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 аеропорту

Студентки 2 курсу ІП-12 групи

спеціальності 121 «Інженерія програмного забезпечення»

Кушнір Ганни Вікторівни

(прізвище та ініціали)

Керівник канд. техн. наук, доцент

Ліщук Катерина Ігорівна

(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 – 2022 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-12

Семестр 3

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Кушнір Ганні Вікторівні

(прізвище, ім’я, по батькові)

1. Тема роботи База даних аеропорту

керівник роботи Ліщук Катерина Ігорівна, доцент, канд. техн. наук

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

1. Строк подання студентом роботи 04.01.2022
2. Вихідні дані до роботи

завдання на розробку бази даних для відстеження фінансової сторони роботи аеропорту

1. 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

1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
2. Дата видачі завдання 08.11.2022

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 14.11.2022 |  |
| 2 | Побудова ER-моделі | 21.11.2022 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 25.11.2022 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 28.11.2022 |  |
| 5 | Створення користувачів бази даних | 03.12.2022 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 10.12.2022 |  |
| 7 | Створення мовою SQL запитів | 20.12.2022 |  |
| 8 | Оптимізація роботи запитів | 25.12.2022 |  |
| 9 | Оформлення пояснювальної записки | 07.01.2023 |  |
| 10 | Захист курсової роботи | 11.01.2023 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент**  Кушнір Г.В.

(підпис ) (прізвище та ініціали)

**Керівник роботи**  Ліщук К.І.

(підпис ) (прізвище та ініціали)

**ЗМІСТ**

[ВСТУП 6](#_Toc123753932)

[1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА 7](#_Toc123753933)

[2 ПОСТАНОВКА ЗАВДАННЯ 8](#_Toc123753934)

[3 ПРОЄКТУВАННЯ БАЗИ ДАНИХ 9](#_Toc123753935)

[3.1 Побудова концептуальної моделі 9](#_Toc123753936)

[3.1.1 Опис інформаційних об’єктів 9](#_Toc123753937)

[3.1.2 Опис атрибутів сутностей 9](#_Toc123753938)

[3.1.3 Опис зв’язків між об’єктами 11](#_Toc123753939)

[3.1.4 ER-модель предметної області 12](#_Toc123753940)

[3.1.5 Модель користувачів бази даних 13](#_Toc123753941)

[3.2 Побудова даталогічної моделі 13](#_Toc123753942)

[3.2.1 Набір відношень бази даних 13](#_Toc123753943)

[3.2.2 Нормалізація бази даних 14](#_Toc123753944)

[3.2.3 Даталогічна схема бази даних 15](#_Toc123753945)

[3.3 Підбиття підсумків за розділом 17](#_Toc123753946)

[4 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ 18](#_Toc123753947)

[4.1 Обґрунтування вибору СУБД 18](#_Toc123753948)

[4.2 Створення бази даних 18](#_Toc123753949)

[4.2.1 SQL-скрипт для створення БД 18](#_Toc123753950)

[4.2.2 Створення таблиць бази даних 19](#_Toc123753951)

[4.3 Схема бази даних 20](#_Toc123753952)

[4.4 Створення користувачів 20](#_Toc123753953)

[4.5 Підбиття підсумків за розділом 21](#_Toc123753954)

[5 РОБОТА З БАЗОЮ ДАНИХ 22](#_Toc123753955)

[5.1 Заповнення бази даних тестовими даними 22](#_Toc123753956)

[5.2 Створення представлень 22](#_Toc123753957)

[5.2.1 Представлення «PurchaseOfTickets» 22](#_Toc123753958)

[5.2.2 Представлення «TicketsEconomyClass» 22](#_Toc123753959)

[5.2.3 Представлення «TicketsBusinessClass» 23](#_Toc123753960)

[5.3 Створення збережених процедур 24](#_Toc123753961)

[5.3.1 Процедура «countIncome» 24](#_Toc123753962)

[5.3.2 Процедура «countExpenses» 24](#_Toc123753963)

[5.3.3 Процедура «whetherTurnedIntoProfit» 24](#_Toc123753964)

[5.4 Створення функцій 25](#_Toc123753965)

[5.4.1 Функція «getFlightReport» 25](#_Toc123753966)

[5.4.2 Функція «getTicketsReportForPeriod» 25](#_Toc123753967)

[5.5 Створення тригерів 26](#_Toc123753968)

[5.5.1 Тригер «on\_insert\_tickets» 26](#_Toc123753969)

[5.5.2 Тригер «on\_insert\_flights» 27](#_Toc123753970)

[5.5.3 Тригер «on\_insert\_salary» 27](#_Toc123753971)

[5.6 Написання SQL-запитів 28](#_Toc123753972)

[5.6.1 Список клієнтів, що придбали квитки 28](#_Toc123753973)

[5.6.2 Ціни на квитки для окремого клієнта 28](#_Toc123753974)

[5.6.3 Інформація про квитки економ-класу для кожного рейсу 28](#_Toc123753975)

[5.6.4 Інформація про квитки бізнес-класу для кожного рейсу 29](#_Toc123753976)

[5.6.5 Інформація по виплатам кожному працівнику 29](#_Toc123753977)

[5.6.6 Перелік літаків аеропорту 30](#_Toc123753978)

[5.6.7 Інформація по кількості квитків, замовлених різними клієнтами 31](#_Toc123753979)

[5.6.8 Кількість працівників на кожній посаді 31](#_Toc123753980)

[5.6.9 Перелік працівників аеропорту 32](#_Toc123753981)

[5.6.10 Екіпаж літака, закріплений за певним рейсом 32](#_Toc123753982)

[5.7 Створення індексів 33](#_Toc123753983)

[5.8 Підбиття підсумків за розділом 33](#_Toc123753984)

[ВИСНОВКИ 34](#_Toc123753985)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 35](#_Toc123753986)

[ДОДАТОК А ТЕКСТИ SQL-СКРИПТІВ ДЛЯ СТВОРЕННЯ ТАБЛИЦЬ 36](#_Toc123753987)

[ДОДАТОК Б СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ 44](#_Toc123753988)

[ДОДАТОК В ЗАПОВНЕННЯ БАЗИ ДАНИХ ТЕСТОВИМИ ДАНИМИ 46](#_Toc123753989)

[ДОДАТОК Д РЕЗУЛЬТАТИ ДОДАВАННЯ ДАНИХ ДО ТАБЛИЦЬ 53](#_Toc123753990)

[ДОДАТОК Е ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ПРЕДСТАВЛЕНЬ 57](#_Toc123753991)

[ДОДАТОК Ж ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР 59](#_Toc123753995)

[ДОДАТОК З ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ФУНКЦІЙ 61](#_Toc123753999)

[ДОДАТОК И ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ТРИГЕРІВ 63](#_Toc123754002)

# ВСТУП

Бази даних є невід’ємною складовою будь-якого підприємства чи бізнесу, вони слугують місцем для збереження всієї важливої інформації про клієнтів та працівників закладу, про доходи та витрати, про різноманітні події, що відбуваються на підприємстві, та про багато-багато іншого.

Актуальність теми даної курсової роботи полягає в тому, що обсяги даних у сучасному світі є невимовно великими, тож зберігати їх у вигляді архівів та паперових журналів стає недоцільним. Тут на допомогу приходять славнозвісні бази даних, котрі надають миттєвий доступ до всієї інформації в одному місці. А оскільки аеропорт містить чималу кількість об’єктів, відомості про які потрібно десь зберігати, створити для нього базу даних є найрозумнішим рішенням.

Дана робота присвячена отриманню практичних навичок з проектування та реалізації реляційних баз даних на прикладі бази даних аеропорту.

Завдання даної курсової роботи зводиться до аналізу предметної області, побудови концептуальної та логічної моделей бази даних, та безпосередньо реалізації спроектованої бази даних з використанням однієї з існуючих систем керування базами даних.

1. ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Основна задача програмного забезпечення, що проєктується, – це відстеження фінансової сторони роботи аеропорту.

Діяльність організована наступним чином: аеропорт продає квитки клієнтам на певний рейс. Кожен рейс характеризується датою та часом відправлення, своїм маршрутом та цінами на квитки. Окрім цього на кожен рейс обирається літак, що здійснюватиме політ, та екіпаж, що керуватиме літаком і буде його обслуговувати.

Будь-який маршрут, у свою чергу, характеризується аеропортом призначення, а кожен літак має свій бортовий номер та модель, яка визначає кількість місць бізнес- та економ-класу.

Усім працівникам аеропорту виплачується заробітня плата та можуть нараховуватися премії чи надбавки.

Клієнтами є різні особи, за якими збирається деяка інформація (прізвище, ім’я, по-батькові, номер та серія паспорту або іншого документу, що посвідчує особу, а також постійна знижка на квитки, якщо така є). При бронюванні квитка на певний рейс клієнт може обрати клас комфорту: бізнес або економ; ціни на квитки кожного з цих класів встановлюються для кожного рейсу окремо.

Продаж квитка клієнтові проводиться лише за наявності вільних місць на рейсі, який відповідає параметрам, котрі вказав клієнт. Також неможливо продати квиток на рейс, час відправлення якого минув.

1. ПОСТАНОВКА ЗАВДАННЯ

Розробити програмне забезпечення для відстеження фінансової сторони роботи аеропорту.

Вхідними даними для даної роботи є інформація про:

* рейси;
* маршрути, якими здійснюються рейси;
* літаки;
* моделі літаків;
* квитки на рейси;
* клієнтів, що купують квитки;
* працівників аеропорту;
* посади, які можуть займати працівники;
* виплати заробітної плати та премій працівникам.

Програмне забезпечення повинно зберігати всю вищеперераховану інформацію у базі даних та перевіряти коректність даних перед їх додаванням; надавати користувачам можливість переглядати, змінювати, додавати та видаляти дані із бази, а також здійснювати наступні дії:

* дізнаватися загальну суму доходів та витрат аеропорту;
* дізнаватися, чи приносить аеропорт прибуток;
* генерувати фінансовий звіт за кожним із рейсів;
* генерувати фінансовий звіт щодо придбаних користувачами квитків за певний період часу;
* генерувати прайс-лист, актуальний для певного користувача (тобто з урахуванням його знижок).

Вихідними даними для даної курсової роботи є спроектована та реалізована база даних, готова до практичного застосування.

1. ПРОЄКТУВАННЯ БАЗИ ДАНИХ
2. Побудова концептуальної моделі
3. Опис інформаційних об’єктів

Під час проведення аналізу предметного середовища даної курсової роботи були виділені наступні множини інформаційних об’єктів (сутностей):

а) «Routes» – встановлені маршрути від аеропорту, для якого створена база даних, до аеропортів призначення; за кожним з цих маршрутів можуть призначити виліт на певну дату;

б) «Flights» – рейси (вильоти), які мають чітко встановлену дату та зарезервований літак;

в) «Planes» – літаки, що здійснюють польоти з даного аеропорту;

г) «AircraftModels» – моделі літаків;

ґ) «Personnel» – усі працівники аеропорту (включаючи пілотів і стюардес);

д) «Aircrew» – екіпажі літаків, закріплені за певними рейсами;

е) «Positions» – посади, які можуть займати робітники аеропорту;

є) «SalaryPayment» - виплати заробітної плати та премій працівникам;

ж) «Clients» – клієнти, що бронюють квитки на рейси;

з) «Tickets» – квитки, куплені клієнтами на певний рейс.

1. Опис атрибутів сутностей

Наступним кроком було визначення основних характеристик кожного інформаційного об’єкта – його атрибутів. Результати даного кроку наведені у таблиці 3.1.

Таблиця 3.1 – Опис атрибутів виділених сутностей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва сутності | Назва атрибута | Опис атрибута |
| 1 | Routes | RouteID | Унікальний ID маршруту |
| DestinationAirport | Назва аеропорту призначення |
| 2 | Flights | FlightID | Унікальний ID рейсу |
| RouteID | ID-номер маршруту |
| PlaneID | ID-номер літака, яким здійснюється рейс |

Продовження таблиці 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва сутності | Назва атрибута | Опис атрибута |
| 2 | Flights | DepartureTime | Час відправлення (вильоту) |
| EconomyPrice | Вартість квитка економ-класу |
| BusinessPrice | Вартість квитка бізнес-класу |
| 3 | Planes | PlaneID | Унікальний ID літака |
| RegistrationNumber | Бортовий номер літака |
| ModelID | ID-номер моделі літака |
| 4 | AircraftModels | ModelID | Унікальний ID моделі |
| Name | Назва моделі |
| NumberOfEconomySeats | Кількість місць економ-класу |
| NumberOfBusinessSeats | Кількість місць бізнес-класу |
| 5 | Personnel | EmployeeID | Унікальний ID працівника |
| Surname | Прізвище працівника |
| Name | Ім’я працівника |
| Patronymic | По-батькові працівника |
| Document | Номер та серія документу, що посвідчує особу |
| PositionID | ID-номер посади, яку займає працівник |
| DateOfEmployement | Дата працевлаштування |
| 6 | Aircrew | FlightID | ID-номер рейсу |
| EmployeeID | ID-номер члена екіпажу (робітника) |
| 7 | Positions | PositionID | Унікальний ID посади |
| Name | Назва посади |
| Salary | Заробітня плата, яку отримують робітники на даній посаді |
| 8 | SalaryPayment | PaymentID | Унікальний ID виплати |
| EmployeeID | ID-номер працівника |
| PaymentDate | Дата здійснення виплати |
| Coefficient | Коефіцієнт виплати відносно заробітної плати |
| 9 | Clients | ClientID | Унікальний ID клієнта |
| Surname | Прізвище клієнта |
| Name | Ім’я клієнта |

Продовження таблиці 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва сутності | Назва атрибута | Опис атрибута |
| 9 | Clients | Patronymic | По-батькові клієнта |
| Document | Номер та серія документу, що посвідчує особу |
| Discount | Постійна знижка клієнта |
| 10 | Tickets | Number | Унікальний номер (ID) квитка |
| FlightID | ID-номер рейсу |
| ClientID | ID-номер клієнта, що купив квиток |
| Comfort | Тип місця (економ- чи бізнес-класу) |
| SeatNumber | Номер місця |
| DateOfPurchase | Дата покупки квитка |

1. Опис зв’язків між об’єктами

Між об’єктами (таблицями) можуть бути встановлені зв’язки наступних типів: один-до-одного, один-до-багатьох або багато-до-багатьох.

