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** 5](#_Toc122967075)

[**1** **АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА** 6](#_Toc122967076)

[**2** **ПОБУДОВА ER-МОДЕЛІ** 10](#_Toc122967077)

[**3** **ПОБУДОВА РЕЛЯЦІЙНОЇ СХЕМИ З ER-МОДЕЛІ** 12](#_Toc122967078)

[**4** **СТВОРЕННЯ БАЗИ ДАНИХ, У ФОРМАТІ ОБРАНОЇ СИСТЕМИ УПРАВЛІННЯ БАЗОЮ ДАНИХ** 13](#_Toc122967079)

[**5 СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ** 13](#_Toc122967080)

[**6** **ІМПОРТ ДАНИХ З ВИКОРИСТАННЯМ ЗАСОБІВ СУБД У СТВОРЕНУ БАЗУ ДАНИХ** 25](#_Toc122967081)

[**7** **СТВОРЕННЯ МОВОЮ SQL ЗАПИТІВ** 28](#_Toc122967082)

[**8** **ОПТИМІЗАЦІЯ РОБОТИ ЗАПИТІВ** 33](#_Toc122967083)

[**ВИСНОВКИ** 44](#_Toc122967084)

[**ПЕРЕЛІК ПОСИЛАНЬ** 44](#_Toc122967085)

[**ДОДАТОК А** **Ошибка! Закладка не определена.**](#_Toc122967086)

[**ДОДАТОК Б** **Ошибка! Закладка не определена.**](#_Toc122967087)

**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 незаброньовані квитки за необхідності.
4. Видалити користувача за виявлення певних порушень з його боку.
5. Виконати операції СD 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.
9. **ПОБУДОВА ER-МОДЕЛІ**

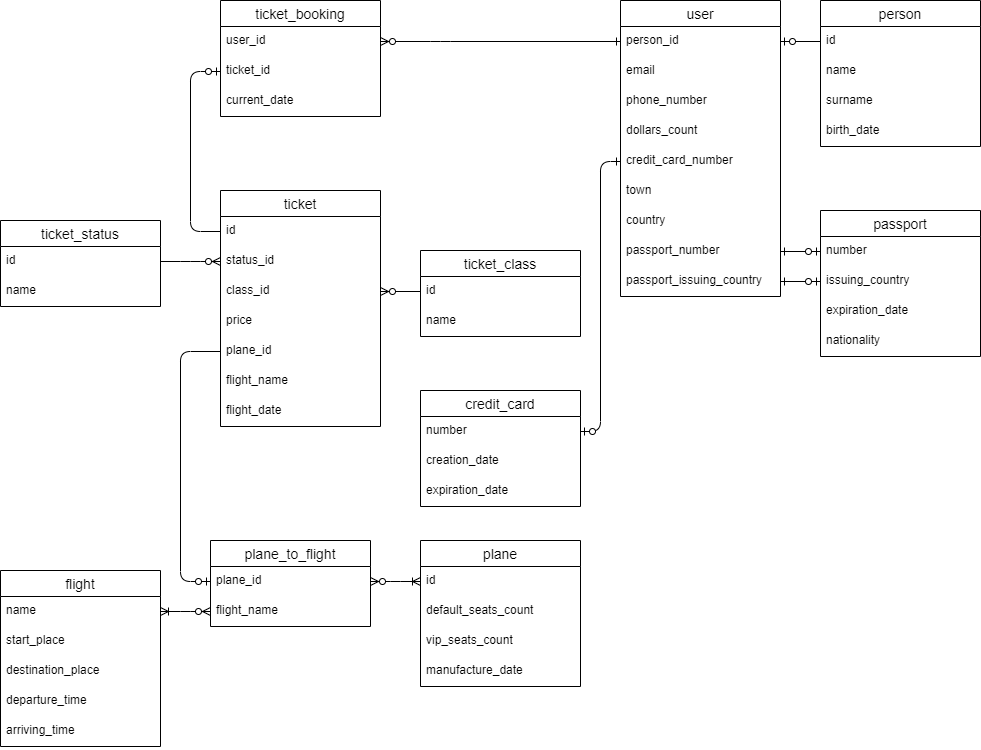


Рисунок 2.1 – ER-діаграма

1. **ПОБУДОВА РЕЛЯЦІЙНОЇ СХЕМИ З ER-МОДЕЛІ**

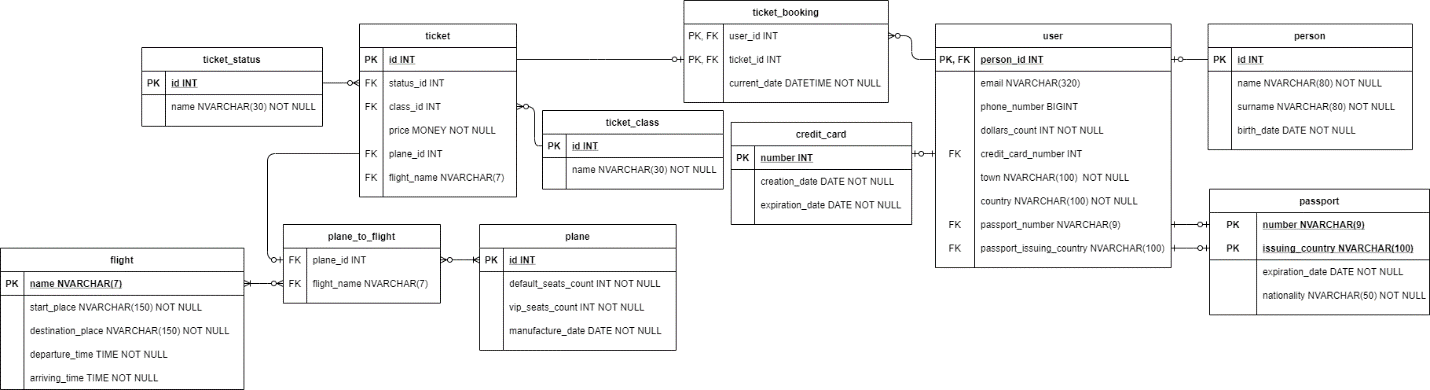


Рисунок 3.1 - реляційна схема

1. **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\_сlass\_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\_сlass\_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\_сlass\_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\_сlass\_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\_сlass\_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;

1. **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;

1. **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\_сlass\_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'),

