Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “ Бази даних”

ІНФОРМАЦІЙНА СИСТЕМА БРОНЮВАННЯ АВІАКВИТКІВ

Керівник , доцент Мазурова О.О.

Студент гр. ПІ-15-2 Джафаров Е.Е.

Комісія: Доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мазурова О.О.

Ст. викл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю.

Ст. викл. \_\_\_\_\_\_\_\_\_\_\_\_\_ Широкопетлєва М.С.

Харків 2016

\_\_\_\_\_\_\_\_\_\_\_\_\_Харківський національний університет радіоелектроніки\_\_\_\_\_\_\_\_\_\_

**Кафедра** \_\_\_Програмної інженерії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дисципліна** \_\_\_Бази даних\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Спеціальність** \_Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Курс** \_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_ **Група** \_\_\_ПІ-15-2\_\_\_\_\_\_\_\_\_ **Семестр**\_\_\_\_ 3\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

\_\_\_\_\_\_\_\_\_\_\_\_*Джафарова Еміля Ельшан огли*\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Тема роботи:** \_\_\_\_Інформаційна система бронювання авіаквитків\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Строк здачі закінченої роботи** \_\_\_23.12.2016\_\_\_\_\_\_\_\_

**3. Вихідні дані для роботи:** методичні вказівки до виконання курсової роботи,\_вимоги до інформаційної системи, предметна область, що пов’язана з\_діяльністю аеропорту.

**4. Зміст розрахунково - пояснювальної записки:** вступ, аналіз предметної області; постановка задачі; проектування бази даних; опис програми; висновки; перелік посилань.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5. Перелік графічного матеріалу:** загальна схема концептуальної моделі, ER-діаграма, структура 1НФ, 2НФ, 3НФ, схема БД в 3НФ, UML-діаграми, копії екранів (“скріншоти”) прикладної програми, приклади звітів прикладної програми\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. Дата видачі завдання** \_\_\_09.09.16 р.\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Аналіз предметної області | 9.09.16 – 24.09.16 | Виконано |
| 2 | Постановка задачі | 20.09.16 – 30.09.16 | Виконано |
| 3 | Побудова ER-діаграми бази даних | 27.09.16 – 15.10.16 | Виконано |
| 4 | Оформлення розділів 1, 2 та 3.1, 3.2 пояснювальної записки | 15.10.16 - 27.10.16 | Виконано |
| 5 | Перша контрольна точка з курсового проекту | 24.10.16 – 28.10.16 | Виконано |
| 6 | Нормалізація бази даних | 20.10.16 - 10.11.16 | Виконано |
| 7 | Створення демо-версії програми | 20.10.16 – 20.11.16 | Виконано |
| 8 | Тестування програми, наповнення бази даних | 15.11.16 - 25.11.16 | Виконано |
| 9 | Друга контрольна точка з курсового проекту | 21.12.16– 02.12.16 | Виконано |
| 10 | Реалізація остаточної версії програми | 1.12.16-15.12.16 | Виконано |
| 11 | Оформлення інших розділів пояснювальної записки | 1.11.16 – 15.12.16 | Виконано |
| 12 | Захист курсового проекту (третя контрольна точка) | 12.12.16- 23.12.16 | Виконано |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *доц. Мазурова О.О.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 42 с., 13 рис., 7 табл., 7 ліст. з кодом,

6 джерел.

Метою роботи є розробка програми “Бронювання авіаквитків”, використовуючи реляційні бази даних.

Методи розробки базуються на використанні середи розробки IntelliJ IDEA, СУБД MySQL 5.7, мови програмування Java, технології доступу до даних JDBC, графічного процесору JavaFX Framework.

В результаті отримана програма під назвою “Бронювання авіаквитків” для роботи з базою даних. Програма дозволяє керувати багатьма аспектами життєдіяльності сучасного аеропорту. Найголовнішими функціями є можливість додавати, редагувати, видаляти рейси та бронювати квитки на обраний рейс.

ПРОГРАМА, БАЗА ДАНИХ, РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ, СУБД MYSQL, НОРМАЛІЗАЦІЯ ВІДНОШЕНЬ, ER-ДІАГРАМА, JDBC, МОВА ПРОГРАМУВАННЯ JAVA, JAVAFX FRAMEWORK, БРОНЮВАННЯ АВІАКВИТКІВ, РЕЙС, ПОЛІТ, БРОНЮВАННЯ, ПАСАЖИР.

ЗМІСТ

Вступ……….………………………………………………………………............6 1 Аналіз та концептуальне моделювання редметної області…………………...8

1.1 Опис функціональної структури інформаційної системи…………..............9

1.2 Опис об’єктів предметної області та зв’язки між ними………...................10

* 1. Опис інформаційних потреб користувачів……………...…….....................13
  2. Опис алгоритмічних залежностей…...…………………....……...................15
  3. Опис обмежень цілісності…………………………………….......................15

1.6 Опис існуючого документообігу предметної області…………...................16

* 1. Лінгвістичні відносини…………………………………………....................17

2 Постановка задачі…………………………………………………...................19

3 Проектування бази даних……………………………………….......................20

3.1 Uml – моделювання……………………………………………......................20

3.2 Побудова ER-діаграми.....................................................................................21

3.3 Побудова схеми реляційної бази даних у 3НФ…...……………...................23

4 Опис програми………………………………………………………................ 27

4.1 Загальні відомості………………………………………………….……….27 4.2 Виклик та завантаження………………………………………….................27

4.3 Призначення та логічна структура програми………….…………………...29

4.4 Опис фізичної моделі бази даних…………………………………………...30

4.5 Опис програмної реалізації………..………………………………………...32 4.6 Опис задачі автоматизації…………………………………………………...37

Висновки………………………………………………………………………… 41

Перелік посилань………………………………………………………………...42

ВСТУП

Важко уявити сучасне життя без перельотів між містами та країнами. Мільйони людей по всьому світу користуються послугами багатьох авіакомпаній. По всій планеті розкидані «термінали прийому літаків» - аеропорти. І кожен з них використовує дуже потужну систему керування авіарейсами.

Метою цього курсового проектування є створення інформаційної системи для робітників аеропорту під назвою «Бронювання авіаквитків». Користувачами цієї системи будуть переважно касири аеропорту. Вони будуть здійснювати бронювання квитків за побажаннями клієнтів – пасажирів. Однак, для повноцінної роботи системи потрібно вносити рейси до бази даних, і цим будуть займатися укладальники розкладу, тобто ті робітники аеропорту, які планують рейси, використовуючи послуги окремих авіакомпаній.

Для того, щоб кожен робітник аеропорту виконував свої і тільки свої зобов’язання була створена система з правами доступу до неї окремих особистостей.

Під бронюванням розуміють, що клієнт (пасажир) не знає заздалегідь своє місце в салоні літака. Йому тільки відомий номер рейсу, дата та час вильоту, клас зручності перельоту (бізнес клас, економ клас, тощо). Своє місце він дізнається вже при проходженні реєстрації. Це загальновідома процедура, яка використовується у багатьох аеропортах світу.

У результаті розроблено програмне забезпечення, яке виконує всі функції, які необхідні для правильного керування життєдіяльністю аеропорту. Воно зменшує кількість роботи робітникам аеропорту. Завдяки гарному та зрозумілому інтерфейсу користувачі системи мають можливість з легкістю маніпулювати рейсами в базі даних: додавати, редагувати, видаляти. А найголовніше - є можливість задовольняти потреби клієнтів (пасажирів), тобто бронювати авіаквитки із усіма їхніми побажаннями.

За основу розробки програми було узято об’єктно-орієнтований підхід, завдяки якому систему досить легко супроводжувати під час її експлуатації. Мовою програмування була обрана Java (SE 8). Для керування базою даних було обрано СУБД MySQL (Версія 5.7). Для розробки візуального графічного інтерфейсу була використана технологія JavaFX 8. Середовищем розробки було програмне забезпечення IntelliJ IDEA.

1 АНАЛІЗ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Сьогодні, у ХХІ столітті, все більше і більше людей користуються послугами авіакомпаній. Мабуть, це пов’язано з дуже швидким ритмом життя сучасної людини. Щодня з різних кутків планети здіймаються у повітря тисячі літаків з десятками тисяч пасажирів на борту. Проте, мало кому відомо, як же організовуються перельоти та що за цим криється. В основі будь-якого аеропорту покладена дуже потужна внутрішня система для керування авіарейсами. Завдяки такій інформаційній системі знижується ймовірність помилок при складанні розкладу перельотів. Великий шмат складних завдань бере на себе комп’ютер. Людині ж тепер не потрібно запам’ятовувати тисячі рейсів для бронювання авіаквитків або укладання розкладу польотів.

Зі збільшенням кількості перельотів постає необхідність в створенні інформаційної системи для керування авіарейсами. Ця система з кожним роком буде ставати все більше і більше за рахунок збільшення кількості зберігання інформації в базах даних. А це в свою чергу призведе до того, що зменшиться рівень якості пошуку потрібної інформації. Тому перед програмістами постає важливе завдання – створити якомога швидкий, зручний, надійний додаток для керування авіарейсами та іншими складовими сучасного аеропорту.

