НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра прикладної математики

КУРСОВА РОБОТА

з курсу «Бази даних та організація надвеликих баз»

на тему:

«Розроблення бази даних аеропорту і віконного додатку**»**

Виконав: студент групи ПМ-32

Кріль Павло Васильович

Науковий керівник:

асистент кафедри ПМ

Польовий Віталій Євгенович

Львів – 2022

Анотація

Кріль П.В.

Курсова роботана тему:

Розроблення бази даних аеропорту і віконного додатку

ПМ Прикладна математика

Національний університет “Львівська Політехніка”Львів 2022

Результатом виконання курсової роботи є створення програми для роботи з базою даних невеликого аеропорту. До переліку основних функцій програми можна віднести насамперед можливість змінювати, редагувати і видаляти записи користувачам з бази даних. З точки зору адміністрування програми, було реалізовано можливість відстеження активності користувачів.

Таким чином було налагоджено повноцінну діяльність системи, тобто всі її елементи правильно функціонують та взаємодіють. В ході розробки інтерфейсу програми було використано мову C#.

Перелік умовних скорочень

1. БД – база даних

Зміст

Вступ

Побудова сучасних інформаційних систем немислимо без використання технологій баз даних, які стали не тільки інструментом зберігання відомостей про предметну область, а й джерелом інформації, на базі якої можна приймати управлінські рішення, вирішувати велику кількість різнорідних завдань в різних областях діяльності людини.

Історія розвитку баз даних налічує кілька десятиліть, але навіть за такий короткий проміжок розвитку бази даних стали одним з ключових елементів комп'ютерних інформаційних систем, надавши користувачам великі можливості по обробці даних. Так, розвиток баз даних призвело до появи таких інформаційних технологій, які є розширенням технології баз даних, як: сховища даних, банки даних, бази знань і багато інших. Основу для всіх цих технологій склали рішення, сформульовані Едгаром Коддом і Пітером Ченом. Придумані ними технології зберігання і обробки даних в електронних системах стали центральними в розвитку баз даних і використовуються досі в більшості систем роботи з даними.

Розділ 1

1.1 Створення схеми БД

В процесі розробки програмного продукту з метою забезпечення максимального зручного оперування даними було використано SQL– реляційна система управління базами даних. В якості середовища для маніпуляцій даними та налагодження роботи БД в цілому було використано Microsoft SQL Server Management Studio.

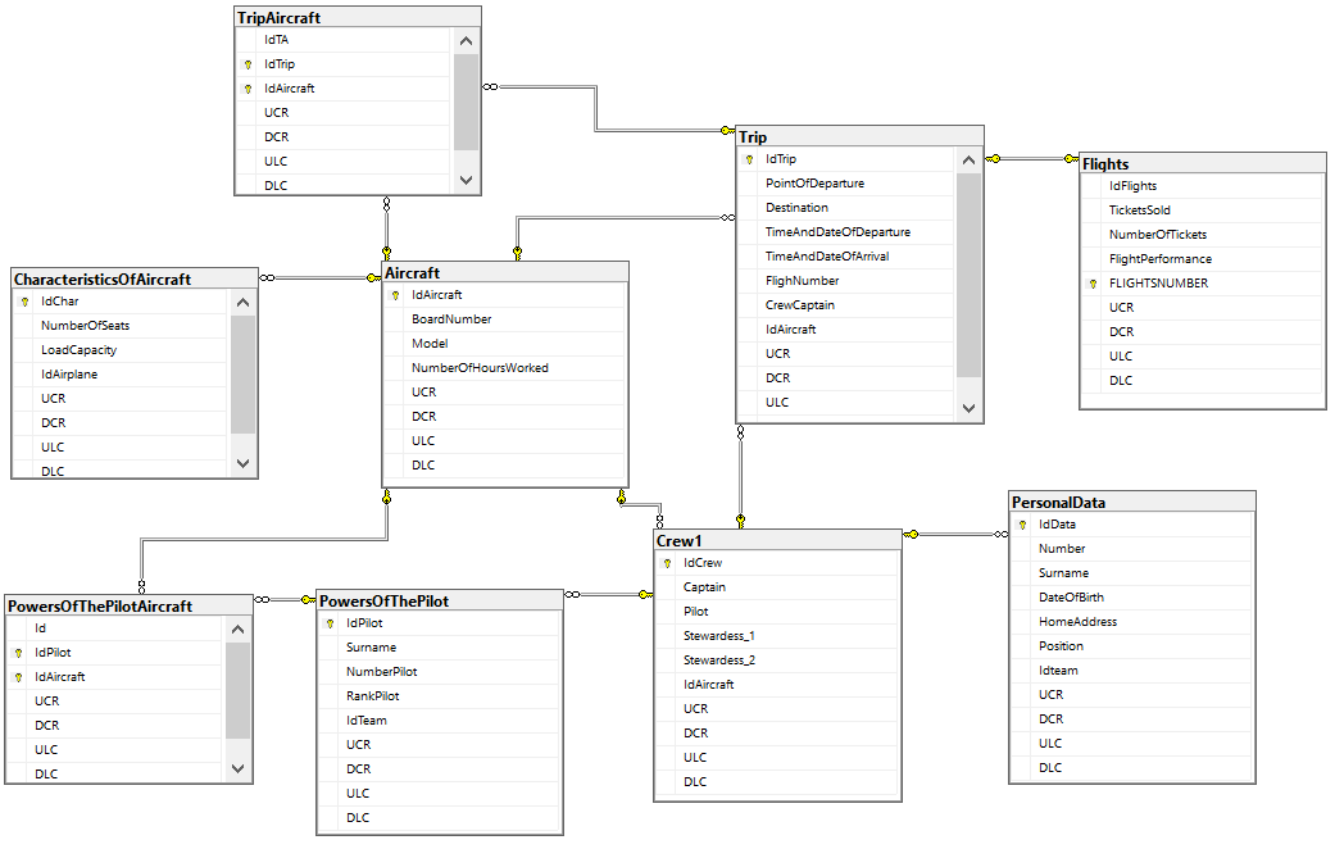


Рисунок 1 Diagram БД

Між таблицями є різні відношення в залежності від виду відношення між табличками “Trip” і “Aircraft”, відношення багато до багатьох, бо рейс може виконуватись на багатьох літаках, тому створюється проміжна табличка “TripAircraft”. Таке ж відношення між табличками “PowersOfThePilot” і “Aircraft” так, як пілот може літати на багатьох літаках, створюється проміжна табличка “PowersOfThePilotAircraft”. Між табличками “Trip ” і “Flights” відношення один до одного, так як інформація про рейс тільки одна для конкретного рейсу. Між табличками “Trip” і “Aircraft” існує ще зв’язок один до багатьох так, як рейс може виконуватись різних літаках. Таблички “CharacteristicsOfAircreft” і “Aircraft” мають відношення один до багатьох, бо літаки одної моделі можуть мати різні характеристики. Між табличками “Crew1 ” і “PersonalData” відношення один до багатьох, бо одна команда має персональні дані про всіх учасників. Таблички “Aircraft” і “Crew1” мають відношення один до багатьох, бо екіпаж може літати на багатьох літаках. Таблички “Crew1” і “Trip” мають відношення один до багатьох, бо один екіпаж може виконувати

багато рейсів. Таблички “Crew1” і “PowersOfThePilot” мають відношення один до багатьох, так як в команді два пілота.

