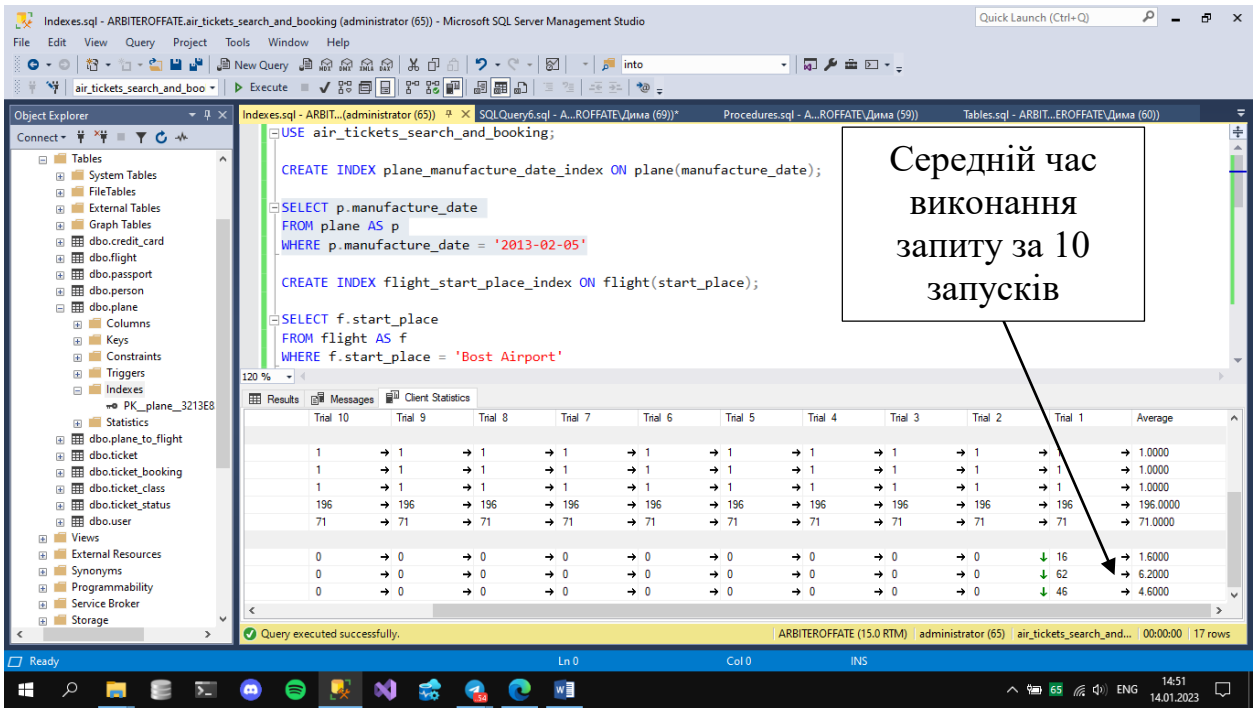


Рисунок 8.1. Середній виконання запиту 10 разів до створення індексу

```
CREATE INDEX plane_manufacture_date_index
    ON plane(manufacture_date);
```