Найголовнішим процесом в аеропорті є бронювання авіаквитків пасажирами. Він складається з декількох етапів, а саме: пошук бажаного рейсу, перевірка касиром вільних місць на вказану дату та час відправлення, заповнення касиром персональних даних пасажира, підтвердження бронювання та друк талону для проходження реєстрації.

Також дуже важливим є процес додання нового рейсу до бази даних. Цим займаються менеджери або планувальники рейсів. Перш ніж додавати новий рейс, вони ознайомлюються із статистичними даними за попередні роки задля того, щоб найефективніше використовувати ресурси авіакомпаній. Після ознайомлення, менеджери заповнюють необхідні дані про рейс (номер, маршрут, класи зручності, дата та час відправлення тощо) і після цього інформація з’являється у розкладі рейсів та польотів.

Жоден аеропорт не може повноцінно працювати без робітників, таких як касири, планувальники рейсів, менеджери та інші. Процес їх взаємодії був би надзвичайно важким без єдиної інформаційної системи. Тому процес керування персоналом є також дуже важливим. Він складається з декількох етапів. Спочатку керівництво аеропорту влаштовує працівника на зацікавлену ним роботу, заповнюючи всі необхідні персональні дані (ПІБ, номер паспорту, ідентифікаційний номер тощо). Далі для працівника створюється акаунт для доступу до системи. Це в першу чергу робиться задля безпеки даних. Керівництво аеропорту може у будь-який момент часу проаналізувати продуктивність робітника та звільнити його за необхідності.

Найголовнішою складовою інформаційної системи є переліт. Він базується на окремому рейсі, тобто один рейс може мати багато перельотів. Переліт відбувається у конкретний момент часу, він є одиничним.

* 1. Опис функціональної структури інформаційної системи

Інформаційна система «Бронювання авіаквитків» реалізовує такі функції:

* додання, редагування, видалення авіарейсів з бази даних;
* знаходження потрібного рейсу, використовуючи фільтрацію за критеріями: дата відправлення, маршрут, класи зручності;
* перегляд інформації стосовно шуканого за номером рейсу (функція пошуку);
* бронювання квитків касиром із усіма побажаннями клієнта;
* можливість для перегляду інформації щодо пасажирів обраного рейсу та її редагування;
* можливість для перегляду різнопланової статистики (кількість перельотів по рокам, найбільш популярні авіакомпанії за останній час, пасажиро обіг за минулі роки тощо) менеджерами або планувальниками рейсів у вигляді статистичних діаграм;
* співробітникам відповідно до їхніх посад надається доступ до системи;
* забезпечення безпеки при вході до системи для співробітників у вигляді унікального логіну та паролю;
* можливість неодноразової зміни паролю для входу до системи.
  1. Опис об’єктів предметної області та зв’язки між ними

Базуючись на аналізі предметної області, інформаційна система повинна облегшити роботу касира та інших робітників аеропорту. Вона буде використовуватися декількома робітниками, кожному з яких буде надано відповідний доступ до системи. Наприклад, касир не зможе редагувати рейси, він має право тільки на їх перегляд. Але зможе вносити персональні дані чергового пасажира до бази даних та бронювати квитки. Обов’язково у системі буде адміністратор, який зможе приймати на роботу людей на відповідні посади, попередньо додавши їх до бази даних.

Основними об’єктами системи є:

* пасажир;
* робітник;
* рейс;
* замовлення.

Об’єкт «пасажир» включає всі необхідні персональні дані пасажира, а саме:

* код пасажира;
* ім’я;
* прізвище;
* по батькові;
* дата народження;
* номер паспорту;
* громадянство;
* знижка на квиток (якщо така існує).

Об’єкт «робітник» включає також всі необхідні персональні дані робітника та інформацію щодо посади, яку обіймає:

* код робітника;
* ім’я;
* прізвище;
* по батькові;
* дата народження;
* номер паспорту;
* податковий код;
* громадянство;
* посада робітника;
* зарплата робітника;
* логін для доступу до системи;
* дата влаштування на роботу.

Об’єкт «рейс» включає в себе таку інформацію:

* код рейсу;
* номер рейсу;
* маршрут;
* дата та час відправлення;
* тривалість польоту;
* авіакомпанія-перевізник;
* вартість перельоту із урахуванням рівнів зручності;
* загальна кількість посадкових місць у класах зручностей.

Об’єкт «замовлення» включає в себе наступну інформацію:

* код пасажира;
* код рейсу;
* код рівня зручності перельоту;
* код касира, який здійснив замовлення;
* дата та час бронювання.

Схема взаємодії головних об’єктів системи представлена на рисунку 1.1.



Рисунок 1.1 – Схема взаємодії об’єктів предметної області

1.3 Опис інформаційних потреб користувачів

Користувачами програмного забезпечення під назвою «Бронювання квитків» є робітники аеропорту. При влаштуванні чергового робітника йому надається посада, а разом і з нею права на доступ до бази даних цього програмного додатку, звісно якщо посада цього потребує. На рисунку 1.2 представлено діаграму інформаційних потреб користувачів.

Для роботи касира з програмою треба надати всі потрібні можливості, які є необхідними для правильної роботи, а саме:

* внесення і редагування персональних даних клієнта (пасажира) до бази даних;
* редагування списку пасажирів обраного рейсу;
* перегляд детальної інформації щодо пасажирів обраного рейсу;
* пошук та фільтрація рейсів за такими полями: маршрут, дата відправлення, класи зручності перельоту;
* зміна паролю для входу до системи.

Для роботи планувальника рейсів або менеджера з програмою треба надати всі необхідні можливості, які можуть потребуватися для ефективної роботи:

* внесення інформації про рейс до бази даних;
* редагування інформації обраного рейсу;
* видалення інформації обраного рейсу;
* перегляд статистики щодо польотів у вигляді діаграм для подальшого планування рейсів (кількість польотів та рейсів по рокам, найпопулярніші авіакомпанії, пасажиро обіг за минулі роки тощо);
* зміна паролю до входу до системи.

Адміністратори системи мають можливість на виконання таких функцій системи:

* влаштовувати людей на роботу;
* звільняти з роботи;
* додавати, редагувати, видаляти нові посади та вказувати зарплату на них;
* додавати типи знижок, а також їхні значення;
* редагування будь-яких даних, наявних в базі даних.

|  |
| --- |
| Перегляд розкладу  Редагування рейсу/польоту  Перегляд рейсової статистики  Вхід  Керування співробітниками  Бронювання квитків  Додання рейсу/польоту  Видалення рейсу/польоту  Керівництво  Касир  Менеджер  Користувач |

Рисунок 1.2 – Діаграма інформаційних потреб користувачів

Також у програмі присутня автоматизація процесу додавання польотів у базу даних. Планувальнику польотів тепер не потрібно вводити декілька разів інформацію, щоб занести до бази даних декілька польотів, які мають ті ж самі характеристики. Він просто вказує маршрут і тривалість здійснення польотів (один рік, три місяці, тощо). А програма запропонує на основі проведеного аналізу бажану кількість польотів та період між ними. Інформаційна система «Бронювання авіаквитків» вираховує сезон року, на який планується створити рейс, та підраховує щільність заповнення салону літака у той самий сезон у минулому і якщо щільність була досить високою, вона запропонує більші значення кількості польотів та менші значення періоду між польотами. Це може бути корисним для планувальників рейсів. Програма сама додасть до бази даних встановлену кількість польотів створюваного рейсу із різними датою та часом польоту.

В результаті програмне забезпечення, піклуючись про робітника аеропорту, зберігає йому багато часу та сил, а також усуває можливі помилки при створенні рейсу.

* 1. Опис алгоритмічних залежностей

Інформаційна система «Бронювання авіаквитків» має алгоритмічні залежності. Бронювання квитків відбувається тільки за наявністю вільних, тобто незаброньованих місць. Для знаходження рейсу за вказаними маршрутом та рівнем зручності система виконує пошук серед всіх рейсів і відображає тільки ті, де кількість вільних місць для вказаного рівня зручності перевищує або дорівнює одиниці (один пасажир може забронювати на власне ім’я тільки один квиток за один переліт) при формуванні пошукового запиту. Значення загальної кількості посадкових місць задається при створенні рейсу.

Для розрахунку остаточної вартості квитка використовується формула: res = cost\*discount. В якій cost – це вартість квитка із урахуванням класу зручності (задається при створенні рейсу), а discount – знижка, яку може отримати постійний клієнт.

* 1. Опис обмежень цілісності

Інформаційна система має наступні обмеження цілісності:

* один касир може здійснити багато замовлень (1:∞);
* один пасажир може забронювати тільки одне місце на один рейс, звісно враховуючи максимальну кількість місць (1:1);
* один рейс може мати багато польотів (1:∞);
* один рейс може мати декілька класів зручності (1:∞);
* знижка може використовуватися багатьма пасажирами (1:∞);
* пасажир може використовувати лише одну знижку (∞:1).