Наступні зображення демонструють стандартний вигляд кожної з таблиць в БД.

Aircraft

Column Name	Data Type	Allow Nulls
IdAircraft	int	☐
BoardNumber	nvarchar(50)	☐
Model	nvarchar(50)	☐
NumberOfHoursWorked	int	☐
UCR	varchar(50)	☑
DCR	datetime	☑
ULC	varchar(50)	☑
DLC	datetime	☑
		☐

Рисунок 1.1. Структура таблиці літаки.

В ході розробки логічної моделі бази даних було насамперед створено згідно предметної області табличку “Aircraft” , яка включає поля для зберігання бортового номера(тип даних nvarchar(50)), моделі (тип даних nvarchar(50)), кількості відпрацьованих годин(тип даних int) для праввильного функціонування і зв’язків табличок створив IdAircraft і надав йому значення PRIMARY KEY (первинний ключ).Також в ній присутні поля для відслідкування роботи користувачів поля UCR,DCR,ULC,DLC і зберігають дані про запис і зміну полів таблички.

CharacteristicsOfAircraft

Column Name	Data Type	Allow Nulls
IdChar	int	☐
NumberOfSeats	int	☐
LoadCapacity	int	☐
IdAirplane	int	☐
UCR	varchar(50)	☑
DCR	datetime	☑
ULC	varchar(50)	☑
DLC	datetime	☑
		☐

Рисунок 1.2. Структура таблиці характеристики літаків.

IdChar – первинний ключ, поле для ідентифікації характеристики літака, тип даних- int.

NumberOfSeats – поле для зберігання кількості місць в літаку, тип даних – int.
LoadCapacity – поле з інформацією про вантажопідйомність літака, тип даних – int.
IdAirplane – зовнішній ключ, поле для ідентифікації літака, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних – varchar(50).
DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.

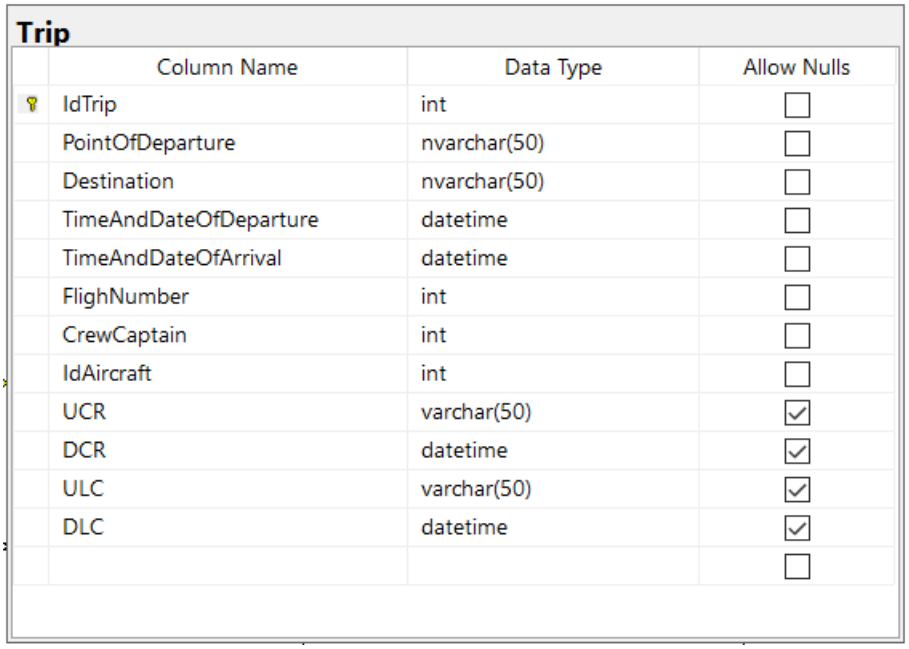


Рисунок 1.3. Структура таблиці Рейси.

IdTrip – первинний ключ, поле для ідентифікації рейсу, тип даних – int.

PointOfDeparture – поле для зберігання пункту відправки літака, тип даних – nvarchar(50).
Destination – поле з інформацією про пункт призначення літака, тип даних – nvarchar(50).
TimeAndDateOfDeparture – зберігає час і дату вильоту, тип даних – datatime.
TimeAndDateOfArrival – зберігає час і дату посадки, тип даних – datatime.
FlighNumber – поле для ідентифікації характеристик рейсу, тип даних – int.
CrewCaptain – поле для ідентифікації команди рейсу, тип даних – int.
IdAircraft – поле для зберігання ідентифікатора літака, що виконує рейс, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних – varchar(50).

DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.

TripAircraft

	Column Name	Data Type	Allow Nulls
	IdTA	int	☐
🔑	IdTrip	int	☐
🔑	IdAircraft	int	☐
	UCR	varchar(50)	☑
	DCR	datetime	☑
	ULC	varchar(50)	☑
	DLC	datetime	☑
			☐

Рисунок 1.4. Структура проміжної таблиці Рейс Літак.

IdTA – поле для ідентифікації даних , тип даних – int.

IdTrip – первинний ключ, поле для ідентифікації рейсу, тип даних – int.
IdAircraft – первинний ключ, поле для ідентифікації літака, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних – varchar(50).
DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.

Crew1

	Column Name	Data Type	Allow Nulls
🔑	IdCrew	int	☐
	Captain	nvarchar(50)	☐
	Pilot	nvarchar(50)	☐
	Stewardess_1	nvarchar(50)	☐
	Stewardess_2	nvarchar(50)	☐
	IdAircraft	int	☐
	UCR	varchar(50)	☑
	DCR	datetime	☑
	ULC	varchar(50)	☑
	DLC	datetime	☑
			☐

Рисунок 1.5. Структура таблиці Екіпаж.

IdCrew – первинний ключ, поле для ідентифікації екіпажу, тип даних- int.

