ВІДЗИВ НАУКОВОГО КЕРІВНИКА НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ

Студента (ки) **Пащенко П.В.**

(*прізвище та ініціали)*

представлену на здобуття освітнього ступеня «бакалавр»
зі спеціальності 123 «Комп'ютерна інженерія»

1. Робота виконана на тему Проект інформаційної системи для моніторингу роботи залізничного транспорту

2. Актуальність. Актуальність теми дипломного проекту зумовлюється необхідністю автоматизації роботи менеджерів (касира, оператора і служби довідок) з метою зниження тимчасових і грошових витрат на виконання стандартних рутинних операцій. Практична значущість роботи визначається розробкою реального програмного засобу, що служить для автоматизації роботи менеджерів касира, оператора і служби довідок.

(зазначити, які питання теоретичного і практичного характеру визначають актуальність роботи)

3. Основний зміст роботи викладено на 67 сторінках тексту, містить 3 таблиці та 19 ілюстрацію. Список використаних літературних джерел налічує 22 найменування.

4. Зміст розділів роботи *(зазначити за кожним розділом, які питання досліджені та з якою метою)*

Розділ 1. Аналітичний огляд літературних та інших джерел. У розділі здійснено аналіз аналогічних розробок, проаналізовано типи баз даних, описано тригери та збережені процедури.

Розділ 2. Проектування інформаційної системи для моніторингу роботи залізничного транспорту. У даному дозділі визначено цілі та задачі проектування, обґрунтувано проектні рішення з інформаційного забезпечення, описано процедури.

Розділ 3. Опис програми та тестування. Прелставлено керівництво користувача, розробку алгоритмів, здійснено тестування програмного забезпечення.

5. Висновки обґрунтовані, повністю відповідають поставленим завданням, мають теоретичне та практичне значення. Практичне значення отриманих результатів полягає у створенні системи управління пасажирськими залізничними перевезеннями, що дозволяє усунути недоліки реального процесу продажу залізничних квитків.

(обґрунтованість, теоретичне і практичне значення)

6. Список використаних літературних джерел. Джерела опрацьовані у повному обсязі, відповідають змісту роботи.

(обсяг і повнота опрацювання за змістом роботи)

7. Додатки

(перелік додатків та повнота їх опрацювання за змістом роботи)

8. Таблиці та ілюстрації Ілюстрації носять інформаційний характер, є доповненням до змісту роботи.

(призначення таблиць та їх використання для підтвердження досліджень в роботі)

9. Якість оформлення роботи на достатньому рівні

10. Ставлення студента (ки) до виконання випускної кваліфікаційної роботи відповідально

11. Висновок

В цілому при написанні випускної кваліфікаційної роботи студент (ка) **Пащенко П.В.**

(прізвище та ініціали)

вміння /невміння/ підібрати необхідну літературу, інформаційну базу, проаналізувати їх, зробити відповідні висновки і оформити свої дослідження у випускній кваліфікаційній роботі.

Випускна кваліфікаційна робота студента (ки) **Пащенка П.В.**

(прізвище та ініціали)

написана на достатньо /недостатньо/ високому рівні та може /не може/ бути допущена до захисту в Екзаменаційній комісії.

Науковий керівник д. п.н., доцент, професор кафедри інженерії програмного забезпечення

(вчений ступінь, звання, посада)

(підпис) Вакалюк Т.А.

(прізвище та ініціали)

«22» 06 2020 р.

РЕЦЕНЗІЯ
на кваліфікаційну роботу
студента 4 курсу групи КІ-2 спеціальності 123 «Комп'ютерна інженерія»
факультету інформаційно-комп'ютерних технологій
Державного університету «Житомирська політехніка»
Пащенка Павла Володимировича
на тему: „Проект інформаційної системи для моніторингу роботи залізничного транспорту"

Кваліфікаційна робота студента Пащенка П.В. „Проект інформаційної системи для моніторингу роботи залізничного транспорту" присвячена досить актуальній тематиці – розробці засобів автоматизації роботи працівників підрозділів залізничного транспорту.

Робота містить реалізований за допомогою засобів середовища Delphi 7 та СУБД MS SQL 2019 проект інформаційної системи та пояснювальну записку наступного складу: вступ, три розділи основного тексту, висновки, список використаної літератури, додатки.

Перший розділ присвячений огляду інформаційних джерел. У розділі проведено аналіз аналогічних розробок, наведено огляд типів баз даних, розглянуто поняття тригерів та збережених процедур. Другий розділ присвячений проектуванню ІС. У розділі визначені цілі і задачі проектування, наведено обгрунтування проектних рішень, наведено передбачуваний набір процедур. Третій розділ присвячений опису програми та тестуванню. У розділі наведено керівництво користувача, наведено опис алгоритмів та результати тестування розробленої ІС.

Зміст роботи в цілому відповідає поставленому завданню. Кваліфікаційна робота виконана на достатньому рівні. Тема роботи розкрита в достатньому обсязі. Студентом продемонстровані достатні фахові компетентності, що отримані як під час навчання, так і у процесі виконання роботи. Матеріал пояснювальної записки викладено відносно грамотно. Ілюстративний матеріал (рисунки, таблиці), що наведений у роботі, оформлений відносно якісно і наведений у відповідних місцях. Оформлення роботи у цілому відповідає вимогам стандартів. Проект ІС реалізований за допомогою засобів застарілого середовища Delphi 7 та СУБД MS SQL 2019. Функціонал є достатнім для демонстрації основних запроектованих рішень.

Позитивних сторін роботи не встановлено.

Недоліками роботи є наступні:

– не досить якісно проведений аналіз діяльності об'єкта дослідження;
– не належно виконано структуризацію матеріалу роботи;
– не наведено проекту структури ІС;
– обрано застарілі засоби розробки Delphi 7;
– не наведено приклади звітів з роботи ІС;
– наявні стилістичні і граматичні помилки, помилки та відхилення від вимог щодо оформлення пояснювальної записки (наприклад, перелік умовних скорочень наведено не в відповідному місці, у змісті пропущено підрозділ 1.1, некоректно відображено таблицю 3.2, помилки при оформленні списку використаної літератури);

Кваліфікаційна робота студента Пащенка П.В. „Проект інформаційної системи для моніторингу роботи залізничного транспорту" умовно відповідає вимогам рівня вищої освіти «бакалавр» спеціальності 123 «Комп'ютерна інженерія» і може бути допущена до захисту в ЕК.

Пропонована оцінка роботи – «задовільно» (за умови успішного захисту).

Рецензент,
Завідувач кафедри комп'ютерної інженерії
та кібербезпеки, к.т.н.
22 червня 2020 р.



А.А. Єфіменко

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему:

«Проект інформаційної системи для моніторингу роботи залізничного транспорту»

Виконав студент 4-го курсу, групи КІ-2
спеціальності 123 «Комп’ютерна інженерія»
_____ Пащенко П.В.
Керівник, доктор педагогічних наук,
доцент, професор кафедри інженерії
програмного забезпечення
_____ Вакалюк Т. А.
Рецензент, завідувач кафедри КІ та КБ.,
к.т.н., доцент
_____ Єфіменко А. А.

Житомир – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ
СПЕЦІАЛЬНІСТЬ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерної інженерії
та кібербезпеки
Єфіменко А. А.
«06» квітня 2020 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу

Студента **Пащенка Павла Валерійовича**

Тема роботи: **Проект інформаційної системи для моніторингу роботи залізничного транспорту**
затверджена Наказом по університету від **«06» квітня 2020 р. № 98с**
Термін здачі студентом закінченої роботи – **1 червня 2020 р.**
Вихідні дані для роботи:

1. Аналіз проблематики, методів та засобів вирішення задач
2. Проектування та розробка програмного забезпечення
3. Опис роботи з програмним додатком та його тестування

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1,2,3	Проф. кафедри ІПЗ, д.п.н. Вакалюк Т. А.	06.04.2020 р.	06.04.2020 р.
2	Проф. кафедри ІПЗ, д.п.н. Вакалюк Т. А.	27.04.2020 р.	27.04.2020 р.
3	Проф. кафедри ІПЗ, д.п.н. Вакалюк Т. А.	18.05.2020 р.	18.05.2020 р.

Керівник _____ Вакалюк Т.А.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	6 квітня 2020 – 7 квітня 2020	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	8 квітня 2020 – 11 квітня 2020	Виконано
3	Формулювання технічного завдання	12 квітня 2020 – 16 квітня 2020	Виконано
4	Опрацювання літературних джерел	17 квітня 2020 – 19 квітня 2020	Виконано
5	Проектування структури	20 квітня 2020 – 24 квітня 2020	Виконано
6	Написання програмного коду	25 квітня 2020 – 30 квітня 2020	Виконано
7	Відлагодження	1 травня 2020 – 4 травня 2020	Виконано
8	Тестування роботи системи	5 травня 2020 – 9 травня 2020	Виконано
9	Аналіз результатів тестування	10 травня 2020 – 13травня 2020	Виконано
10	Написання пояснювальної записки	25 травня 2020 – 29 травня 2020	Виконано
11	Підготовка презентації та виступу	30 травня 2020 – 31 травня 2020	Виконано
12	Отримання відгуку керівника та рецензії на кваліфікаційну роботу	4 червня 2020 – 8 червня 2020	Виконано
13	Захист	21 червня 2020	Виконано

Студент _____ Пащенко П.В.

Керівник _____ Вакалюк Т.А.

РЕФЕРАТ

Кваліфікаційна робота складається з проекту інформаційної системи для моніторингу роботи залізничного транспорту та пояснювальної записки до нього. Пояснювальна записка до роботи містить 67 сторінок, 19 ілюстрацій та 3 таблиці.

Метою роботи є розробка проекту інформаційної системи для моніторингу роботи залізничного транспорту.

В роботі визначено основні завдання на розробку інформаційної системи, яка забезпечить автоматизацію роботи менеджерів залізниці (касира, оператора і служби довідок) з метою зниження тимчасових і грошових витрат на виконання стандартних рутинних операцій.

У першому розділі проведено аналітичний огляд літературних та інших джерел. Описано теоретичний матеріал.

У другому розділі продемонстрований процес проектування інформаційної системи для моніторингу роботи залізничного транспорту.

У третьому розділі описано тестування готового програмного забезпечення.

КЛЮЧОВІ СЛОВА: БАЗИ ДАНИХ, ДАНІ, СУБД, ІНФОРМАЦІЯ, СИСТЕМА, ЗАПИТИ.