Зв’язок ***один-до-одного*** (1:1) виявляє себе, коли одному значенню поля однієї таблиці відповідає єдине значення поля другої таблиці та, навпаки, одному значенню поля другої таблиці – єдине значення поля першої.

Зв’язок ***один-до-багатьох*** (1:Б) має місце, коли одному значенню поля першої таблиці може відповідати декілька значень поля другої таблиці, а кожному значенню поля другої таблиці – тільки єдине значення поля першої.

Зв’язок ***багато-до-багатьох*** (Б:Б) має місце, коли кожному значенню поля першої таблиці відповідає декілька значень поля другої таблиці й кожному значенню другої таблиці відповідає декілька значень першої таблиці.[1]

Структурні зв’язки між виділеними інформаційними об’єктами зображено у таблиці 3.2.

Таблиця 3.2 – Зв’язки між сутностями

|  |  |  |  |
| --- | --- | --- | --- |
| Номер зв’язку | Головна таблиця | Дочірня таблиця | Тип зв’язку |
| 1 | AircraftModels | Planes | 1:Б |
| 2 | Planes | Flights | 1:Б |

Продовження таблиці 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| Номер зв’язку | Головна таблиця | Дочірня таблиця | Тип зв’язку |
| 3 | Positions | Personnel | 1:Б |
| 4 | Flights | Tickets | 1:Б |
| 5 | Clients | Tickets | 1:Б |
| 6 | Routes | Flights | 1:Б |
| 7 | Personnel | SalaryPayment | 1:Б |
| 8 | Flights | Aircrew | 1:Б |
| 9 | Personnel | Aircrew | 1:Б |

1. ER-модель предметної області

На рисунку 3.1 представлена ER-діаграма (у нотації Баркера) предметної області «База даних аеропорту», на якій відображені всі сутності та їх атрибути, а також символом «#» позначено, які атрибути будуть ключовими.

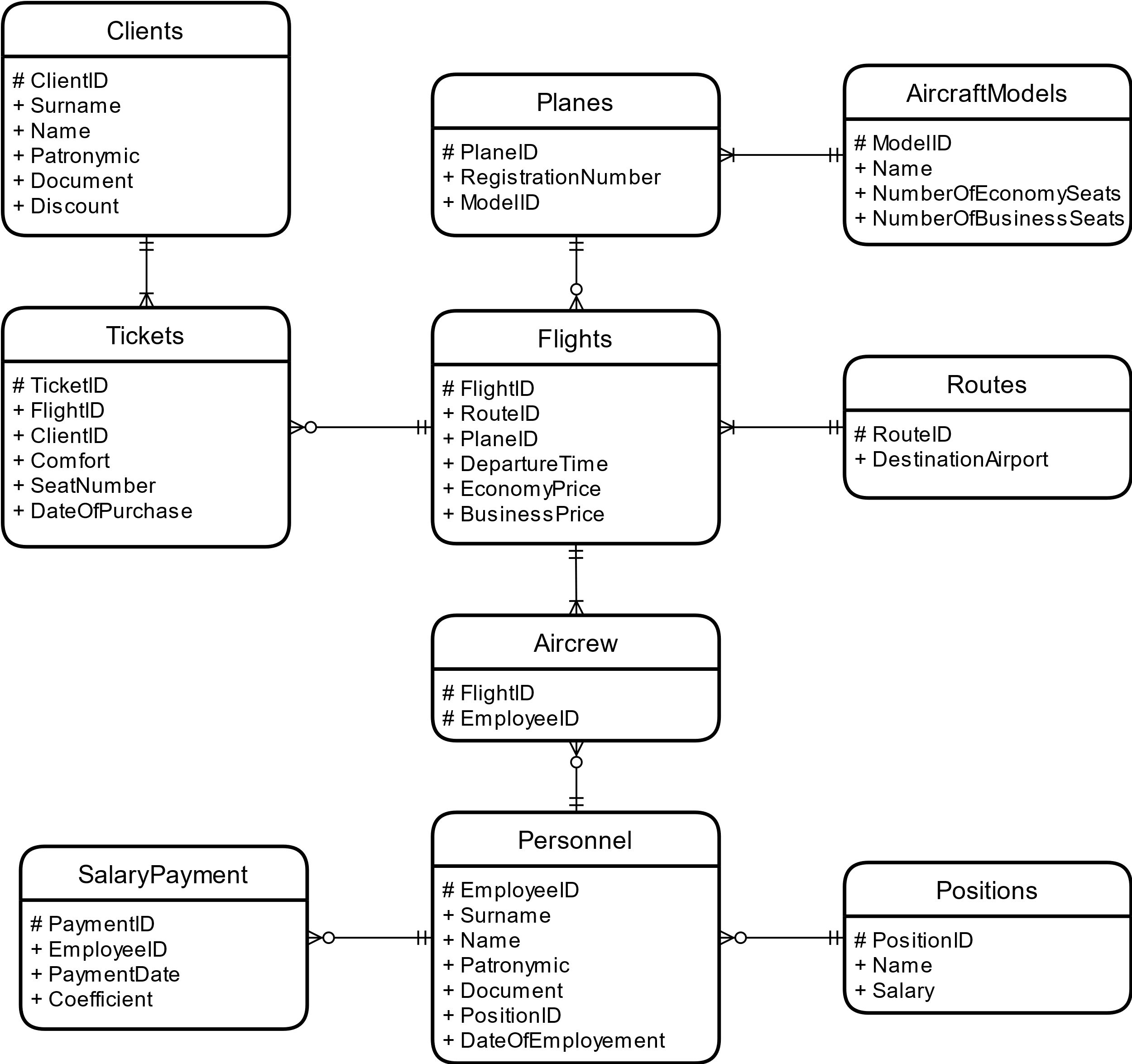


Рисунок 3.1 – ER-діаграма сформованої предметної області

1. Модель користувачів бази даних

З урахуванням побудованої ER-моделі предметного середовища, з’являється необхідність реалізації наступних користувачів бази даних:

а) Власник бази даних – користувач, який має повну владу над створеною базою даних: може її видаляти, змінювати місце її збереження, назву; додавати та видаляти таблиці; керувати всіма даними, що зберігаються в БД; а також реєструвати нових користувачів бази даних, надавати їм права та привілеї.

б) Адміністратор бази даних – користувач, що має право створювати, змінювати та видаляти об’єкти (таблиці, запити і т.д.) у базі даних; іншими словами, це користувач, що керує структурою бази даних, але не додає самі дані.

в) Записувач даних – користувач, який може модифікувати будь-які дані в будь-якій таблиці (у тому числі видаляти старі чи додавати нові дані).

г) Читач даних – користувач, що може переглядати дані будь-якої таблиці бази даних.

1. Побудова даталогічної моделі
2. Набір відношень бази даних

Відношення (структурні зв’язки) між виділеними інформаційними об’єктами зручно подати у вигляді реляційної схеми бази даних:

а) відношення, утворені з множин сутностей (нижнім підкреслюванням виділені первинні ключі, а курсивом – зовнішні):

1. Routes(RouteID, DestinationAirport);
2. Flights(FlightID, *RouteID*, *PlaneID*, DepartureTime, EconomyPrice, BusinessPrice);
3. Planes(PlaneID, RegistrationNumber, *ModelID*);
4. AircraftModels(ModelID, Name, NumberOfEconomySeats, NumberOfBusinessSeats);
5. Personnel(EmployeeID, Surname, Name, Patronymic, Document, *PositionID*, DateOfEmployement);
6. Aircrew(*FlightID*, *EmployeeID*);
7. Positions(PositionID, Name, Salary);
8. SalaryPayment(PaymentID, *EmployeeID*, PaymentDate, Coefficient);
9. Clients(ClientID, Surname, Name, Patronymic, Document, Discount);
10. Tickets(TicketID, Number, *FlightID*, *ClientID*, Comfort, SeatNumber, DateOfPurchase);

б) відношення, утворені зі зв’язків між множинами сутностей:

1. Has(PlaneID, ModelID);
2. IsCarriedOutOn(FlightID, PlaneID);
3. Takes(EmployeeID, PositionID);
4. For(TicketID, FlightID);
5. WasPurchasedBy(TicketID, ClientID);
6. Follows(FlightID, RouteID);
7. WasPaidTo(PaymentID, EmployeeID);
8. Aircrew – слабка множина сутностей, що поєднує множини сутностей Flights та Personnel, тому відношення, що відповідають підтримуючим зв’язкам множини Aircrew, не створюються.
9. Нормалізація бази даних

Нормалізація бази даних – один з найважливіших кроків при моделюванні БД. Нормалізація потрібна для усунення надмірності даних, тобто видалення повторень даних в різних рядках однієї таблиці.

1. Перша нормальна форма

Відношення знаходиться в ***першій нормальній формі*** тоді і тільки тоді, коли домен кожного атрибута містить лише нероздільні значення, а значення кожного атрибута містить лише одне значення з цього домену.[2]

Усі атрибути побудованих відношень мають атомарні та неподільні домени, а отже, кожне з реляційних відношень бази даних (а відповідно, кожна із таблиць) перебуває у першій нормальній формі (1НФ).

1. Друга нормальна форма

Таблиця, що перебуває в першій нормальній формі, перебуває в ***другій нормальній формі*** тоді й тільки тоді, коли всі її неключові атрибути функціонально залежні від потенційного ключа в цілому. У разі, якщо 1НФ таблиця не має складних потенційних ключів (таких, що складаються більш ніж з одного атрибута), тоді вона автоматично перебуватиме в 2НФ.[3]

Усі таблиці (відношення), окрім таблиці Aircrew, містять прості потенційні ключі, тобто такі, що складаються з одного атрибута. Звідси можна зробити висновок, що всі вони перебувають у другій нормальній формі.

Відношення ж Aircrew має складний ключ із двох атрибутів, але, оскільки неключові атрибути відсутні, умова перебування у 2НФ все ще виконується.

1. Третя нормальна форма

За Коддом таблиця знаходиться в ***третій нормальній формі*** тоді й лише тоді, коли відношення R (таблиця) знаходиться в 2НФ і кожен неключовий атрибут відношення R нетранзитивно (безпосередньо) залежить від кожного потенційного ключа в R.[4]

Після детального перегляду і перевірки атрибутів кожного з відношень не було знайдено жодної транзитивної залежності між атрибутами. Тож можна підсумувати, що всі відношення перебувають у третій нормальній формі.

1. Даталогічна схема бази даних

У таблицях 3.3 – 3.12 наведені описи сутностей сформованої бази даних із зазначенням їх атрибутів, обраних первинних ключів та орієнтовних типів даних.