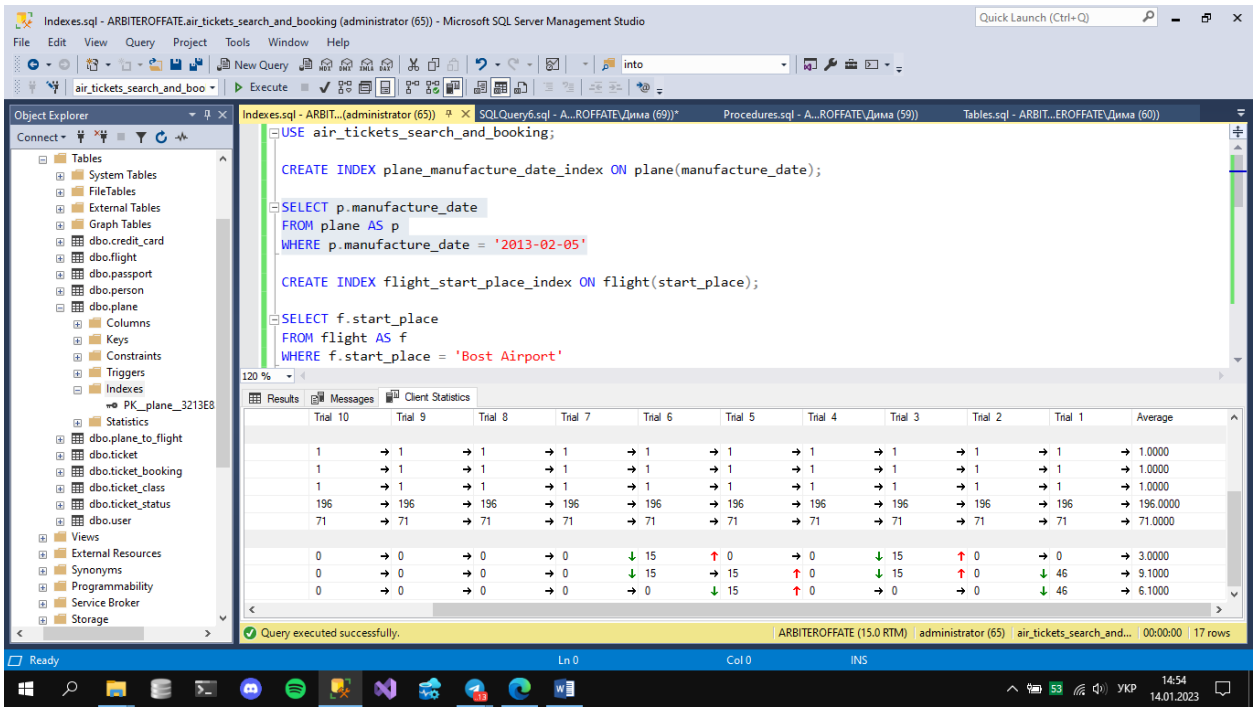


Рисунок 8.2. Середній виконання запиту 10 разів після створення індексу

```
SELECT p.manufacture_date
FROM plane AS p
WHERE p.manufacture_date = '2013-02-05'
```

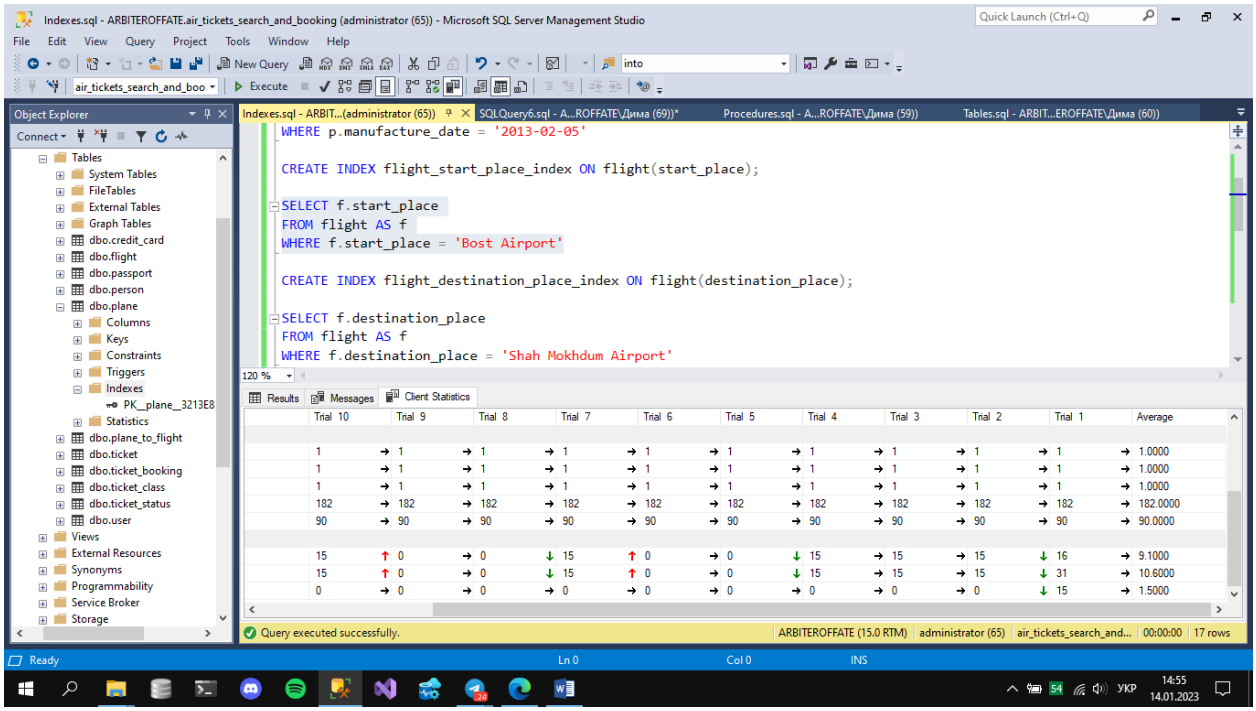


Рисунок 8.3. Середній виконання запиту 10 разів до створення індексу

```
CREATE INDEX flight_start_place_index
    ON flight(start_place);
```