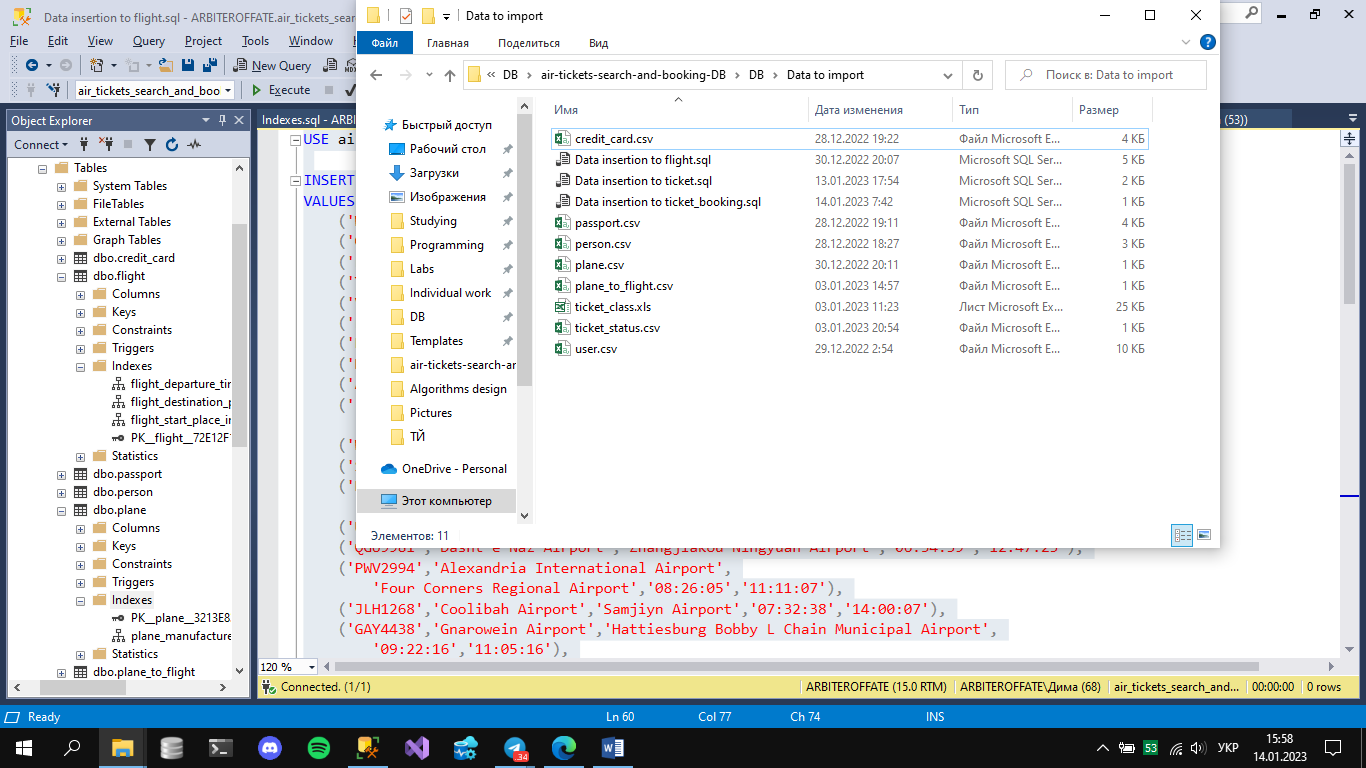
('NTQ4658','So Loureno do Sul Airport','Kambalda Airport','08:18:16','10:34:54'),

('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**



1. **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 'Ha%'

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\_сlass\_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\_сlass\_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

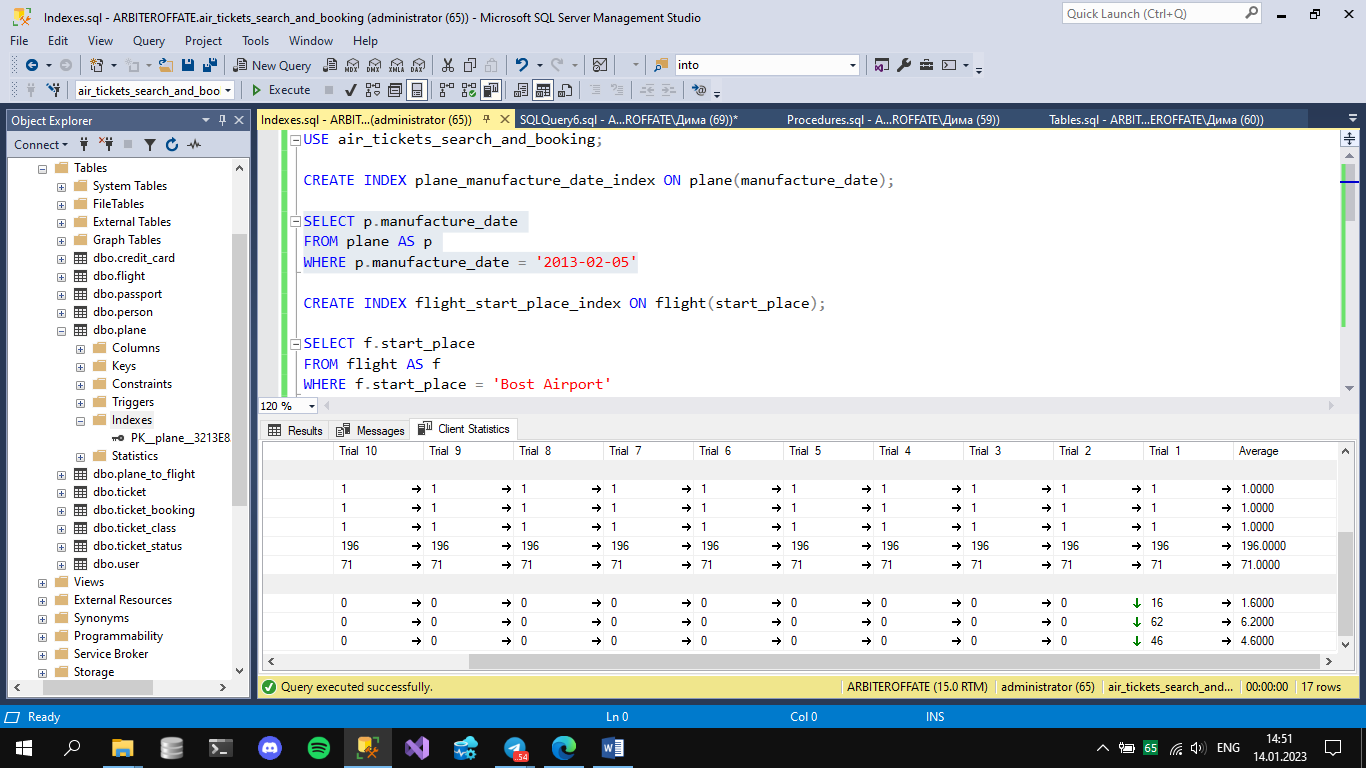
WHERE ts.[name] = 'Booked'