Зм.	Арк.	№ докум.	Підпис	Дата	ЖИТОМИРСЬКА ПОЛІТЕХНІКА. 20.123.20.000 - ПЗ		
Розроб.		Пащенко П.В.		20.06	Проект інформаційної системи для моніторингу роботи залізничного транспорту	Літ.	Арк. 4 / Аркушів 56
Керівник		Вакалюк Т.А..		2020.			
Н.контроль		Єфіменко А.А.		22.06.20		ФІКТ, група КІ-2	
Рецензент		Єфіменко А.А.		22.06.20			
Зав. каф.		Єфіменко А.А.		22.06.20			

ABSTRACT

As part of the final work, an information system was developed for traffic management and railway transport management. The purpose of the work is to optimize the work of railway station employees and create a convenient way to order and print tickets.

The software based on the implementation of databases helps in managing basic operations such as ticket sales in various directions, managing the schedule of railway transport, basic operations with the carriage of passengers.

The information system is a software product provided by the database server.

The first section provides an analytical review of literature and other sources. Theoretical material is described.

The second section demonstrates the process of designing an information system for monitoring the operation of railway transport.

The third section describes the testing of ready-made software.

KEY WORDS: DATABASE, DATA, DBMS, INFORMATION, SYSTEM, REQUESTS

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

ПЗ – Програмне забезпечення

ІС – Інформаційна система

СУБД – Система управління базами даних

ПК – Первинний ключ

ВК – Вторинний ключ

SQL – Structure Query Language

АСУ – Автоматизована система управління

АРМ – Автоматизоване робоче місце

БНД – Банк даних

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6
ВСТУП 8
РОЗДІЛ 1. Аналітичний огляд літературних та інших джерел 10
1.2 Аналіз аналогічних розробок 13
1.3 Типи баз даних 14
1.4. Тригери 23
1.5. Збережені процедури 24
ВИСНОВОК ДО РОЗДІЛУ 1 27
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ МОНІТОРИНГУ РОБОТИ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ 28
2.1 Визначення цілей та задач проектування 28
2.2 Обґрунтування проектних рішень з інформаційного забезпечення 29
2.3. Процедури 30
ВИСНОВОК ДО РОЗДІЛУ 2 33
РОЗДІЛ 3. ОПИС ПРОГРАМИ ТА ТЕСТУВАННЯ 34
3.1 Керівництво користувача 34
3.2 Розробка алгоритмів 36
3.2.1 Проектний розділ 37
3.2.2 Інформаційне забезпечення комплексу завдань та використання класифікаторів та системи кодування 38
3.2.3 Характеристика нормативно-довідкової і вхідної оперативної інформації 39
3.2.5 Характеристика даталогічної моделі БД 41
3.2.6 Характеристика результатної інформації 42
3.2.7 Програмне забезпечення комплексу задач 44
3.2.8 Структурна схема пакета 45
3.2.9 Технологічне забезпечення комплексу задач 47
3.3 Тестування програмного забезпечення 47
ВИСНОВКИ ДО РОЗДІЛУ 3 52
ВИСНОВКИ 53
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 54
ДОДАТКИ 56

ВСТУП

В даний час важко уявити будь-яку сферу діяльності людини, де б не стояла проблема створення і використання інформаційних систем. Сьогодні такі системи стали нагальною потребою, і попит на грамотних фахівців в цій області стає дедалі більшим. А оскільки вся будівля інформаційних систем базується на концепції баз даних, то природно, що без більш-менш детального знайомства з основами дисципліни "Бази даних" в наш час неможливо бути не тільки кваліфікованим програмістом, але навіть і грамотним користувачем комп'ютерів. Тому можна сміливо сказати, що навички роботи в цій галузі не тільки підвищують інтелектуальний потенціал користувача, але є в цьому питанні одним з основних факторів.

Історія досліджень систем баз даних - це, по суті, історія розвитку додатків, які досягли виняткової продуктивності і зробили величезний вплив на економіку. Якщо ще 20 років тому ця сфера була всього лиш областю фундаментальних наукових досліджень, то тепер на дослідженнях баз даних заснована ціла індустрія інформаційних послуг. Досягнення в дослідженнях баз даних стали основою фундаментальних розробок комунікаційних систем, транспорту і логістики, фінансового менеджменту, систем з базами знань, методів доступу до наукової літератури, а також великої кількості цивільних і військових програм.

Актуальність теми дипломного проекту зумовлюється необхідністю автоматизації роботи менеджерів (касира, оператора і служби довідок) з метою зниження тимчасових і грошових витрат на виконання стандартних рутинних операцій. Практична значущість роботи визначається розробкою реального програмного засобу, що служить для автоматизації роботи менеджерів касира, оператора і служби довідок.

Основні завдання дослідження:

– аналіз поточного стану процесу обробки інформації при роботі касира, оператора і служби довідок;

– аналіз аналогічних продуктів

– проектування і реалізація АСУ пасажирськими залізничними перевезеннями.

АСУ пасажирськими залізничними перевезеннями можна розділити на АРМ оператора, АРМ касира і довідкову, які повинні забезпечувати виконання таких основних функцій:

– ведення довідників вагонів, типів і категорій вагонів, місць по вагонах, довідників поїздів, станцій;

– формування складу поїзда;

– складання та планування шляху прямування поїзда;

– складання розкладу руху поїзда;

– продаж квитка;

– облік проданих квитків;

– видачу інформації про склади, що мають задані станцію відправлення та призначення;

– видачу інформації про шлях прямування поїзда і квитків в наявності;

– формування документів:

– проїзний документ (квиток);

– інформація про склад поїзда по вагонах;

– інформація про шлях прямування поїзда;

– інформація про продані квитки для заданого періоду.

Метою роботи є створення системи управління пасажирськими залізничними перевезеннями, що дозволяє усунути недоліки реального процесу продажу залізничних квитків.

Відповідно до мети було поставлено такі завдання:

1. Підвищення ступеня достовірності обробки інформації;
2. Підвищення ступеня захищеності інформації;
3. Автоматизація роботи менеджера залізниці

Обєкт дослідження – робота залізниці

Предмет дослідження – ефективне управління ресурсами та співробітниками.

РОЗДІЛ 1. Аналітичний огляд літературних та інших джерел

1.1 Визначення

Стрижневі ідеї сучасних інформаційних технологій базуються на концепції баз даних.

Відповідно до цієї концепції, основою інформаційних технологій є дані, які повинні бути організовані в бази даних з метою адекватного відображення мінливого реального світу і задоволення інформаційних потреб користувачів.

Одним з найважливіших понять в теорії баз даних є поняття інформації. Під інформацією розуміються будь-які відомості про будь-яку подію, процеси, об'єкти. До інформації може належати все, що може цікавити користувача будь-якого рівня.

Дані - це інформація, представлена в певному вигляді, що дозволяє автоматизувати її збір, зберігання і подальшу обробку людиною або інформаційним засобом. Для комп'ютерних технологій дані - це інформація в дискретно - фіксованому вигляді, зручна для зберігання, обробки на ЕОМ, а також для передачі по каналах зв'язку.

База даних (БД) - сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв’язки між їх елементами. БД складається з безлічі пов'язаних файлів.

Система управління базами даних (СКБД) - сукупність мовних та програмних засобів, призначених для створення, ведення і сумісного використання БД багатьма користувачами.

Автоматизована інформаційна система (АІС) - це система, яка реалізує автоматизований збір, обробку, маніпулювання даними, що функціонує на основі ЕОМ та інших технічних засобів і включає відповідне програмне забезпечення (ПО) і персонал. Надалі в цій якості буде використовуватися термін інформаційна система (ІС), який має на увазі поняття автоматизована.

ІС може функціонувати самостійно або служити компонентом більш складної системи.

Кожна ІС в залежності від її призначення має справу з тією чи іншою частиною реального світу, яку прийнято називати предметною областю (ПрО) системи. Виявлення ПрО - це необхідний початковий етап розробки будь-якої ІС. Саме на цьому етапі визначаються інформаційні потреби всієї сукупності користувачів майбутньої системи, які, в свою чергу, зумовлюють зміст її бази даних. По області застосування ІС можна розділити на системи, які використовуються в освіті, виробництві, бізнесі, науці та інших областях.

Банк даних (БНД) є різновидом ІС. БНД - це система спеціальним чином організованих даних: баз даних, програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого накопичення і колективного багатоцільового використання даних.

Проектована БД повинна мати певні властивості. Назвемо основні властивості БД.

Цілісність. У кожен момент часу існування БД відомості, що містяться в ній, повинні бути несуперечливі. Цілісність БД досягається внаслідок введення обмежень цілісності, зокрема, до них належать обмеження, пов'язані з нормалізацією БД. Бажано відстежувати діапазон допустимих значень, співвідношення між значеннями в полях, особливості написання формату.

Існують обмеження, що працюють тільки при видаленні записів.

Наприклад, не можна видаляти запис, пов'язану з іншого, що не видаляється записом.

1. Відновленість. Дана властивість передбачає можливість відновлення БД після збою системи або окремих видів псування системи. Так само як перевірка наявності файлів, складових додатків. В основному властивість відновлюваності забезпечується дублюванням БД і використанням техніки підвищеної надійності.
2. Безпека. Безпека БД передбачає захист даних від навмисного і ненавмисного доступу, модифікації або руйнування. Дії заборони на

залізницях несанкціонованого доступу, захист від копіювання та криптографічний захист. Також необхідні і чисто адміністративні заходи, наприклад, обмеження доступу до носіїв інформації.

3. Ефективність. Властивість ефективності зазвичай розуміється як:
   - мінімальний час реакції на запит користувача;
   - мінімальні потреби в пам'яті;
   - поєднання цих параметрів.

Граничні розміри і експлуатаційні обмеження. Граничні розміри, а також інші обмеження, що накладаються експлуатацією даної БД, можуть істотно вплинути на проектне рішення.

Створення баз даних, підтримка їх в цілісному, несуперечливому стані, забезпечення безпеки їх використання та збереження інформації аж до відновлення її після різних збоїв, надання різних інформаційних послуг користувачам і багато іншого забезпечується системами управління баз даних, яким присвячений наступний підрозділ книги.

Модель даних - це деяка абстракція, в якій відображаються найважливіші аспекти функціонування виділеної предметної області, а другорядні - ігноруються. Модель даних включає в себе набір понять для опису даних, зв'язків між ними і обмежень, що накладаються на дані. У моделі даних розрізняють три головні складові:

– структурну частину, що визначає правила породження допустимих для даної СУБД видів структур даних;

– керуючу частину, що визначає можливі операції над такими структурами;

– класи обмежень цілісності даних, які можуть бути реалізовані засобами цієї системи.