Captain – поле для зберігання інформації про капітана екіпажу, тип даних – nvarchar(50).
Pilot - поле з інформацією про пілота екіпажу, тип даних – nvarchar(50).
Stewardess_1 – поле з інформацією про стюардесу екіпажу, тип даних –

nvarchar(50).
Stewardess_2 – поле з інформацією про другу стюардесу екіпажу, тип даних – nvarchar(50).
FlighNumber – поле для ідентифікації характеристик рейсу, тип даних – int.
CrewCaptain – поле для ідентифікації команди рейсу, тип даних – int.
IdAircraft – поле для зберігання ідентифікатора літака на якому літає екіпаж, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних – varchar(50).
DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.

PersonalData

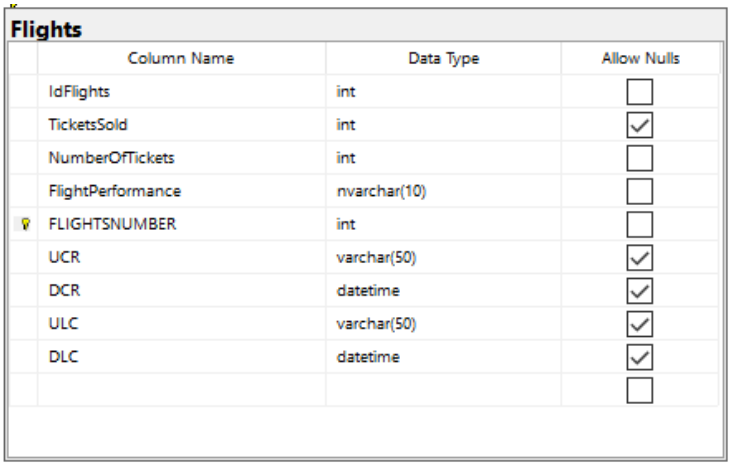
	Column Name	Data Type	Allow Nulls
🔑	IdData	int	☐
	Number	int	☐
	Surname	nvarchar(50)	☐
	DateOfBirth	date	☐
	HomeAddress	nvarchar(50)	☐
	Position	nvarchar(50)	☐
	Idteam	int	☐
	UCR	varchar(50)	☑
	DCR	datetime	☑
	ULC	varchar(50)	☑
	DLC	datetime	☑
			☐

Рисунок 1.6. Структура таблиці Персональні дані.

IdData – первинний ключ, поле для ідентифікації персональних даних, тип даних- int.

Number – поле для зберігання номера члена екіпажу, тип даних – int.
Surname – поле з інформацією про прізвище члена екіпажу, тип даних – nvarchar(50).
DataOfBirth – поле з інформацією про дату народження члена екіпажу, тип даних – data.
HomeAddress – поле з інформацією адресу, тип даних – nvarchar(50).
Position – поле для зберігання даних про роль члена екіпажу, тип даних – nvarchar(50).
Idteam – поле для зберігання ідентифікатора команди екіпажу, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних – varchar(50).

DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.



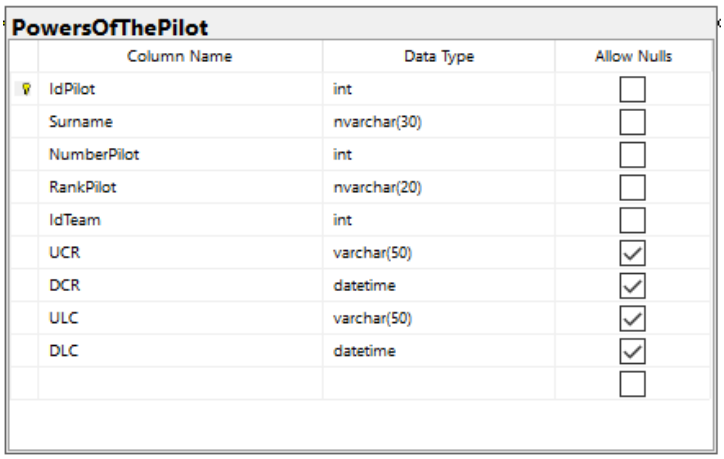
Flights

Column Name	Data Type	Allow Nulls
IdFlights	int	☐
TicketsSold	int	☑
NumberOfTickets	int	☐
FlightPerformance	nvarchar(10)	☐
FLIGHTSNUMBER	int	☐
UCR	varchar(50)	☑
DCR	datetime	☑
ULC	varchar(50)	☑
DLC	datetime	☑
		☐

Рисунок 1.7. Структура таблиці Рейси.

IdFlights – поле для ідентифікації даних про рейс, тип даних- int.

TicketsSold – поле для зберігання кількості проданих квитків на рейс, тип даних – int.
NumberOfTickets – поле з інформацією про кількість квитків, тип даних – int.
FlightPerformance – поле з інформацією про те чи відбувся рейс чи ні, тип даних – nvarchar(10).
FLIGHTSNUMBER – первинний ключ, поле для ідентифікації рейсу, тип даних – int.
UCR – поле з ім’ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім’я користувача, що останнім змінив даний запис, тип даних – varchar(50).
DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.
В табличці передбачено Check обмеження яке спрацьовує для колонки FlightPerformance, якщо кількість проданих квитків на рейс менша за половину, то рейс не виконується.



PowersOfThePilot

Column Name	Data Type	Allow Nulls
IdPilot	int	☐
Surname	nvarchar(30)	☐
NumberPilot	int	☐
RankPilot	nvarchar(20)	☐
IdTeam	int	☐
UCR	varchar(50)	☑
DCR	datetime	☑
ULC	varchar(50)	☑
DLC	datetime	☑
		☐

Рисунок 1.8. Структура таблиці Повноваження пілота.

IdPilot – первинний ключ, поле для ідентифікації пілота, тип даних – int.
Surname – поле з інформацією про прізвище пілота, тип даних – nvarchar(30).
NumberPilot – поле з інформацією про номер пілота, тип даних – int.
RankPilot – поле з інформацією рейтинг пілота, тип даних – nvarchar(20).
IdTeam – поле для ідентифікації команди пілота, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних – varchar(50).
DLC – поле з датою та часом останньої модифікації даного запису, тип даних – datatime.

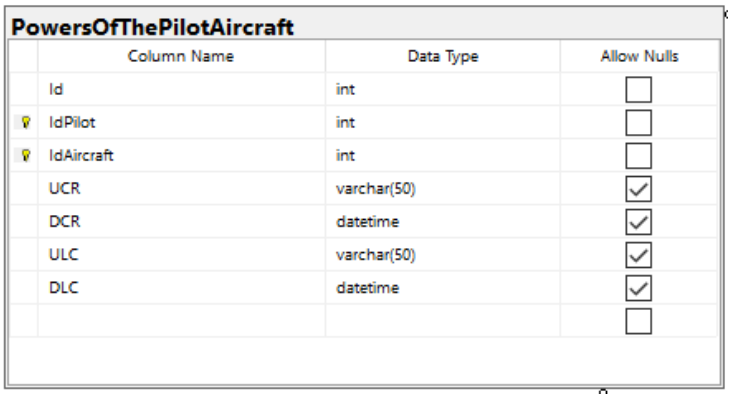


Рисунок 1.9. Структура проміжної таблиці Повноваження пілота Літак .