Таблиця 3.3 – Даталогічний опис таблиці Routes

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| RouteID | Так | INT |
| DestinationAirport | Ні | NVARCHAR |

Таблиця 3.4 – Даталогічний опис таблиці Flights

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| FlightID | Так | INT |
| RouteID | Ні | INT |
| PlaneID | Ні | INT |
| DepartureTime | Ні | DATETIME |
| EconomyPrice | Ні | MONEY |
| BusinessPrice | Ні | MONEY |

Таблиця 3.5 – Даталогічний опис таблиці Planes

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| PlaneID | Так | INT |
| RegistrationNumber | Ні | INT |
| ModelID | Ні | INT |

Таблиця 3.6 – Даталогічний опис таблиці AircraftModels

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| ModelID | Так | INT |
| Name | Ні | NVARCHAR |
| NumberOfEconomySeats | Ні | INT |
| NumberOfBusinessSeats | Ні | INT |

Таблиця 3.7 – Даталогічний опис таблиці Personnel

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| EmployeeID | Так | INT |
| Surname | Ні | NVARCHAR |
| Name | Ні | NVARCHAR |
| Patronymic | Ні | NVARCHAR |
| Document | Ні | NVARCHAR |
| PositionID | Ні | INT |
| DateOfEmployement | Ні | DATE |

Таблиця 3.8 – Даталогічний опис таблиці Positions

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| PositionID | Так | INT |
| Name | Ні | NVARCHAR |
| Salary | Ні | MONEY |

Таблиця 3.9 – Даталогічний опис таблиці SalaryPayment

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| PaymentID | Так | INT |
| EmployeeID | Ні | INT |
| PaymentDate | Ні | DATE |
| Coefficient | Ні | REAL |

Таблиця 3.10 – Даталогічний опис таблиці Aircrew

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| FlightID | Так | INT |
| EmployeeID | Так | INT |

Таблиця 3.11 – Даталогічний опис таблиці Clients

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| ClientID | Так | INT |
| Surname | Ні | NVARCHAR |
| Name | Ні | NVARCHAR |
| Patronymic | Ні | NVARCHAR |
| Document | Ні | NVARCHAR |
| Discount | Ні | REAL |

Таблиця 3.12 – Даталогічний опис таблиці Tickets

|  |  |  |
| --- | --- | --- |
| Назва атрибута | Первинний ключ | Тип даних |
| Number | Так | INT |
| FlightID | Ні | INT |
| ClientID | Ні | INT |
| Comfort | Ні | NVARCHAR(6) |
| SeatNumber | Ні | INT |
| DateOfPurchase | Ні | DATE |

1. Підбиття підсумків за розділом

Отже, у даному розділі курсової роботи було здійснено проектування бази даних для предметної області «База даних аеропорту». Був проведений детальний аналіз середовища, за результатами якого були побудовані ER-модель (модель «сутність-зв’язок») та даталогічна модель області, які будуть використані у подальшому для реалізації бази даних у конкретній СУБД. У ході роботи були виділені основні інформаційні об’єкти (сутності), їх атрибути та ключі, а також визначені обмеження цілісності, що накладаються на кожну із сутностей. Урешті-решт були встановлені структурні зв’язки, тобто відношення між таблицями бази даних, які забезпечать можливість здійснення всіх необхідних запитів.

1. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ
2. Обґрунтування вибору СУБД

Система управління базами даних, яка буде використана при реалізації змодельованої бази даних, повинна відповідати наступним вимогам:

* має бути представлена можливість створення реляційної бази даних;
* можливість створення таблиць, встановлення на них обмежень для збереження цілісності БД, а також можливість додавання зовнішніх ключів для встановлення структурних зв’язків між таблицями;
* підтримка SQL-скриптів та запитів;
* а також підтримка багатокористувацької моделі БД.

Microsoft SQL Server – це одна з найпоширеніших серверних СУБД. Вона відповідає всім вищезазначеним вимогам і містить в собі реалізований майже весь функціонал SQL.

Одна з переваг MS SQL – простота в налагодженні роботи сервера, а також створенні і супроводженні БД. Не можна не згадати також про легкість у вивченні, оскільки на просторах Інтернету можна знайти детальну документацію до всього функціоналу даної СУБД.

При написанні SQL-скриптів буде використана мова структурованих запитів Transact-SQL (T-SQL), яка підтримується СУБД Microsoft SQL Server.

1. Створення бази даних
2. SQL-скрипт для створення БД

Текст SQL-скрипту для створення БД:

USE master GO

IF EXISTS (

SELECT \*

FROM sys.databases

WHERE name = N'KR\_DB'

)

DROP DATABASE KR\_DB

GO

CREATE DATABASE KR\_DB

GO

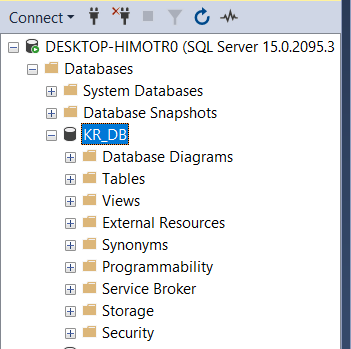
На рисунку 4.1 наведений результат виконання вищевказаного скрипту (у списку баз даних серверу почала відображатись БД з назвою «KR\_DB»).

Рисунок 4.1 – Відображення створеної БД у списку баз даних сервера

1. Створення таблиць бази даних

Перед початком створення таблиць необхідно вказати, в яку базу даних потрібно їх додавати. Це було зроблено за допомогою наступної команди:

USE [KR\_DB]

У додатку А наведені SQL-скрипти, що використовувалися для створення кожної із таблиць бази даних та їх обмежень цілісності.

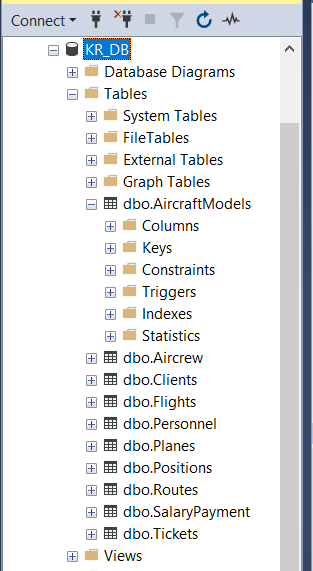
На рисунку 4.2 наведений результат додавання таблиць до бази даних «KR\_DB» - у переліку таблиць БД почали відображатися створені таблиці .

Рисунок 4.2 – Відображення таблиць у списку таблиць бази даних «KR\_DB»

1. Схема бази даних

На рисунку 4.3 наведена схема побудованої бази даних, яка була згенерована засобами СУБД. На даній схемі відображені всі таблиці БД та зв’язки між ними, а також назви, типи та значення за замовчуванням полів кожної з таблиць.

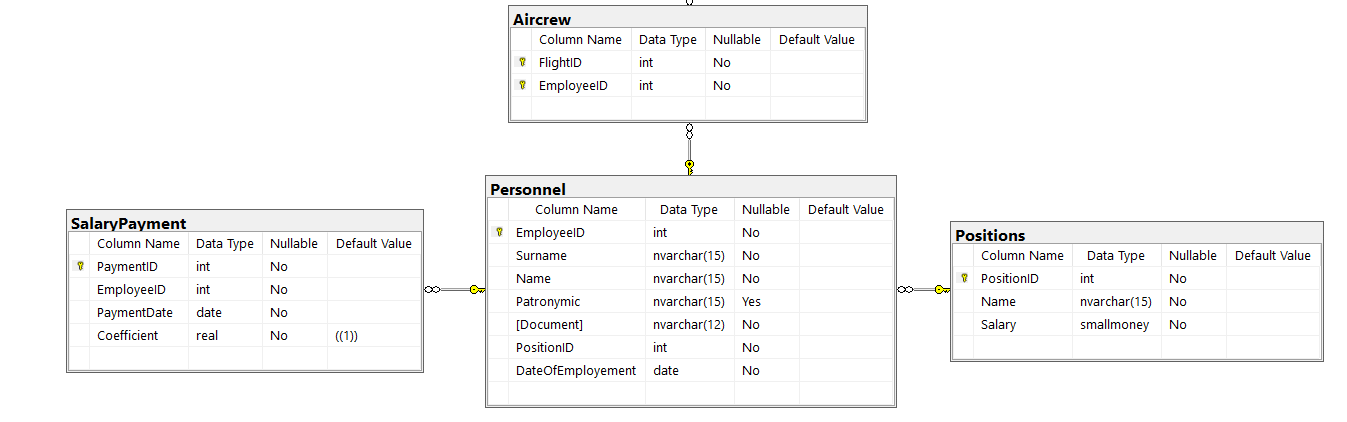
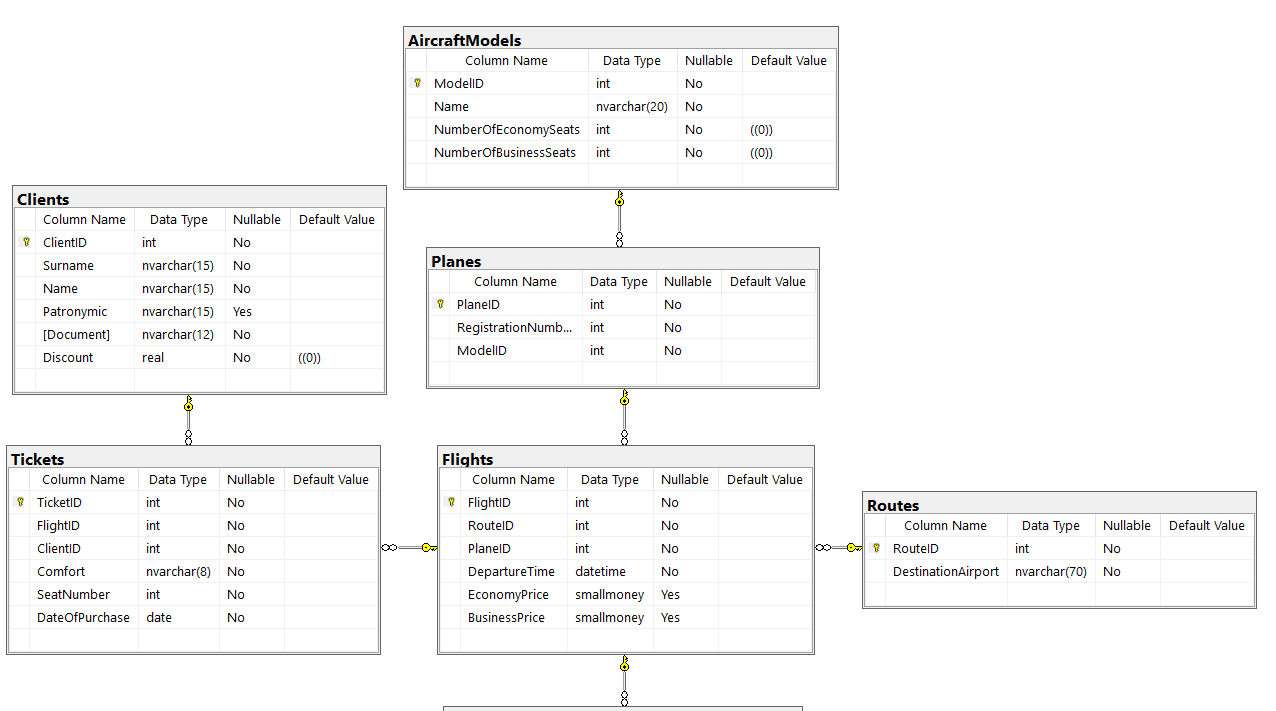


Рисунок 4.3 – Схема БД, згенерована засобами СУБД

1. Створення користувачів

Користувач «Власник БД», який був описаний у пункті 3.1.3 даної курсової роботи, був реалізований з використанням SQL-скрипту, представленого у підрозділі В.1 додатку В.

Перевірити, чи було створено бажаного користувача, можна за допомогою наступної команди:

EXEC sp\_helpuser

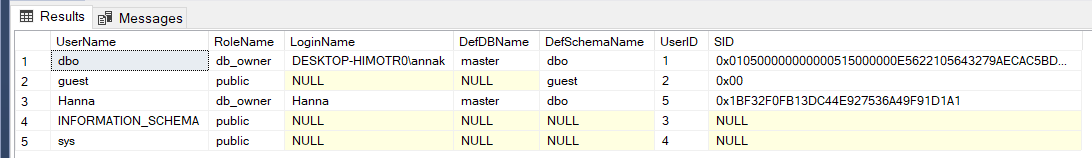
На рисунку 4.4 наведений результат виконання вищезазначеної команди. Можна побачити, що до списку користувачів бази даних був доданий користувач «Hanna» з привілеями власника бази даних.

Рисунок 4.4 – Результат додавання користувача «Hanna»

Тексти SQL-скриптів для створення решти змодельованих користувачів також наведені у додатку В.

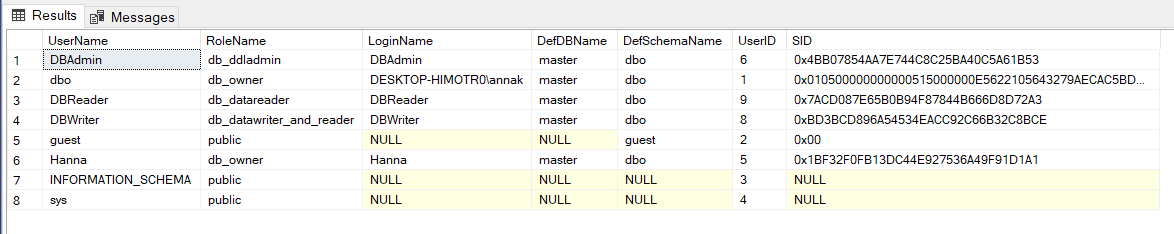
На рисунку 4.5 приведений результат виконання скриптів для створення користувачів бази даних «KR\_DB» (повторний виклик команди «sp\_helpuser» відобразив усіх створених користувачів: «Hanna», «DBAdmin», «DBWriter» та «DBReader»).

Рисунок 4.5 – Результат додавання користувачів до бази даних

1. Підбиття підсумків за розділом

Отже, у розділі «Реалізація бази даних» спроектована раніше модель БД була втілена з використанням системи управління базами даних Microsoft SQL Server. За побудованими концептуальною та логічною моделями БД були написані SQL-скрипти для створення таблиць та встановлення на них обмежень цілісності даних. Також на основі розробленої багатокористувацької моделі БД були створені відповідні користувачі та ролі, що мають власні права та привілеї при управлінні базою даних.