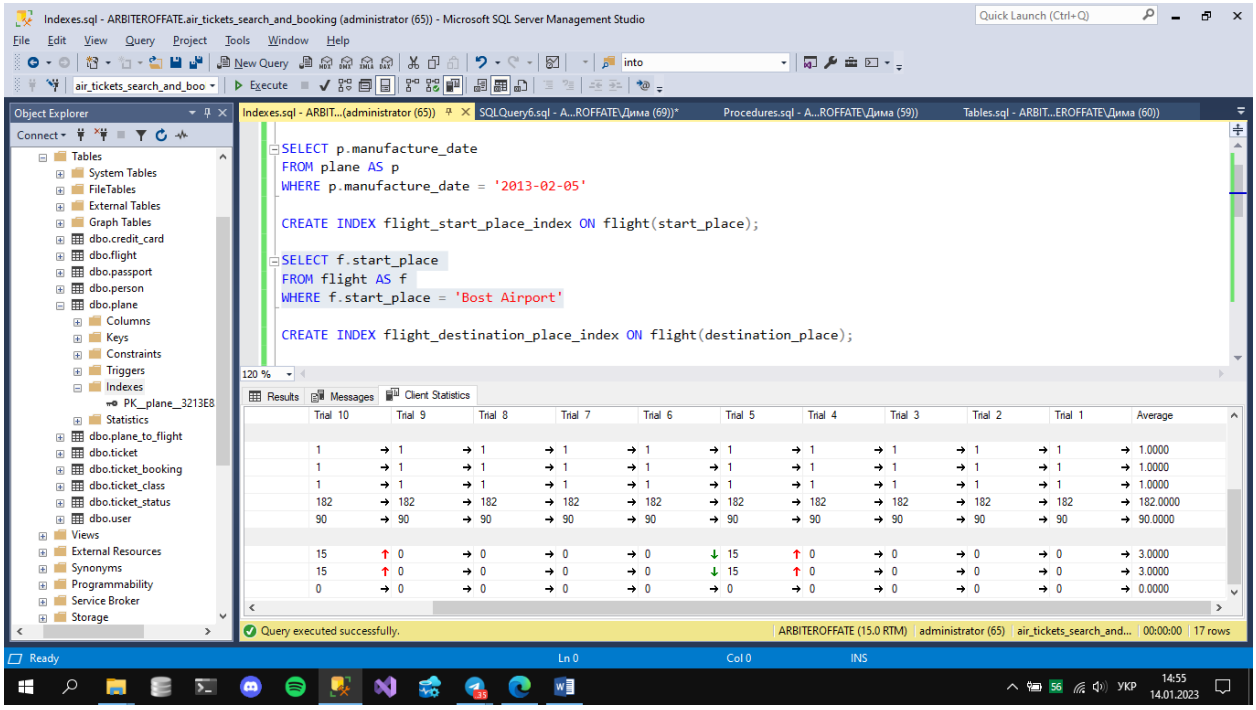


Рисунок 8.4. Середній виконання запиту 10 разів після створення індексу

```
SELECT f.start_place
FROM flight AS f
WHERE f.start_place = 'Bost Airport'
```