В системі «Бронювання авіаквитків» для точної ідентифікації рейсу

використовується номер рейсу. Він є унікальним ідентифікатором рейсу. Для точного визначення польоту унікальним ідентифікатором служить номер рейсу, на якому базується політ, та дата і час відправлення. Для точної ідентифікації людини використовується її паспортні дані. Окреме замовлення можна визначити так: номер рейсу + дата та час відправлення + клас зручності перельоту + паспортні дані пасажира.

* 1. Опис існуючого документообігу предметної області

Інформаційна система «Бронювання квитків» формує перепустку для проходження реєстрації за деякий час до вильоту. Ця перепустка включає в себе:

* персональні дані про пасажира;
* дата та час вильоту;
* рівень зручності перельоту (економ клас, бізнес клас, тощо).

Як у більшості аеропортах світу пасажир достеменно не знає своє посадкове місце у салоні літака. Воно йому стає відомо при проходженні реєстрації. Тому до перепустки для проходження реєстрації не включається номер місця.

Керівники аеропорту можуть переглядати результат роботи касирів. Для цього інформаційна система складає за запитом адміністратора звіт щодо продуктивності роботи касирів (мається на увазі можливість перегляду кількості замовлень, яку здійснив кожен касир).

Планувальники рейсів отримують статистичний звіт перевезень за деякий період. Цей звіт можна передавати безпосередньо головним планувальникам рейсів – авіакомпаніям. А вони в свою чергу у майбутньому будуть формувати тільки ті рейси, на які є попит. Цей звіт включає в себе наступну інформацію:

* номер рейсу;
* розглядуваний період;
* кількість перевезених пасажирів.
  1. Лінгвістичні відносини

Пасажир або клієнт – людина, яка бажає здійснити переліт, попередньо забронювавши місце в бажаному класі зручності.

Клас зручності або рівень зручності – ступінь якості обслуговування пасажирів (бізнес клас, економ клас, перший клас, тощо). Від нього залежить вартість авіаквитка.

Рейс – переміщення транспортного засобу (літак) з точки відправлення до точки прибуття. Характеризується періодичністю або розкладом та тривалістю перельоту.

Переліт або політ – «окремий випадок» рейсу. Має також характеристику переміщення. А ще характеризується конкретною датою та часом відправлення.

Планувальник рейсів або менеджер – співробітник аеропорту, який коректує розклад.

Касир – співробітник аеропорту, який здійснює бронювання за побажаннями клієнтів.

Замовлення – бронювання клієнтом квитка. Як правило, включає дату та час вильоту, клас зручності перельоту та персональні дані про пасажира.

2 ПОСТАНОВКА ЗАДАЧІ

Програмна система «Бронювання авіаквитків» створюється задля зменшення зусиль касирів та менеджерів аеропорту у повсякденному житті. Ця система виконує левову частку роботи замість їх, завдяки котрій робітники можуть працювати більш ефективніше, ніж раніше.

Дана інформаційна система повинна включати наступні функції:

* авторизація при запуску для безпечної та надійної роботи;
* можливість додання, редагування, видалення рейсів і польотів;
* можливість спрощення роботи планувальникам рейсів за рахунок програмної системи (задача автоматизації);
* можливість бронювання квитків;
* можливість фільтрації польотів згідно з потребами клієнтів по таким критеріям: маршрут, дата та час відправлення, наявність бажаного класу зручності;
* можливість для швидкого пошуку детальної інформації по введеному унікальному номеру рейсу;
* можливість влаштовування на роботу до аеропорту людей;
* можливість надання новим співробітникам необхідних згідно їхній посаді прав доступу до системі;
* можливість перегляду різнопланових статистичних відомостей у вигляді візуальних діаграм (кількість здійснених перельотів за попередні роки, пасажиро обіг за минулі роки, популярність авіакомпаній, продуктивність роботи касирів тощо);
* можливість формування необхідних звітів (рейсова та польотна відомості) для менеджерів або керівників аеропорту та талонів для проходження реєстрації для пасажирів;
* можливість неодноразової зміни паролю користувачів системи;
* можливість надання знижок клієнтам.

3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

3.1 UML – моделювання

На основі концептуального моделювання, проведеного у першому розділі можна виділити таких основних акторів системи: клієнт, касир, адміністратор та планувальник рейсів. Касир здійснює бронювання квитків, а також має доступ до перегляду інформації щодо рейсів. Планувальник коректує рейси згідно з наданими даними про них авіакомпаніями. Адміністратор приймає або звільняє робітників. Клієнт бронює місце на бажаний рейс. На рисунку 3.1 приведена діаграма USE CASE, яка досить гарно ілюструє функції і варіанти взаємодії акторів із системою.

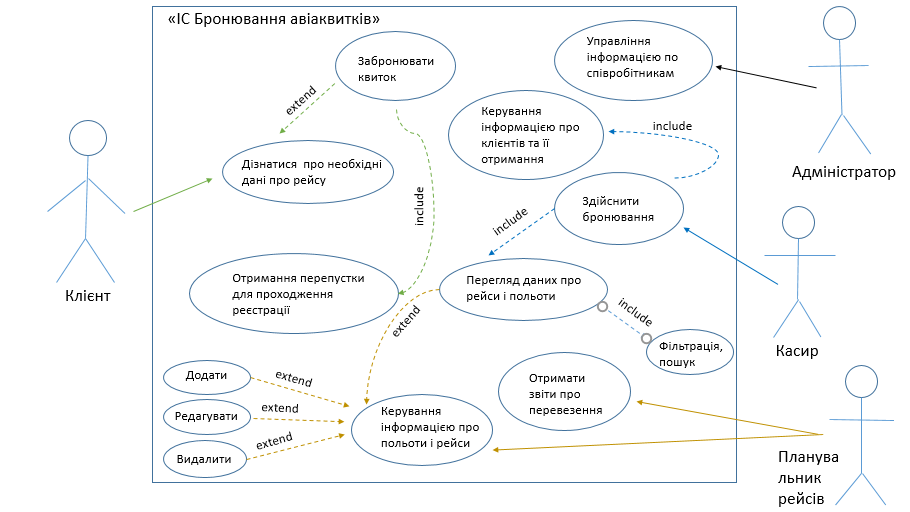


Рисунок 3.1 – Діаграма варіантів use case

системи «Бронювання авіаквитків»

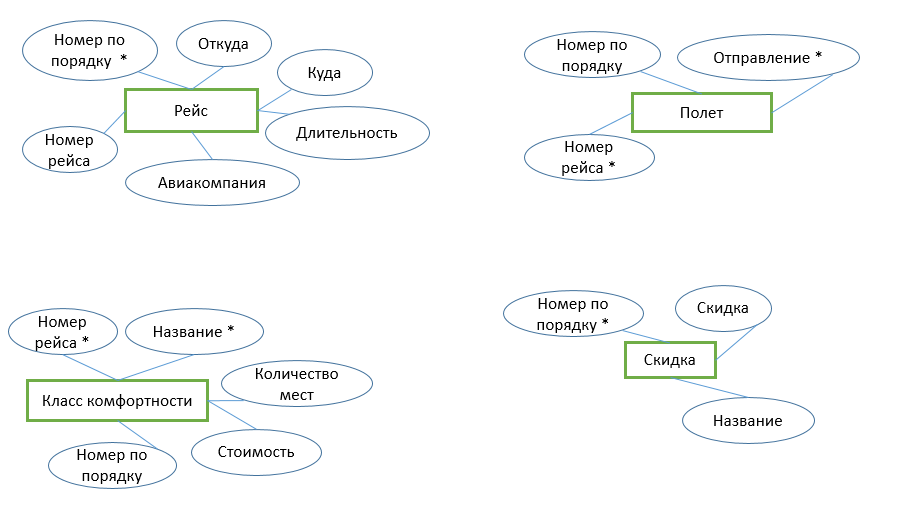
3.2 Побудова ER-діаграми

Виходячи із поставленої задачі, треба створити структуру бази даних, яка б дозволила з легкістю керувати життєдіяльністю аеропорту. Основуючись на тих об’єктах та їхніх атрибутах, які були представлені у першому розділі, а також на проведеній нормалізації бази даних, можна візуально зобразити останню. База даних складається із сімох сутностей, які між собою мають переважно 1:М зв’язок.

У базі даних присутні наступні сутності:

* рейс (унікальний ідентифікатор, номер рейсу, звідки, куди, авіакомпанія, тривалість польоту);
* політ (унікальний ідентифікатор, номер рейсу, дата та час відправлення);
* клас зручності (унікальний ідентифікатор, назва, кількість місць, номер рейсу, вартість);
* пасажир (унікальний ідентифікатор, ім’я, прізвище, по батькові, дата народження, національність, номер паспорту, знижка);
* знижка (унікальний ідентифікатор, назва, знижка);
* робітник (унікальний ідентифікатор, ім’я, прізвище, по батькові, дата народження, національність, номер паспорту, ІПН, посада, логін для входу до системи);
* замовлення (унікальний ідентифікатор, номер пасажира, номер польоту, номер класу зручності, номер касира, дата та час бронювання).