Кожна СУБД підтримує ту чи іншу модель даних. Необхідно відзначити, що поняття моделі даних фактично увійшло в побут фахівців в області БД тільки разом з появою реляційного підходу. Усі попередні системи не грунтувалися на

будь-яких абстрактних моделях. Абстрактні уявлення попередніх систем з'явилися пізніше на основі аналізу та виявлення загальних ознак у різних системах.

Попередні (кореляційні) СУБД активно використовувалися протягом багатьох років. Свого часу в подібні системи багато компаній інвестували великі кошти. Вони застосовувалися довше, ніж використовується будь-яка з реляційних СУБД, а деякі з ранніх систем використовуються навіть в наш час. За час їх існування накопичені величезні бази даних, і однією з актуальних проблем інформаційних систем є використання цих систем спільно з сучасними системами.

Отже, по суті, модель даних, що підтримується механізмами СУБД, повністю визначає безліч конкретних баз даних, які можуть бути створені засобами цієї системи, а також способи модифікації стану БД з метою відображення тих змін, які відбуваються в предметній області.

1.2 Аналіз аналогічних розробок

Під час опрацювання джерел та пошуку аналогічних систем в мережі Інтернет було виявлено досить багато аналогічних розробок різного плану і напрямів використання. При аналізі їх призначення досліджено, що деякі з них є досить застарілими і на сьогоднішній день вже не використовуються.

Загалом в фінальну добірку потрапили ті розробки, які є конкуруючими на сьогоднішньому ринку реалізацій мереж для спілкування до яких відноситься найбільше користувачів та сучасна молодь.

Отже, в вибірці аналогічних розробок представлені ті ресурси, які допомагають замовити пасажирські квитки для подорожей залізничним транспортом. Далі в таблиці наведені такі системи:

Таблиця 1.1

Аналогічні системи

№п/п	Назва системи	Опис
1	poezd.ua	Агентство «АРТ-ТУР» (офіційно зареєстрована товарна марка - Українське авіаційне транспортне агентство «АРТ-ТУР») засновано 5 березня 1998 року. Основний вид діяльності агентства - бронювання і продаж авіаквитків. Продаж і доставка залізничних квитків по Києву
2	booking.uz.gov.ua/ru/	Акціонерне товариство «Українська залізниця» – національний перевізник вантажів та пасажирів. Метою діяльності товариства є задоволення потреб у безпечних та якісних залізничних перевезеннях, забезпечення ефективного функціонування та розвитку залізничного транспорту, створення умов для підвищення конкурентоспроможності галузі тощо.
3	gd.tickets.ua/	Tickets.ua - частина холдингу TTN, одного з провідних і динамічно розвиваються туристичних онлайн-агентств Європи.
4	businessvisit.com.ua	Туристична компанія активно веде свою діяльність на ринку України в сфері ділового туризму. Компанія є універсальним консолідатором з оформлення авіаквитків, залізничних квитків, готелів для туристичних агентів по Україні.

1.3 Типи баз даних

Типи баз даних, які називаються також моделями БД або сімействами БД, є шаблони і структури, які використовуються для організації даних в системі керування базами даних (СКБД). Вибір типу вплине на те, які операції зможе

виконувати додаток, як будуть представлені дані, на функції СКБД для розробки і рантайму.

Почнемо з трьох типів БД, які все ще можуть зустрічатися в спеціалізованих середовищах, але в основному замінені надійними і продуктивними альтернативами.