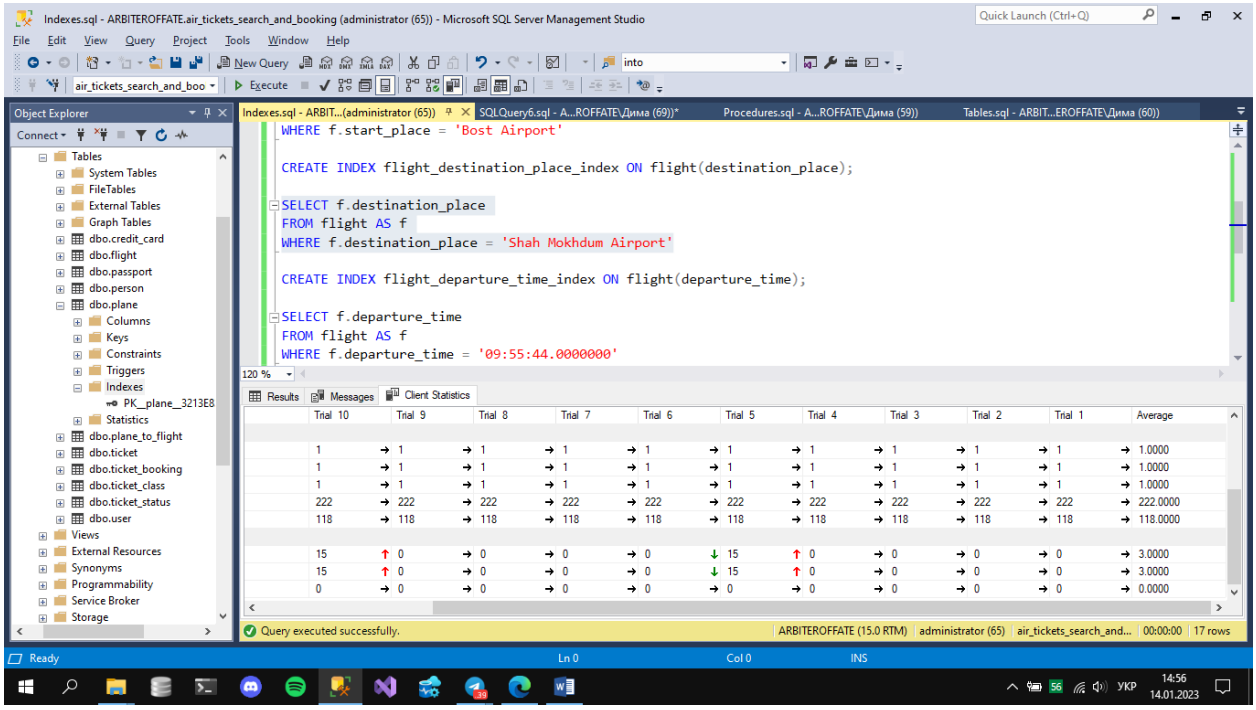


Рисунок 8.5. Середній виконання запиту 10 разів до створення індексу

```
CREATE INDEX flight_destination_place_index
    ON flight(destination_place);
```

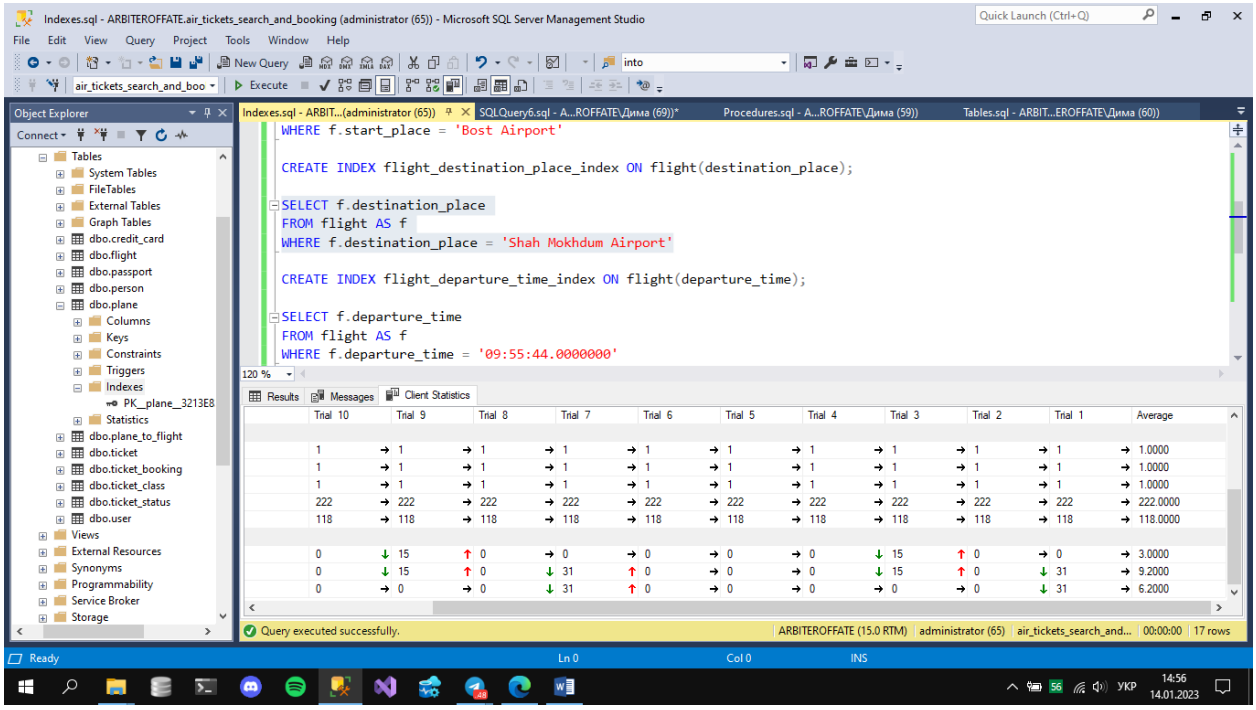


Рисунок 8.6. Середній виконання запиту 10 разів після створення індексу

```
SELECT f.destination_place
FROM flight AS f
WHERE f.destination_place = 'Shah Mokhdum Airport'
```