GROUP BY t.flight\_date

ORDER BY t.flight\_date DESC

1. **ОПТИМІЗАЦІЯ РОБОТИ ЗАПИТІВ**

USE air\_tickets\_search\_and\_booking;



Середній час виконання запиту за 10 запусків

Рисунок 8.1. Середній виконання запиту 10 разів до створення індексу

CREATE INDEX plane\_manufacture\_date\_index

ON plane(manufacture\_date);

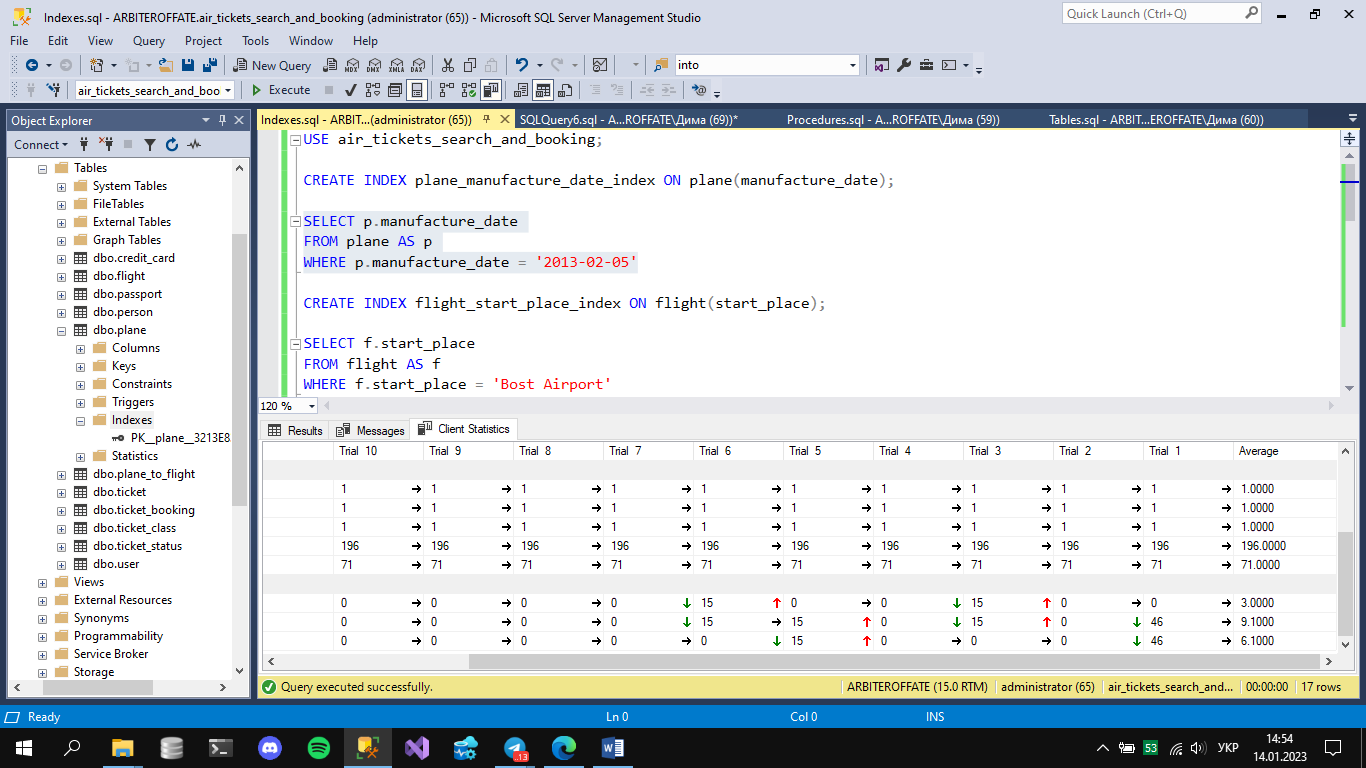


Рисунок 8.2. Середній виконання запиту 10 разів після створення індексу

SELECT p.manufacture\_date

FROM plane AS p

WHERE p.manufacture\_date = '2013-02-05'

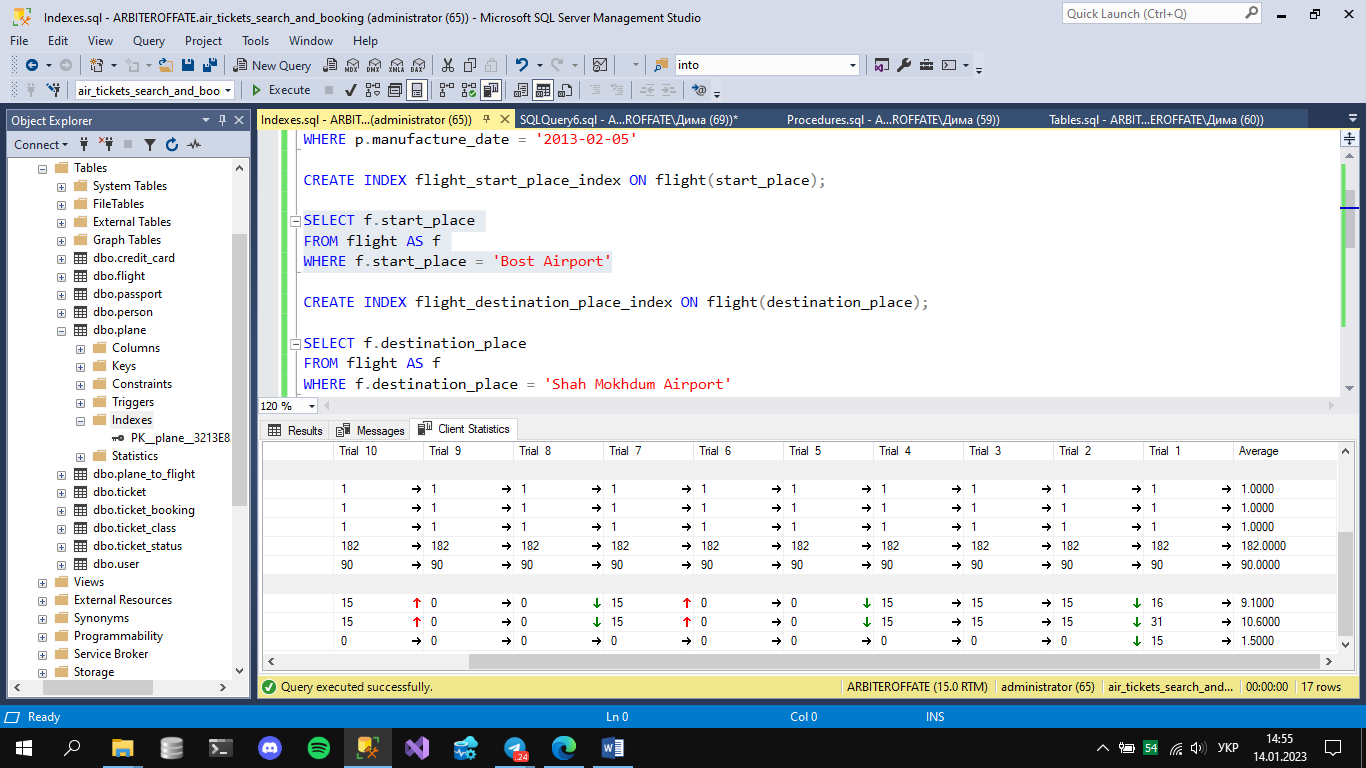


Рисунок 8.3. Середній виконання запиту 10 разів до створення індексу

CREATE INDEX flight\_start\_place\_index

ON flight(start\_place);