1. РОБОТА З БАЗОЮ ДАНИХ
2. Заповнення бази даних тестовими даними

Для початку роботи з побудованою базою даних необхідно додати до неї тестові дані. Тексти SQL-скриптів для додавання даних до таблиць бази даних представлені у додатку В. Результати виконання цих скриптів (таблиці з вставленими даними) наведені у додатку Д.

1. Створення представлень
2. Представлення «PurchaseOfTickets»

Для спрощення подальшої роботи зі збереженими процедурами та функціями було створено представлення «PurchaseOfTickets», яке слугує для подання придбаних квитків із вказанням ID рейсу, ID самого квитка, класу місця та ціни, яку заплатив конкретний користувач (тобто з урахуванням його знижок).

Текст SQL-скрипту, що був використаний при створенні даного представлення, наведений у додатку Е.

Перевіримо зміст представлення, запустивши наступну команду:

SELECT \* FROM PurchaseOfTickets

При запиті сформованого представлення програмне забезпечення видало наступну таблицю (рис. 5.1):

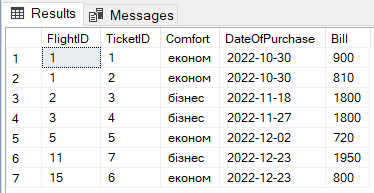


Рисунок 5.1 – Представлення «PurchaseOfTickets»

1. Представлення «TicketsEconomyClass»

Представлення «TicketsEconomyClass» було створено для відображення кількості заброньованих та доступних квитків економ-класу за кожним із запланованих та здійснених рейсів.

Текст SQL-скрипту, що був використаний для створення даного представлення, також наведений у додатку Е.

Зміст представлення перевіряємо наступною командою:

SELECT \* FROM TicketsEconomyClass

Результат виконання даного запиту наведений на рисунку 5.2.

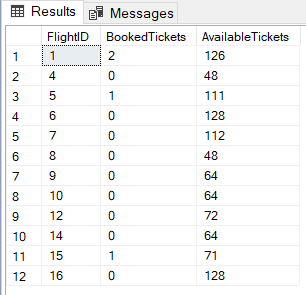


Рисунок 5.2 – Представлення «TicketsEconomyClass»

1. Представлення «TicketsBusinessClass»

Представлення «TicketsBusinessClass» було створено для відображення кількості заброньованих та доступних квитків бізнес-класу за кожним із запланованих та здійснених рейсів.

Текст SQL-скрипту, що був використаний для створення даного представлення, також наведений у додатку Е.

Для перевірки змісту сформованого представлення запустимо наступну команду:

SELECT \* FROM TicketsBusinessClass

Результат її виконання зображений на рисунку 5.3.

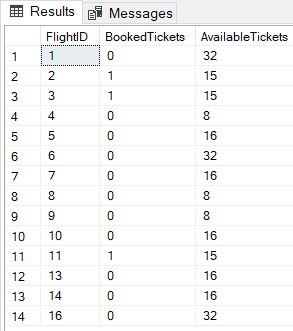


Рисунок 5.3 – Представлення «TicketsBusinessClass»

1. Створення збережених процедур
2. Процедура «countIncome»

Процедура «countIncome» була створена для того, щоб обчислювати загальний дохід, що приносить аеропорт (прибуток від продажу квитків). Скрипти для її створення та тестування наведені у додатку Ж. У роботі дана процедура використовує створене раніше представлення «PurchaseOfTickets».

Результат роботи даної процедури наведений на рисунку 5.4.



Рисунок 5.4 – Результат виконання процедури «countIncome»

1. Процедура «countExpenses»

Процедура «countExpenses» була створена для того, щоб обчислювати загальні витрати, що несе аеропорт (на виплати заробітної плати працівникам). Скрипти для її створення та тестування також наведені у додатку Ж.

Результат роботи даної процедури наведений на рисунку 5.5.



Рисунок 5.5 – Результат виконання процедури «countExpenses»

1. Процедура «whetherTurnedIntoProfit»

Для перевірки, чи вийшов аеропорт в плюс (доходи перевищили витрати) була створена процедура «whetherTurnedIntoProfit», скрипт до якої наведений у додатку Ж. Результатом її виконання є виведене повідомлення про фінансовий стан аеропорту. У роботі дана процедура використовує створені раніше процедури «countIncome» та «countExpenses».

Для перевірки правильності роботи вищезазначеної процедури, запустимо наступну команду:

EXEC whetherTurnedIntoProfit

На рисунку 5.6 можна побачити результат виконання даної команди, з якого видно, що аеропорт поки вийшов у мінус, оскільки витрати на виплату заробітної плати перевищують доходи від продажу квитків.



Рисунок 5.6 – Результат роботи процедури «whetherTurnedIntoProfit»

1. Створення функцій
2. Функція «getFlightReport»

База даних аеропорту повинна надавати користувачам можливість відслідковувати дохід за кожним із рейсів, тобто дізнаватися кількість проданих квитків та заробіток із них. Для цієї задачі була написана функція «getFlightReport», скрипт для створення якої наведений у додатку З.

Правильність роботи функції перевіримо наступною командою:

SELECT \* FROM getFlightReport(1)

За отриманим виводом (рис. 5.7) можна зробити висновок, що функція працює коректно і надає звіт по запитаному рейсу. При обрахунку доходу вона також враховує знижки, які може мати кожен із клієнтів, що призводить до отримання точного результату.

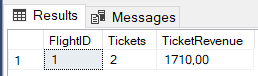


Рисунок 5.7 – Результат виклику функції «getFlightReport»

1. Функція «getTicketsReportForPeriod»

Програмне забезпечення, яке проєктується, також повинно забезпечувати користувачів можливістю отримувати фінансовий звіт по квиткам, проданим впродовж певного періоду часу. Тож функція «getTicketsReportForPeriod», скрипт до якої також наведений у додатку З, повертає таблицю, що містить загальну кількість квитків, проданих у заданий період часу, а також дохід з продажу цих квитків. У роботі ця та попередня функції використовують створене раніше представлення «PurchaseOfTickets».

Перевіримо роботу функції, запустивши наступну команду:

SELECT \* FROM getTicketsReportForPeriod('2022-10-20', '2022-11-20')

Результат виконання обчислень наведений на рисунку 5.8.

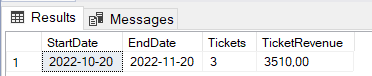


Рисунок 5.8 – Результат виклику функції «getTicketsReportForPeriod»

1. Створення тригерів
2. Тригер «on\_insert\_tickets»

У ході роботи з базою даних було виявлено необхідність у створенні тригера, який при кожному додаванні даних до таблиці «Tickets» буде перевіряти, чи доступне дане місце у літаку на даному рейсі (чи не виходить його номер за загальну кількість місць відповідного класу у літаку), а також, чи дата купівлі квитка не перевищує дату здійснення самого рейсу (оскільки неможливо купити квиток на рейс після відправки літака).

SQL-скрипти, що були використані для створення цього та інших тригерів, а також для проведення тестування створених тригерів, наведені у додатку И.

На рисунку 5.9 наведений приклад роботи створеного тригера при намаганні додати квиток на місце, якого не існує.

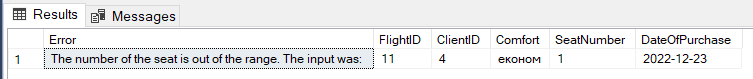


Рисунок 5.9 – Приклад роботи тригера «on\_insert\_tickets» для хибних даних

Наступний приклад роботи створеного тригера – спроба додати квиток на рейс, який вже відправився (рис. 5.10).

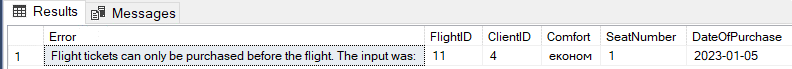


Рисунок 5.10 – Приклад роботи тригера «on\_insert\_tickets» для хибних даних

І останній приклад роботи тригера – додавання коректних даних до таблиці «Tickets». Дані були правильно ідентифіковані та, відповідно, додані до заданої таблиці (рис. 5.11).

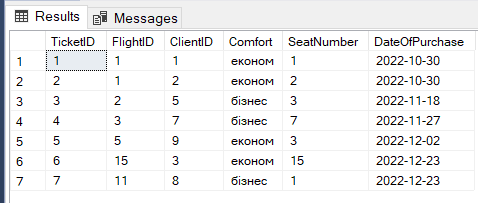


Рисунок 5.11 – Приклад роботи тригера «on\_insert\_tickets» для коректних даних

1. Тригер «on\_insert\_flights»

Програмне забезпечення також повинно при додаванні даних до таблиці «Flights» перевіряти, чи введена ціна на місця певного класу, якщо такі місця передбачено моделлю літака, що здійснює рейс.

Приклад роботи даного тригера, що спрацював при виконанні скрипта із додатку И, наведений на рисунку 5.12. Дійсно, у літака, що має «PlaneID» рівним «1», вказана кількість місць економ- та бізнес-класу, яка відмінна від нуля, тому при створенні рейсу з даним літаком повинна бути встановлена ціна на місця відповідних класів.

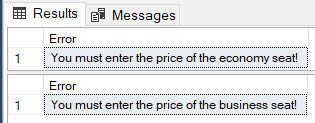


Рисунок 5.12 – Результат роботи тригера «on\_insert\_flights»

1. Тригер «on\_insert\_salary»

Ще одна задача даного програмного забезпечення – впевнитися, що заробітня плата не була виплачена працівнику до його працевлаштування, тобто що дата виплати йде після дати влаштування на роботу. Для цього був створений тригер «on\_insert\_salary», приклад роботи якого наведений на рисунку 5.13.

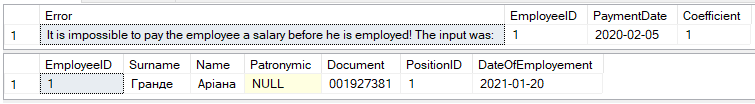


Рисунок 5.13 – Результат роботи тригера «on\_insert\_salary»

1. Написання SQL-запитів
2. Список клієнтів, що придбали квитки

Запит, що виводить список клієнтів, котрі вже придбали квитки:

SELECT \*

FROM [Clients]

WHERE [ClientID] IN (SELECT [ClientID] FROM [Tickets])

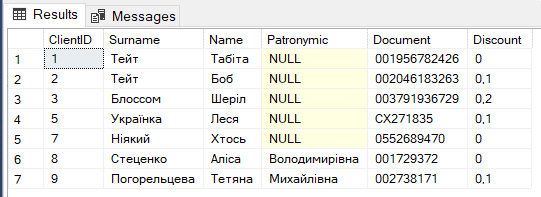


Рисунок 5.14 – Результат виконання запиту

1. Ціни на квитки для окремого клієнта

Запит, що виводить ціни на квитки на рейси, що ще не відправлені, з урахуванням знижки, що має конкретний клієнт:

DECLARE @ClientID INT

SET @ClientID = 9

SELECT F.[FlightID], F.[EconomyPrice] \* (1 - C.[Discount]) [EconomyPrice],

F.[BusinessPrice] \* (1 - C.[Discount]) [BusinessPrice]

FROM [Clients] C, [Flights] F

WHERE C.[ClientID] = @ClientID AND F.[DepartureTime] > CURRENT\_TIMESTAMP

SELECT CURRENT\_TIMESTAMP CurrTime

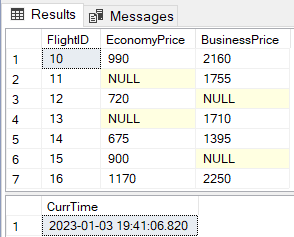


Рисунок 5.15 - Результат виконання запиту

1. Інформація про квитки економ-класу для кожного рейсу

Запит, що надає перелік рейсів з указанням цін на квитки економ-класу та кількістю вільних місць відповідного класу:

SELECT F.[FlightID], F.[DepartureTime], R.[DestinationAirport],

F.[EconomyPrice], V.[AvailableTickets]

FROM [Flights] F, [Routes] R, TicketsEconomyClass V

WHERE F.[RouteID] = R.[RouteID] AND F.[FlightID] = V.[FlightID]

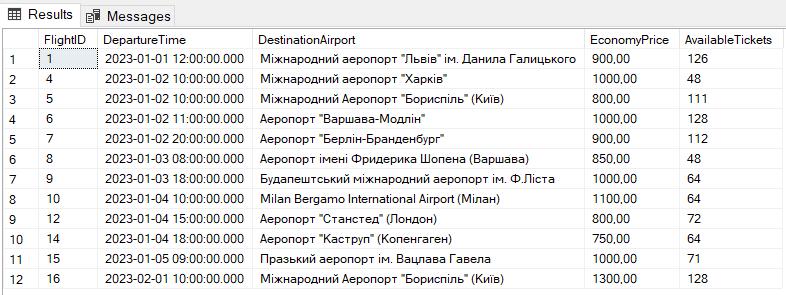


Рисунок 5.16 – Результат виконання запиту

1. Інформація про квитки бізнес-класу для кожного рейсу

Запит, що надає перелік рейсів з указанням цін на квитки бізнес-класу та кількістю вільних місць відповідного класу:

SELECT F.[FlightID], F.[DepartureTime], R.[DestinationAirport],

F.[BusinessPrice], V.[AvailableTickets]

FROM [Flights] F, [Routes] R, TicketsBusinessClass V

WHERE F.[RouteID] = R.[RouteID] AND F.[FlightID] = V.[FlightID]

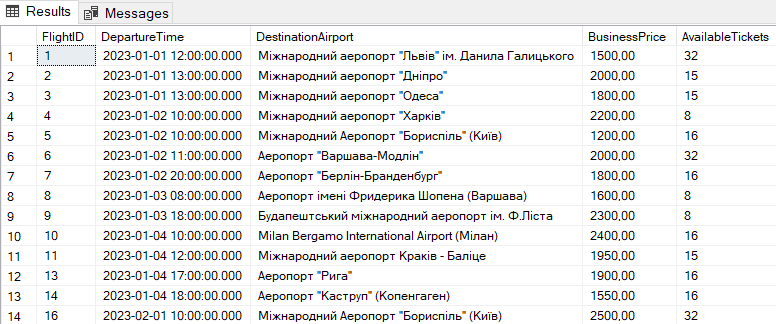


Рисунок 5.17 – Результат виконання запиту

1. Інформація по виплатам кожному працівнику

Запит, що виводить загальну суму виплат (заробітня плата та премії) за кожним із працівників аеропорту.