ER-діаграму бази даних можна побачити на рисунку 3.2. Вона побудована за нотацією Чена.



1

1

М

1

М

1

1



М

1

М

1

М

М

М

Рисунок 3.2 – ER-діаграма структури бази даних

3.3 Побудова схеми реляційної бази даних у 3НФ

Нормалізація – це розділення таблиці на дві або більш, таких, що мають ліпші властивості під час включення, зміни та видалення даних. Кінцева мета нормалізації – отримання такого проекту бази даних, в якому кожен факт з’являється лише в одному місті, тобто виключена надмірність інформації.

В схемі БД наявні наступні функціональні залежності (див. рис. 3.3):

1. залежності від первинного ключа: { id-пасажира, id-польоту } => (id-класу, тип класу, кількість місць, вартість квитка, id-рейсу, звідки, куди, тривалість, авіакомпанія, номер рейсу, час та дата відправлення, ім’я, прізвище, по батькові, номер паспорту, дата народження, громадянство, id-знижки, тип знижки, знижка);
2. залежності від id-польоту: { id-польоту} => (id-рейсу, час та дата відправлення, номер рейсу, звідки, куди, тривалість, авіакомпанія);
3. залежності від id-пасажира: {id-пасажира} => (ім’я, прізвище, по батькові, номер паспорту, дата народження, громадянство, id-знижки, тип знижки, знижка);
4. залежності від id-класу: { id-класу } => (тип класу, кількість місць, вартість квитка, id-рейсу, номер рейсу, звідки, куди, тривалість, авіакомпанія);
5. залежності від id-рейсу: { id-рейсу } => (номер рейсу, звідки, куди, авіакомпанія, тривалість польоту);
6. залежності від id-знижки: { id-знижки } => (тип знижки, знижка).

ID – Класу зручн.

Тип класу

Кількість місць

Вартість

ID - Рейсу

Номер

Звідки

Куди

Авіакомпанія

Тривалість свісвіспольпопольоту

#

ID - Польоту

Відправлення

#

ID-пасажира

Ім’я

Прізвище

По батькові

Дата народж.

Громадянство

Номер паспорту

ID - Знижки

Тип знижки

Знижка

Рисунок 3.3 – Відношення Т, яке влючає всі атрибути схеми БД

Відношення Т знаходиться у першій нормальній формі (1NF), тому що виконується правило атомарності проте не знаходиться у другій нормальній формі (2NF), бо не кожен неключовий атрибут повністю функціонально залежить від первинного ключа відношення (наприклад, атрибут «номер паспорту» функціонально залежить не тільки від первинного ключа, а й від атрибуту «id-пасажира»). Для того, щоб привести відношення Т до другої нормальної форми необхідно виділити ще одне відношення «Пасажир», побудоване з наступних атрибутів: id-пасажира, ім’я, прізвище, по батькові, дата народження, номер паспорту, громадянство. У відношенні «Пасажир» первинним ключом є id-пасажира. На рисунку 3.4 зображено 2НФ.

#

ID-пасажира

#

ID - Рейсу

ID-пасажира

Номер

Ім’я

Звідки

Прізвище

Куди

По батькові

Авіакомпанія

Дата народж.

Тривалість свісвіспольпопольоту

Громадянство

#

ID - Польоту

Номер паспорту

Відправлення

ID - Знижки

ID – Класу зручн.

Тип знижки

Тип класу

Знижка

Кількість місць

Вартість

#

ID – Класу зручн.

Рисунок 3.4 – Друга нормальна форма (2NF)

Для того, щоб отримати схему бази даних у третій нормальній формі необхідно позбутися від транзитивних залежностей між неключовими атрибутами, тобто слід винести в окремі відношення наступні атрибути: id-класу, тип класу, вартість квитку, кількість посадкових місць – у відношення «клас зручності»; id-польотів, відправлення – у відношення «політ»; id-знижки, тип знижки, знижка – у відношення «знижка».

Після нормалізації відношення Т схема бази даних налічує не одну таблицю, а сім. Кожна з яких не має транзитивних залежностей між неключовими атрибутами, тобто знаходяться у третій нормальній формі. На рисунку 3.5 представлено схему бази даних, яка знаходитья у 3НФ.

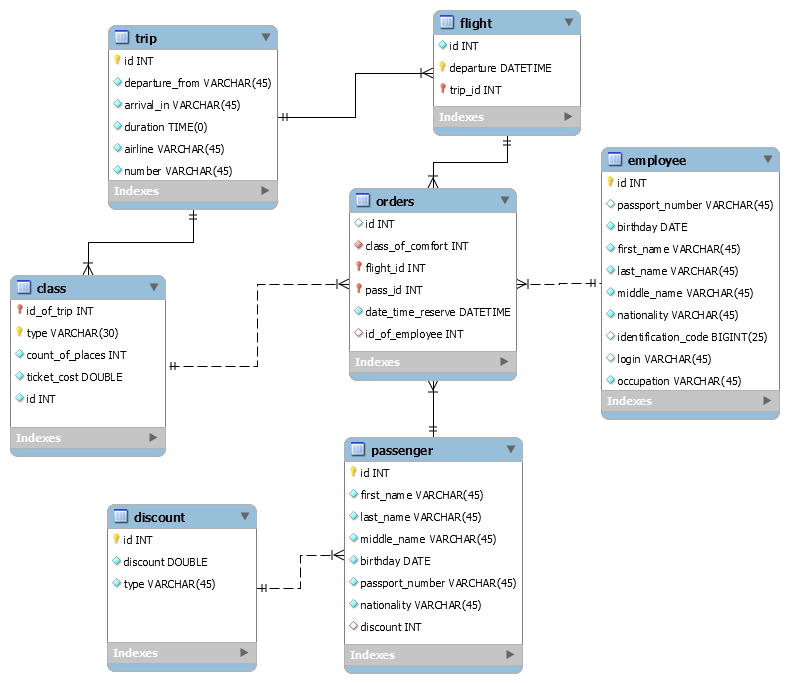


Рисунок 3.5 – Схема бази даних у третій нормальній формі (3NF)

4 ОПИС ПРОГРАМИ

4.1 Загальні відомості

Програмна система «Бронювання авіаквитків» була створена завдяки мові програмування Java. В основу покладений об’єктно-орієнтований підхід, завдяки якому програму дуже легко супроводжувати під час її використання.

У якості інструменту для збереження даних обрано реляційну модель баз даних, яка дозволяє чітко та лаконічно маніпулювати даними. Для керування базою даних використовується СУБД MySQL 5.7, яка є однією з найпопулярніших на теперішній момент.

Згідно з потребами користувачів програма містить всі необхідні розділи: робота з рейсами та польотами, робота з клієнтами (пасажирами). Задля безпеки даних програмна реалізація вищезазначеної системи дозволяє керувати доступом до бази даних. Інформаційною системою можуть користуватися: касири, менеджери або планувальники рейсів і польотів, та керівництво аеропорту. Завдяки ергономічному та мінімалістичному інтерфейсу, користувачі мають можливість з легкістю та комфортністю працювати з даними. Для роботи програми необхідна наявність на комп’ютері СУБД MySQL. Також необхідно 25 мб вільного дискового простору. Розширення монітору має бути більшим за 1024х768.

4.2 Виклик та завантаження

При запуску програми надається можливість обрати свою посаду та виконати вхід до системи. Якщо були введені недостовірні дані, програма повідомить користувача про це. Логін надається касирові при влаштуванні на роботу до аеропорту. Пароль він задає сам. Аналогічно для планувальника рейсів.

Зв’язок між базою даних та програмним кодом здійснюється завдяки JDBC (Java Database Connectivity). Для підключення необхідно використовувати спеціальний драйвер від JDBC для конкретної СУБД, в даному випадку MySQL. Фрагмент коду, який підключає БД до програми представлений у лістингу 4.1. Якщо були введені правильні дані, то виконається з’єднання із базою даних та відкривається вікно для роботи нею. На рисунку 4.1 представлено вікно програми для авторизації.

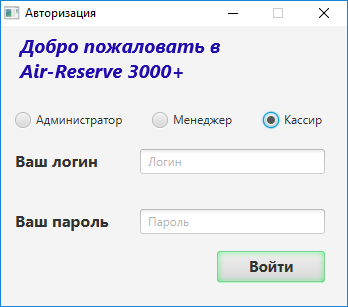


Рисунок 4.1 – Вікно для авторизації

|  |
| --- |
| **private** Connection **connection** = **null**; **private** Statement **statement** = **null**; **private** ResultSet **resultSet** = **null**;  **public** ConnectionToDB(String login, String password) **throws** Exception {  String url = **"jdbc:mysql://localhost:3306/airport2?useSSL=false"**;  Class.*forName*(**"com.mysql.jdbc.Driver"**).newInstance();  **connection** = DriverManager.*getConnection*(url, login, password);  **statement** = **connection**.createStatement(); } |

Лістинг 4.1 – Підключення до бази даних

4.3 Призначення та логічна структура програми

Найголовніше призначення програми – це поліпшення праці робітникам аеропорту. Цей додаток зберігає дуже багато часу при бронюванні квитків. Відтепер для бронювання необхідно усього одна хвилина.