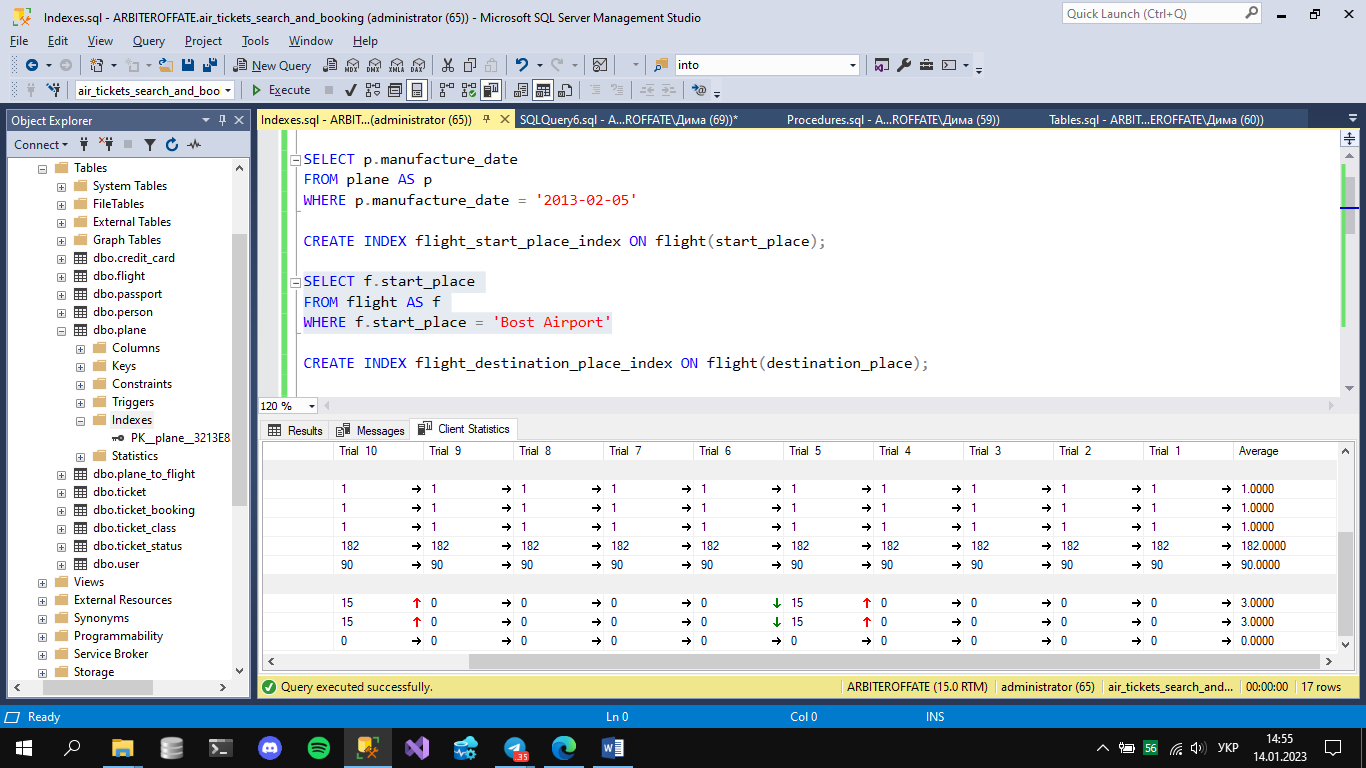


Рисунок 8.4. Середній виконання запиту 10 разів після створення індексу

SELECT f.start\_place

FROM flight AS f

WHERE f.start\_place = 'Bost Airport'