SELECT E.[EmployeeID], ROUND(SUM(P.[Salary] \* S.[Coefficient]), 2) Payment

FROM [Personnel] E

INNER JOIN [Positions] P

ON E.[PositionID] = P.[PositionID]

INNER JOIN [SalaryPayment] S

ON E.[EmployeeID] = S.[EmployeeID]

GROUP BY E.[EmployeeID]

ORDER BY Payment DESC

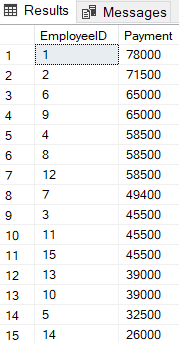


Рисунок 5.18 – Результат виконання запиту

1. Перелік літаків аеропорту

Запит, що виводить список літаків з вказанням їх бортового номеру та моделі:

SELECT [PlaneID], [RegistrationNumber], (SELECT M.[Name] FROM [AircraftModels] M

WHERE M.[ModelID] = P.[ModelID]) Model

FROM [Planes] P

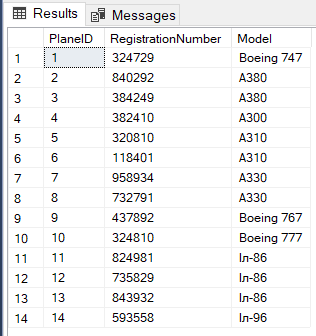


Рисунок 5.19 – Результат виконання запиту

1. Інформація по кількості квитків, замовлених різними клієнтами

Запит, що виводить список клієнтів із зазначенням кількості квитків, придбаних кожним із них:

SELECT C.[ClientID] ID, C.[Name] + ' ' + C.[Surname] FullName,

C.[Document], COUNT(T.[TicketID]) BookedTickets

FROM [Clients] C

LEFT JOIN [Tickets] T

ON C.[ClientID] = T.[ClientID]

GROUP BY C.[ClientID], C.[Name] + ' ' + C.[Surname], C.[Document]

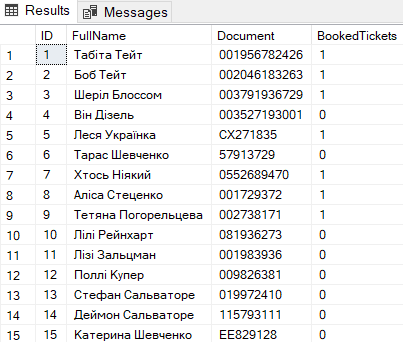


Рисунок 5.20 – Результат виконання запиту

1. Кількість працівників на кожній посаді

SELECT P.[PositionID], P.[Name] PositionName,

COUNT(E.[EmployeeID]) EmployeesNumber

FROM [Personnel] E, [Positions] P

WHERE E.[PositionID] = P.[PositionID]

GROUP BY P.[PositionID], P.[Name]

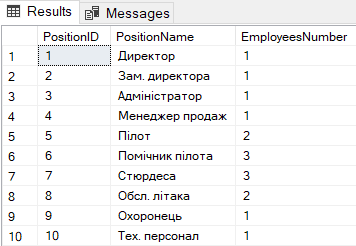


Рисунок 5.21 – Результат виконання запиту

1. Перелік працівників аеропорту

Запит, що виводить список працівників аеропорту із вказанням посад, які вони займають, та їх заробітної плати:

SELECT E.[Surname] + ' ' + E.[Name] FullName, E.[Document], E.[DateOfEmployement],

P.[Name] Position, P.[Salary]

FROM [Personnel] E, [Positions] P

WHERE E.[PositionID] = P.[PositionID]

ORDER BY FullName

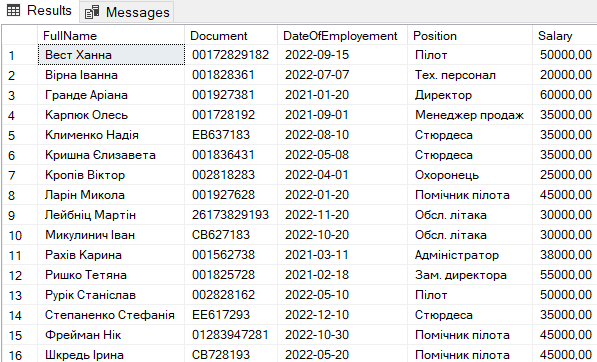


Рисунок 5.22 – Результат виконання запиту (частина)

1. Екіпаж літака, закріплений за певним рейсом

Запит, що виводить перелік працівників, що будуть супроводжувати рейс.

DECLARE @FlightID INT

SET @FlightID = 1

SELECT A.[FlightID], P.[Name] EmployeePosition, E.[EmployeeID],

E.[Surname] + ' ' + E.[Name] FullName

FROM [Aircrew] A, [Personnel] E, [Positions] P

WHERE A.[FlightID] = @FlightID AND

A.[EmployeeID] = E.[EmployeeID] AND

E.[PositionID] = P.[PositionID]

ORDER BY EmployeePosition, FullName

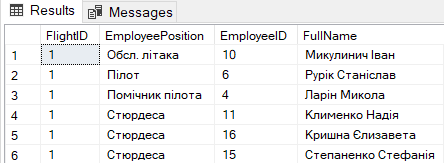


Рисунок 5.23 – Результат виконання запиту

1. Створення індексів

Для оптимізації роботи запитів були створені індекси. Для прикладу візьмемо наступний SQL-запит, що виводить інформацію про рейси:

SELECT F.[FlightID], F.[DepartureTime], R.[DestinationAirport],

F.[EconomyPrice]

FROM [Flights] F

INNER JOIN [Routes] R

ON F.[RouteID] = R.[RouteID]

Час виконання даного запиту без використання індексу показаний на рисунку 5.24.

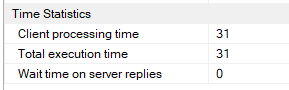


Рисунок 5.24 – Час виконання запиту без використання індексу

Запустимо скрипт для створення індексу для таблиці «Flights»:

CREATE INDEX flights\_index ON [Flights] ([FlightID], [RouteID]);

Після цього виконання запиту займає в 4 рази менше часу (рис. 5.25).

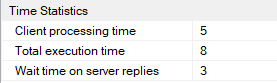


Рисунок 5.25 – Час виконання запиту з використанням індексу

1. Підбиття підсумків за розділом

Отже, у даному розділі були сформовані SQL-скрипти створення функцій, збережених процедур та представлень, необхідних для виконання різноманітних операцій у базі даних, а також скрипти створення тригерів, необхідних для забезпечення цілісності даних. Разом з цим були написані запити для надання потрібної інформації користувачам. Створені запити були оптимізовані завдяки використанню індексів, що пришвидшують роботу з базою даних і дозволяють отримувати доступ до даних миттєво.

# ВИСНОВКИ

У даній курсовій роботі було досліджено методи проєктування та реалізації реляційних баз даних, закріплено отримані під час вивчення дисципліни теоретичні знання.

Згідно з поставленою задачею було розроблено програмне забезпечення для відстеження фінансової сторони роботи аеропорту.

Під час виконання даної роботи було проаналізовано предметне середовище, визначено сутності та атрибути, а також зв’язки між об’єктами. Була побудована ER-модель заданої предметної області, на основі якої була побудована реляційна схема бази даних (тобто виділені первинні та зовнішні ключі сутностей, визначені обмеження для підтримки цілісності). З використанням засобів мови SQL були розроблені скрипти для реалізації спроектованої бази даних та заповнення її тестовими даними. Також були написані запити, функції, збережені процедури та тригери, необхідні для забезпечення всього функціоналу бази даних, необхідного користувачам.

У матеріалах до курсової роботи наведені всі сформовані SQL-скрипти та приклади їх виконання на тестовому наборі даних.

Таким чином, можна зробити висновок, що розроблене програмне забезпечення відповідає всім поставленим функціональним та нефункціональним вимогам, а також виконує всі поставлені задачі. Отже, результуюча база даних готова до практичного застосування.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Білоусова Л.І., Муравка А.С., Олефіренко Н.В. Інформатика 10-11: навч. посіб. Харків: Факт, 2009. 352 с.
2. Перша нормальна форма. Матеріал з Вікіпедії – вільної енциклопедії // URL: <https://uk.wikipedia.org/wiki/Перша_нормальна_форма> (дата звернення: 27.12.2022)
3. Друга нормальна форма. Матеріал з Вікіпедії – вільної енциклопедії // URL: <https://uk.wikipedia.org/wiki/Друга_нормальна_форма> (дата звернення: 27.12.2022)
4. Третя нормальна форма. Матеріал з Вікіпедії – вільної енциклопедії // URL: <https://uk.wikipedia.org/wiki/Третя_нормальна_форма> (дата звернення: 27.12.2022)

ДОДАТОК А ТЕКСТИ SQL-СКРИПТІВ ДЛЯ СТВОРЕННЯ ТАБЛИЦЬ

1. Текст SQL-скрипту для створення таблиці «Routes»

CREATE TABLE [Routes]

(

[RouteID] INT IDENTITY(1, 1) NOT NULL,

[DestinationAirport] NVARCHAR(70) NOT NULL,

CONSTRAINT [PK\_Routes] PRIMARY KEY CLUSTERED ([RouteID])

)

-- Встановити обмеження для таблиці [Routes]

ALTER TABLE [dbo].[Routes]

WITH CHECK ADD CONSTRAINT [CK\_Routes\_DestinationAirport]

CHECK ([DestinationAirport] <> '')

GO

1. Текст SQL-скрипту для створення таблиці «AircraftModels»

CREATE TABLE [AircraftModels]

(

[ModelID] INT IDENTITY(1, 1) NOT NULL,

[Name] NVARCHAR(20) NOT NULL,

[NumberOfEconomySeats] INT NOT NULL,

[NumberOfBusinessSeats] INT NOT NULL,

CONSTRAINT [PK\_AircraftModels] PRIMARY KEY CLUSTERED ([ModelID])

)

-- Встановити обмеження для таблиці [AircraftModels]

ALTER TABLE [dbo].[AircraftModels]

WITH CHECK ADD CONSTRAINT [CK\_AircraftModels\_Name]

CHECK ([Name] <> '')

GO

ALTER TABLE [dbo].[AircraftModels]

WITH CHECK ADD CONSTRAINT [CK\_AircraftModels\_NumberOfSeats]

CHECK (([NumberOfEconomySeats] >= 0) AND ([NumberOfBusinessSeats] >= 0))

GO

ALTER TABLE [dbo].[AircraftModels]

WITH CHECK ADD CONSTRAINT [DF\_AircraftModels\_NumberOfEconomySeats]

DEFAULT 0 FOR [NumberOfEconomySeats]

GO

ALTER TABLE [dbo].[AircraftModels]

WITH CHECK ADD CONSTRAINT

[DF\_AircraftModels\_NumberOfBusinessSeats]

DEFAULT 0 FOR [NumberOfBusinessSeats]

GO

1. Текст SQL-скрипту для створення таблиці «Planes»

CREATE TABLE [Planes]

(

[PlaneID] INT IDENTITY(1, 1) NOT NULL,

[RegistrationNumber] INT NOT NULL,

[ModelID] INT NOT NULL,

CONSTRAINT [PK\_Planes] PRIMARY KEY CLUSTERED ([PlaneID])

)

-- Встановити зв'язок FK між таблицями [Planes] та [AircraftModels]

-- Зв'язок: [Planes].[ModelID] -> [AircraftModels].[ModelID]

ALTER TABLE [dbo].[Planes]

WITH CHECK ADD CONSTRAINT [FK\_Planes\_AircraftModels]

FOREIGN KEY([ModelID])

REFERENCES [dbo].[AircraftModels] ([ModelID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити обмеження для таблиці [Planes]

ALTER TABLE [dbo].[Planes]