Id – поле для ідентифікації даних , тип даних- int.

IdPilot – первинний ключ, поле для ідентифікації пілота, тип даних – int.
IdAircraft – первинний ключ, поле для ідентифікації літака, тип даних – int.
UCR – поле з ім'ям користувача, що створив даний запис, тип даних – varchar(50).
DCR – поле з датою та часом створення даного запису, тип даних – datatime.
ULC – поле для зберігання ім'я користувача, що останнім змінив даний запис, тип даних - varchar(50).
DLC – поле з датою та часом останньої модифікації даного запису, тип даних - datatime.

1.2 Написання SQL-запитів

Ключове слово SELECT означає запит на подання інформації. Вона буде подана у вигляді результуючої таблиці стрічки якої задовольнятимуть умові. Стовпчики, на основі яких формуються результуючі або перевіряється умова, повинні належати таблицям перерахованим у списку.
Якщо список результуючих стовпчиків співпадає із списком єдиної таблиці-джерела, то такий список зручно представляти у скороченому вигляді за допомогою зірочки – *.
Порядок результуючих стовпчиків може бути довільним.
Sql запити нам потрібні для зручності в роботі і економії часу так як за допомогою 1 запиту можна зекономити багато часу якби ми шукали і рахували необхідні дані в ручну в нашій БД.

SELECT на базі таблиці CharacteristicsOfAircraft з використанням сортування по спаданню, з накладенням умов зі зв'язками OR та AND.
Запит:

```
SELECT * FROM CharacteristicsOfAircraft
  WHERE IdChar<7 AND IdAirplane<=5 OR NumberOfSeats< 700
    ORDER BY IdAirplane  DESC.
```

	IdChar	NumberOfSeats	LoadCapacity	IdAirplane	UCR	DCR	ULC	DLC
1	10	51	2400	10	NULL	NULL	DESKTOP-4SCTOQE\Admin	2022-06-07 15:25:47.650
2	9	300	2500	9	NULL	NULL	NULL	NULL
3	8	250	2300	8	NULL	NULL	NULL	NULL
4	5	50	2200	5	NULL	NULL	NULL	NULL
5	4	800	3100	4	NULL	NULL	NULL	NULL
6	3	150	2100	3	NULL	NULL	NULL	NULL
7	2	70	4000	2	NULL	NULL	NULL	NULL
8	1	50	3000	1	NULL	NULL	NULL	NULL

Рисунок 1.10. Результат виконання запиту.

SELECT на базі таблиці Flights з виводом кількості доступних білетів в колонку результату NumberFreePlaces.
Запит:

```
SELECT DISTINCT NumberOfTickets,FLIGHTSNUMBER, NumberOfTickets-TicketsSold AS
NumberFreePlaces FROM Flights
```

	NumberOfTickets	FLIGHTSNUMBER	NumberFreePlaces
1	50	1	26
2	50	2	30
3	50	3	20
4	750	4	350
5	50	5	40
6	50	6	19
7	300	7	148

Рисунок 1.11. Результат виконання запиту.

SELECT з використанням оператора Like.

Запит:

```
SELECT* FROM [PersonalData ]
WHERE Surname Like '%е%';
```

	IdData	Number	Surname	DateOfBirth	HomeAddress	Position	Idteam	UCR	DCR	ULC	DLC
1	2	2	Семенець	1970-06-21	м. Львів Генерала Чупринки 10	Pilot	1	NULL	NULL	NULL	NULL
2	3	1	Семенів	1985-04-23	м. Львів Генерала Чупринки 12	Captain	1	NULL	NULL	NULL	NULL
3	4	3	Семенюк	1998-05-05	м. Львів Генерала Чупринки 16	Stewardess_1	1	NULL	NULL	NULL	NULL

Рисунок 1.12. Результат виконання запиту.

Ієрархічний SELECT запит буде видавати всіх пілотів хто може керувати літаком з IdAircraft = 3.

Запит:

```
SELECT PowersOfThePilotAircraft.IdPilot,Litak.IdAircraft
FROM PowersOfThePilotAircraft INNER JOIN PowersOfThePilotAircraft Litak
ON PowersOfThePilotAircraft.IdAircraft=Litak.IDPilot WHERE Litak.IdAircraft= 3
```

	IdPilot	IdAircraft
1	1	3
2	3	3
3	7	3

Рисунок 1.13. Результат виконання запиту.

SELECT запит типу CrossTab буде видавати інформацію про кількість літаків, які можуть здійснити необхідні нам рейси.

Запит:

```
SELECT * From TripAircraft
PIVOT
(COUNT(IdAircraft)
FOR IdTrip
IN ([1],[3] ,[4],[5], [10])
 ) pvt;
```

	1	3	4	5	10
1	3	3	0	3	5

Рисунок 1.13. Результат виконання запиту.

1.3 Підпрограми СУБД Oracle

Збережені процедури - це об'єкти бази даних, в яких закладено алгоритм у вигляді набору SQL інструкцій. Іншими словами, можна сказати, що збережені процедури - це програми всередині бази даних. Збережені процедури використовуються для збереження на сервері повторно використовуваного коду, нам потрібно повторно викликати нашу процедуру яка викликає функцію для рейтингу кожен місяць рейтинг змінюється щоб не писати її кожен раз ми створили збережену процедуру і просто її запускаєм.

```
CREATE OR ALTER PROCEDURE CalculateAllRanks AS
BEGIN
    UPDATE PowersOfThePilot SET RankPilot = dbo.CalculatePilotRank(IdPilot)
END
GO
```