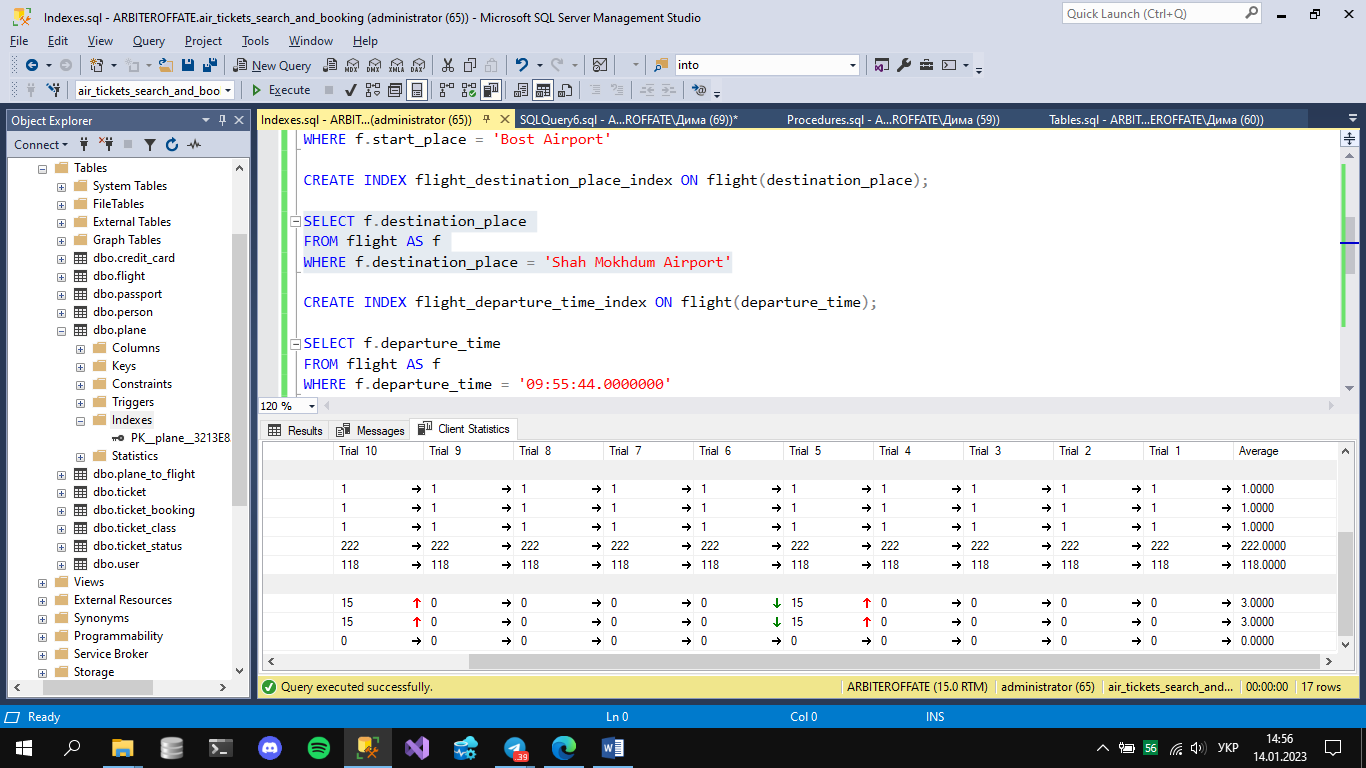


Рисунок 8.5. Середній виконання запиту 10 разів до створення індексу

CREATE INDEX flight\_destination\_place\_index

ON flight(destination\_place);

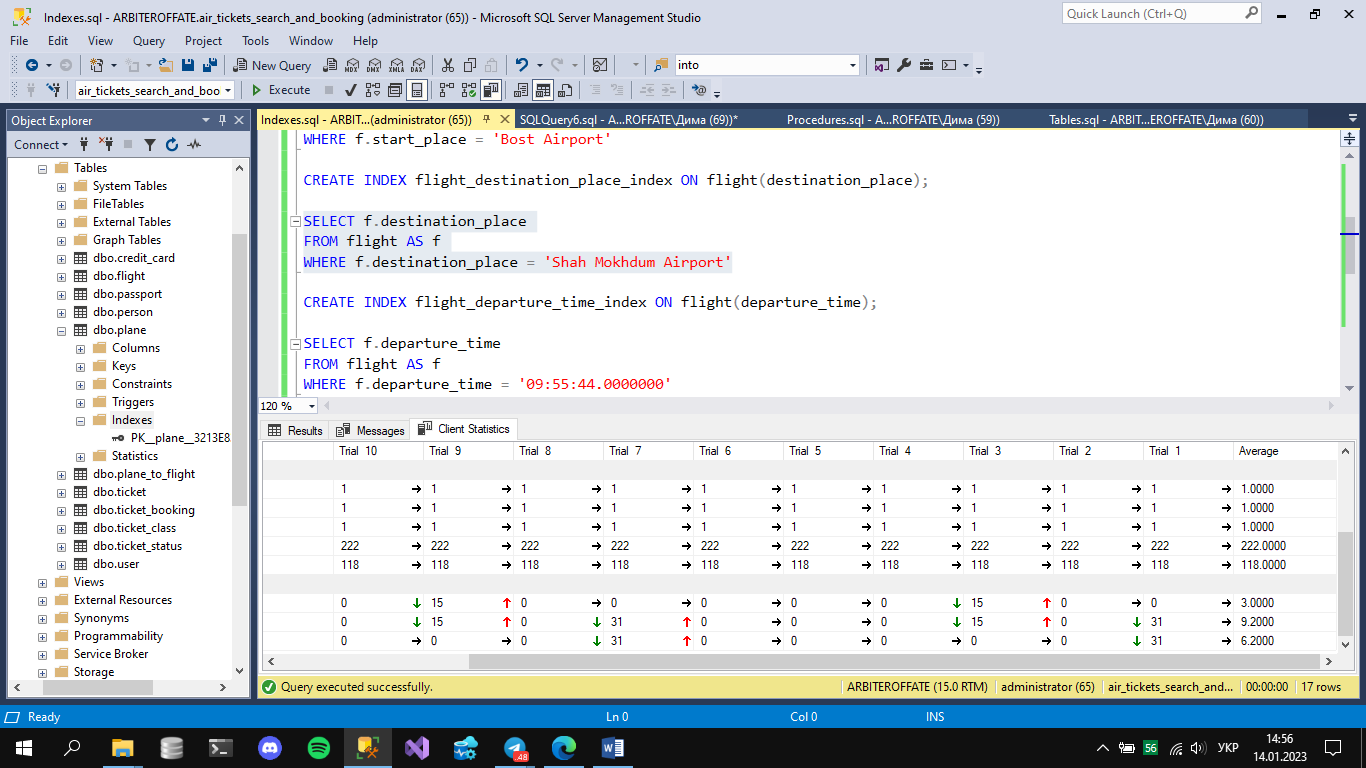


Рисунок 8.6. Середній виконання запиту 10 разів після створення індексу

SELECT f.destination\_place

FROM flight AS f

WHERE f.destination\_place = 'Shah Mokhdum Airport'