Програму можна розділити на три основні модулі:

* модуль для касира (бронювання авіаквитків, формування реєстраційного талону);
* модуль для менеджера або планувальника рейсів (додавання, редагування, видалення рейсів/польотів; перегляд статистичної інформації; друк наступних документів: рейсова відомість, польотна відомість);
* модуль для адміністратора (влаштовування, звільнення робітників; редагування інформації знижок).

Кожен із вищезазначених модулів реалізовує певний функціонал інформаційної системи. Модуль касира реалізовує можливість для пошуку та фільтрації рейсів і дозволяє бронювати квитки для пасажирів. Модуль для менеджера допомагає керувати рейсами та польотами і переглядати статистичні дані щодо них. Модуль для адміністратора дозволяє влаштовувати людей на роботу до аеропорту та керувати знижками, які надаються клієнтам.

Для того щоб відкрити певний модуль потрібно виконати вхід до системи з відповідного акаунту. Наприклад, щоб увійти до модуля касира необхідно ввести логін і пароль будь-якого касира, наявного в системі.

Програмою можуть користуватися багато користувачів, а отже потрібно було створити систему для їх авторизації, що і було зроблено. Перш за все це було необхідно для безпеки та цілісності даних. При влаштовуванні людини на роботу вона отримує доступ до системи (логін та пароль). Паролі хешуються вбудованими в СУБД функціями. Хешування паролів необхідно перш за все для надійності та конфіденційності даних. Таким чином у програмі реалізовано захист від несанкціонованого доступу.

4.4 Опис фізичної моделі бази даних

У якості СУБД було обрано MySQL 5.7. Цей продукт є одним з найпопулярніших на теперішній момент. Саме через це його активно підтримують і видають нові оновлення для нього. Безумовним плюсом є те, що MySQL є безкоштовним продуктом (open source database).

Із урахуванням усіх вимог до функціональної складової інформаційної системи «Бронювання авіаквитків», була створена база даних, яка складається з сімох таблиць. Нижче наведено подробиці щодо окремої створеної таблиці  (табл. 4.1 – 4.7).

Таблиця 4.1 – Таблиця рейсів (trip):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id | Integer | \* | Код рейсу |
| 2 | departure\_from | Varchar(45) |  | Звідки відправлення |
| 3 | arrival\_in | Varchar(45) |  | Куди прямує |
| 4 | duration | Datetime |  | Тривалість польоту |
| 5 | airline | Varchar(45) |  | Авіакомпанія-перевізник |
| 6 | number | Varchar(45) |  | Номер рейсу |

Таблиця 4.2 – Таблиця класів зручності (class):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id\_of\_trip | Integer | \* | Код рейсу |
| 2 | type | Varchar (45) | \* | Тип класу |
| 3 | сount\_of\_places | Integer |  | Кількість місць |
| 4 | ticket\_cost | Double |  | Вартість квитка |
| 5 | id | Integer |  | Код класу зручності |

Таблиця 4.3 – Таблиця польотів (flight):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id | Integer |  | Код польоту |
| 2 | departure | Varchar(45) | \* | Дата та час відправлення |
| 3 | trip\_id | Integer | \* | Код рейсу |

Таблиця 4.4 – Таблиця замовлень (orders):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id | Integer |  | Код замовлення |
| 2 | class\_of\_comfort | Integer |  | Код класу зручності |
| 3 | flight\_id | Integer | \* | Код польоту |
| 4 | pass\_id | Integer | \* | Код пасажира |
| 5 | id\_of\_employee | Integer |  | Код касира |
| 6 | date-time-reserve | Datetime |  | Дата та час бронювання |

Таблиця 4.5 – Таблиця пасажирів (passenger):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id | Integer | \* | Код пасажира |
| 2 | first\_name | Varchar (45) |  | Ім’я |
| 3 | last\_name | Varchar (45) |  | Прізвище |
| 4 | middle\_name | Varchar (45) |  | По батькові |
| 5 | nationality | Varchar (45) |  | Громадянство |
| 6 | passport\_number | Varchar (45) |  | Номер паспорту |
| 7 | birthday | Date |  | Дата народження |
| 8 | discount | Integer |  | Код знижки |

Таблиця 4.6 – Таблиця робітників (employee):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id | Integer | \* | Код пасажира |
| 2 | first\_name | Varchar (45) |  | Ім’я |
| 3 | last\_name | Varchar (45) |  | Прізвище |
| 4 | middle\_name | Varchar (45) |  | По батькові |
| 5 | nationality | Varchar (45) |  | Громадянство |
| 6 | passport\_number | Varchar (45) |  | Номер паспорту |
| 7 | birthday | Date |  | Дата народження |
| 8 | identification\_code | Long |  | Податковий код |
| 9 | login | Varchar (45) |  | Логін для входу |
| 10 | occupation | Varchar (45) |  | Посада службовця |

Таблиця 4.7 – Таблиця знижок (discount):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип даних | Ключ | Опис |
| 1 | id | Integer | \* | Код знижки |
| 2 | type | Varchar (45) |  | Тип знижки |
| 3 | discount | Double |  | Значення знижки |

4.5 Опис програмної реалізації

Програмна система «Бронювання авіаквитків» складається із багатьох екранних форм. Для логічного їх групування були створені модулі, які розділяють функції цього додатку. Касир взаємодіє із програмою за допомогою вікна «панель касира». На цій панелі касир може виконувати свої звичайні обов’язки. За допомогою зручного інтерфейсу він має можливість фільтрувати польоти за такими критеріями: маршрут, дата відправлення та клас зручності перельоту. Панель для персональних даних пасажира розміщена зліва, бо саме вона є однією з найбільш вживаних на формі. Якщо клієнт одного разу вже користувався послугами нашого аеропорту, то не потрібно повторно вводити всі його персональні дані. Достатньо тільки ввести номер паспорту. Завдяки випадаючим списками стає можливим найшвидша взаємодія із системою. При бажанні клієнта касир може відмінити бронювання. На рисунку 4.2 представлено вікно для взаємодії касира з програмою.

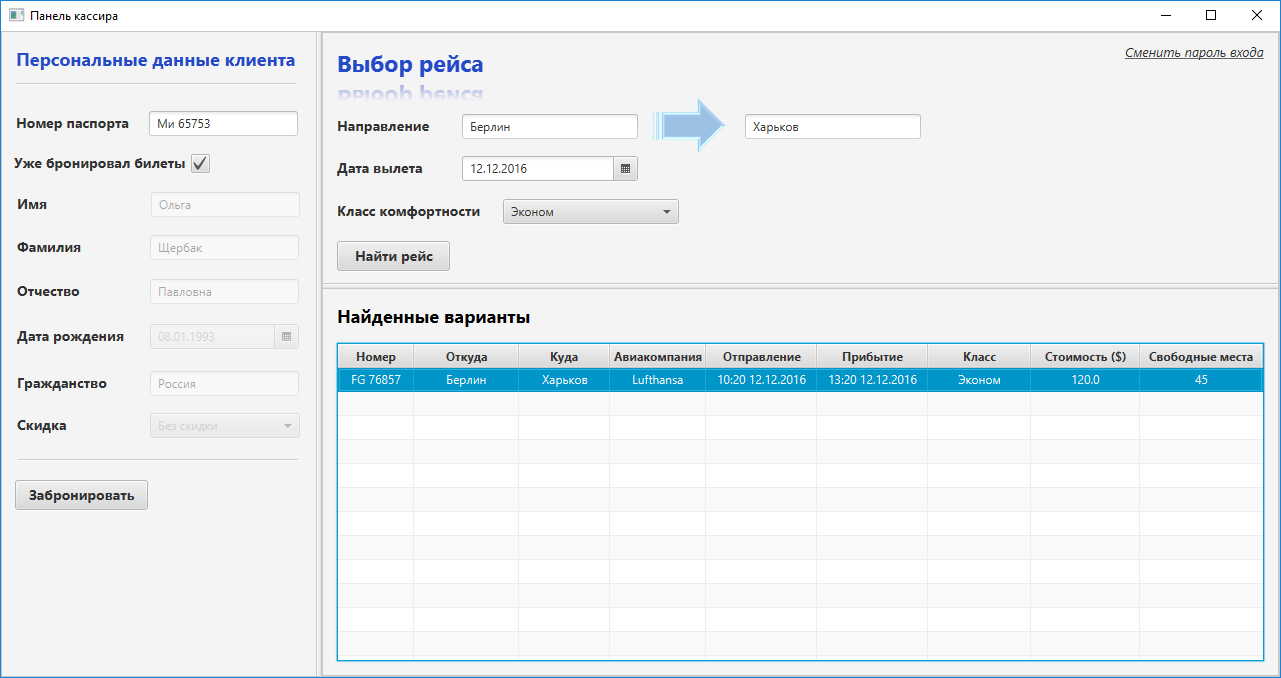


Рисунок 4.2 – Панель касира

SQL – запит, який представлено у лістингу 4.2, робить вибірку із бази даних, основуючись на тих критеріях, які були введені касиром.