Рисунок 1.14. Процедура яка оновлює усі стовпці RankPilot для всіх пілотів.

```
CREATE OR ALTER FUNCTION CalculatePilotRank(@pid INT)
RETURNS INT
AS
BEGIN
    DECLARE @hoursin INT                -- Кількість годин в польоті
    DECLARE @flights INT                -- Кількість польотів
    DECLARE @aircrafts INT              -- Кількість літаків якими пілот може літати
    DECLARE @pilotName NVARCHAR(30)     -- Ім'я пілота
    -- Константи для формули
    DECLARE @hoursin_value FLOAT = 0.1
    DECLARE @flights_value FLOAT = 10
    DECLARE @aircrafts_value FLOAT = 3

    SET @pilotName = (SELECT TOP 1 Surname FROM PowersOfThePilot WHERE IdPilot = @pid)  -- Визначення імені пілота
    -- Визначення кількості годин (Сумма різниці [час приліту]-[час виліту] для усіх польотів, у яких пілотом був пілот з ім'ям @pilotName)
    SET @hoursin =
    (SELECT SUM(DATEDIFF(hour, TimeAndDateOfDeparture, TimeAndDateOfArrival)) FROM Trip
    LEFT JOIN
    Crew1 ON Trip.CrewCaptain = Crew1.IdCrew
    WHERE Crew1.Pilot = @pilotName GROUP BY Pilot)
    -- Визначення кількості польотів (Кількість польотів, у яких пілотом був пілот з ім'ям @pilotName)
    SET @flights =
    (SELECT COUNT(*) FROM Trip
    LEFT JOIN
    Crew1 ON Trip.CrewCaptain = Crew1.IdCrew
    WHERE Crew1.Pilot = @pilotName GROUP BY Pilot)
    -- Визначення кількості літаків (Кількість літаків, у яких айді пілота дорівнює @pid)
    SET @aircrafts = (SELECT COUNT(IdAircraft) FROM PowersOfThePilotAircraft WHERE IdPilot = @pid)
    -- Визначення рейтингу за формулою [кількість годин]/10 + [кількість польотів]*10 + [кількість літаків]*3
    RETURN ((@hoursin*@hoursin_value) + @flights*@flights_value + @aircrafts*@aircrafts_value)
END
GO
```

Рисунок 1.15. Функція визначення рейтингу.

В функції ми декларуєм змінні: кількість годин в польоті, кількість польотів, кількість літаків якими може літати і ім’я пілота.Також задекларував константи для кожного з параметру і призначив їм значення. Згідно з беремо дані з таблиці Trip і шукаєм різницю між часов вильоту і часом прибуття для всіх пілотів з ім’ям @pilotName.Наступний параметр визначаємо кількість вильотів пілота з ім’ям @pilotName. Знаходим з проміжної таблички “PowersOfThePilotAircraft” кількість літаків на яких може літати пілот. Процедура повертає знайдені наші значення помножені на декларовані константи.

Для виклику процедури використовуєм команду: EXEC CalculateAllRanks.

```
(5 rows affected)

Completion time: 2022-06-14T16:30:12.3670307+03:00
```

Рисунок 1.16. Повідомлення про успішне виконання процедури.

IdPilot	Surname	NumberPilot	RankPilot	IdTeam
1	Семенець	6	33	1
2	Шевчук	2	38	2
3	Бондаренко	3	19	3
4	Коваль	4	19	4
5	Руденко	5	21	5

Рисунок 1.17. Оновлений рейтинг пілотів після виконання CalculateAllRanks.

1.4 Тригери в СУБД Oracle

Три́гер — це збережена процедура особливого типу, яку користувач не викликає явно, а використання якої обумовлено настанням визначеної події (дії) у реляційній базі даних:

- додаванням INSERT,
- вилученням рядка в заданій таблиці DELETE,
- або зміною даних у певному стовпці заданої таблиці UPDATE.

Тригери застосовуються для забезпечення цілісності даних і реалізації складної бізнес-логіки. Тригер запускається сервером автоматично при спробі зміни даних у таблиці, з якою він пов'язаний. Всі здійснені ним модифікації даних розглядаються як виконані в транзакції, в якій виконано дію, що викликало спрацьовування тригера. Відповідно, у разі виявлення помилки або порушення цілісності даних може відбутися відкат цієї транзакції.

```
CREATE OR ALTER TRIGGER [dbo].[Crew3daysCheck] ON [dbo].[Crew1] FOR INSERT, UPDATE AS
BEGIN
    DECLARE @pilot NVARCHAR(30) = (SELECT Pilot FROM inserted)

    IF
    (SELECT COUNT(*) FROM Trip AS T1
    CROSS JOIN Trip AS T2
    WHERE (T1.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Pilot = @pilot)) AND

    (  (T1.TimeAndDateOfArrival >= T2.TimeAndDateOfDeparture AND T1.TimeAndDateOfDeparture <= T2.TimeAndDateOfArrival) OR
        DATEDIFF(day, T1.TimeAndDateOfArrival, T2.TimeAndDateOfDeparture) BETWEEN 0 AND 3 OR
        DATEDIFF(day, T2.TimeAndDateOfArrival, T1.TimeAndDateOfDeparture) BETWEEN 0 AND 3)) != 0
    BEGIN
        RAISERROR('Повинна бути перерва 3 дні між вильотами!', 10, 1)
        ROLLBACK TRANSACTION
    END
END
```

Рисунок 1.18. Тригер визначення що у пілота перерва між вильотами 3 дні.

Тригер спрацьовує для таблички Crew1, ми декларуєм ім’я пілота і перевіряєм чи між датою прильоту пілота з заданим іменем, і нового призначення на рейс пройшло 3 дні, якщо перерва менша то виводимо помилку.

Перевірка чи працює наш тригер, якщо ми перепризначимо пілота, але перерва між вильотами менша 3 днів отримаємо повідомлення:

Error Message: The transaction ended in the trigger. The batch has been aborted.

Повинна бути перерва 3 дні між вильотами!

Рисунок 1.19. Спрацював тригер.

Триггер визначає, що одна і та сама команда не буде призначена на декілька рейсів одночасно, він не дозволяє призначити кільком рейсам одну і ту саму команду, викликається при операціях INSERT, UPDATE.
Тут нам потрібна додаткова функція, яка повинна відслідковувати перетини між вильотами:

```
CREATE OR ALTER FUNCTION dateOverlap(@d1s DATETIME, @d1e DATETIME, @d2s DATETIME, @d2e DATETIME)
RETURNS BIT AS
BEGIN
    IF (@d1s <= @d2e) AND (@d1e >= @d2s)
        RETURN 1
    RETURN 0
END
GO
```

Рисунок 1.20. Функція відслідкування перетину між рейсами.