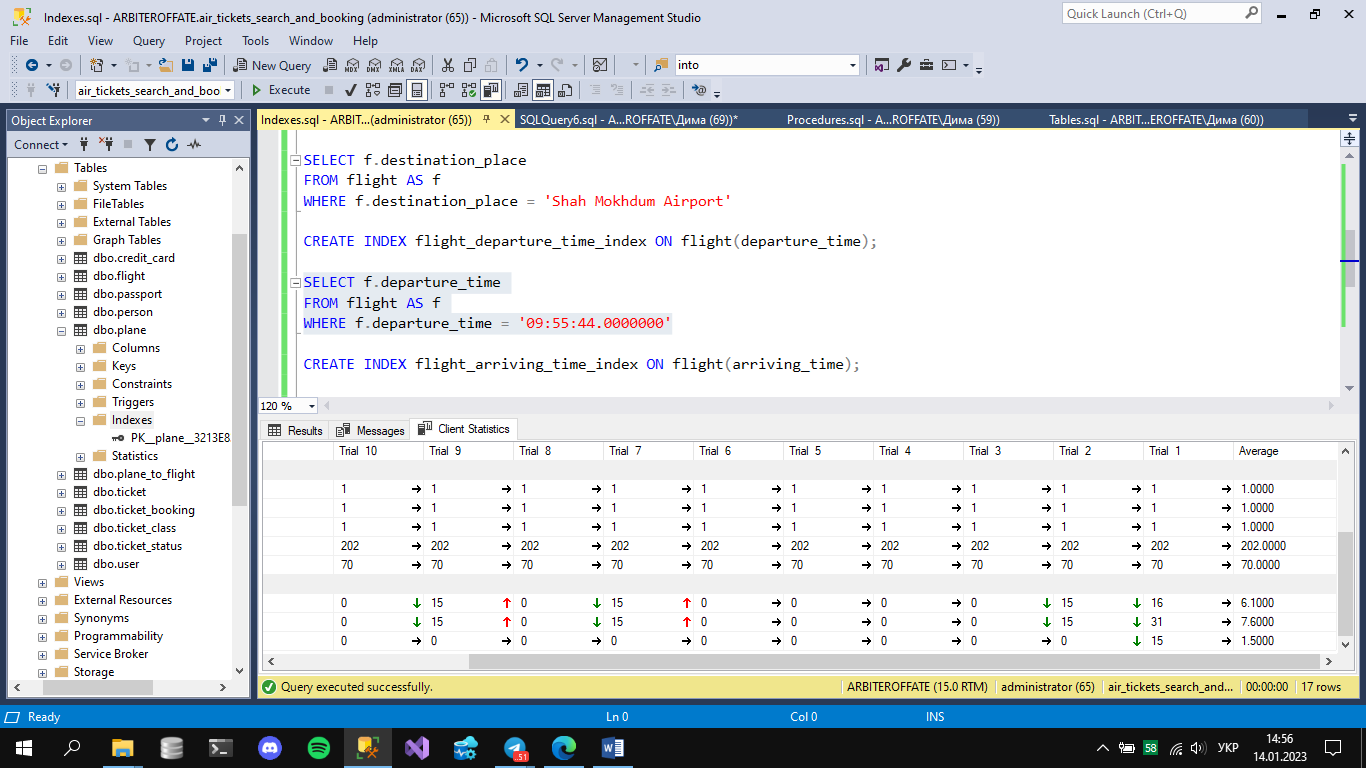


Рисунок 8.7. Середній виконання запиту 10 разів до створення індексу

CREATE INDEX flight\_departure\_time\_index

ON flight(departure\_time);

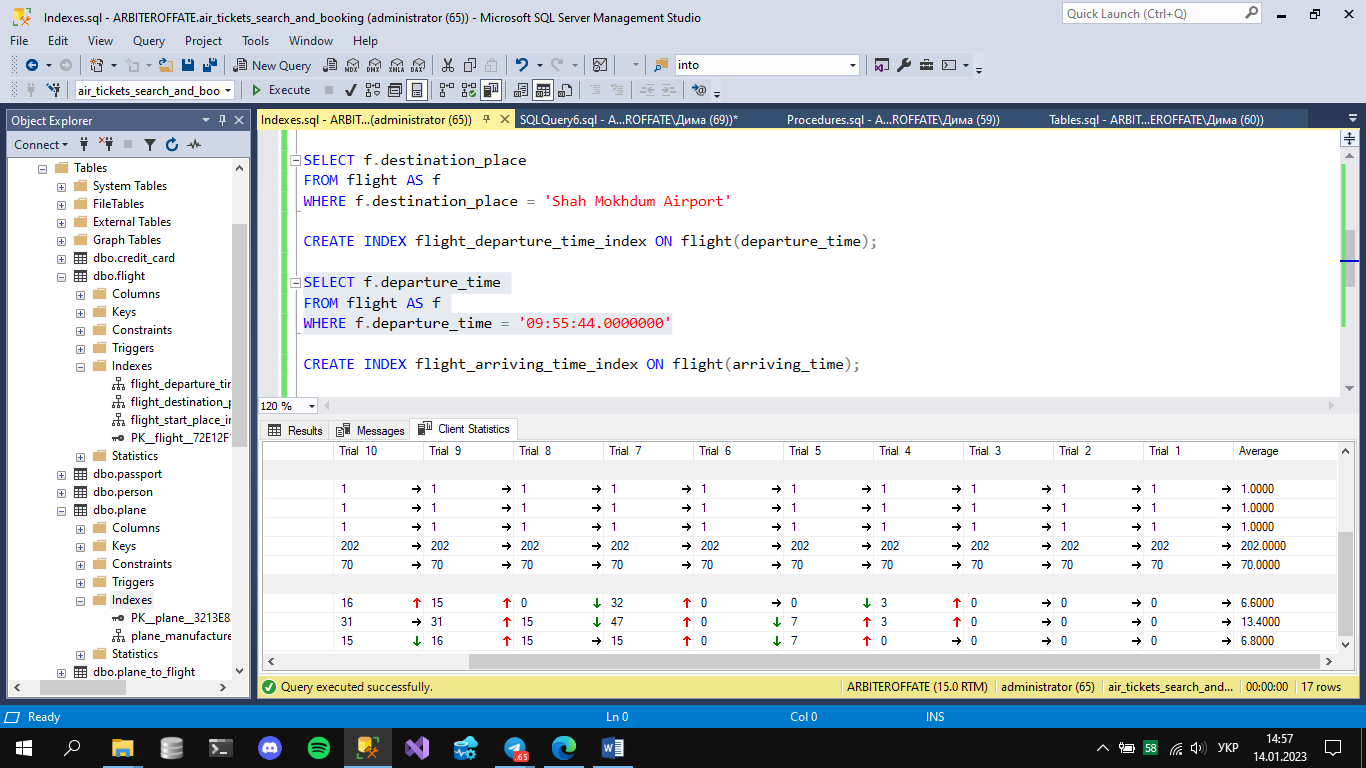


Рисунок 8.8. Середній виконання запиту 10 разів після створення індексу

SELECT f.departure\_time

FROM flight AS f

WHERE f.departure\_time = '09:55:44.0000000'