1. Прості структури даних

Перший і найпростіший спосіб зберігання даних - текстові файли. Метод застосовується і сьогодні для роботи з невеликими обсягами інформації. Для поділу полів використовується спеціальний символ: кома або крапка з комою в csv-файлах датасетів, двокрапка або пробіл в * nix-подібних системах:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
bob:x:1000:1000:Bob Smith,,,:/home/bob:/bin/bash
```

Рис.1.1 Прості структури даних

Наслідки: Обмежений тип і рівень складності інформації, що зберігається;

Важко встановити зв'язку між компонентами даних; Відсутність функцій паралелізму; Практичні тільки для систем з невеликими вимогами до читання і запису; Використовуються для зберігання конфігураційних даних; Немає необхідності в сторонньому програмному забезпеченні.

Приклади: /etc/passwd и /etc/fstab в *nix-системах

csv-файли

2. Ієрархічні бази даних

На відміну від текстових таблиць, в наступному типі БД з'являються зв'язки між об'єктами. В ієрархічних базах даних кожний запис має одного «батька». Це створює деревоподібну структуру, в якій записи класифікуються за їхнім стосункам з ланцюжком батьківських записів.

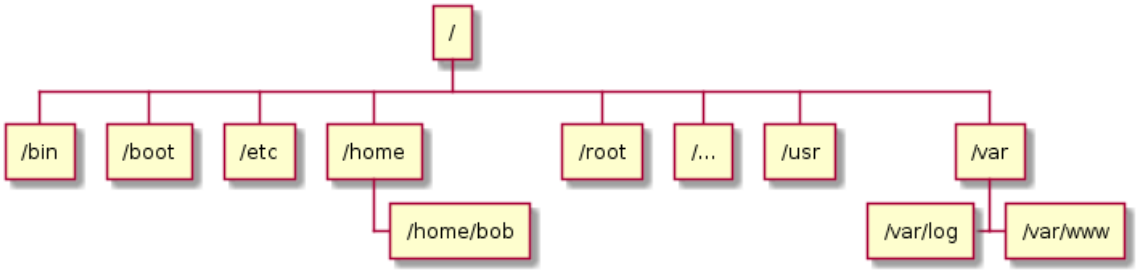


Рис. 1.2 Приклад побудови ієрархічних зв'язків

Наслідки: Інформація організована у вигляді дерева з відносинами «предок-нащадок»; Кожен запис може мати не більше одного з батьків;

Зв'язки між записами виконані у вигляді фізичних покажчиків;

Неможливо реалізувати відносини «багатьох до багатьох».

Приклади: файлові системи, DNS, LDAP

3. Мережеві бази даних

Мережеві бази даних розширюють функціональність ієрархічних: записи можуть мати більше одного батька. А це означає, що можна моделювати складні відносини.

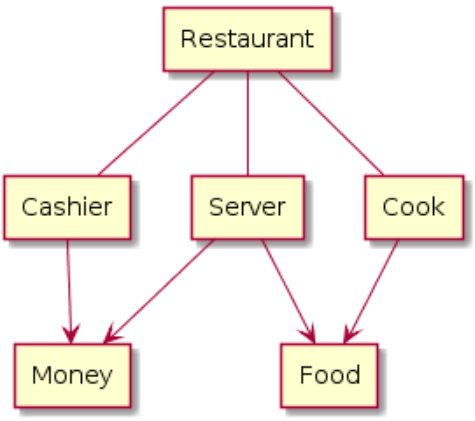


Рис. 1.3 Приклад зв'язків в мережевий базі даних

Наслідки: Мережеві бази даних подаються не деревом, а загальним графом

Обмежені тими самими шаблонами доступу, що ієрархічні БД

Приклади: IDMS

II. реляційні БД

4. SQL бази даних

Реляційні бази даних - найстаріший тип досі широко використовуваних БД загального призначення. Дані та зв'язки між даними організовані за допомогою таблиць. Кожен стовпець у таблиці має ім'я і тип. Кожен рядок представляє окремий запис або елемент даних в таблиці, який містить значення для кожного з стовпців.

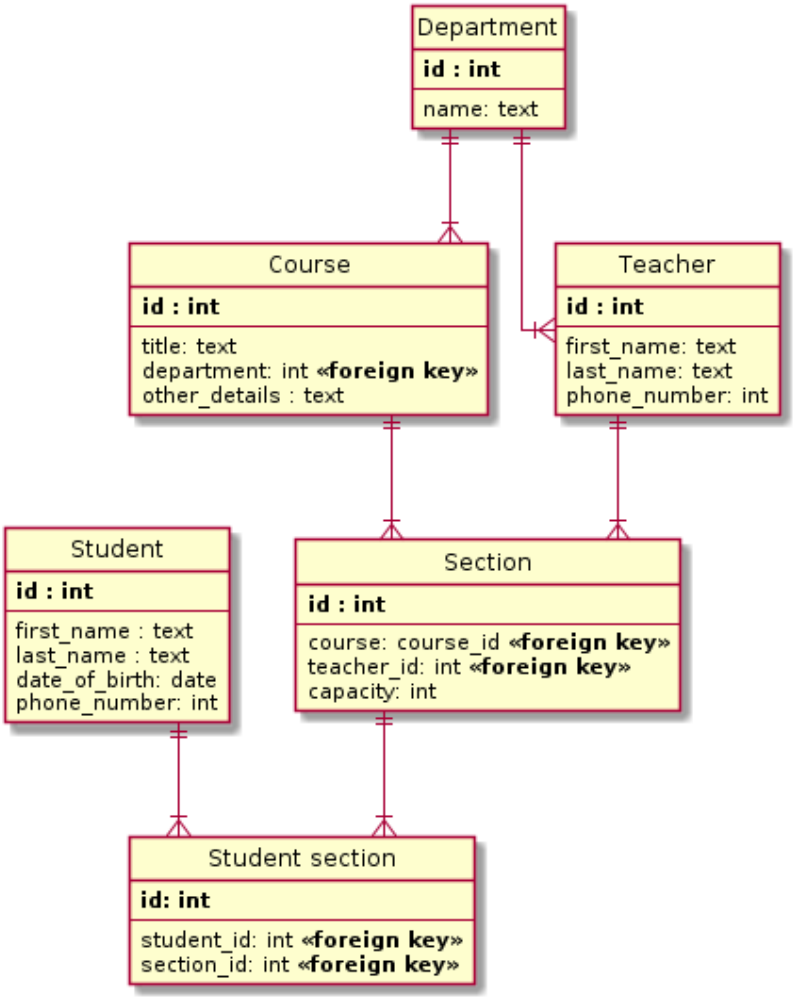


Рис. 1.4 Приклад бази даних SQL

Наслідки: Поле в таблиці, що називається зовнішнім ключем, може містити посилання на стовпці в інших таблицях, що дозволяє їх з'єднувати;

Високоорганізована структура і гнучкість робить реляційні БД потужними і адаптуються до різних типів даних; Для доступу до даних використовується мова структурованих запитів (SQL); Надійний вибір для багатьох додатків.

Приклади: MySQL, MariaDB, PostgreSQL, SQLite

III. NoSQL бази даних

NoSQL - група типів БД, що пропонують підходи, відмінні від стандартного реляційного шаблону. Говорячи NoSQL, мають на увазі або "не-SQL», або «не тільки SQL», щоб уточнити, що іноді допускається SQL-подібний запит.

5. Бази даних «ключ-значення»

У базах даних «ключ-значення» для зберігання інформації ви представляєте ключ і об'єкт даних, який потрібно зберегти. Наприклад, JSON-об'єкт, зображення або текст. Щоб зробити запит, відправляєте ключ і отримуєте blob-об'єкт.

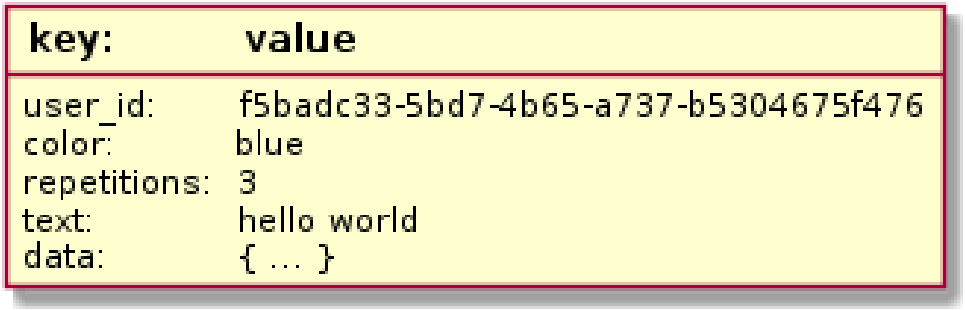


Рис. 1.5 Приклад бази даних «ключ-значення»

Наслідки: Сховища забезпечують швидкий доступ з незначними витратами;

Часто зберігають дані конфігурацій і інформацію про стан даних, представлених словниками або хешем; Немає жорсткої схеми відносини між даними, тому в таких БД часто зберігають одночасно різні типи даних; Розробник відповідає за визначення схеми іменування ключів і за те, щоб значення мало відповідний тип / формат.

Приклади: Redis, memcached, etcd.

6. Документна база даних

Документні бази даних (також документоорієнтовані БД або сховища документів), спільно використовують базову семантику доступу і пошуку

сховищ ключів і значень. Такі БД також використовують ключ для унікальної ідентифікації даних. Різниця між сховищами «ключ-значення» і документною БД полягає в тому, що замість зберігання blob-об'єктів, документоорієнтовані бази зберігають дані в структурованих форматах - JSON, BSON або XML.

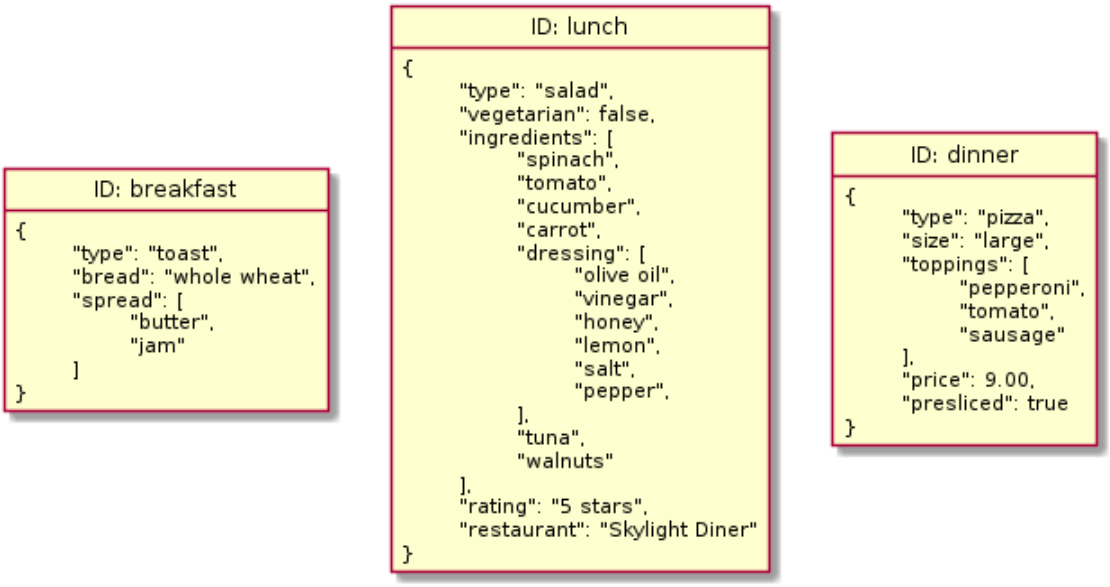


Рис. 1.6 Приклад документної БД

Наслідки: База даних не наказує певний формат або схему; Кожний документ може мати свою внутрішню структуру; Документні БД є хорошим вибором для швидкої розробки; В будь-який момент можна змінювати властивості даних, не змінюючи структуру або самі дані.

Приклади: MongoDB, RethinkDB

7. Графова база даних

Замість зіставлення зв'язків з таблицями і зовнішніми ключами, графові бази даних встановлюють зв'язки, використовуючи вузли, ребра і властивості.

Графові бази представляють дані у вигляді окремих вузлів, які можуть мати будь-яку кількість пов'язаних з ними властивостей.

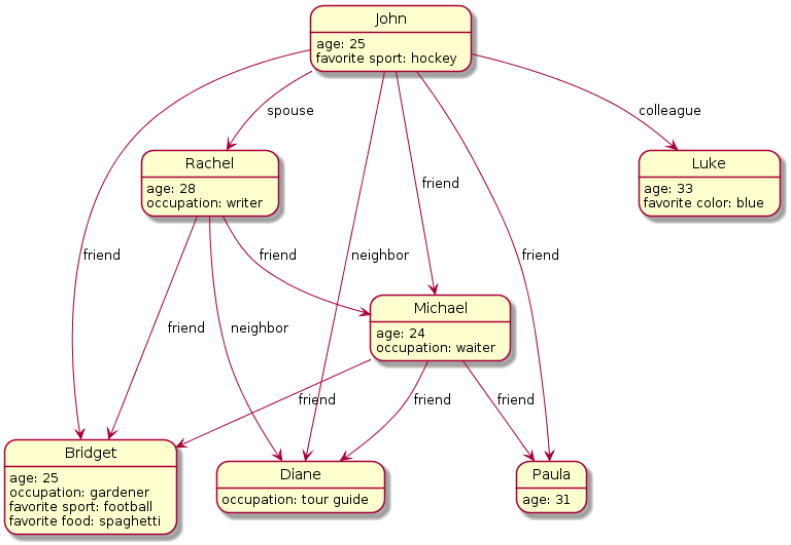


Рис. 1.7 Приклад графової БД

Наслідки: Виглядають аналогічно мережевим; Фокусуються на зв'язках між елементами; Явно відображають зв'язки між типами даних; Не вимагають покрокового обходу для переміщення між елементами; Немає обмежень в типах зв'язків, що представляються.

Приклади: Neo4j, JanusGraph, Dgraph

8. Стовбчикові бази даних

Стовбчикові бази даних (також нереляційні колоночні сховища або бази даних з широкими стовпцями) належать до сімейства NoSQL БД, але зовні схожі на реляційні БД. Як і реляційні, стовпчикові БД зберігають дані, використовуючи рядки і стовпці, але з іншим зв'язком між елементами.

У реляційних БД всі рядки повинні відповідати фіксованій схемою. Схема визначає, які стовпчики будуть в таблиці, типи даних та інші критерії. У стовпчикових базах замість таблиць є структури - «стовпчик сімейства». Сімейства містять рядки, кожна з яких визначає власний формат. Рядок складається з унікального ідентифікатора, що використовується для пошуку, за яким слідують набори імен та значень стовпців.

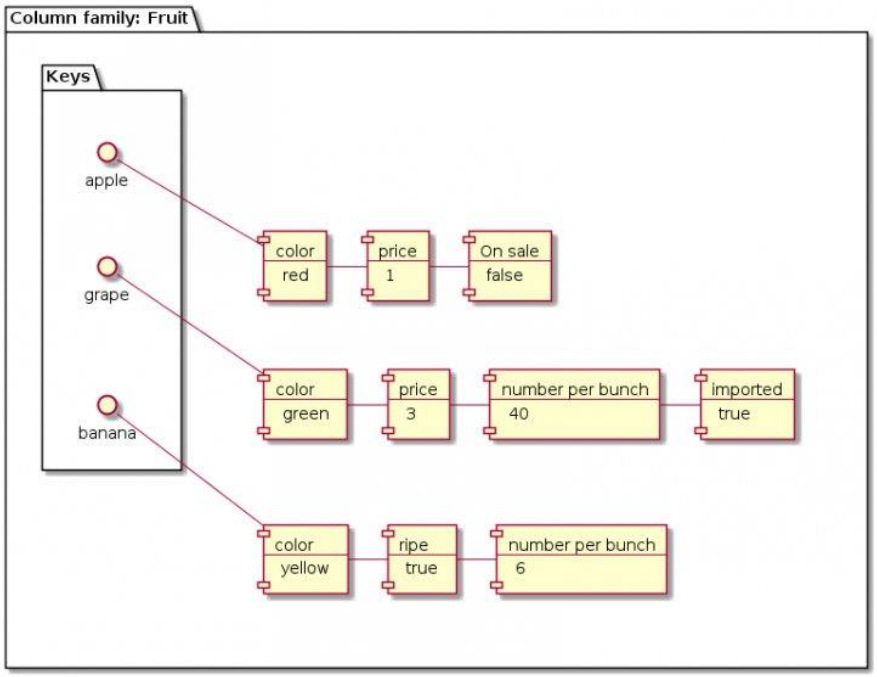


Рис. 1.8 Приклад стовпчикової БД

Наслідки: БД зручні при роботі з додатками, що вимагають високої продуктивності; Дані та метадані доступні по одному ідентифікатору; Гарантоване розміщення всіх даних з рядка в одному кластері, що спрощує сегментацію і масштабування даних.

Приклади: Cassandra, HBase

9. Бази даних часових рядів

Бази даних часових рядів створені для збору і управління елементами, що змінюються з плином часу. Більшість таких БД організовані в структури, які записують значення для одного елемента. Наприклад, можна створити таблицю для відстеження температури процесора. Усередині кожне значення буде складатися з тимчасової мітки і показника температури. У таблиці може бути декілька метрик.

Time	CPU Temp	System Load	Memory Usage %
2019-10-31T03:48:05+00:00	37	0.85	92
2019-10-31T03:48:10+00:00	42	0.87	90
2019-10-31T03:48:15+00:00	33	0.74	87
2019-10-31T03:48:20+00:00	34	0.72	77
2019-10-31T03:48:25+00:00	40	0.88	81
2019-10-31T03:48:30+00:00	42	0.89	82
2019-10-31T03:48:35+00:00	41	0.88	82

Рис. 1.9 Приклад бази даних часових рядів

Наслідки: орієнтовані на запис; призначені для обробки постійного потоку вхідних даних; продуктивність залежить від кількості відслідковуваних елементів, інтервалу опитування між записом нових значень і фактичного корисного навантаження даних.

Приклади: OpenTSDB, Prometheus, InfluxDB, TimescaleDB

Комбіновані типи

NewSQL і багатомодельні БД є різними типами баз даних, але вирішують одну групу проблем, викликаних полярними підходами SQL або NoSQL-стратегії. Чому б не об'єднати переваги обох груп?

10. NewSQL бази даних

NewSQL бази даних успадковують реляційну структуру і семантику, але побудовані з використанням більш сучасних, масштабованих конструкцій. Мета - забезпечити більшу масштабованість, ніж реляційні БД, і більш високі гарантії узгодженості, ніж в NoSQL. Компроміс між узгодженістю і доступністю є фундаментальною проблемою розподілених баз даних, описаних теоремою CAP.

Наслідки: можливість горизонтального масштабування; висока доступність;

велика продуктивність і реплікація; невеликий функціонал і гнучкість;

чимале споживання ресурсів і необхідність спеціалізованих знань для роботи з базою даних.

Приклади: MemSQL, VoltDB, Spanner, Calvin. CockroachDB, FaunaDB, YugabyteDB

11. Багатомодельні бази даних