Триггер визначення що одна і та сама команда не буде призначена на кілька рейсів одночасно:

```
CREATE OR ALTER TRIGGER CheckTrips ON Crew1 FOR INSERT, UPDATE AS
BEGIN
        DECLARE @captain NVARCHAR(30) = (SELECT Captain FROM inserted)  -- Отримуємо капітана зміненої команди
        DECLARE @pilot NVARCHAR(30) = (SELECT Pilot FROM inserted)-- Отримуємо пілота зміненої команди
        DECLARE @s1 NVARCHAR(30) = (SELECT Stewardess_1 FROM inserted)-- Отримуємо першу стюардесу зміненої команди
        DECLARE @s2 NVARCHAR(30) = (SELECT Stewardess_2 FROM inserted)-- Отримуємо другу стюардесу зміненої команди
        -- Перевірка перетину польотів капітана
        -- Для перевірки, генеруємо усі можливі комбінації перетинів дат польотів за допомогою CROSS JOIN таблиці Trip
        IF
        (SELECT COUNT(*)
        FROM Trip AS T1
        CROSS JOIN Trip AS T2
        -- Враховуємо тільки ті записи, де капітан той, який в зміненої команди
        WHERE T1.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Captain = @captain) AND T2.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Captain = @captain) AND
        -- Відкидаємо те значення, яке вставлене
        T1.IdTrip != T2.IdTrip AND
        -- Якщо дати перетинаються
        dbo.dateOverlap(T1.TimeAndDateOfDeparture, T1.TimeAndDateOfArrival, T2.TimeAndDateOfDeparture, T2.TimeAndDateOfArrival) = 1) != 0 -- Якщо хоча б одна дата перетинається - скасовуємо зміни
        BEGIN
                RAISERROR('Дати капітана перетинаються!', 10, 1)
                ROLLBACK
        END
        -- Перевірка перетину польотів пілота
        IF
        (SELECT COUNT(*)
        FROM Trip AS T1
        CROSS JOIN Trip AS T2
        WHERE T1.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Pilot = @pilot) AND T2.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Pilot = @pilot) AND
        T1.IdTrip != T2.IdTrip AND
        dbo.dateOverlap(T1.TimeAndDateOfDeparture, T1.TimeAndDateOfArrival, T2.TimeAndDateOfDeparture, T2.TimeAndDateOfArrival) = 1) != 0
        BEGIN
                RAISERROR('Дати пілота перетинаються!', 10, 1)
                ROLLBACK
        END

        -- Перевірка перетину польотів першої стюардеси
        IF
        (SELECT COUNT(*)
        FROM Trip AS T1
        CROSS JOIN Trip AS T2
```

```
        WHERE T1.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Stewardess_1 = @s1) AND
T2.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Stewardess_1 = @s1) AND
        T1.IdTrip != T2.IdTrip AND
        dbo.dateOverlap(T1.TimeAndDateOfDeparture, T1.TimeAndDateOfArrival, T2.TimeAndDateOfDeparture,
T2.TimeAndDateOfArrival) = 1) != 0
        BEGIN
                RAISERROR('Дати першої стюардеси перетинаються!', 10, 1)
                ROLLBACK
        END

        -- Перевірка перетину польотів другої стюардеси
        IF
        (SELECT COUNT(*)
        FROM Trip AS T1
        CROSS JOIN Trip AS T2
        WHERE T1.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Stewardess_2 = @s2) AND
T2.CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Stewardess_2 = @s2) AND
        T1.IdTrip != T2.IdTrip AND
        dbo.dateOverlap(T1.TimeAndDateOfDeparture, T1.TimeAndDateOfArrival, T2.TimeAndDateOfDeparture,
T2.TimeAndDateOfArrival) = 1) != 0
        BEGIN
                RAISERROR('Дати другої стюардеси перетинаються!', 10, 1)
                ROLLBACK
        END
END
GO
```

Перевіримо чи працює тригер при призначенні стюардеси на на два рейси.

```
Error Message: The transaction ended in the trigger. The batch has
been aborted.
Дати першої стюардеси перетинаються!
```

Рисунок 1.21. Спрацював тригер.

```
CREATE OR ALTER TRIGGER [dbo].[timeCheck] ON [dbo].[Trip] FOR INSERT, UPDATE AS
BEGIN
    DECLARE @pilot NVARCHAR(30) = (SELECT Pilot FROM Crew1 WHERE IdCrew = (SELECT TOP 1 CrewCaptain FROM inserted))

    DECLARE @d1 DATETIME = (SELECT TOP 1 TimeAndDateOfDeparture FROM inserted)
    DECLARE @d2 DATETIME = (SELECT TOP 1 TimeAndDateOfArrival FROM inserted)

    IF ((SELECT COUNT(*) FROM Trip WHERE
    CrewCaptain IN (SELECT IdCrew FROM Crew1 WHERE Pilot = @pilot) AND
    IdTrip != (SELECT IdTrip FROM inserted) AND
    ((@d2 >= TimeAndDateOfDeparture AND @d1 <= TimeAndDateOfArrival) OR
        DATEDIFF(day, @d2, TimeAndDateOfDeparture) BETWEEN 0 AND 3 OR
        DATEDIFF(day, TimeAndDateOfArrival, @d1) BETWEEN 0 AND 3)) != 0)
    BEGIN
        RAISERROR('Повинна бути перерва 3 дні між вильотами!', 10, 1)
        ROLLBACK TRANSACTION
    END
END
```

Рисунок 1.22. Тригер для перевірки перерви пілота в 3 дні для таблички Trip.

Тригер, який фіксує хто і коли створив запис. Покажемо на прикладі однієї таблички для всіх інших він відрізняється тільки назвою таблички і тригера:

```
CREATE OR ALTER TRIGGER CharacteristicsOfAircraft_Insert ON CharacteristicsOfAircraft AFTER INSERT AS
BEGIN
        UPDATE CharacteristicsOfAircraft SET UCR = SUSER_NAME(), DCR = GETDATE() WHERE IdChar =
(SELECT Top 1 IdChar FROM inserted)
END
```

```
GO

CREATE OR ALTER TRIGGER CharacteristicsOfAircraft_Update ON CharacteristicsOfAircraft AFTER UPDATE
AS
BEGIN
	UPDATE CharacteristicsOfAircraft SET ULC = SUSER_NAME(), DLC = GETDATE() WHERE IdChar =
(SELECT Top 1 IdChar FROM inserted)
END
GO
```

IdChar	NumberOf...	LoadCapac...	IdAirplane	UCR	DCR	ULC	DLC
1	50	3000	1	*NULL*	*NULL*	*NULL*	*NULL*
2	70	4000	2	*NULL*	*NULL*	*NULL*	*NULL*
3	150	2100	3	*NULL*	*NULL*	*NULL*	*NULL*
4	800	3100	4	*NULL*	*NULL*	*NULL*	*NULL*
5	50	2200	5	*NULL*	*NULL*	*NULL*	*NULL*
6	700	3200	6	*NULL*	*NULL*	*NULL*	*NULL*
7	750	3500	7	*NULL*	*NULL*	*NULL*	*NULL*
8	250	2300	8	*NULL*	*NULL*	*NULL*	*NULL*
9	300	2500	9	*NULL*	*NULL*	*NULL*	*NULL*
10	51	2400	10	*NULL*	*NULL*	DESKTOP-4...	2022-06-14 ...
13	35	1370	11	DESKTOP-4...	2022-06-14 ...	DESKTOP-4...	2022-06-14 ...