|  |
| --- |
| SQL = **"SELECT trip.number, trip.departure-from, trip.arrival-in, "** +  **"flight.departure, trip.duration, "** +  **" class.ticket\_cost AS cost, trip.airline,"** +  **" class.type, class.count-of-places -"** +  **" (SELECT IFNULL(COUNT(pass-id),0) FROM orders"** +  **" WHERE class-of-comfort=class.id AND flight-id = flight.id)"** +  **" FROM flight JOIN trip JOIN class"** +  **" ON flight.trip-id = trip.id "** +  **" AND class.id-of-trip = trip.id "** +  **" AND trip.departure-from LIKE '"** + departureFrom + **"%'"** +  **" AND trip.arrival-in LIKE '"** + arrivalIn + **"%'"** +  **" AND flight.departure > NOW()"** +  **" AND DATE(flight.departure)= '"** + flightDate + **"'"** +  **" AND class.type='"** + comfortClass + **"'"** +  **" ORDER BY flight.departure ASC, class.type ASC LIMIT 100"**; |

Лістинг 4.2 – SQL-запит, який шукає польоти

Для роботи менеджера із програмним додатком використовується вікно менеджера. У ньому є можливість додавати польоти та рейси до бази даних, редагувати і видаляти їх. Також є можливість для перегляду статистичних даних у вигляді діаграм. На рисунку 4.3 представлено вікно менеджера. Менеджери виконують найголовніші функції аеропорту, адже без їхньої праці база даних була би порожньою. Для максимально комфортної їх роботи використовуються випадаючі списки із найймовірнішими значеннями. Для заощадження часу та зусиль менеджерів програмна система дозволяє один раз ввести інформацію про рейс і вказати часові границі його виконання. Система сама потурбується за розподілення польотів створюваного рейсу. У лістингу 4.3 наведено SQL-запит для відображення польотів.

|  |
| --- |
| SQL = **"SELECT number, departure-from, arrival-in, "** +  **"airline, duration, departure "** +  **"FROM trip JOIN flight "** +  **"ON flight.trip-id = trip.id "** +  **"WHERE departure > NOW() "** +  **"ORDER BY departure ASC, number ASC LIMIT 200"**; |

Лістинг 4.3 – SQL-запит для відображення польотів

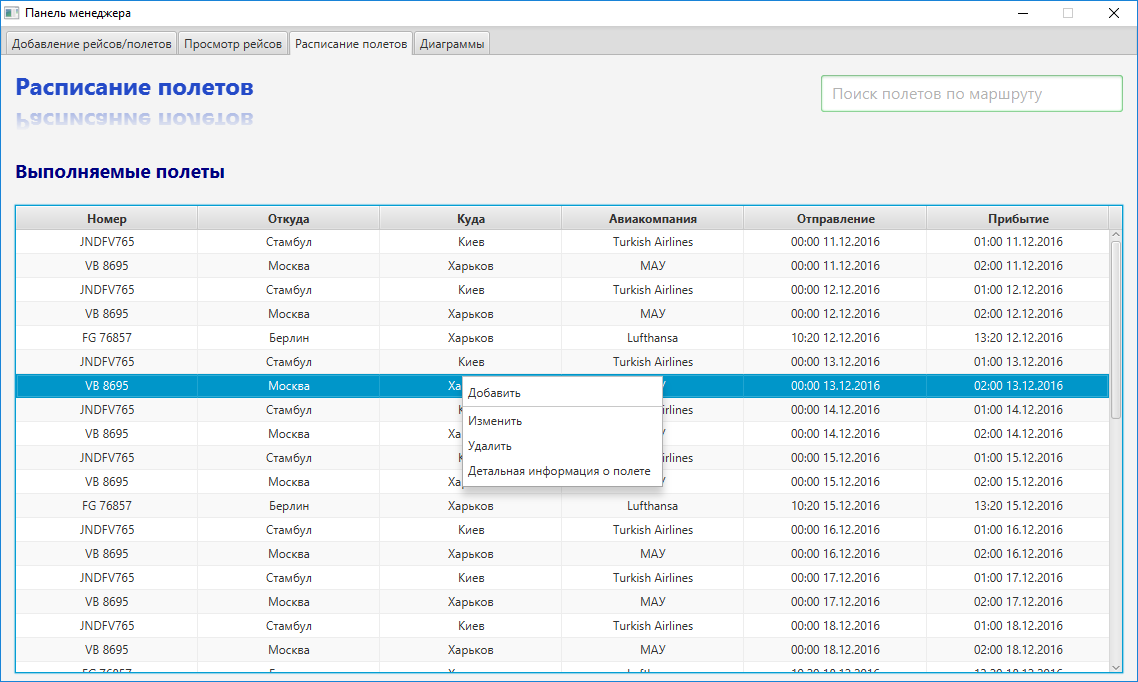


Рисунок 4.3 – Панель менеджера

Для отримання статистичної інформації слід перейти на закладку «Діаграми». У лістингу 4.4 представлено програмний код, який виконується при формуванні статистики «Найпопулярніші авіакомпанії за минулі 5 років».

На рисунку 4.4 наведений результат цієї статистики.

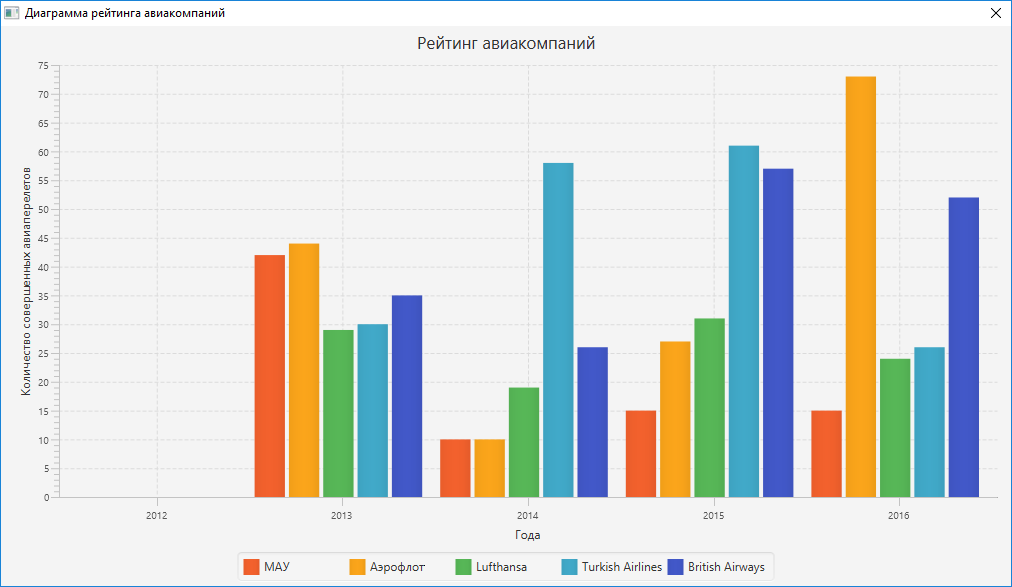


Рисунок 4.4 – Діаграма популярності авіакомпаній за останні 5 років

|  |
| --- |
| ArrayList<Integer> years = **new** ArrayList<>(); LocalDate begin = LocalDate.*now*();  **for** (**int** i = 0; i < 5; i++) {  years.add(begin.getYear());  begin = begin.minusYears(1); } **final** CategoryAxis xAxis = **new** CategoryAxis(); **final** NumberAxis yAxis = **new** NumberAxis(); xAxis.setLabel(**"Года"**); yAxis.setLabel(**"Количество обслуженых пассажиров"**); **final** BarChart<String ,Number> barChart = **new** BarChart<String, Number>(xAxis, yAxis); barChart.setTitle(**"Диаграмма популярности авиакомпаний"**);  Scene scene = **new** Scene(barChart, 900, 550);  HashMap<String, Integer> hashMap = **new** HashMap<>();  String SQL = **"SELECT IFNULL(COUNT(orders.id), 0) AS pass\_count, trip.airline "** + **"FROM trip JOIN flight JOIN orders "** + **"ON trip.id = flight.trip\_id "** + **"AND orders.flight\_id = flight.id "** + **"WHERE YEAR(departure) BETWEEN (YEAR(NOW()) - 5) AND YEAR(NOW())"** + **"GROUP BY trip.airline "** + **"ORDER BY IFNULL(COUNT(pass\_id), 0) DESC LIMIT 5"**;  ResultSet resultSet = **connectionToDB**.retriveData(SQL); **while** (resultSet.next()) {  hashMap.put(resultSet.getString(**"trip.airline"**),  resultSet.getInt(**"pass\_count"**)); } **for** (String name : hashMap.keySet()) {  XYChart.Series<String, Number> ser = **new** XYChart.Series<>();  ser.setName(name);   **for** (**int** i = years.size() - 1; i >= 0; i--)  {  String sql = **"SELECT IFNULL(COUNT(pass\_id),0) AS pass\_count "** + **"FROM trip JOIN flight JOIN orders "** + **"ON trip.id = flight.trip\_id "** + **"AND orders.flight\_id = flight.id "** + **"WHERE trip.airline = '"** + name + **"' "** + **"AND YEAR(departure) = "** + years.get(i) + **" "** + **"AND departure < NOW()"**;  ResultSet resSet = **connectionToDB**.retriveData(sql);  **while** (resSet.next())  {  ser.getData().add(**new** XYChart.Data<>(years.get(i) + **""**,  resSet.getLong(**"pass\_count"**)));  }  }  barChart.getData().add(ser); } |