Багатомодельні бази даних - бази, які поєднують функціональні можливості кількох видів БД. Переваги такого підходу очевидні - одна і та ж система може використовувати різні уявлення для різних типів даних.

Спільне розміщення даних з декількох типів БД в одній системі дозволяє виконувати нові операції, які в іншому випадку були б ускладнені або неможливі. Наприклад, багатомодельні бази можуть дозволити користувачам отримати доступ до даних, що зберігаються в різних типах БД, і управляти ними в рамках одного запиту, а також підтримують узгодженість даних при виконанні операцій, що змінюють інформацію відразу в декількох системах.

Наслідки: допомагають зменшити навантаження на СУБД; дозволяють розширюватися до нових моделей у міру зміни потреб без внесення змін до базової інфраструктури; забезпечують безперервний доступ і простий розподіл даних; мають лінійну масштабованість і прості для розробки.

Приклади:ArangoDB, OrientDB, Couchbase

1.4. Тригери

Тригери - методи, за допомогою яких можна забезпечувати цілісність бази даних навіть у тих випадках, коли вона використовується безліччю додатків.

Тригер - це спеціальний вид збереженої процедури, яка автоматично виконується при кожній спробі змінити її дані. Тригери забезпечують цілісність даних, запобігаючи їх несанкціонованих або неправильних змін.

Припустимо, що в базі даних є таблиці, пов'язані через поле Surname. Наприклад, це можуть бути таблиця клієнтів підприємства і їх замовлень. Розумно визначити тригер, який при кожній спробі видалити запис клієнта, перевірить наявність у нього замовлень і дозволить видалити цей запис тільки при їх відсутності. Звичайно, подібне завдання можна вирішити за допомогою засобів декларативної посилальної цілісності. Однак, за допомогою тригерів можна створювати більш складні робочі правила. Можна створити тригер, який при кожному додаванні запису в таблицю замовлень, аналізує попередні

замовлення цього ж клієнта і визначає прийнятний термін оплати даного замовлення.

Тригери не приймають параметрів і не повертають значень. Вони виконуються неявно. Тобто тригер запускається тільки при спробі зміни даних.

1.5. Збережені процедури

Даний розділ присвячений одному з механізмів підвищення ефективності функціонування інформаційних систем, який базується на використанні збережених процедур.

Процедура - це послідовність компілювання операторів Transact-SQL, що зберігаються в системній базі даних SQL Server. Збережені процедури попередньо відкомпільовані, тому ефективність їх виконання вища, ніж у звичайних запитах. Збережені процедури працюють безпосередньо на сервері і добре вкладаються в модель клієнт - сервер.

Існує два види процедур: системні і призначені для користувача.

Системні збережені процедури призначені для отримання інформації з системних таблиць і виконання різних службових операцій і особливо корисні при адмініструванні бази даних. Їх імена починаються з sp_ (stored procedure).

Призначені для користувача процедури, що створюються безпосередньо розробниками або адміністраторами бази даних.

Корисність збережених процедур визначається в першу чергу високою (у порівнянні зі звичайними T-SQL запитами) швидкістю їх виконання. Крім того, вони є засобом систематизації часто виконуваних операцій. При виконанні вперше збереженої процедури можна виділити ряд етапів.

1. Процедура розбивається на окремі компоненти лексичним аналізатором виразів.

2. Компоненти, що посилаються на об'єкти бази даних (таблиці, індекси, уявлення і т. п.) зіставляються з цими об'єктами з попередньою перевіркою їх існування. Цей процес носить назву- дозвіл посилань.

3. В системній таблиці syscomments зберігається вихідний текст процедури, а в таблиці sysobjects - її назва.

4. Створюється попередній план виконання запиту. Він називається нормалізованим планом або деревом запиту і зберігається в системній таблиці sysprocedures.

5. При першому виконанні процедури, що дерево запиту зчитується і остаточно оптимізується. Виконується раніше створений план процедури.

Така схема дає можливість при повторних викликах не витрачати час на синтаксичний аналіз, дозвіл посилань і компіляцію дерева запитів. А при наступних викликах виконується тільки п'ятий крок. Причому план збереженої процедури після першого виконання міститься в швидкодіючому процедурному кеші. Це означає, що під час виклику процедури швидкість його зчитування буде дуже високою.

Використання збережених процедур мають ще ряд додаткових переваг.

– -Збережені процедури дозволяють виділяти правила в окрему структуру. Надалі ці правила використовуються багатьма додатками, утворюючи стійкий до помилок інтерфейс даних. Перевага такого підходу полягає в тому, що можна здійснювати зміни правил тільки для окремої частини об'єктів бази даних, а не для всіх її додатків.

– Використання збережених процедур значно підвищує продуктивність запитів. Однак найбільший її приріст досягається при виконанні багаторазово повторюваних операцій, коли план запиту постійно зберігається в системному кеші.

– Збережені процедури можуть приймати аргументи при запуску і повертати значення (у вигляді результуючих наборів даних).

– Збережені процедури можуть запускатися за розкладом (в режимі автоматичного виконання), що задається при запуску SQL Server.

– Збережені процедури використовуються для вилучення або зміни даних в будь-який час.

– Збережені процедури, на відміну від тригерів, викликаються явно. Тобто при безпосередньому зверненні до процедури з програми, сценарію, пакета або завдання.

Збережені процедури - потужний засіб обробки даних. Системні збережені процедури відіграють дуже важливу роль в адмініструванні і підтримці бази даних. Призначені для користувача процедури застосовуються при вирішенні практично будь-яких завдань. Крім того, користувач може отримати право виконання процедури, навіть якщо він не має права доступу до об'єктів, до яких звертається процедура.

Отже, у першому розділі було проаналізовано аналогічні системи. Було наведено теоретичну інформацію. Визначення, що стосуються баз даних. Інформація про види баз даних. Також було ознайомлено з тригерами та збереженими процедурами.

Зміна типів даних, що зберігаються, вимоги до швидкості і продуктивності привели до триваючого розширення типів баз даних. При цьому кожен з них продовжує бути потрібним у своїй ніші, де взаємозв'язки між даними асоціюються з певною схемою будови бази даних.

ВИСНОВОК ДО РОЗДІЛУ 1

У даному розділі було розглянуто та проаналізовано літературні джерела, наведено приклади аналогічних систем.

Також дано визначення основним поняттям. Проаналізовано типи баз даних. Було розглянуто поняття “тригери” та “збережені процедури”.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ МОНІТОРИНГУ РОБОТИ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

2.1 Визначення цілей та задач проектування

В рамках даної роботи планується автоматизація роботи менеджера з продажу ж/д квитків (касира, оператора і служби довідок). В проекті менеджери виконують такі дії:

оператор: облік складів поїздів, їх шляху проходження, розкладу і проданих квитків; касир: продаж квитків; працівник довідкової: видачу інформації про склади, що мають задані станції відправлення та призначення, видачу інформації про шлях прямування поїзда і квитки в наявності.

АРМ оператора повинна забезпечувати виконання таких основних функцій:

ведення довідників вагонів, типів і категорій вагонів, місць по вагонах, довідників поїздів, станцій; формування складу поїзда; складання та планування шляху прямування поїзда; складання розкладу руху поїзда; облік проданих квитків; видачу інформації про шлях прямування поїзда і квитках в наявності;

формування документів: інформація про склад поїзда по вагонах; інформація про шлях прямування поїзда; інформація про продані квитки для заданого періоду.

АРМ касира повинна забезпечувати виконання таких основних функцій:

видачу інформації про склади, що мають задані станці відправлення та призначення; продаж квитка; друк проїзного документа (квитка).

Довідкова повинна забезпечувати виконання таких основних функцій:

видачу інформації про склади, що мають задані станції відправлення та призначення; видачу інформації про шлях прямування поїзда і квитки в наявності;

видачу інформації про склад обраного поїзда по вагонах.

2.2 Обґрунтування проектних рішень з інформаційного забезпечення

В даному дипломному проекті передбачається створення клієнт-серверного додатка, і відповідно, використання СУБД на сервері, що забезпечує розподілений доступ багатьом користувачам.