WITH CHECK ADD CONSTRAINT [CK\_Planes\_RegistrationNumber]

CHECK ([RegistrationNumber] > 0)

GO

ALTER TABLE [dbo].[Planes]

WITH CHECK ADD CONSTRAINT [UQ\_Planes\_RegistrationNumber]

UNIQUE ([RegistrationNumber])

GO

1. Текст SQL-скрипту для створення таблиці «Flights»

CREATE TABLE [Flights]

(

[FlightID] INT IDENTITY(1, 1) NOT NULL,

[RouteID] INT NOT NULL,

[PlaneID] INT NOT NULL,

[DepartureTime] DATETIME NOT NULL,

[EconomyPrice] SMALLMONEY NULL,

[BusinessPrice] SMALLMONEY NULL,

CONSTRAINT [PK\_Flights] PRIMARY KEY CLUSTERED ([FlightID])

)

-- Встановити зв'язок FK між таблицями [Flights] та [Routes]

-- Зв'язок: [Flights].[RouteID] -> [Routes].[RouteID]

ALTER TABLE [dbo].[Flights]

WITH CHECK ADD CONSTRAINT [FK\_Flights\_Routes]

FOREIGN KEY([RouteID])

REFERENCES [dbo].[Routes] ([RouteID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити зв'язок FK між таблицями [Flights] та [Planes]

-- Зв'язок: [Flights].[PlaneID] -> [Planes].[PlaneID]

ALTER TABLE [dbo].[Flights]

WITH CHECK ADD CONSTRAINT [FK\_Flights\_Planes]

FOREIGN KEY([PlaneID])

REFERENCES [dbo].Planes ([PlaneID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити обмеження для таблиці [Flights]

ALTER TABLE [dbo].[Flights]

WITH CHECK ADD CONSTRAINT [CK\_Flights\_Price]

CHECK (([BusinessPrice] >= 0 OR [BusinessPrice] IS NULL)

AND ([EconomyPrice] >= 0 OR [EconomyPrice] IS NULL))

GO

1. Текст SQL-скрипту для створення таблиці «Clients»

CREATE TABLE [Clients]

(

[ClientID] INT IDENTITY(1, 1) NOT NULL,

[Surname] NVARCHAR(15) NOT NULL,

[Name] NVARCHAR(15) NOT NULL,

[Patronymic] NVARCHAR(15) NULL,

[Document] NVARCHAR(12) NOT NULL,

[Discount] REAL NOT NULL,

CONSTRAINT [PK\_Clients] PRIMARY KEY CLUSTERED ([ClientID])

)

-- Встановити обмеження для таблиці [Clients]

ALTER TABLE [dbo].[Clients]

WITH CHECK ADD CONSTRAINT [CK\_Clients\_Surname]

CHECK ([Surname] <> '')

GO

ALTER TABLE [dbo].[Clients]

WITH CHECK ADD CONSTRAINT [CK\_Clients\_Name]

CHECK ([Name] <> '')

GO

ALTER TABLE [dbo].[Clients]

WITH CHECK ADD CONSTRAINT [UQ\_Clients\_Document]

UNIQUE ([Document])

GO

ALTER TABLE [dbo].[Clients]

WITH CHECK ADD CONSTRAINT [CK\_Clients\_Discount]

CHECK ([Discount] BETWEEN 0 AND 1)

GO

ALTER TABLE [dbo].[Clients]

WITH CHECK ADD CONSTRAINT [DF\_Clients\_Discount]

DEFAULT 0 FOR [Discount]

GO

1. Текст SQL-скрипту для створення таблиці «Tickets»

CREATE TABLE [Tickets]

(

[TicketID] INT IDENTITY(1, 1) NOT NULL,

[FlightID] INT NOT NULL,

[ClientID] INT NOT NULL,

[Comfort] NVARCHAR(8) NOT NULL,

[SeatNumber] INT NOT NULL,

[DateOfPurchase] DATE NOT NULL,

CONSTRAINT [PK\_Tickets] PRIMARY KEY CLUSTERED ([TicketID])

)

-- Встановити зв'язок FK між таблицями [Tickets] та [Flights]

-- Зв'язок: [Tickets].[FlightID] -> [Flights].[FlightID]

ALTER TABLE [dbo].[Tickets]

WITH CHECK ADD CONSTRAINT [FK\_Tickets\_Flights]

FOREIGN KEY([FlightID])

REFERENCES [dbo].[Flights] ([FlightID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити зв'язок FK між таблицями [Tickets] та [Clients]

-- Зв'язок: [Tickets].[ClientID] -> [Clients].[ClientID]

ALTER TABLE [dbo].[Tickets]

WITH CHECK ADD CONSTRAINT [FK\_Tickets\_Clients]

FOREIGN KEY([ClientID])

REFERENCES [dbo].[Clients] ([ClientID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити обмеження для таблиці [Tickets]

ALTER TABLE [dbo].[Tickets]

WITH CHECK ADD CONSTRAINT [CK\_Tickets\_Comfort]

CHECK (LOWER([Comfort]) IN ('бізнес', 'економ'))

GO

ALTER TABLE [dbo].[Tickets]

WITH CHECK ADD CONSTRAINT [UQ\_Tickets]

UNIQUE ([FlightID], [Comfort], [SeatNumber])

GO

1. Текст SQL-скрипту для створення таблиці «Positions»

CREATE TABLE [Positions]

(

[PositionID] INT IDENTITY(1, 1) NOT NULL,

[Name] NVARCHAR(15) NOT NULL,

[Salary] SMALLMONEY NOT NULL,

CONSTRAINT [PK\_Positions] PRIMARY KEY CLUSTERED ([PositionID])

)

-- Встановити обмеження для таблиці [Positions]

ALTER TABLE [dbo].[Positions]

WITH CHECK ADD CONSTRAINT [CK\_Positions\_Name]

CHECK ([Name] <> '')

GO

ALTER TABLE [dbo].[Positions]

WITH CHECK ADD CONSTRAINT [CK\_Positions\_Salary]

CHECK ([Salary] >= 0)

GO

1. Текст SQL-скрипту для створення таблиці «Personnel»

CREATE TABLE [Personnel]

(

[EmployeeID] INT IDENTITY(1, 1) NOT NULL,

[Surname] NVARCHAR(15) NOT NULL,

[Name] NVARCHAR(15) NOT NULL,

[Patronymic] NVARCHAR(15) NULL,

[Document] NVARCHAR(12) NOT NULL,

[PositionID] INT NOT NULL,

[DateOfEmployement] DATE NOT NULL,

CONSTRAINT [PK\_Personnel] PRIMARY KEY CLUSTERED ([EmployeeID])

)

-- Встановити зв'язок FK між таблицями [Personnel] та [Positions]

-- Зв'язок: [Personnel].[PositionID] -> [Positions].[PositionID]

ALTER TABLE [dbo].[Personnel]

WITH CHECK ADD CONSTRAINT [FK\_Personnel\_Positions]

FOREIGN KEY([PositionID])

REFERENCES [dbo].[Positions] ([PositionID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити обмеження для таблиці [Clients]

ALTER TABLE [dbo].[Personnel]

WITH CHECK ADD CONSTRAINT [CK\_Personnel\_Surname]

CHECK ([Surname] <> '')

GO

ALTER TABLE [dbo].[Personnel]

WITH CHECK ADD CONSTRAINT [CK\_Personnel\_Name]

CHECK ([Name] <> '')

GO

ALTER TABLE [dbo].[Personnel]

WITH CHECK ADD CONSTRAINT [UQ\_Personnel\_Document]

UNIQUE ([Document])

GO

1. Текст SQL-скрипту для створення таблиці «Aircrew»

CREATE TABLE [Aircrew]

(

[FlightID] INT NOT NULL,

[EmployeeID] INT NOT NULL,

CONSTRAINT [PK\_Aircrew] PRIMARY KEY CLUSTERED ([FlightID], [EmployeeID])

)

-- Встановити зв'язок FK між таблицями [Aircrew] та [Flights]

-- Зв'язок: [Aircrew].[FlightID] -> [Flights].[FlightID]

ALTER TABLE [dbo].[Aircrew]

WITH CHECK ADD CONSTRAINT [FK\_Aircrew\_Flights]

FOREIGN KEY([FlightID])

REFERENCES [dbo].[Flights] ([FlightID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити зв'язок FK між таблицями [Aircrew] та [Personnel]

-- Зв'язок: [Aircrew].[EmployeeID] -> [Personnel].[EmployeeID]

ALTER TABLE [dbo].[Aircrew]

WITH CHECK ADD CONSTRAINT [FK\_Aircrew\_Personnel]

FOREIGN KEY([EmployeeID])

REFERENCES [dbo].[Personnel] ([EmployeeID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

1. Текст SQL-скрипту для створення таблиці «SalaryPayment»

CREATE TABLE [SalaryPayment]

(

[PaymentID] INT IDENTITY(1, 1) NOT NULL,

[EmployeeID] INT NOT NULL,

[PaymentDate] DATE NOT NULL,

[Coefficient] REAL NOT NULL

CONSTRAINT [PK\_SalaryPayment] PRIMARY KEY CLUSTERED ([PaymentID])

)

-- Встановити зв'язок FK між таблицями [SalaryPayment] та [Personnel]

-- Зв'язок: [SalaryPayment].[EmployeeID] -> [Personnel].[EmployeeID]

ALTER TABLE [dbo].[SalaryPayment]

WITH CHECK ADD CONSTRAINT [FK\_SalaryPayment\_Personnel]

FOREIGN KEY([EmployeeID])

REFERENCES [dbo].[Personnel] ([EmployeeID])

ON UPDATE CASCADE ON DELETE CASCADE

GO

-- Встановити обмеження для таблиці [SalaryPayment]

ALTER TABLE [dbo].[SalaryPayment]

WITH CHECK ADD CONSTRAINT [DF\_SalaryPayment\_Coefficient]

DEFAULT 1 FOR [Coefficient]

GO

ALTER TABLE [dbo].[SalaryPayment]

WITH CHECK ADD CONSTRAINT [CK\_SalaryPayment\_Coefficient]

CHECK ([Coefficient] > 0)

GO

ДОДАТОК Б СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ

1. Текст SQL-скрипту для створення користувача «Власник БД»

-- Створення імені користувача (логіну) Hanna з паролем "Iamtheowner".

CREATE LOGIN Hanna

WITH PASSWORD = 'Iamtheowner';

GO

-- Створення користувача БД для логіну, створеного вище.

CREATE USER Hanna FOR LOGIN Hanna;

GO

-- Надання користувачу Hanna відповідної ролі власника БД db\_owner.

EXEC sp\_addrolemember 'db\_owner', 'Hanna'

1. Текст SQL-скрипту для створення користувача «Адміністратор БД»

CREATE LOGIN DBAdmin

WITH PASSWORD = 'Iamtheadmin';

GO

CREATE USER DBAdmin FOR LOGIN DBAdmin;

GO

EXEC sp\_addrolemember 'db\_ddladmin', 'DBAdmin'

1. Текст SQL-скрипту для створення користувача «Записувач даних»

CREATE ROLE db\_datawriter\_and\_reader

GO

GRANT INSERT, UPDATE, DELETE, SELECT TO db\_datawriter\_and\_reader

GO

CREATE LOGIN DBWriter

WITH PASSWORD = 'Iamthedatawriter';

GO

CREATE USER DBWriter FOR LOGIN DBWriter;

GO

EXEC sp\_addrolemember 'db\_datawriter\_and\_reader', 'DBWriter'

1. Текст SQL-скрипту для створення користувача «Читач даних»

CREATE LOGIN DBReader

WITH PASSWORD = 'Iamthedatareader';

GO

CREATE USER DBReader FOR LOGIN DBReader;

GO

EXEC sp\_addrolemember 'db\_datareader', 'DBReader'

ДОДАТОК В ЗАПОВНЕННЯ БАЗИ ДАНИХ ТЕСТОВИМИ ДАНИМИ

1. Текст SQL-скрипту для заповнення даними таблиці «Routes»

INSERT INTO [Routes]

([DestinationAirport])

VALUES

('Міжнародний аеропорт "Львів" ім. Данила Галицького'),

('Міжнародний аеропорт "Дніпро"'),

('Міжнародний аеропорт "Одеса"'),

('Міжнародний аеропорт "Харків"'),

('Міжнародний Аеропорт "Бориспіль" (Київ)'),

('Аеропорт "Варшава-Модлін"'),

('Аеропорт "Берлін-Бранденбург"'),

('Аеропорт імені Фридерика Шопена (Варшава)'),

('Будапештський міжнародний аеропорт ім. Ф.Ліста'),

('Milan Bergamo International Airport (Мілан)'),

('Міжнародний аеропорт Краків - Баліце'),

('Аеропорт "Станстед" (Лондон)'),

('Аеропорт "Рига"'),

('Аеропорт "Каструп" (Копенгаген)'),

('Празький аеропорт ім. Вацлава Гавела')

;

1. Текст SQL-скрипту для заповнення даними таблиці «AircraftModels»

INSERT INTO [AircraftModels]

([Name], [NumberOfEconomySeats], [NumberOfBusinessSeats])