Лістинг 4.4 – Програмний код для отримання статистики про найпопулярніші авіакомпанії за останні 5 років

Для взаємодії адміністраторів із системою використовується панель адміністратора. Вона дозволяє з легкістю влаштовувати та звільняти персонал, а також додавати, редагувати, видаляти знижки для клієнтів. Також є можливість перегляду статистики щодо продуктивності роботи касирів з метою найточнішої оцінки їх праці. У лістингу 4.5 наведено SQL – запит, який виконується при влаштуванні касира на роботу. На рисунку 4.5 приведена панель адміністратора.

|  |
| --- |
| SQL = **"FLUSH PRIVILEGES;"**; **connectionToDB**.executeStatement(SQL);  SQL = **"CREATE USER paymaster"** + **login**.getText() + **"@localhost IDENTIFIED BY '123456';"**; **connectionToDB**.executeStatement(SQL);  SQL = **" GRANT INSERT,UPDATE,DELETE ON airport2.passenger "** +  **" TO 'paymaster"** + **login**.getText() +**"'@'localhost';"**; **connectionToDB**.executeStatement(SQL);  SQL = **" GRANT INSERT,UPDATE,DELETE ON airport2.orders "** +  **" TO 'paymaster"** + **login**.getText() +**"'@'localhost';"**; **connectionToDB**.executeStatement(SQL);  SQL = **"GRANT SELECT ON airport2.\* TO 'paymaster"** + **login**.getText() +**"'@'localhost';"**; **connectionToDB**.executeStatement(SQL); |

Лістинг 4.5 – SQL-запит додання нового касира

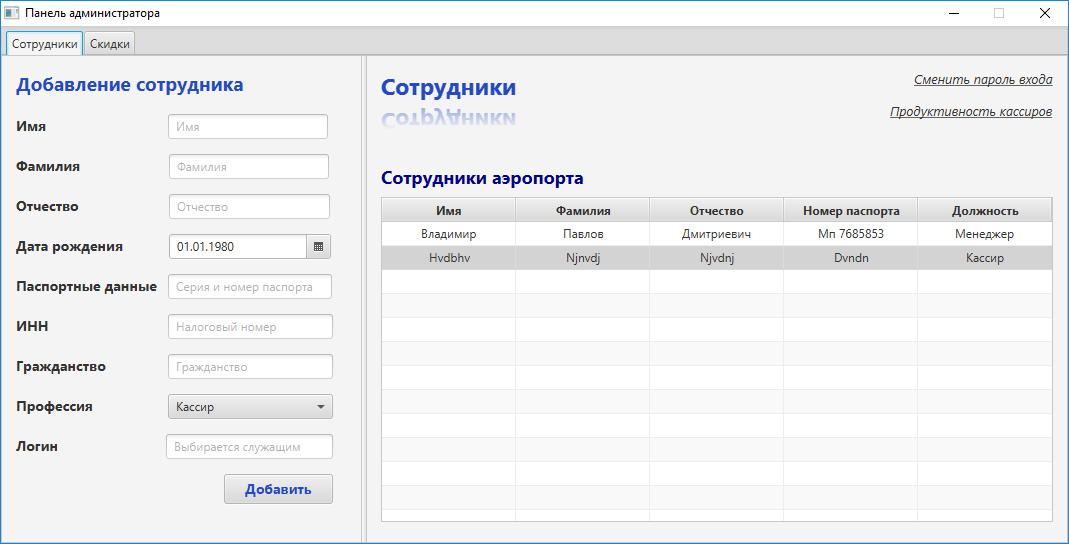


Рисунок 4.5 – Панель адміністратора

Для перегляду статистичних даних щодо продуктивності роботи касирів необхідно натиснути на посилання «Продуктивність касирів». Після натискання відкриється нове вікно із статистикою. У лістингу 4.6 представлено SQL-запит, на базі якого формується ця статистика.

|  |
| --- |
| SQL = **"SELECT IFNULL(COUNT(orders.id), 0), passport-number "** + **"FROM employee LEFT OUTER JOIN orders "** + **"ON orders.id-of-employee = employee.id "** + **"WHERE occupation = 'Кассир' "** + **"GROUP BY passport-number "** + **"ORDER BY IFNULL(COUNT(orders.id), 0)"**; |

Лістинг 4.6 – SQL-запит, формуючий статистику продуктивності касирів

4.6 Опис задачі автоматизації

Жоден аеропорт світу не здатен повноцінно функціонувати без програмної системи. Комп’ютерна програма виконує майже моментально ті дії, які б людина виконувала набагато довше. Інформаційна система автоматизує багато процесів, які потребують неабияких зусиль людини.

Програмний додаток «Бронювання авіаквитків» не є виключенням. Він також має на меті автоматизацію процесів керування розкладом та бронювання авіаквитків. Завдяки реалізованій задачі автоматизації планувальнику рейсів не потрібно виконувати одну ж ту саму дію, щоб додати польоти окремого рейсу. Для того, щоб створити новий рейс (наприклад Харків-Київ) необхідно вказати часові рамки його виконання (наприклад 1.2.2017 – 1.5.2017) і період між польотами (наприклад кожну добу). Додаток у мить додасть до бази даних необхідну інформацію, тим самим значно заощадивши час й зусилля менеджера. Однак для більш точнішого коректування розкладу у планувальника рейсів є можливість видаляти окремі польоти (наприклад якщо політ прийшовся на 1-ше січня) і вставляти окремі польоти, вибравши необхідний рейс. Задача автоматизації полягає у тому, щоб підказувати планувальникам рейсів бажаний період між польотами. Для цього програмна система «Бронювання авіаквитків» переглядає минулорічні рейси з таким самим маршрутом і такими ж часовими рамками, а далі підраховує наскільки пустими були салони літаків. Тобто значущість задачі автоматизації полягає не тільки в спрощенні роботи менеджерам, а й в оцінці рентабельності польотів окремих рейсів задля максимально ефективного використання ресурсів авіакомпаній. Результат задачі автоматизації будується на оцінці середніх значень по всім підходящим польотам, незважаючи на авіакомпанії.

Сам процес задачі автоматизації можна поділити на декілька пунктів:

* знайти у базі даних номери рейсів, які мають той самий маршрут і ті ж самі часові рамки створюваного рейсу;
* підрахувати загальну кількість посадкових місць по кожному знайденому номеру рейсу;
* підрахувати кількість заброньованих пасажирами місць по кожному знайденому номеру рейсу;
* підрахувати кількість польотів по кожному знайденому номеру рейсу;
* знайти середні значення і дати оцінку рентабельності рейсу.

Програмний код задачі автоматизації наведено у лістингу 4.7. На рисунку 4.6 приведений результат автоматизації.