У проектованій системі завдання будуть реалізовані відповідним програмним модулем, кожен з яких матиме можливість виконання ряду операцій, таких як коригування інформації, пошук даних, формування звітів. Процеси заповнення, коригування та пошуку даних будуть реалізовані однотипним способом у всіх модулях. При додаванні і редагуванні даних застосовуватиметься контроль правильності введення користувачем інформації, шляхом її порівняння з необхідним типом, діапазоном зміни можливих значень, маскою введення і т.д.

Для додавання і редагування даних будуть використані екранні форми з усіма необхідними полями введення, а також поясненнями і керуючими елементами (наприклад, кнопками), призначеними для вироблення управляючих впливів (збереження, скасування змін) і навігації (переміщенню) по БД. Для забезпечення пошуку даних також будуть використані екранні форми, що дозволяють задавати різні значення (діапазони значень), що цікавить, з контролем допустимості значень умов пошуку. Пошук інформації можна буде робити на частковий, або повний збіг умов пошуку. У програмних модулях системи передбачається наявність всіх необхідних звітних форм для формування і друку документів встановленої форми. Всі звіти генеруються автоматично, використовуючи вибірки інформації з БД.

Вхідними документами для розробленої програми є запити користувачів довідкової системи, запити оператора та касира.

Вихідними документами, що розробляються є проїзний документ (квиток), інформація про склад поїзда по вагонах, інформація про шлях прямування поїзда, інформація про продані квитки для заданого періоду.

Основна ідея реляційної моделі даних полягає в тому, щоб представити будь-який набір даних у вигляді двовимірної таблиці. У найпростішому випадку реляційна модель описує єдину двовимірну таблицю, але частіше за все ця модель описує структуру і взаємовідносини між декількома різними таблицями.

Отже, метою інформаційної системи є обробка даних про об'єкти реального світу, з урахуванням зв'язків між об'єктами. У теорії БД дані часто називають атрибутами, а об'єкти сутностями. Об'єкт, атрибут і зв'язок фундаментальні поняття ІС.

Об'єкт (або сутність) – це щось існуюче і помітне, тобто об'єктом можна назвати те «щось», для якого існують назва і спосіб відрізняти один подібний об'єкт від іншого.

Атрибут (або дане) – це деякий показник, який характеризує якийсь об'єкт і приймає для конкретного екземпляра об'єкта деяке числове, текстове або інше значення. Інформаційна система оперує наборами об'єктів, спроектованими стосовно до даної предметної області, використовуючи при цьому конкретні значення атрибутів (даних) тих чи інших об'єктів.

Реляційною вважається така база даних, в якій всі дані представлені для користувача у вигляді прямокутних таблиць значень даних, і всі операції над базою даних зводяться до маніпуляцій з таблицями.

Таблиця складається з стовпців (полів) і рядків (записів); має ім'я, унікальність всередині бази даних. Таблиця відображає тип об'єкта реального світу (сутность), а кожен її рядок конкретний об'єкт.

Для створення бази даних передбачається використовувати MS SQL Server 2019.

2.3. Процедури

Передбачуваний набір процедур включає в себе:

1) процедури отримання даних з бази даних, відповідно до вимог предметної області (дані таблиць, відомості з певних запитів до таблиць і т.п.);

2) процедури, що підтримують редагування даних і коректний запис їх в базу даних;

3) процедури, що підтримують видалення даних з бази відповідно до потреб користувача;

4) процедури, що підтримують забезпечення цілісності бази даних;

5) процедури, що підтримують захист бази даних від несанкціонованого доступу;

6) процедури, що відповідають за контроль введеної в ПК інформації, що зменшують відсоток помилок оператора при введенні даних;

7) інші процедури.

Для розробки програмного засобу, який здійснює доступ до даних бази використовується Delphi 7.0.

Оскільки в Delphi програма будується візуальним образом, всі ці компоненти мають своє графічне представлення в поле форм для того, щоб можна було ними відповідним чином оперувати. Але для працюючої програми видимими залишаються тільки візуальні компоненти. Компоненти згруповані на сторінках палітри за своїми функціями.

Delphi дозволяє розробникам налаштувати середовище для максимальної зручності. Можна легко змінити палітру компонентів, інструментальну лінійку, а також налаштувати виділення синтаксису кольором.

У Delphi можна визначити свою групу компонентів і розмістити їх на сторінці палітри, а якщо виникне необхідність, перегрупувати компоненти або видалити невикористані.

Інтелектуальний редактор. Редагування програм можна здійснювати, використовуючи запис і виконання макросів, роботу з текстовими блоками, що настроюють комбінації клавіш і колірне виділення рядків.

Щодо графічного відладчика, Delphi володіє потужним, вбудованим в редактор графічним відладчиком, що дозволяє знаходити і усувати помилки в коді. Можна перевірити і змінити змінні, за допомогою покрокового виконання в точності зрозуміти поведінку програми. Якщо ж потрібні можливості більш

тонкого налагодження, можна використовувати окремо доступний Turbo Debugger, перевіривши асемблерні інструкції та регістри процесора.

Отже, у другому розділі було обґрунтовано використання програмного забезпечення для реалізації проекту.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі було визначено цілі та задачі проектування.

Обґрунтовано використання програмного забезпечення для реалізації проекту Delphi 7.0. та системи управління базами даних Microsoft SQL Server 2019.

Було розглянуто передбачуваний у проекті набір процедур.

РОЗДІЛ 3. ОПИС ПРОГРАМИ ТА ТЕСТУВАННЯ

3.1 Керівництво користувача

Екранні форми додатка наведені на рисунках в додатках А.

Для запуску програми запустіть файл Station.exe. Після цього на екрані з'явиться головна форма вибору АРМ.

Для запуску АРМ оператора натисніть відповідну кнопку на головній формі. У вікні «АРМ оператора» можна виконувати наступне:

- редагування довідника станцій;
- редагування довідника вагонів і місць в них;
- редагування типів і категорій вагонів, а також класів місць в них;
- формування складу поїзда;
- формування шляху прямування поїзда;
- складання розкладу руху поїзда;
- облік проданих квитків;
- визначення вільних місць для обраного складу і часу проходження.

Для редагування довідника станцій слід натиснути кнопку «Станції» на формі «АРМ оператора». Після цього на екрані з'явиться форма «Станції», яка припускає додавання, видалення і редагування списку станцій, а також навігацію по записах за допомогою відповідних кнопок роботи з базою даних і навігації по ній.

Для редагування довідника вагонів і місць в них слід натиснути кнопку «Вагони та місця» на формі «АРМ оператора». Після цього на екрані з'явиться форма «Вагони», яка припускає додавання, видалення і редагування списку вагонів, а також навігацію по записах за допомогою відповідних кнопок роботи з базою даних і навігації по ній. Для редагування списку місць вагона слід вибрати його з таблиці «Вагони» на форми і далі працювати з таблицею «Місця вагона» тієї ж форми.

Для редагування довідників типів і категорій вагонів, а також класів місць в них слід натиснути кнопку «Вагони» на формі «Вагони». Після цього на екрані з'явиться форма «Типи, категорії, класи вагонів», яка припускає додавання, видалення і редагування даних, а також навігацію по записах за допомогою відповідних кнопок роботи з базою даних і навігації по ній.

Для формування складу поїзда слід натиснути кнопку «розклад і склади» на формі «АРМ оператора». Після цього на екрані з'явиться форма «Потяги», яка припускає додавання, видалення і редагування списку поїздів, а також навігацію по записах за допомогою відповідних кнопок роботи з базою даних і навігації по ній. Для редагування складу поїзда слід натиснути кнопку «Склад поїзда» на даній формі. Після цього з'явиться форма «Формування складу поїзда», на якій у верхній таблиці відображений список всіх вагонів, а в нижній склад обраного поїзда. Для додавання вагона до складу або його видалення зі складу слід скористатися відповідною кнопкою зі стрілкою.