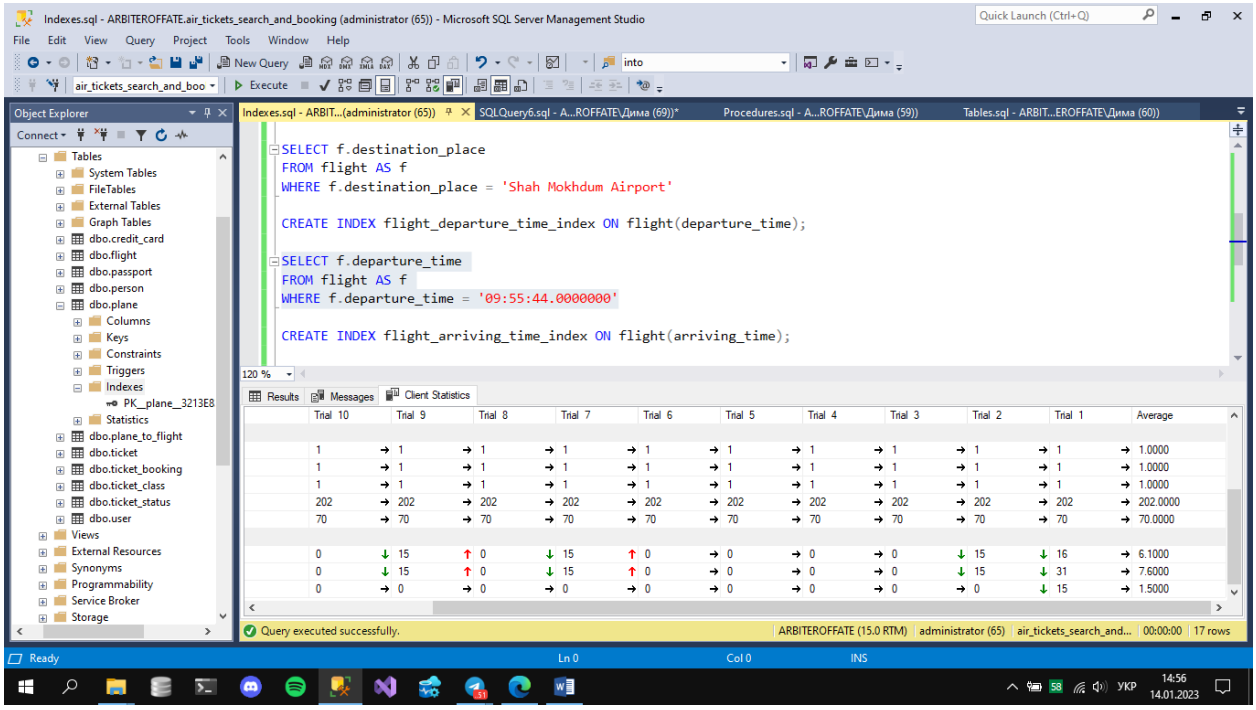


Рисунок 8.7. Середній виконання запиту 10 разів до створення індексу

```
CREATE INDEX flight_departure_time_index
    ON flight(departure_time);
```

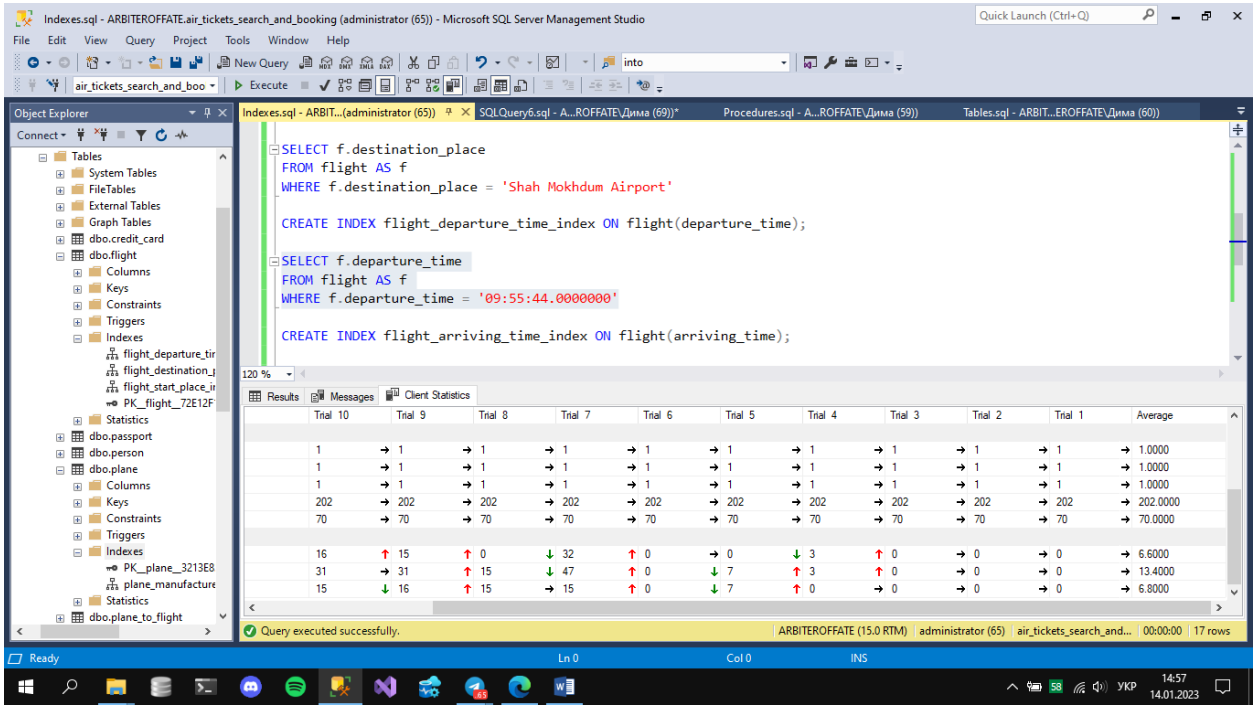


Рисунок 8.8. Середній виконання запиту 10 разів після створення індексу

```
SELECT f.departure_time
FROM flight AS f
WHERE f.departure_time = '09:55:44.0000000'
```