Рисунок 1.23. Результат роботи системного тригера.

1.5 Адміністрування БД

Адміністрування бази даних позначає весь набір заходів, що виконуються адміністратором бази даних, щоб гарантувати, що база даних завжди доступна за потребою. Інші тісно пов'язані завдання та ролі - це безпека баз даних, моніторинг баз даних та усунення несправностей та планування подальшого зростання.

Запит для створення та видалення користувача AM_GUEST, який повинен мати привілей з’єднуватися з базою даних:

```
USE Kursova
CREATE LOGIN AM_GUEST
WITH PASSWORD = '1638hr_gjs',DEFAULT_DATABASE=Kursova
GO
CREATE USER AM_GUEST
FOR LOGIN AM_GUEST;
GO
DROP USER AM_GUEST;
GO
DROP LOGIN AM_GUEST
EXECUTE AS USER = 'AM_GUEST';
GO
```

Розроблений набір типових повноважень та привілеїв, необхідних для роботи з базою даних, запити для створення типових ролей користувачів бази даних:

```
CREATE ROLE Administrator;
CREATE ROLE Editor;
CREATE ROLE Moderator;
CREATE ROLE Filler;
CREATE ROLE Reader;
```

Ролі для нашої предметнох області:

```
CREATE ROLE Director;--директор
CREATE ROLE Paymaster;--касир
CREATE ROLE RegistrationWorker;--працівник з реєстрації
CREATE ROLE Pilot;-- пілот
CREATE ROLE Stewardess;--стюардеса
CREATE ROLE ChiefEngineer;-- головний інженер
```

Запити для надання необхідних привілеїв ролям :

```
GRANT CONTROL ON DATABASE ::Kursova TO Administrator;--«Адміністратор» — має усі повноваження та привілеї для роботи з базою даних;
GRANT SELECT, INSERT, UPDATE, DELETE TO Editor;--«Редактор» — має привілеї для читання (SELECT) та модифікації (INSERT, UPDATE, DELETE) даних усіх таблиць бази даних;
GRANT SELECT, DELETE TO Moderator;--«Модератор» — має лише привілеї для вибірки (SELECT) та видалення (DELETE) даних/змін внесених іншими користувачами;
GRANT SELECT, INSERT ON Trip TO Filler;--«Наповнювач» — має привілеї для вибірки (SELECT) та внесення (INSERT) даних, що належать до окремих таблиць чи представлень;
GRANT SELECT, INSERT ON Flights TO Filler;
GRANT SELECT ON Flights TO Reader;--«Читач» — має привілеї лише для вибірки (SELECT) даних з окремих таблиць чи представлень.
GRANT SELECT ON CharacteristicsOfAircraft TO Reader;
GRANT SELECT ON Crew1 TO Reader;
GRANT SELECT ON Trip TO Reader;
--для нашої предметної області
GRANT SELECT,UPDATE TO Director;
GRANT SELECT, UPDATE, INSERT,DELETE ON Trip TO Paymaster;
GRANT SELECT, UPDATE, INSERT,DELETE ON Flights TO Paymaster;
GRANT SELECT TO Paymaster;
GRANT SELECT, UPDATE, INSERT,DELETE ON Flights TO RegistrationWorker;
```

```
GRANT SELECT, UPDATE, INSERT,DELETE ON Crew1 TO RegistrationWorker;
GRANT SELECT TO RegistrationWorker;
GRANT SELECT TO Pilot;
GRANT SELECT TO Stewardess;
GRANT SELECT, UPDATE, INSERT,DELETE ON Aircraft TO ChiefEngineer;
GRANT SELECT, UPDATE, INSERT,DELETE ON CharacteristicsOfAircraft TO ChiefEngineer;
GRANT SELECT, UPDATE, INSERT,DELETE ON Aircraft TO ChiefEngineer;
```

Для користувача AM_GUEST:

Запити для призначення декількох ролей (наприклад, «Читач» та «Наповнювач) та перевірити можливість застосування привілей, наданих через відповідні ролі.

```
ALTER ROLE Reader ADD MEMBER AM_GUEST
GO
ALTER ROLE Filler ADD MEMBER  AM_GUEST
GO
```

Запит для відкликання однієї із наданих ролей:

```
ALTER ROLE  Filler DROP MEMBER AM_GUEST

GO
```

Запит для явного призначення привілеїв для читання і модифікації деякої таблиці та перевірити результати цього призначення:

```
GRANT SELECT ON Trip TO AM_GUEST;
GRANT INSERT ON Trip TO AM_GUEST;
GRANT UPDATE ON Trip TO AM_GUEST;
```

Приклад запиту для відкликання привілей у ролі, яка призначена користувачу, та перевірити результати цього відкликання:

```
REVOKE SELECT ON Crew1  TO Reader;
```

Запит для явного відкликання привілей для читання і модифікації деякої таблиці та перевірити результати цього відкликання.

```
REVOKE SELECT, UPDATE ON Trip TO AM_GUEST;
```

Приклад запити для створення користувачів з нашої предметної області мати привілей з’єднуватися з базою даних:

```
CREATE LOGIN Pavlo
  WITH PASSWORD = '78956Wece',DEFAULT_DATABASE=Kursova
 GO
CREATE USER Pavlo
FOR LOGIN Pavlo;
GO
ALTER ROLE Director ADD MEMBER Pavlo
GO
CREATE LOGIN Oleh
  WITH PASSWORD = '1580YUi09',DEFAULT_DATABASE=Kursova
 GO
CREATE USER Oleh
FOR LOGIN Oleh;
GO
ALTER ROLE Paymaster ADD MEMBER Oleh
GO
CREATE LOGIN Kolia
  WITH PASSWORD = 'cvr128&8u',DEFAULT_DATABASE=Kril_Pavlo
```

```
 GO
CREATE USER Kolia
FOR LOGIN Kolia;
GO
ALTER ROLE RegistrationWorker ADD MEMBER Kolia
GO
CREATE LOGIN Christina
  WITH PASSWORD = '150!nfW3e',DEFAULT_DATABASE=Kursova
 GO
CREATE USER Christina
FOR LOGIN Christina;
GO
ALTER ROLE Stewardess ADD MEMBER Christina
GO
CREATE LOGIN Ivan
  WITH PASSWORD = '541cm_89t-9A',DEFAULT_DATABASE=Kursova
 GO
CREATE USER Ivan
FOR LOGIN Ivan;
GO
ALTER ROLE Pilot ADD MEMBER Ivan
GO
```

Розділ 2. Розробка віконного додатку

В ході розробки віконного додатку було насамперед створено дизайн. На форму додано tabControl і для кожної таблички новий tabPage, додали всі необхідні кнопки, поля і label для всіх табличок. І потім підключено БД через Entity.