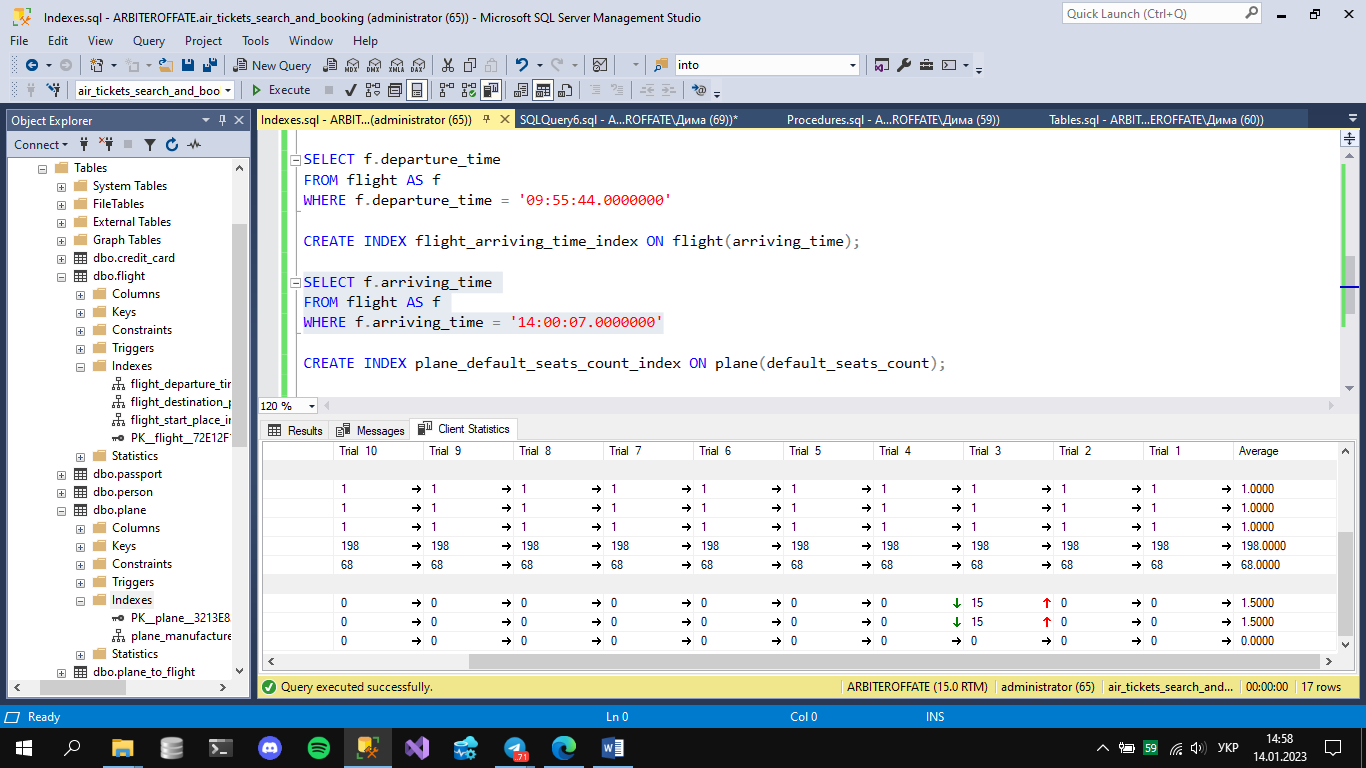


Рисунок 8.9. Середній виконання запиту 10 разів до створення індексу

CREATE INDEX flight\_arriving\_time\_index

ON flight(arriving\_time);

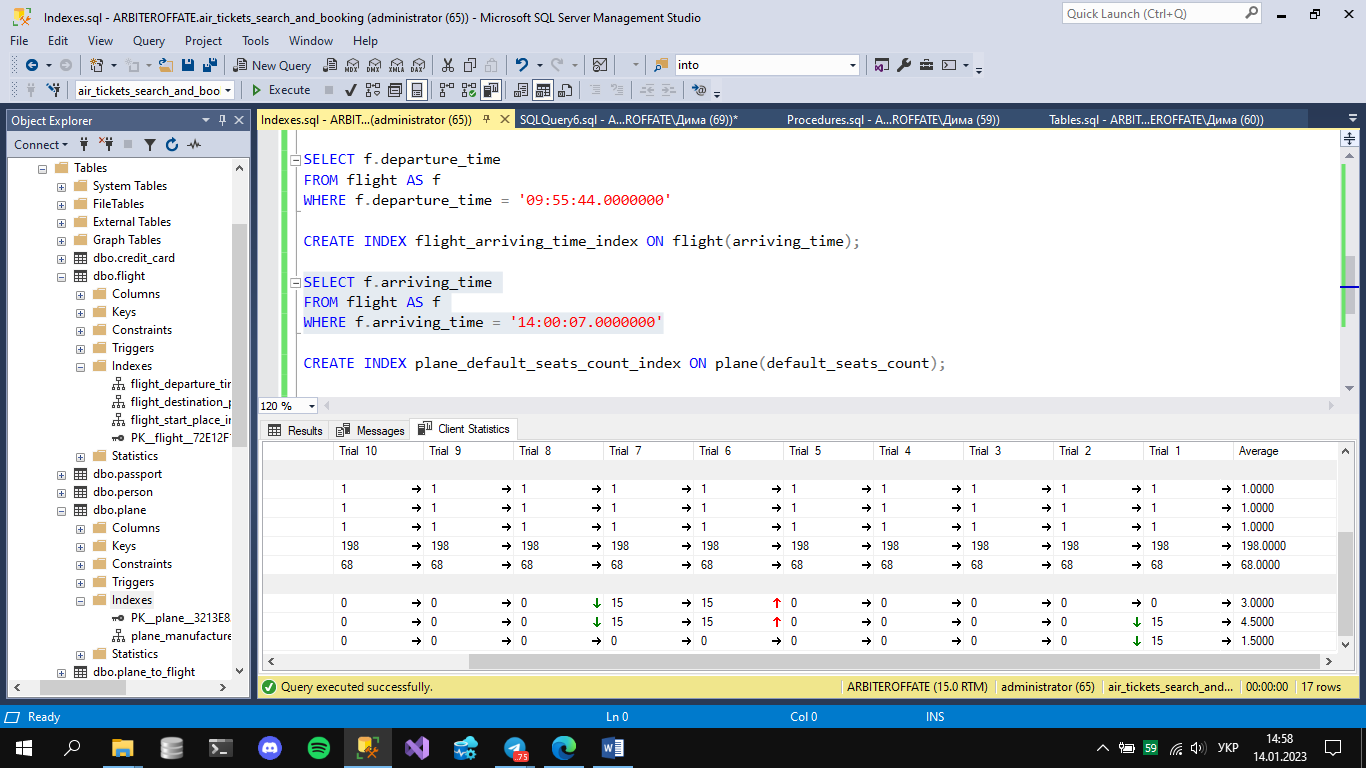


Рисунок 8.10. Середній виконання запиту 10 разів після створення індексу

SELECT f.arriving\_time

FROM flight AS f

WHERE f.arriving\_time = '14:00:07.0000000'