VALUES

('Boeing 747', 128, 32),

('А380', 112, 16),

('A300', 64, 16),

('A310', 0, 16),

('A330', 48, 8),

('A340', 48, 12),

('Boeing 767', 64, 8),

('Boeing 777', 64, 16),

('Boeing 787', 36, 16),

('Іл-86', 72, 0),

('Іл-96', 32, 8)

;

1. Текст SQL-скрипту для заповнення даними таблиці «Planes»

INSERT INTO [Planes]

([RegistrationNumber], [ModelID])

VALUES

(324729, 1),

(840292, 2),

(384249, 2),

(382410, 3),

(320810, 4),

(118401, 4),

(958934, 5),

(732791, 5),

(437892, 7),

(324810, 8),

(824981, 10),

(735829, 10),

(843932, 10),

(593558, 11)

;

1. Текст SQL-скрипту для заповнення даними таблиці «Flights»

INSERT INTO [Flights]

([RouteID], [PlaneID], [DepartureTime], [EconomyPrice], [BusinessPrice])

VALUES

(1, 1, '2023-01-01 12:00', 900, 1500),

(2, 5, '2023-01-01 13:00', null, 2000),

(3, 6, '2023-01-01 13:00', null, 1800),

(4, 7, '2023-01-02 10:00', 1000, 2200),

(5, 3, '2023-01-02 10:00', 800, 1200),

(6, 1, '2023-01-02 11:00', 1000, 2000),

(7, 2, '2023-01-02 20:00', 900, 1800),

(8, 8, '2023-01-03 08:00', 850, 1600),

(9, 9, '2023-01-03 18:00', 1000, 2300),

(10, 10, '2023-01-04 10:00', 1100, 2400),

(11, 6, '2023-01-04 12:00', null, 1950),

(12, 12, '2023-01-04 15:00', 800, null),

(13, 5, '2023-01-04 17:00', null, 1900),

(14, 4, '2023-01-04 18:00', 750, 1550),

(15, 11, '2023-01-05 09:00', 1000, null)

;

1. Текст SQL-скрипту для заповнення даними таблиці «Clients»

INSERT INTO [Clients]

([Surname], [Name], [Patronymic], [Document])

VALUES

('Тейт', 'Табіта', null, '001956782426'),

('Тейт', 'Боб', null, '002046183263'),

('Блоссом', 'Шеріл', null, '003791936729'),

('Дізель', 'Він', null, '003527193001'),

('Українка', 'Леся', null, 'CX271835'),

('Шевченко', 'Тарас', 'Григорович', '57913729'),

('Ніякий', 'Хтось', null, '0552689470'),

('Стеценко', 'Аліса', 'Володимирівна', '001729372'),

('Погорельцева', 'Тетяна', 'Михайлівна', '002738171'),

('Рейнхарт', 'Лілі', null, '081936273'),

('Зальцман', 'Лізі', null, '001983936'),

('Купер', 'Поллі', null, '009826381'),

('Сальваторе', 'Стефан', null, '019972410'),

('Сальваторе', 'Деймон', null, '115793111'),

('Шевченко', 'Катерина', 'Григорівна', 'EE829128')

;

GO

UPDATE [Clients]

SET [Discount] = 0.1

WHERE [ClientID] IN (2, 5, 6, 9)

GO

UPDATE [Clients]

SET [Discount] = 0.2

WHERE [ClientID] IN (3, 12)

GO

1. Текст SQL-скрипту для заповнення даними таблиці «Tickets»

INSERT INTO [Tickets]

([FlightID], [ClientID], [Comfort], [SeatNumber], [DateOfPurchase])

VALUES

(1, 1, 'економ', 1, '2022-10-30'),

(1, 2, 'економ', 2, '2022-10-30'),

(2, 5, 'бізнес', 3, '2022-11-18'),

(3, 7, 'бізнес', 7, '2022-11-27'),

(5, 9, 'економ', 3, '2022-12-02')

;

1. Текст SQL-скрипту для заповнення даними таблиці «Positions»

INSERT INTO [Positions]

([Name], [Salary])

VALUES

('Директор', 60000),

('Зам. директора', 55000),

('Адміністратор', 38000),

('Менеджер продаж', 35000),

('Пілот', 50000),

('Помічник пілота', 45000),

('Стюрдеса', 35000),

('Обсл. літака', 30000),

('Охоронець', 25000),

('Тех. персонал', 20000)

;

1. Текст SQL-скрипту для заповнення даними таблиці «Personnel»

INSERT INTO [Personnel]

([Surname], [Name], [Patronymic], [Document], [PositionID], [DateOfEmployement])

VALUES

('Гранде', 'Аріана', null, '001927381', 1, '2021-01-20'),

('Ришко', 'Тетяна', 'Миколаївна', '001825728', 2, '2021-02-18'),

('Карпюк', 'Олесь', null, '001728192', 4, '2021-09-01'),

('Ларін', 'Микола', 'Валерійович', '001927628', 6, '2022-01-20'),

('Кропів', 'Віктор', 'Вікторович', '002818283', 9, '2022-04-01'),

('Рурік', 'Станіслав', null, '002828162', 5, '2022-05-10'),

('Рахів', 'Карина', 'Андріївна', '001562738', 3, '2021-03-11'),

('Шкредь', 'Ірина', 'Вікторівна', 'CB728193', 6, '2022-05-20'),

('Вест', 'Ханна', null, '00172829182', 5, '2022-09-15'),

('Микулинич', 'Іван', 'Іванович', 'CB627183', 8, '2022-10-20'),

('Клименко', 'Надія', 'Степанівна', 'EB637183', 7, '2022-08-10'),

('Фрейман', 'Нік', null, '01283947281', 6, '2022-10-30'),

('Лейбніц', 'Мартін', null, '26173829193', 8, '2022-11-20'),

('Вірна', 'Іванна', 'Артемівна', '001828361', 10, '2022-07-07'),

('Степаненко', 'Стефанія', 'Степанівна', 'EE617293', 7, '2022-12-10'),

('Кришна', 'Єлизавета', 'Андріївна', '001836431', 7, '2022-05-08')

;

1. Текст SQL-скрипту для заповнення даними таблиці «Aircrew»

INSERT INTO [Aircrew]

([FlightID], [EmployeeID])

VALUES

(1, 6),

(1, 4),

(1, 11),

(1, 15),

(1, 16),

(1, 10),

(5, 9),

(5, 12),

(5, 11),

(5, 15),

(5, 16)

;

1. Текст SQL-скрипту для заповнення даними таблиці «SalaryPayment»

INSERT INTO [SalaryPayment]

([EmployeeID], [PaymentDate], [Coefficient])

VALUES

(1, '2022-11-30', 1),

(2, '2022-11-30', 1),

(3, '2022-11-30', 1),

(4, '2022-11-30', 1),

(5, '2022-11-30', 1),

(6, '2022-11-30', 1),

(7, '2022-11-30', 1),

(8, '2022-11-30', 1),

(9, '2022-11-30', 1),

(10, '2022-11-30', 1),

(11, '2022-11-30', 1),

(12, '2022-11-30', 1),

(13, '2022-11-30', 1),

(14, '2022-11-30', 1),

(15, '2022-11-30', 1),

(1, '2022-12-20', 0.3),

(2, '2022-12-20', 0.3),

(3, '2022-12-20', 0.3),

(4, '2022-12-20', 0.3),

(5, '2022-12-20', 0.3),

(6, '2022-12-20', 0.3),

(7, '2022-12-20', 0.3),

(8, '2022-12-20', 0.3),

(9, '2022-12-20', 0.3),

(10, '2022-12-20', 0.3),

(11, '2022-12-20', 0.3),

(12, '2022-12-20', 0.3),

(13, '2022-12-20', 0.3),

(14, '2022-12-20', 0.3),

(15, '2022-12-20', 0.3)

;

ДОДАТОК Д РЕЗУЛЬТАТИ ДОДАВАННЯ ДАНИХ ДО ТАБЛИЦЬ

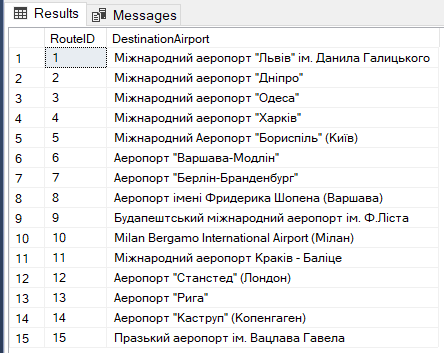


Рисунок Д.1 – Результат додавання даних до таблиці «Routes»

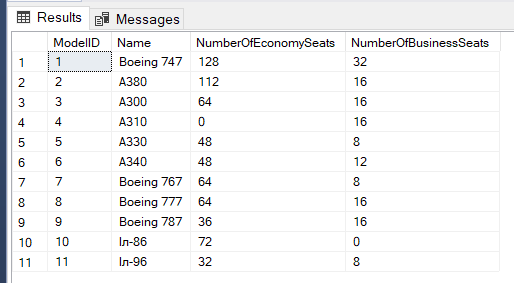


Рисунок Д.2 – Результат додавання даних до таблиці «AircraftModels»

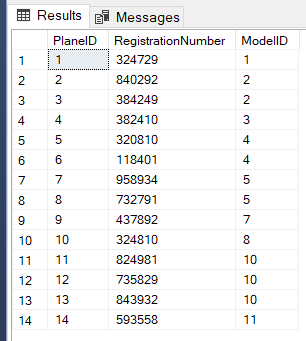


Рисунок Д.3 – Результат додавання даних до таблиці «Planes»

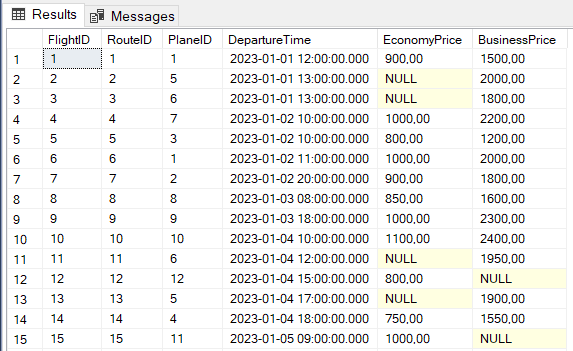


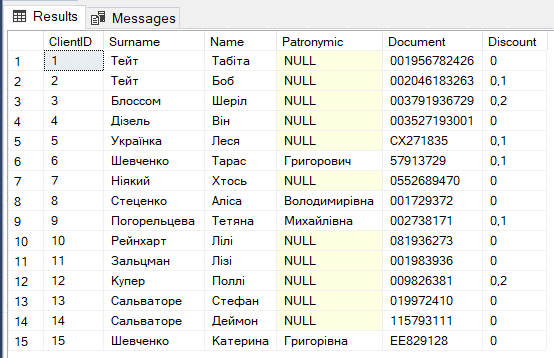
Рисунок Д.4 – Результат додавання даних до таблиці «Flights» 

Рисунок Д.5 – Результат додавання даних до таблиці «Clients»

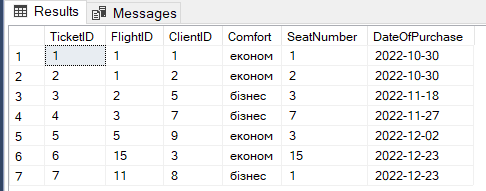


Рисунок Д.6 – Результат додавання даних до таблиці «Tickets»

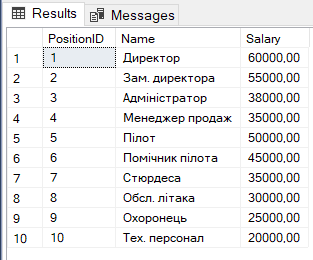


Рисунок Д.7 – Результат додавання даних до таблиці «Positions»

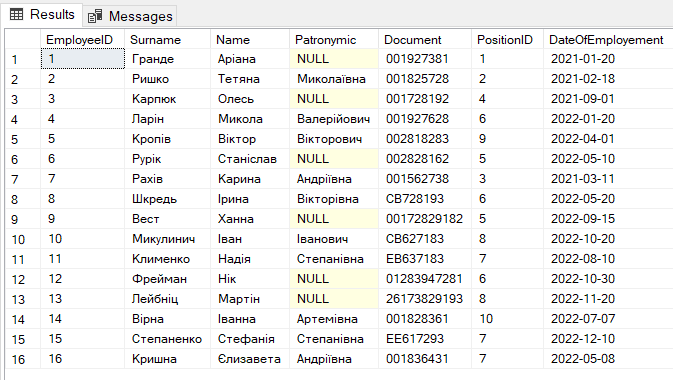


Рисунок Д.8 – Результат додавання даних до таблиці «Personnel»

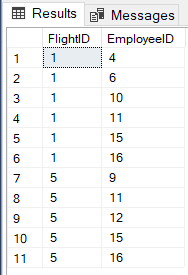


Рисунок Д.9 – Результат додавання даних до таблиці «Aircrew»