|  |
| --- |
| DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*(**"yyyy-MM-dd HH:mm"**);  LocalDateTime begin = LocalDateTime.*parse*(dateTimeBegin, formatter).minusYears(1); LocalDateTime end = LocalDateTime.*parse*(dateTimeEnd, formatter).minusYears(1);  String SQL = **"SELECT DISTINCT number FROM flight JOIN trip "** + **"ON flight.trip\_id = trip.id "** + **"WHERE flight.departure >= '"** + begin.format(formatter) + **"' "** + **"AND flight.departure <= '"** + end.format(formatter) + **"' "** + **"AND trip.departure\_from = '"** + depFrom + **"' "** + **"AND trip.arrival\_in = '"** + arrIn + **"'"**;   ResultSet resultSet = **connectionToDB**.retriveData(SQL);  LinkedList<String> flightNumbers = **new** LinkedList<>();  **while** (resultSet.next())  {  flightNumbers.add(resultSet.getString(**"number"**));  }   ArrayList<Integer> passengerCount = **new** ArrayList<>();  ArrayList<Integer> placeCount = **new** ArrayList<>();  ArrayList<Integer> flightCount = **new** ArrayList<>();  ArrayList<Long> secondsCount = **new** ArrayList<>();   **for** (String number : flightNumbers)  {  SQL = **"SELECT SUM(count\_of\_places) AS all\_places "** + **"FROM trip JOIN class "** + **"ON trip.id = class.id\_of\_trip "** + **"WHERE number = '"** + number + **"' "**;  resultSet = **connectionToDB**.retriveData(SQL);  resultSet.first();  **int** countOfPlaces = resultSet.getInt(**"all\_places"**);   SQL = **"SELECT IFNULL(COUNT(pass\_id), 0) AS pass\_count "** + **"FROM orders JOIN flight JOIN trip "** + **"ON trip.id = flight.trip\_id "** + **"AND orders.flight\_id = flight.id "** + **"WHERE number = '"** + number + **"'"**;  resultSet = **connectionToDB**.retriveData(SQL);  resultSet.first();  **int** countOfPassenger = resultSet.getInt(**"pass\_count"**);   SQL = **"SELECT IFNULL(COUNT(flight.id), 0) AS flight\_count "** + **"FROM flight JOIN trip "** + **"ON trip.id = flight.trip\_id "** + **"WHERE number = '"** + number + **"'"**;  resultSet = **connectionToDB**.retriveData(SQL);  resultSet.first();  **int** countOfFlights = resultSet.getInt(**"flight\_count"**);   passengerCount.add(countOfPassenger);  placeCount.add(countOfPlaces);  flightCount.add(countOfFlights);    SQL = **"SELECT TIMESTAMPDIFF(MINUTE, '"** + begin.format(formatter) +   **"', '"** + end.format(formatter) + **"')"**;  resultSet = **connectionToDB**.retriveData(SQL);  resultSet.first();  secondsCount.add(resultSet.getLong(1) / countOfFlights);  }   **double** avgCountPassengers = 0;  **double** avgCountPlaces = 0;  **double** avgCountFlights = 0;  **double** avgCountMinutes = 0;   **for** (**int** i = 0; i < passengerCount.size(); i++)  {  avgCountPassengers += passengerCount.get(i);  avgCountFlights += flightCount.get(i);  avgCountPlaces += placeCount.get(i);  avgCountMinutes += secondsCount.get(i);  }   avgCountPassengers = avgCountPassengers / passengerCount.size();  avgCountFlights = avgCountFlights / flightCount.size();  avgCountPlaces = avgCountPlaces / placeCount.size();  avgCountMinutes = avgCountMinutes / secondsCount.size();  **double** result = avgCountPassengers / (avgCountFlights \* avgCountPlaces);  **if** (avgCountMinutes < 60)  {  **period**.getSelectionModel().selectFirst();  **valuePeriod**.setText((**int**) (avgCountMinutes / result) + **""**);  }  **else if** (avgCountMinutes >= 60 && avgCountMinutes < 1440)  {  **period**.getSelectionModel().select(1);  **valuePeriod**.setText((**int**) (avgCountMinutes / (60 \* result)) + **""**);  }  **else if** (avgCountMinutes >= 1440 && avgCountMinutes < 10080)  {  **period**.getSelectionModel().select(2);  **valuePeriod**.setText((**int**) (avgCountMinutes / (60 \* 24 \* result)) + **""**);  }  **else if** (avgCountMinutes >= 10080 && avgCountMinutes < 302400)  {  **period**.getSelectionModel().select(3);  **valuePeriod**.setText((**int**) (avgCountMinutes / (60 \* 24 \* 30 \* result))+**""**);  }  **else** {  **period**.getSelectionModel().select(4);  **valuePeriod**.setText((**int**) (avgCountMinutes / (60 \* 24 \* 30 \* 12 \*  result))+**""**);  } |

Лістинг 4.7 – Програмний код задачі автоматизації

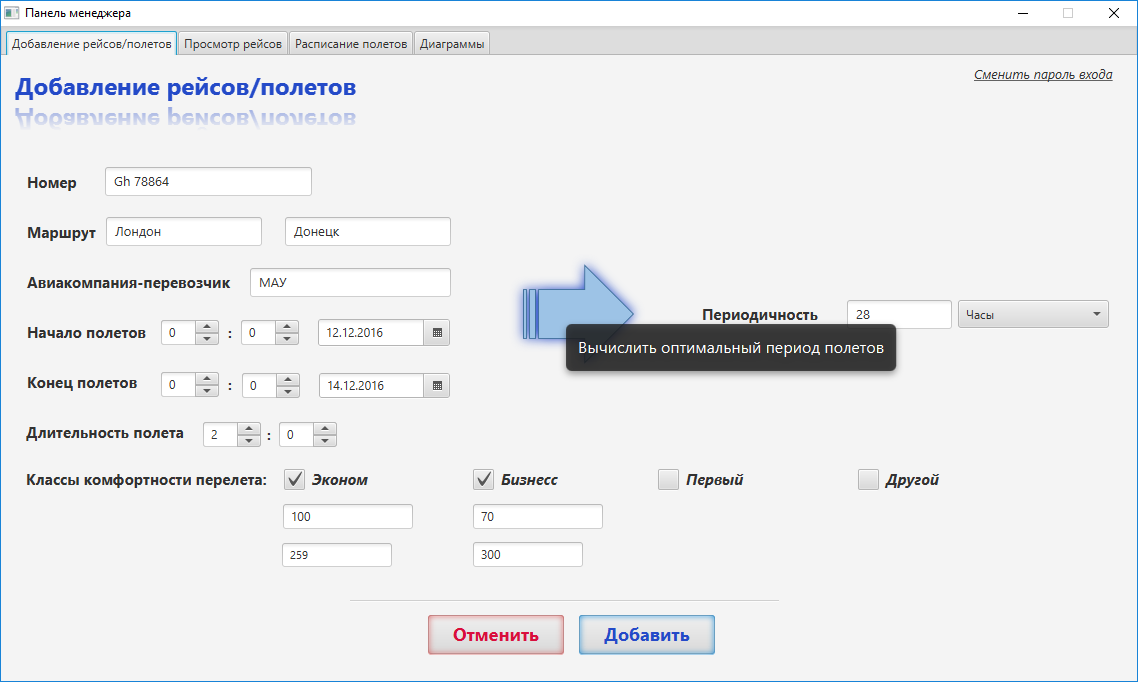


Рисунок 4.6 – Результат автоматизації

ВИСНОВКИ

Під час виконання курсового проектування створювалась інформаційна система «Бронювання авіаквитків». Ця система спрощує роботу службовців в аеропорту і дозволяє керувати багатьма аспектами життєдіяльності аеропорту.

Завдяки зручному та ергономічному інтерфейсу, процес керування рейсами та польотами став неабияк легшим та зрозумілішим. Задля спрощення процесу додання нової інформації щодо рейсів була реалізована його автоматизація. Інформаційна система аналізує минулорічні польоти на ефективність та рентабельність. Це, в свою чергу, заощадить кошти авіакомпаній у майбутньому, бо відтепер літаки не будуть літати з напівпустим салоном. Система запропонує бажаний період між польотами, заснований на минулорічних даних і менеджеру вже не потрібно буде самому відшукувати та аналізувати великі об’єми даних.

Програмний додаток був написаний на мові програмування Java, а це свідчить про те, що ним можна користуватися на будь-якій операційній системі персональних комп’ютерів (Linux, Windows, MacOS тощо).

Під час написання програми були повторені та засвоєні такі технології: об’єктно-орієнтоване програмування, реляційні бази даних і нормалізація відношень в них, технологія доступу до баз даних із Java-коду JDBC (Java Database Connectivity), мова запитів SQL та деякі особливості СУБД MySQL.

В результаті виконання всіх етапів курсового проектування створена програмна реалізація інформаційної системи «Бронювання авіаквитків», яка має досить корисний функціонал, направлений на поліпшення праці робітників аеропорту.

Плани на майбутнє: реалізувати додаток у просторі Інтернет, тобто створити сайт для маніпулювання різними аспектами аеропорту, використовуючи Spring Framework та СУБД Oracle. Додати невелику нейрону мережу для «розумного» пошуку рейсів та польотів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Джеймс Грофф, Пол Вайнберг, Эндрю Оппель. SQL: полное руководство 3-е издание. – М.: Издательский дом «Вильямс», 2015 г. - 959 с.
2. Брюс Эккель. Философия Java 4-е издание. – М.: Издательский дом «Вильямс», 2015 г. – 1170 с.
3. Герберт Шилдт. Java 8 Полное руководство 9-е издание. – М.: Издательский дом «Вильямс», 2015 г. – 1377 с.
4. Ульман Дж.Д. Основы систем баз данных. - М.: Финансы и статистика, 1983 г. - 334 с.
5. Дейт К. Дж. Введение в системы баз данных, 7-е изд. - М.: Изд. Дом «Вильямс», 2001 г. - 846 с.
6. Методичні вказівки до курсової роботи з дисципліни «Бази даних » для студентів усіх форм навчання спеціальності 6.050103 – “Програмна інженерія” / Упоряд.: О.О.Мазурова, М.С.Широкопетлєва, Ю.Ю.Черепанова.– Харків: ХНУРЕ, 2015. – 42 с.