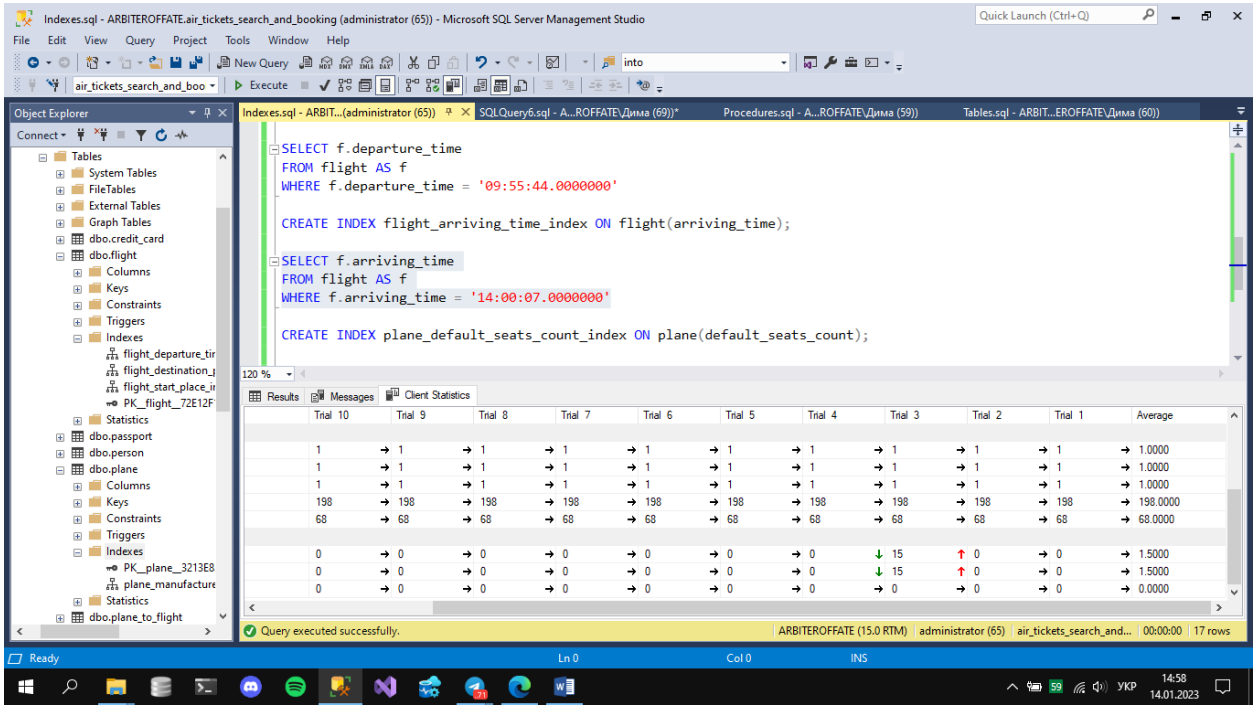


Рисунок 8.9. Середній виконання запиту 10 разів до створення індексу

```
CREATE INDEX flight_arriving_time_index
    ON flight(arriving_time);
```