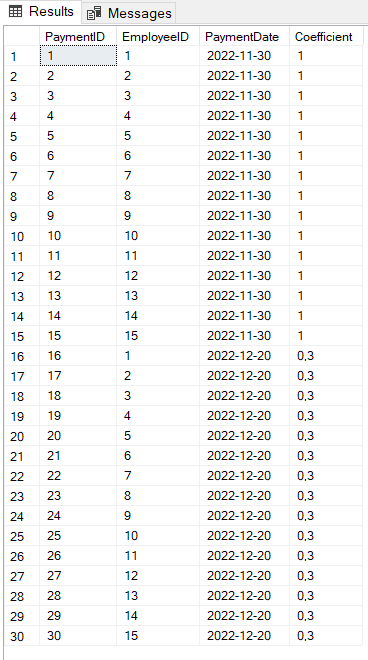


Рисунок Д.10 – Результат додавання даних до таблиці «SalaryPayment»

ДОДАТОК Е ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ПРЕДСТАВЛЕНЬ

1. Текст SQL-скрипту створення представлення «PurchaseOfTickets»

CREATE OR ALTER VIEW PurchaseOfTickets

AS SELECT F.[FlightID], T.[TicketID], T.[Comfort], T.[DateOfPurchase],

ISNULL(F.[EconomyPrice], 0) \* (1 - C.[Discount]) Bill

FROM [Flights] F, [Tickets] T, [Clients] C

WHERE T.[FlightID] = F.[FlightID] AND

LOWER(T.[Comfort]) = 'економ' AND T.[ClientID] = C.[ClientID]

UNION

SELECT F.[FlightID], T.[TicketID], T.[Comfort], T.[DateOfPurchase],

ISNULL(F.[BusinessPrice], 0) \* (1 - C.[Discount]) Bill

FROM [Flights] F, [Tickets] T, [Clients] C

WHERE T.[FlightID] = F.[FlightID] AND

LOWER(T.[Comfort]) = 'бізнес' AND T.[ClientID] = C.[ClientID]

GO

1. Текст SQL-скрипту створення представлення «TicketsEconomyClass»

CREATE OR ALTER VIEW TicketsEconomyClass

AS SELECT F.[FlightID], COUNT(T.TicketID) BookedTickets,

MAX(M.[NumberOfEconomySeats]) - COUNT(T.TicketID) AvailableTickets

FROM [Flights] F

LEFT JOIN [Tickets] T

ON F.[FlightID] = T.[FlightID] AND T.[Comfort] = 'економ'

LEFT JOIN [Planes] P

ON F.[PlaneID] = P.[PlaneID]

INNER JOIN [AircraftModels] M

ON P.[ModelID] = M.[ModelID] AND M.[NumberOfEconomySeats] > 0

GROUP BY F.[FlightID]

GO

1. Текст SQL-скрипту створення представлення «TicketsBusinessClass»

CREATE OR ALTER VIEW TicketsBusinessClass

AS SELECT F.[FlightID], COUNT(T.TicketID) BookedTickets,

MAX(M.[NumberOfBusinessSeats]) - COUNT(T.TicketID) AvailableTickets

FROM [Flights] F

LEFT JOIN [Tickets] T

ON F.[FlightID] = T.[FlightID] AND T.[Comfort] = 'бізнес'

LEFT JOIN [Planes] P

ON F.[PlaneID] = P.[PlaneID]

INNER JOIN [AircraftModels] M

ON P.[ModelID] = M.[ModelID] AND M.[NumberOfBusinessSeats] > 0

GROUP BY F.[FlightID]

GO

ДОДАТОК Ж ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР

1. Текст SQL-скрипту створення та тестування процедури «countIncome»

CREATE OR ALTER PROCEDURE countIncome @Income MONEY OUTPUT

AS

BEGIN

SELECT @Income = SUM([Bill])

FROM PurchaseOfTickets;

RETURN;

END

GO

DECLARE @Result MONEY;

EXEC countIncome @Income = @Result OUTPUT;

PRINT 'The income is: ' + CONVERT(varchar(10), @Result);

GO

1. Текст SQL-скрипту створення та тестування процедури «countExpenses»

CREATE OR ALTER PROCEDURE countExpenses @Expenses MONEY OUTPUT

AS

BEGIN

SELECT @Expenses = ROUND(SUM(P.[Salary] \* S.[Coefficient]), 2)

FROM [SalaryPayment] S, [Personnel] E, [Positions] P

WHERE S.[EmployeeID] = E.[EmployeeID] AND

E.[PositionID] = P.[PositionID]

RETURN;

END

GO

DECLARE @Result MONEY;

EXEC countExpenses @Expenses = @Result OUTPUT;

PRINT 'The expenses are: ' + CONVERT(varchar(10), @Result);

GO

1. Текст SQL-скрипту створення процедури «whetherTurnedIntoProfit»

CREATE OR ALTER PROCEDURE whetherTurnedIntoProfit

AS

BEGIN

DECLARE @Income MONEY, @Expenses MONEY;

EXEC countIncome @Income = @Income OUTPUT;

EXEC countExpenses @Expenses = @Expenses OUTPUT;

IF @Income > @Expenses

PRINT 'Congratulations, your airport is profitable!'

ELSE

PRINT 'Unfortunately, your airport has not yet turned a profit...'

END

GO

ДОДАТОК З ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ФУНКЦІЙ

1. Текст SQL-скрипту створення функції «getFlightReport»

CREATE OR ALTER FUNCTION getFlightReport (@FlightID INT)

RETURNS @Report TABLE (

[FlightID] INT NOT NULL,

[Tickets] INT NOT NULL,

[TicketRevenue] MONEY NOT NULL

)

AS

BEGIN

INSERT @Report

SELECT @FlightID, COUNT(\*),

(SELECT ISNULL(SUM(V.[Bill]), 0) FROM PurchaseOfTickets V

WHERE V.[FlightID] = @FlightID)

FROM [Tickets]

WHERE [Tickets].[FlightID] = @FlightID

RETURN;

END

GO

1. Текст SQL-скрипту створення функції «getTicketsReportForPeriod»

CREATE OR ALTER FUNCTION getTicketsReportForPeriod (@Start NVARCHAR(10), @End NVARCHAR(10))

RETURNS @Report TABLE (

[StartDate] NVARCHAR(10) NOT NULL,

[EndDate] NVARCHAR(10) NOT NULL,

[Tickets] INT NOT NULL,

[TicketRevenue] MONEY NOT NULL

)

AS

BEGIN

INSERT @Report

SELECT @Start, @End, COUNT(\*),

(SELECT ISNULL(SUM(V.[Bill]), 0) FROM PurchaseOfTickets V

WHERE V.[DateOfPurchase] BETWEEN @Start AND @End)

FROM [Tickets]

WHERE [Tickets].[DateOfPurchase] BETWEEN @Start AND @End

RETURN;

END

GO

ДОДАТОК И ТЕКСТИ SQL-СКРИПТІВ ДЛЯ ТРИГЕРІВ

1. Створення тригера «on\_insert\_tickets»

И.1.1 Текст SQL-скрипту, що слугує для створення тригера

CREATE OR ALTER TRIGGER on\_insert\_tickets

ON [Tickets]

INSTEAD OF INSERT

AS

BEGIN

DECLARE @FlightID INT, @ClientID INT, @Comfort NVARCHAR(8),

@SeatNumber INT, @Date DATE

SELECT @FlightID = [FlightID], @ClientID = [ClientID], @Comfort = [Comfort],

@SeatNumber = [SeatNumber], @Date = [DateOfPurchase] FROM inserted

IF DATEDIFF(DAY, @Date, (SELECT F.[DepartureTime] FROM [Flights] F

WHERE F.[FlightID] = @FlightID)) < 0

SELECT 'Flight tickets can only be purchased before the flight. The input was:'

Error, @FlightID FlightID, @ClientID ClientID, @Comfort Comfort,

@SeatNumber SeatNumber, @Date DateOfPurchase

ELSE IF LOWER(@Comfort) = 'економ' AND NOT @SeatNumber BETWEEN 1

AND (

SELECT [AircraftModels].[NumberOfEconomySeats]

FROM [Flights], [Planes], [AircraftModels]

WHERE @FlightID = [Flights].[FlightID]

AND [Flights].[PlaneID] = [Planes].[PlaneID]

AND [Planes].[ModelID] = [AircraftModels].[ModelID]

)

OR

LOWER(@Comfort) = 'бізнес' AND NOT @SeatNumber BETWEEN 1 AND (

SELECT [AircraftModels].[NumberOfBusinessSeats]

FROM [Flights], [Planes], [AircraftModels]

WHERE @FlightID = [Flights].[FlightID]

AND [Flights].[PlaneID] = [Planes].[PlaneID]

AND [Planes].[ModelID] = [AircraftModels].[ModelID]

)

SELECT 'The number of the seat is out of the range. The input was:' Error,

@FlightID FlightID, @ClientID ClientID, @Comfort Comfort, @SeatNumber

SeatNumber, @Date DateOfPurchase

ELSE

INSERT INTO [Tickets]

([FlightID], [ClientID], [Comfort], [SeatNumber], [DateOfPurchase])

VALUES

(@FlightID, @ClientID, @Comfort, @SeatNumber, @Date)

END

GO

И.1.2 Тексти SQL-скриптів для тестування тригера

INSERT INTO [Tickets]

([FlightID], [ClientID], [Comfort], [SeatNumber], [DateOfPurchase])

VALUES

(11, 4, 'економ', 1, '2022-12-23')

;

INSERT INTO [Tickets]

([FlightID], [ClientID], [Comfort], [SeatNumber], [DateOfPurchase])

VALUES

(11, 4, 'економ', 1, '2023-01-05')

;

INSERT INTO [Tickets]

([FlightID], [ClientID], [Comfort], [SeatNumber], [DateOfPurchase])

VALUES

(15, 3, 'економ', 15, '2022-12-23'),

(11, 8, 'бізнес', 1, '2022-12-23')

;

SELECT \* FROM [Tickets]

GO

1. Створення тригера «on\_insert\_flights»

И.2.1 Текст SQL-скрипту, що слугує для створення тригера

CREATE OR ALTER TRIGGER on\_insert\_flights

ON [Flights]

INSTEAD OF INSERT

AS

BEGIN

DECLARE @RouteID INT, @PlaneID INT, @DepartureTime DATETIME,

@EconomyPrice SMALLMONEY, @BusinessPrice SMALLMONEY

SELECT @RouteID = [RouteID], @PlaneID = [PlaneID],

@DepartureTime = [DepartureTime], @EconomyPrice = [EconomyPrice],

@BusinessPrice = [BusinessPrice] FROM inserted

IF @EconomyPrice IS NULL AND

(SELECT M.[NumberOfEconomySeats] FROM [AircraftModels] M, [Planes]

P WHERE P.PlaneID = @PlaneID AND M.ModelID = P.ModelID) > 0

SELECT 'You must enter the price of the economy seat!' Error

IF @BusinessPrice IS NULL AND

(SELECT M.[NumberOfBusinessSeats] FROM [AircraftModels] M, [Planes]

P WHERE P.PlaneID = @PlaneID AND M.ModelID = P.ModelID) > 0

SELECT 'You must enter the price of the business seat!' Error

ELSE

INSERT INTO [Flights]

([RouteID], [PlaneID], [DepartureTime], [EconomyPrice], [BusinessPrice])

VALUES

(@RouteID, @PlaneID, @DepartureTime, @EconomyPrice, @BusinessPrice)

END

GO

И.2.2 Тексти SQL-скриптів для тестування тригера

INSERT INTO [Flights]

([RouteID], [PlaneID], [DepartureTime], [EconomyPrice], [BusinessPrice])

VALUES

(5, 1, '2023-02-01 10:00', null, null)

;

GO

INSERT INTO [Flights]

([RouteID], [PlaneID], [DepartureTime], [EconomyPrice], [BusinessPrice])

VALUES

(5, 1 , '2023-02-01 10:00', 1300, 2500)

;

GO

SELECT \* FROM [Flights]

GO

1. Створення тригера «on\_insert\_salary»

И.3.1 Текст SQL-скрипту, що слугує для створення тригера

CREATE OR ALTER TRIGGER on\_insert\_salary

ON [SalaryPayment]

INSTEAD OF INSERT

AS

BEGIN

DECLARE @EmployeeID INT, @PaymentDate DATE,

@Coefficient REAL

SELECT @EmployeeID = [EmployeeID], @PaymentDate = [PaymentDate],

@Coefficient = [Coefficient] FROM inserted

IF DATEDIFF(DAY, @PaymentDate, (SELECT [DateOfEmployement]

FROM [Personnel] E WHERE E.[EmployeeID] = @EmployeeID)) > 0

SELECT 'It is impossible to pay the employee a salary before he is employed!

The input was:' Error, @EmployeeID EmployeeID, @PaymentDate

PaymentDate, @Coefficient Coefficient

ELSE

INSERT INTO [SalaryPayment]

([EmployeeID], [PaymentDate], [Coefficient])

VALUES

(@EmployeeID, @PaymentDate, @Coefficient)

END

GO

И.3.2 Текст SQL-скрипту для тестування тригера

INSERT INTO [SalaryPayment]

([EmployeeID], [PaymentDate], [Coefficient])

VALUES

(1, '2020-02-05', 1)

;

SELECT \* FROM [Personnel] WHERE [EmployeeID] = 1