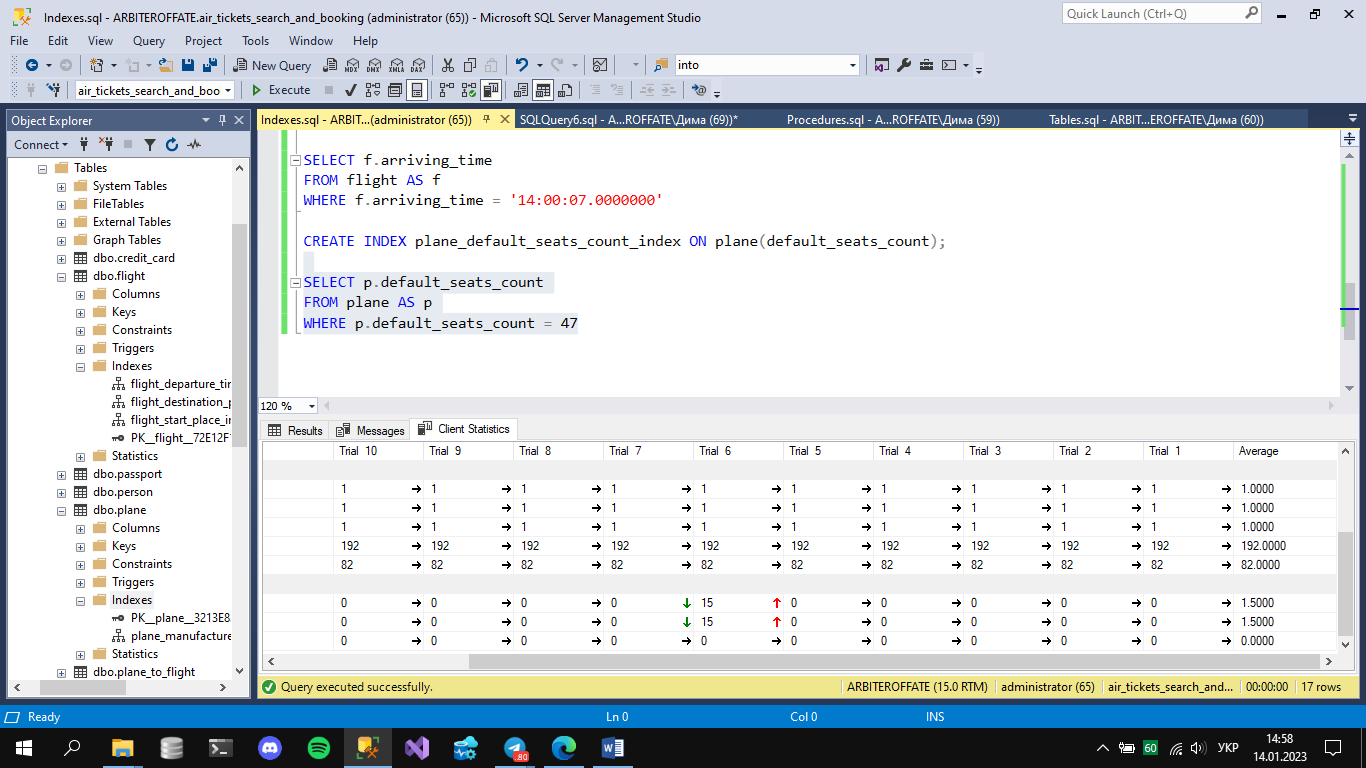


Рисунок 8.11. Середній виконання запиту 10 разів до створення індексу

CREATE INDEX plane\_default\_seats\_count\_index

ON plane(default\_seats\_count);

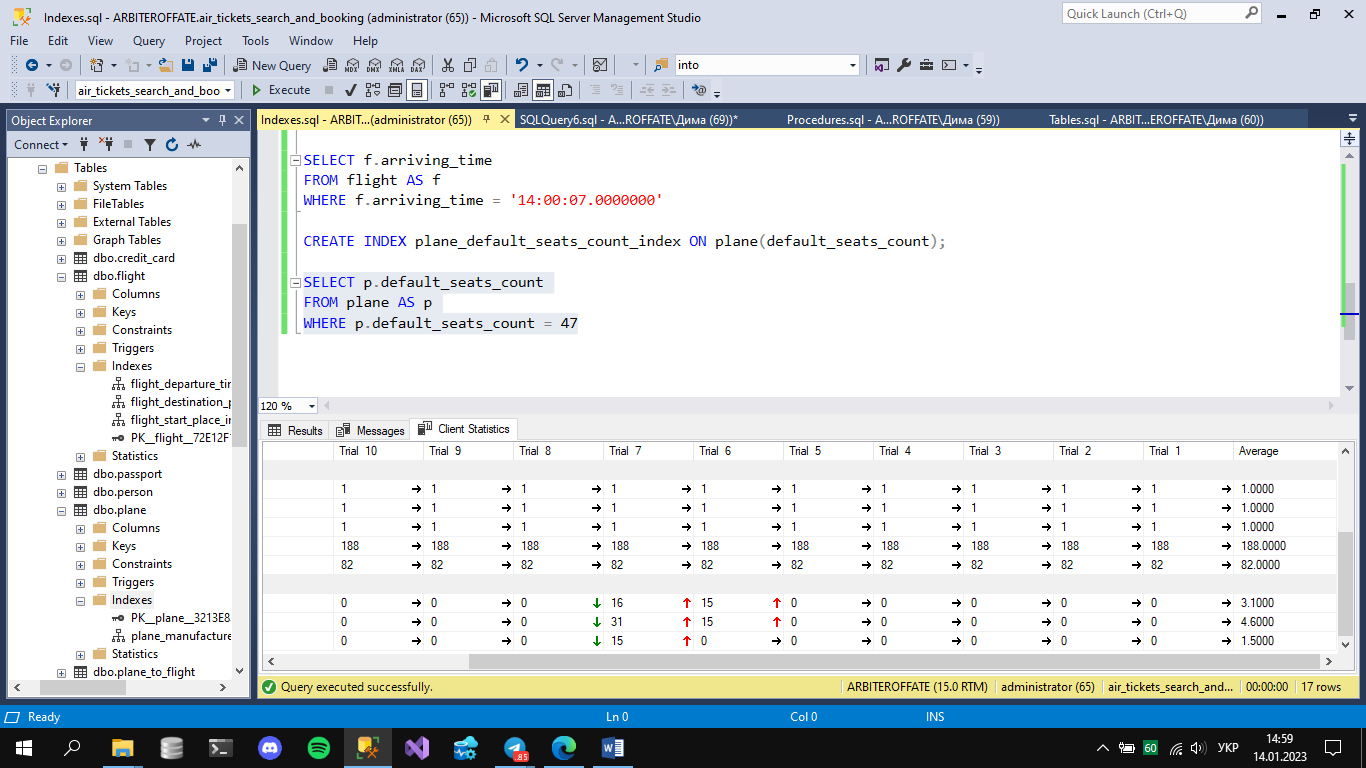


Рисунок 8.12. Середній виконання запиту 10 разів після створення індексу

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)](https://flylib.com/books/en/2.96.1.35/1/)  
         (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](https://www.mockaroo.com/)

(дата звернення – 26.12.2022)