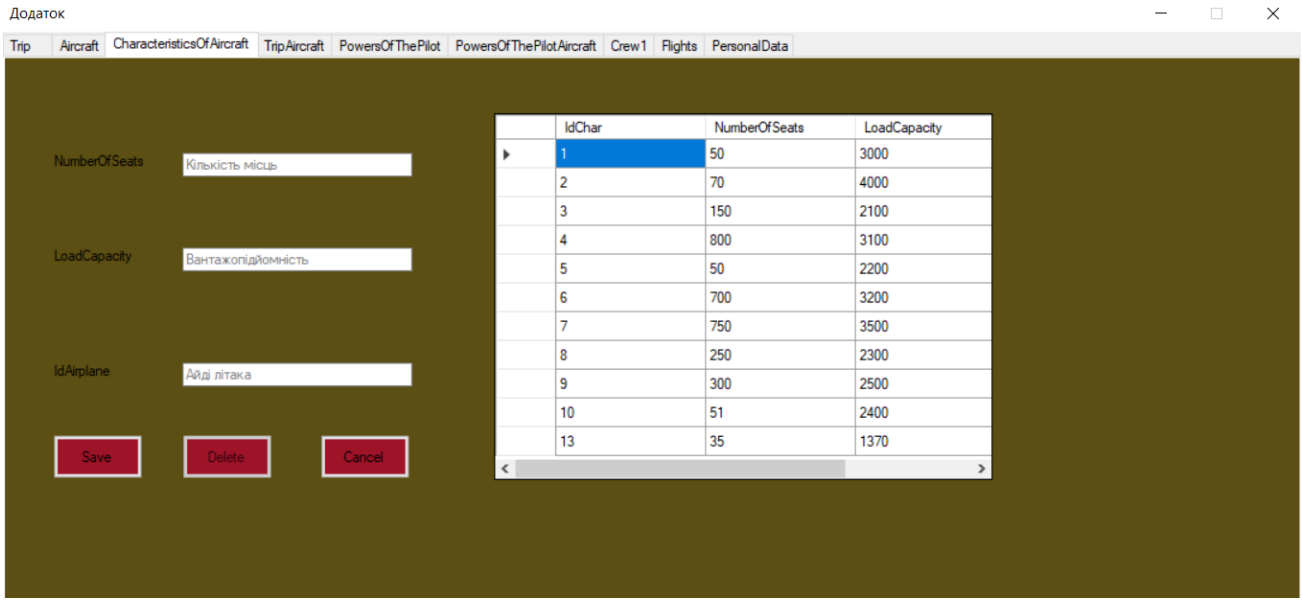


Рисунок 2. Форма для віконного додатку.

Створюєм спочатку нові моделі для роботи з таблицями БД:

```
Trip model = new Trip();
Aircraft models = new Aircraft();
CharacteristicsOfAircraft mode = new CharacteristicsOfAircraft();
TripAircraft mod = new TripAircraft();
PowersOfThePilot mo = new PowersOfThePilot();
PowersOfThePilotAircraft m = new PowersOfThePilotAircraft();
Crew1 model7 = new Crew1();
Flight model8 = new Flight();
PersonalData_ model9 = new PersonalData_();
```

При загрузці форми запускаємо функцію, яка додає дані з таблиць в dataGridView за допомогою Entity:

```
void PopulateDataGridView()
    {
        using (KursovaEntities db = new KursovaEntities())
        {
            var characteristicsOfAircraft = from t in db.CharacteristicsOfAircrafts
          select new
            {
                IdChar = t.IdChar,
                NumberOfSeats = t.NumberOfSeats,
                LoadCapacity = t.LoadCapacity,
                IdAirplane = t.IdAirplane,
                UCR = t.UCR,
                DCR = t.DCR,
                ULC = t.ULC,
                DLC = t.DLC
```

```
                    };
                dataGridView3.DataSource = characteristicsOfAircraft.ToList();
                    }
                }
```

Для збереження даних в таблицю БД використаєм клік по кнопці за допомогою Entity:

```
        private void button7_Click(object sender, EventArgs e)
            {
                mode.NumberOfSeats = int.Parse(textBox11.Text);
                mode.LoadCapacity = int.Parse(textBox12.Text);
                mode.IdAirplane = int.Parse(textBox13.Text);
                using (KursovaEntities db = new KursovaEntities())
                {
                    if (mode.IdChar == 0)//Insert
                        db.CharacteristicsOfAircrafts.Add(mode);
                    else //Update
                        db.Entry(mode).State = EntityState.Modified;
                    db.SaveChanges();
                }
                Clear();
                PopulateDataGridView();
                MessageBox.Show("Submitted Successfully");
            }
```

Видалення даних з таблиці БД з використанням Entity:

```
        private void button8_Click(object sender, EventArgs e)
            {
                if (MessageBox.Show("Are You Sure to Delete this Record ?", "EF CRUD Operation",
                MessageBoxButtons.YesNo) == DialogResult.Yes)
                    {
                        using (KursovaEntities db = new KursovaEntities())
                        {
                            var entry = db.Entry(mode);
                             if (entry.State == EntityState.Detached)
                                db.CharacteristicsOfAircrafts.Attach(mode);
                            db.CharacteristicsOfAircrafts.Remove(mode);
                            db.SaveChanges();
                             PopulateDataGridView();
                             Clear();
                             MessageBox.Show("Deleted Successfully");
                          }
                    }
            }
```

Оновлення даних в таблицях реалізовано через двойний клік по DataGridView:

```
    private void dataGridView3_DoubleClick(object sender, EventArgs e)
        {
          if (dataGridView3.CurrentRow.Index != -1)
           {
            mode.IdChar = Convert.ToInt32(dataGridView3.CurrentRow.Cells["IdChar"].Value);
            using (KursovaEntities db = new KursovaEntities())
            {
                mode = db.CharacteristicsOfAircrafts.Where(x => x.IdChar == mode.IdChar).FirstOrDefault();
                textBox11.Text = Convert.ToString(mode.NumberOfSeats);
                textBox12.Text = Convert.ToString(mode.LoadCapacity);
                textBox13.Text = Convert.ToString(mode.IdAirplane);
             }
            button7.Text = "Update";
            button8.Enabled = true;
           }
```