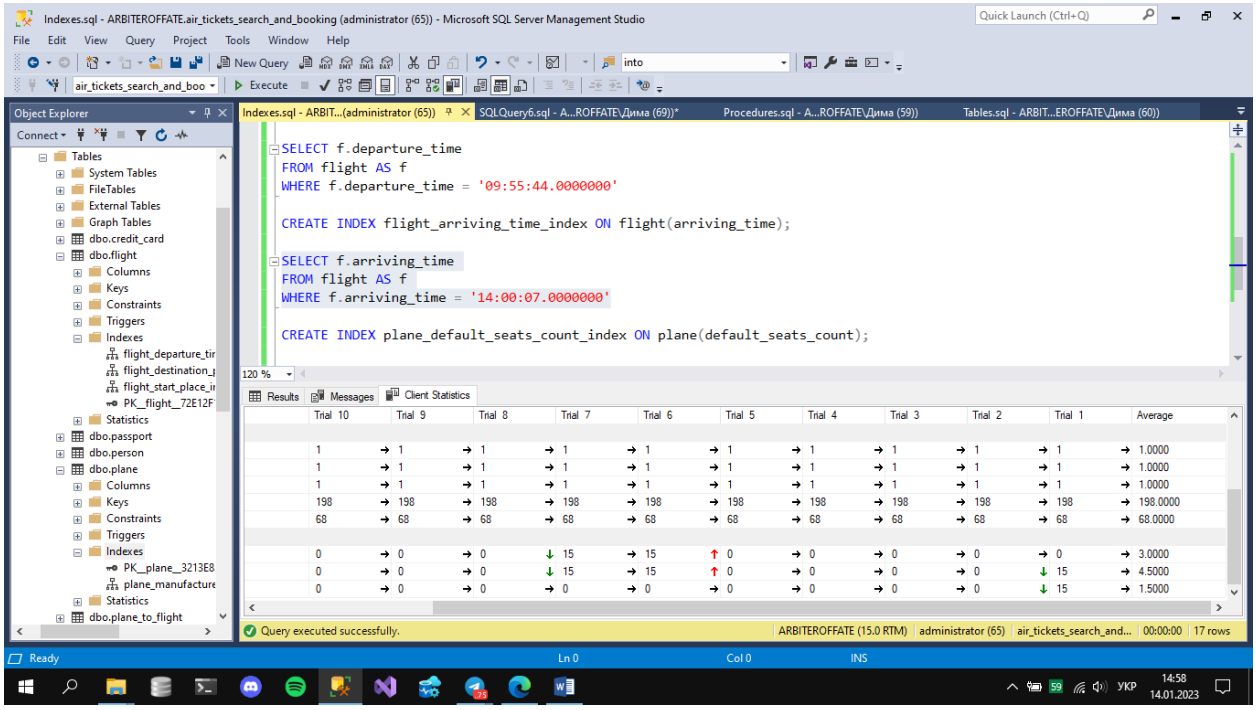


Рисунок 8.10. Середній виконання запиту 10 разів після створення індексу

```
SELECT f.arriving_time
FROM flight AS f
WHERE f.arriving_time = '14:00:07.0000000'
```

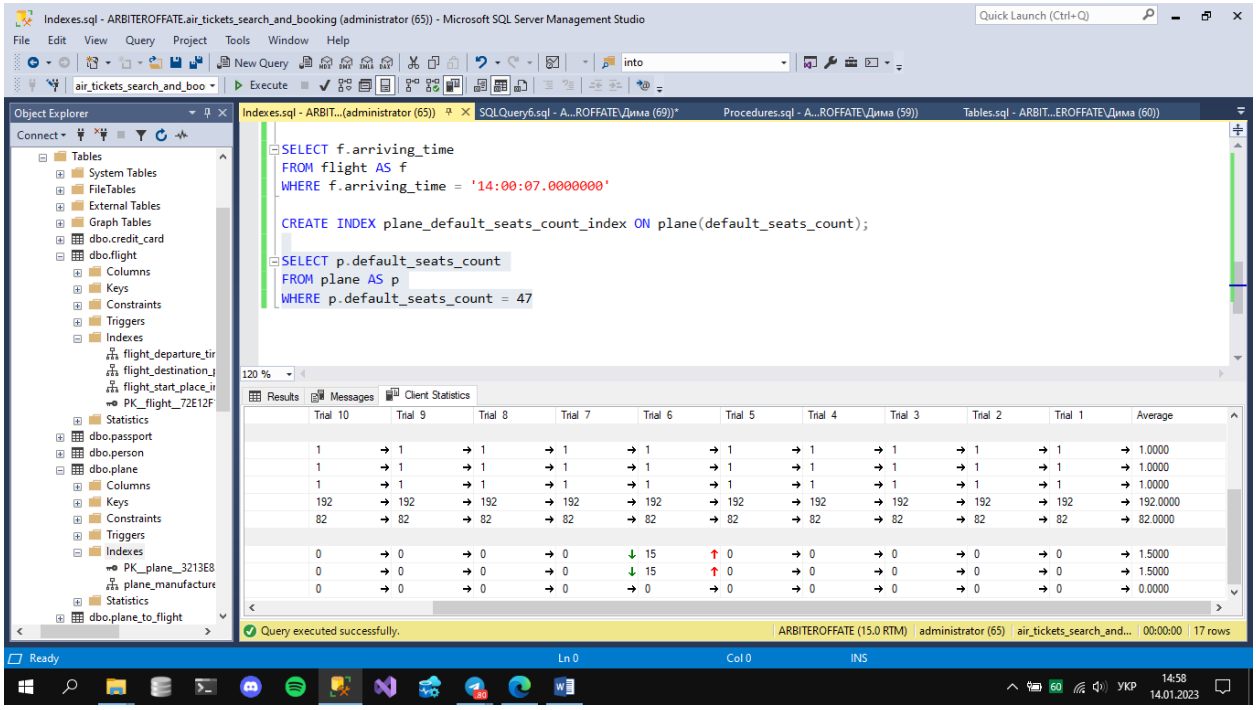


Рисунок 8.11. Середній виконання запиту 10 разів до створення індексу

```
CREATE INDEX plane_default_seats_count_index
    ON plane(default_seats_count);
```

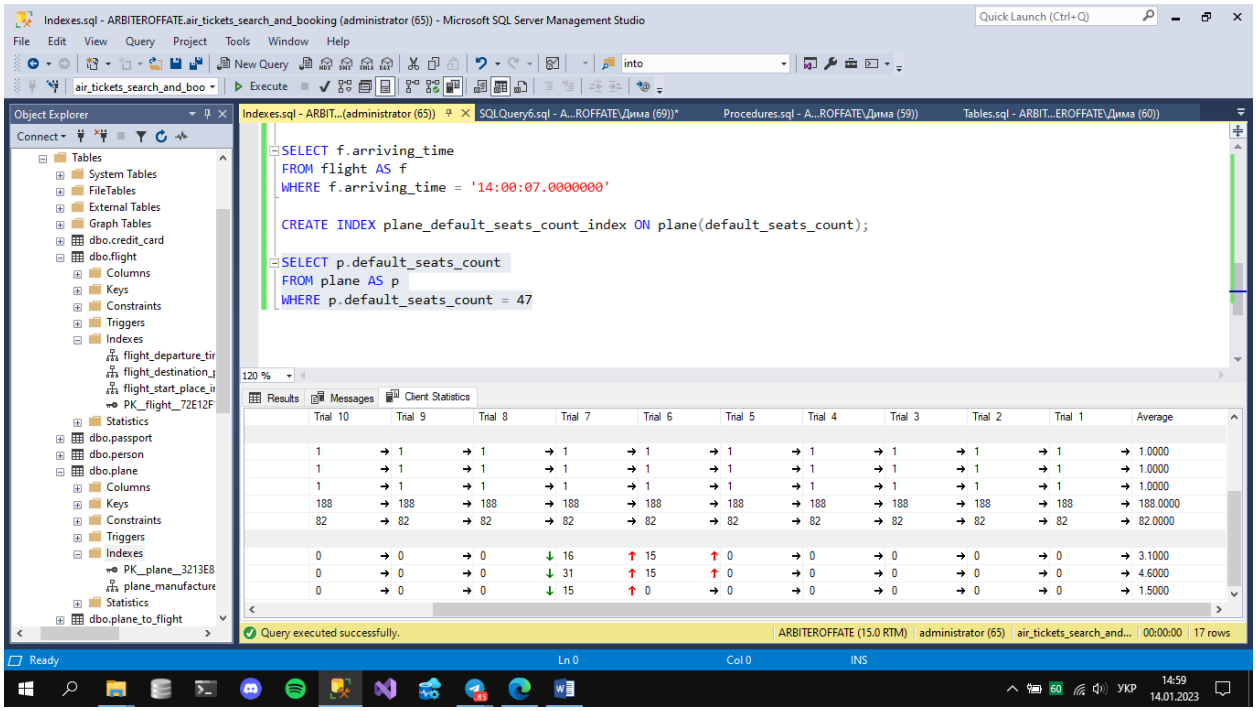


Рисунок 8.12. Середній виконання запиту 10 разів після створення індексу

```
SELECT p.default_seats_count
FROM plane AS p
WHERE p.default_seats_count = 47
```

ВИСНОВКИ

Під час виконання цієї курсової роботи я зміг здобути багато практичних навичок у проектуванні та розробці БД.

Мною було описане предметне середовище БД про бронювання та пошук авіаквитків. Частину ідей проєктування я взяв на веб-сайті з переліку посилань. Я розписав кожну таблицю, котру у майбутньому створив, у вигляді маркерованого списку та коментарів до нього; зобразив принципи основної логіки БД.

За написаним сценарієм, я створив ER-діаграму, а пізніше і реляційну схему з ER-моделі. Проте, через брак досвіду, мені довелося трохи редагувати її структуру під час створення самої БД за допомогою скриптів.

Процес збагачення БД новими можливостями проходив наступним чином:

- Під час написання скриптів для створення таблиць, я додумував логіку, паралельно із цим створюючи додаткові функції, тригери, процедури й інше;
- Після успішного формування «скелету» БД, я скористався наведеним у переліку посилань веб-сайтом для генерації даних, аби імпортувати їх у існуючі таблиці. Оскільки генератор не може створити деякі дані, мені довелося самому писати відповідні скрипти.

Пізніше, я створив користувачів для цієї БД з відповідними їм ролями, надавши їм доступ до скоєння певних дій в залежності від ролі.

Після фінальних налаштувань БД та виправлень помилок я написав створив запити для ролей користувача, працівника та адміністратора.

В кінці, виконання запитів на вибірку даних у відповідних таблицях було оптимізоване за рахунок створення індексів для обраних стовпців.

ПЕРЕЛІК ПОСИЛАНЬ

1. Опис предметної області:

   Структура бази даних: [Електронний ресурс].

   Режим доступу: The Airline Ticket Booking System Example | Sams Teach Yourself BEA WebLogic Server 7.0 in 21 Days (flylib.com)

   (дата звернення – 24.12.2022)

2. Генерація даних для таблиць БД

   Генерація вмісту БД: [Електронний ресурс].

   Режим доступу: Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel

   (дата звернення – 26.12.2022)