Для формування шляху прямування поїзда і складання розкладу слід натиснути кнопку «Розклад і склади» на формі «АРМ оператора». Після цього на екрані з'явиться форма «Потяги», яка допускає додавання, видалення і редагування списку поїздів, а також навігацію по записах за допомогою відповідних кнопок роботи з базою даних і навігації по ній, забезпечених спливаючими підказ-ками. Для редагування шляху прямування поїзда слід натиснути кнопку «Шлях і розклад» на даній формі. Після цього з'явиться форма «Формування шляху прямування поїзда і розкладу», на якій в лівій таблиці відображений список всіх станцій, а в правій станції на шляху прямування обраного поїзду. Для додавання станції в маршрут або її видалення з шляху прямування поїзда слід скористатися відповідною кнопкою зі стрілкою. Для кожної станції проходження задається розклад, коли заданий поїзд прибуває і відбуває на дану станцію.

Для обліку проданих квитків слід натиснути кнопку «продані квитки» на формі «АРМ оператора». Після цього на екрані з'явиться одноіменна форма, яка припускає перегляд списку проданих квитків, фільтрацію і сортування цього

списку, друк, а також навігацію по записах за допомогою відповідних кнопок роботи з базою даних.

Для запуску АРМ касира натисніть відповідну кнопку на головній формі. У вікні «АРМ касира» можна виконати наступне:

– отримання інформації про склади, що мають задані станцію відправлення та призначення;

– продаж квитка;

– друк проїзного документа (квитка).

Для отримання інформації про склади слід натиснути кнопку «Інформація про рейси» на формі «АРМ касира». Після цього на екрані з'явиться форма «АРМ касира - пошук поїзда», яка припускає пошук необхідного поїзда для заданих дат відправлення і прибуття та станцій відправлення і прибуття за допомогою відповідних кнопок роботи з базою даних. Для продажу квитка на обраний поїзд потрібно натиснути кнопку «Продаж квитка» на даній формі.

Для продажу квитка слід натиснути кнопку «Продаж квитка» на формі «АРМ касира» або формі «АРМ касира - пошук поїзда». Після цього на екрані з'явиться форма «Продаж квитка», що дозволяє продати квиток на обраний поїзд і роздрукувати проїзний документ.

Для запуску Довідкової натисніть відповідну кнопку на головній формі. У вікні «Довідкова - пошук поїзда» можна виконати наступне:

– отримання інформації про шлях прямування поїзда і квитках в наявності;

– отримання інформації про склад обраного поїзда по вагонах.

3.2 Розробка алгоритмів

Алгоритми роботи програми є стандартними алгоритмами роботи з базою даних. В основному всі алгоритми роботи пов’язані з введенням даних від користувача, перевірці введеної інформації на предмет порушення цілісності

даних і занесення введеної інформації в саму базу, якщо введені відомості не порушують цілісності.

3.2.1 Проектний розділ

Функціональне проектування системи

Функціональне проектування було проведено за допомогою CASE засоби Computer Associates BPwin 4.0.

Структурно-функціональний аналіз процесу продажу квитків був виконаний на основі методології IDEF0.

Контекстна діаграма приведена на рисунку 3.2.

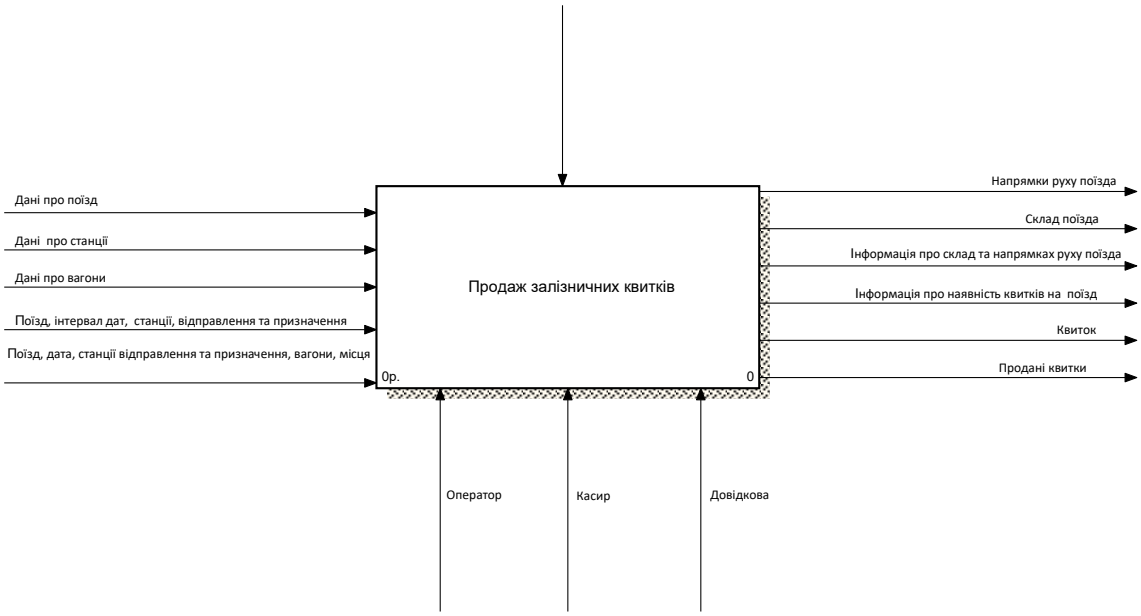


Рис. 3.1 Контекстна діаграма

Входом для системи є дані про потяг, станції, вагони, дані для отримання довідки (поїзд, діапазон дат, станції відправлення і призначення), дані для продажу квитка (поїзд, дата, станції відправлення і призначення, вагони, місця).

Виходом для системи є:

- звіт «Шлях проходження»;
- звіт «Склад поїзда»;
- інформація про склад і шлях проходження обраного поїзда (довідка);
- інформація про наявність квитків на обраний поїзд (довідка);

– квиток;

– звіт «продані квитки».

На цьому рівні виконуються наступні функції:

– формування шляху прямування поїзда;

– формування складу поїзда;

– видача довідкової інформації;

– продаж квитків;

– аналіз продажу квитків.

3.2.2 Інформаційне забезпечення комплексу завдань та використання класифікаторів та системи кодування

Класифікатор - це механізм (елемент моделі), що описує певні риси структури і поведінки системи. До класифікаторів відносяться класи, типи даних, інтерфейси, підсистеми. Найбільш загальними класифікаторами є класи. Всі інші класифікатори визначаються щодо їх схожості з класами, з урахуванням їх обмежень за змістом або використання. При цьому кожен вид класифікатора представлений в метамоделі своїм власним класом. Велика частина властивостей класу є і у класифікаторів, проте кожен з них має свої обмеження.

В процесі проектування завдання були використані класифікатори, інформація яких об'єктивно відображає існуючі відносини між об'єктами і забезпечує порівняння показників за якісними і кількісними ознаками. У задачі використовуються класифікатори, перелік і опис основних преставлені в таблиці 3.1

Таблиця 3.1

Методи класифікації та кодування реквізитів

Найменування реквізиту	Довжина кода в знаках	Система класифікації	Метод кодування	Структура кода
№ проїзного документу (квитка)	6	Ієрархічна	Порядковий	XXXXXX довільний номер
№ поїзду	7	Змішана	Структуро-ваний	XXX_AA довільний номер, довільні букви
№ вагона	6	Змішана	Структуро-ваний	XXXXXA довільний номер + довільна буква

3.2.3 Характеристика нормативно-довідкової і вхідної оперативної інформації

Вхідними даними для програми є дані, які вводить користувач:

-довідкові (вагони, потяги, склади, квитки, розклад, шлях слідування);

- поточні (дані про продаж квитка, зміни складу поїзда, розкладу або шляху прямування).

У проектованій системі вирішуються завдання ,реалізовані відповідним програмним модулем, кожен з яких має можливість виконання ряду операцій, таких ,як коригування інформації, формування звітів, побудова діаграми за даними з таблиці. Процеси заповнення, коригування реалізовані однотипним способом у всіх модулях. При додаванні і редагуванні даних застосовується контроль правильності введення користувачем інформації, шляхом її порівняння з необхідним типом, діапазоном зміни можливих значень, маскою введення і т.д.

Для додавання і редагування даних використані екранні форми з усіма необхідними полями введення, а також поясненнями і керуючими елементами, призначеними для вироблення управляючих впливів (збереження, скасування змін) і навігації (переміщенню) по БД. У кожному з програмних модулів системи

при необхідності є можливість для формування звітності та діаграми. Всі звіти генеруються автоматично, використовуючи вибірки інформації з БД.

3.2.4 Характеристика бази даних

Характеристика інфологічної моделі БД

У системі, можна виділити наступні сутності: класи вагону, типи вагону, категорії вагону, станція, вагон, потяг, склад, шлях слідування, розклад, місце, квиток, місця, квитки.

Нормалізація передбачає визначення необхідних атрибутів з подальшим створенням з них нормалізованих таблиць, заснованих на функціональних залежностях між цими атрибутами. Ставлення, в якому на перетині кожного рядка і кожного стовпця міститься єдине значення, знаходиться в 1-й нормальній формі. При цьому необхідно, щоб відношення мало первинний ключ. Друга нормальна форма застосовується до відносин зі складовими ключами, тобто до таких відносин, первинний ключ яких складається з двох або більше атрибутів. Ставлення з первинним ключем на основі єдиного атрибута завжди знаходиться в 2-й нормальній формі. Ставлення, яке знаходиться в 1-й нормальній формі і кожен атрибут , який не входить до складу первинного ключа, залежить тільки від повного значення ключа і не залежить від окремого атрибута, що входить до складу первинного ключа, має другу нормальну форму (кожен неключовий атрибут функціонально повно залежить від ключа).

Відношення знаходиться в 3-й нормальній формі, якщо воно представлене в 2-й нормальній формі і не має що атрибутів, які перебували б у транзитивній функціональній залежності від цього первинного ключа.

Розроблена модель знаходиться в 3-й нормальній формі, так як:

– атрибути сутностей є атомарними;

– кожен неключовий атрибут функціонально повно залежить від первинного ключа;

– в моделі відсутні транзитивні залежності неключових атрибутів від ключа.

3.2.5 Характеристика даталогічної моделі БД

Фізичне проектування вироблено з урахуванням всіх особливостей обраної СУБД. В якості СУБД обраний MS SQL Server 2019.

Дані в базі повинні мати властивість цілісності. Під цілісністю розуміється правильність даних в будь-який момент часу. Підтримка цілісності бази даних може розглядатися як захист даних від невірних змін або руйнування (це питання не відноситься до незаконних змін і руйнувань, які є проблемою безпеки).

Виділяють три групи правил цілісності:

1) Цілісність сутностей. Об'єкту або сутності реального світу в реляційних базах даних відповідають кортежі відносин. Вимога полягає в тому, щоб будь-який кортеж будь-якого відношення відрізнити від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення має володіти первинним ключем. Ця вимога автоматично задовольняється, якщо в системі не порушуються базові властивості відносин. Первинний ключ - це мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності;

2) Цілісність по посиланнях. База даних не повинна містити неузгоджених значень зовнішніх ключів. Правило стверджує, що якщо В посилається на А, тоді А має існувати. Кажуть, що ставлення, в якому визначено зовнішній ключ, посилається на відповідне відношення, в якому такий же атрибут є первинним ключем. Вимога цілісності по посиланнях, або вимога зовнішнього ключа полягає в тому, що для кожного значення зовнішнього ключа, що з'являється в посиланнях , повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (тобто ні на що не вказувати);

3) Цілісність, що визначається користувачем. У користувача (або у розробника) бази даних повинна бути можливість визначити, які операції повинні бути заборонені, а які дозволені, чи потрібні для дозволених операцій компенсуючі, і якщо так, то які (тобто можливість каскадного видалення).

У структурі бази даних враховані основні правила цілісності. Кожна сутність ідентифікується унікальним ключем, і розроблена система зовнішніх ключів. База даних не містить неузгоджених значень зовнішніх ключів, тобто при роботі з записами відбувається каскадне оновлення пов'язаних полів і каскадне видалення зв'язаних записів;

4) Цілісність, що визначається користувачем, підтримується обмеженнями в таблицях бази даних на введення невід'ємних значень, а також забезпеченням вибору значень зовнішніх ключів зі списків без дозволу варіантів введення неприпустимого значення.

Таким чином розробляється інформаційна система,що підтримує забезпечення цілісності даних. Зокрема, при вставці або додаванні записів з повторюваними значеннями первинного ключа програма видасть попередження і не дозволить операції вставки здійснитися. Крім того, програма забезпечує роботу зі зв'язаними полями. Наприклад, якщо дані видаляються з основної таблиці, то програма видалить відповідні записи і в зв'язаних таблицях. Те ж саме при оновленні або вставці даних. Тобто, програма не дозволяє існувати в базі непов'язаним даними, які «засмічують» базу.

3.2.6 Характеристика результатної інформації

Кожна сутність і кожне відношення трансформується в таблицю. Імена сутностей і відносин стають іменами таблиць, а імена атрибутів - іменами колонок. Опис таблиць бази даних приведено в таблиці 3.2.

Таблиця 3.2

Опис таблиць бази даних

Найменування таблиці	Поле	Тип	Первинний ключ	Зовнішній ключ
Билет	ДатаПрибID	Long Integer	Ні	Так
	Стоимость	Currency	Ні	Ні
	ФИО	Text(30)	Ні	Ні
	ДатаОтпрID	Long Integer	Ні	Так
	NБилета	Long Integer	Так	Ні
Вагон	ТипID	Long Integer	Ні	Так

	КатегорияID	Long Integer	Ні	Ні
	СтанцияВладелецID	Long Integer	Ні	Так
	КлассID	Long Integer	Ні	Так

Продовження Таблиці 3.2

	NВагона	Text(6)	Так	Ні
КатегорииВагона	НазваниеКатегории	Text(15)	Ні	Ні
	ID	Long Integer	Так	Ні
КлассыВагона	НазваниеКласса	Text(4)	Ні	Ні
	ID	Long Integer	Так	Ні
МестаБилета	NБилета	Long Integer	Ні	Так
	МестоID	Long Integer	Ні	Так
	ID	Long Integer	Так	Ні
Место	NМеста	Byte	Ні	Ні
	ТипМеста	Text(4)	Ні	Ні
	NВагонаМесто	Text(6)	Так	Так
	ID	Long Integer	Так	Ні
Поезд	ДатаНачалаСледования	Date/Time	Ні	Ні
	ДатаОкончанияСледования	Date/Time	Ні	Ні
	НазваниеПоезда	Text(30)	Ні	Ні
	NПоезда	Text(6)	Так	Ні
ПутьСледования	СтанцияID	Long Integer	Ні	Так
	ВремяПрибытия	Date/Time	Ні	Ні
	ВремяОтбытия	Date/Time	Ні	Ні
	NПоездаПуть	Text(6)	Ні	Так
	ID	Long Integer	Так	Ні
Расписание	ПутьID	Long Integer	Ні	Так
	Дата	Date/Time	Ні	Ні
	ID	Long Integer	Так	Ні
Состав	НомерВСоставе	Byte	Ні	Ні
	ОтСтанцииID	Long Integer	Ні	Так
	ДоСтанцииID	Long Integer	Ні	Так
	ID	Long Integer	Так	Ні

	NВагонаСостав	Text(6)	Ні	Так
	NПоездаСостав	Text(6)	Ні	Так

Продовження Таблиці 3.2

Станция	Страна	Text(15)	Ні	Ні
	Область	Text(30)	Ні	Ні
	НазваниеСтанции	Text(30)	Ні	Ні
	ID	Long Integer	Так	Ні
ТипыВагона	НазваниеТипа	Text(20)	Ні	Ні
	ID	Long Integer	Так	Ні

Кількість екранних форм і їх зміст спроектовано так, щоб забезпечити можливість виведення всієї результатної інформації, а також результатів запитів користувача до бази даних. Передбачені видача результатів як на екран ПЕОМ, так і на принтер за бажанням користувача. Вся результатна інформація проекту автоматизації представляється у вигляді звітів і екранних форм. Нижче розглянуті деталі кожного з результатних документів.

Вихідними документами, що розробляється є:

- звіт «Шлях проходження»;
- звіт «Склад поїзда»;
- квиток;
- звіт «Продані квитки».

Екранні форми розробленої системи наведені в додатку А. Форми вихідних документів наведені в додатку Б.

3.2.7 Програмне забезпечення комплексу задач

Процес обробки даних передбачається здійснювати в діалоговому режимі. Даний режим обробки найбільш зрозумілий користувачеві, а значить підвищує ефективність роботи з програмою.

Інтерфейс комплексу завдань вирішено побудувати на основі розподілу за операціями обробки. Необхідно вибудувати ланцюжок взаємозв'язку екранних

форм. Для цього потрібно визначити послідовність переходу від однієї форми до іншої.

3.2.8 Структурна схема пакета

Структура пакета прикладних програм наступна:

1) Керуючий модуль, який призначений для перетворення програми на вхідній мові (ПВМ) в послідовність викликів обробних модулів формування меню користувача і забезпечення з'єднання пунктів меню з обробними і обслуговуючими модулями;

2) Обробні і одночасно обслуговуючі модулі, які призначені для взаємодії користувача з пакетом керуючих модулів, а також для реалізації алгоритму вирішення задач, тобто для перетворення даних, що становлять інформаційну базу пакетів.

3) Обробні модулі, для реалізації алгоритму вирішення задач, тобто для перетворення даних, що становлять інформаційну базу пакетів.

Структура пакета прикладних програм представлена на рисунку 3.2

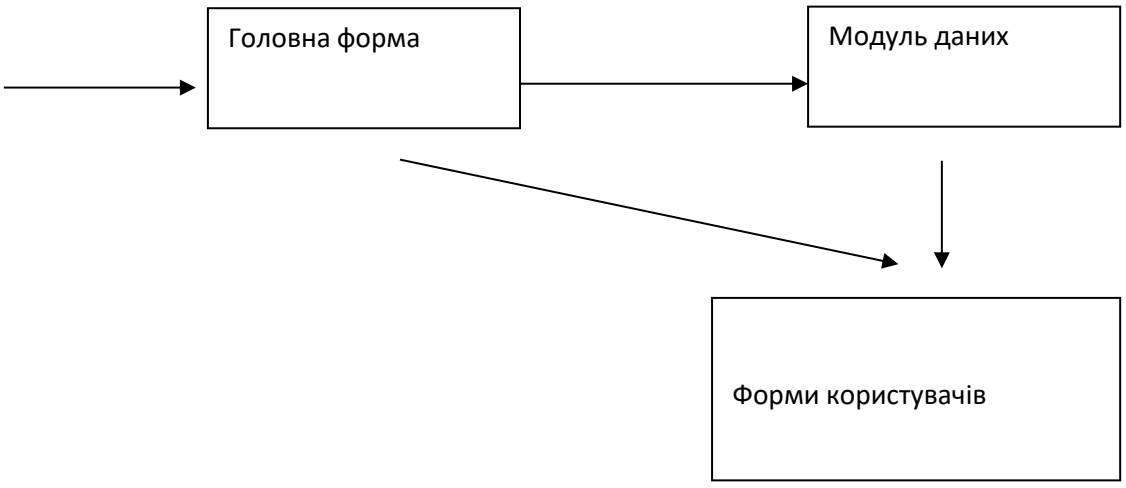


Рисунок 3.2 Структура пакета прикладних програм

Технологія внутрімашинної організації задається послідовністю реалізованих процедур схем взаємозв'язку програмних модулів і інформаційних масивів. Така схема являє собою декомпозицію загального процесу вирішення

задачі на окремі процедури перетворення масивів, іменованими модулями (це введення, контроль, перезапис інформації з одного носія на інший, сортування, ущільнення даних, редагування, накопичення, висновок на друк і т.п.) . Структуру програм можна описати основними блоками, представленими на рисунку 3.3

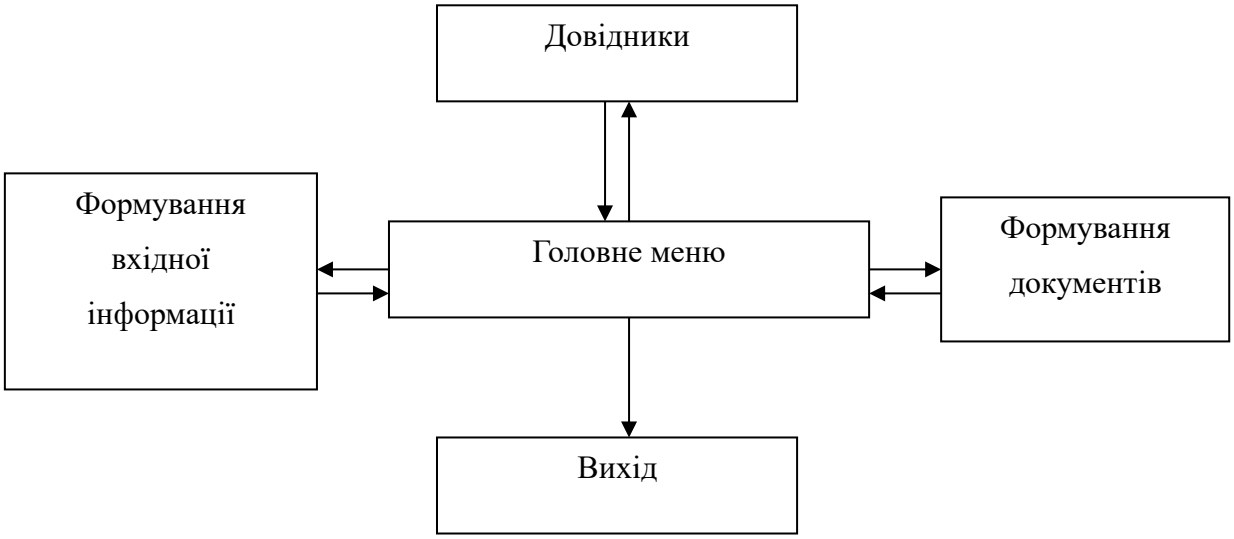


Рис. 3.3 Схема основних модулів програми

Модуль «Головне меню» призначений для запуску основних процедур програми і завершення роботи з програмою.

Модуль роботи з довідниками включає в себе довідники:

- категорії вагонів;
- типи вагонів;
- класи місць у вагонах;
- вагони;
- поїзди;
- станції.

Призначення даного модуля є пошук і підготовка довідкової інформації.

Модуль «Формування вхідної інформації» призначений для введення первинних даних та перегляду раніше занесених. Даний модуль реалізує завдання продажу квитків і видачі довідкової інформації за допомогою спеціальних форм.

Модуль «Формування документів» виконує функції формування друкованих форм. У модулі зберігаються шаблони для друку документів, таких як квиток, інформація про шляху прямування поїзда і т.д. Звіти формуються використовуючи запити, які обробляють вихідну інформацію відповідно до заданих параметрів користувача.

3.2.9 Технологічне забезпечення комплексу задач

Збір інформації здійснюється наступним чином: довідкова інформація вноситься відразу після впровадження проекту і в майбутньому заноситься в міру надходження відомостей. Оперативна інформація вноситься в базу даних у міру надходження відповідних документів і відомостей: дані про продані квитки при продажу квитка, шлях прямування поїзда при формуванні шляху і т.д.
Після запуску програми користувач натискає на головній кнопкової формі кнопку, що викликає необхідну йому в даний момент екранну форму, на якій він вводить оперативні або довідкові дані, або формує необхідні звіти. Після введення даних або формування звіту, користувач може завершити роботу з програмою, або відкрити іншу екранну форму для подальшої роботи.

3.3 Тестування програмного забезпечення

Запускаючи програмний додаток, на екрані з’являється головне меню програми. У головному меню можна побачити чотири кнопки. Такі як “АРМ касира”, “АРМ оператора”, “Довідкова”. Також наявна кнопка “Вихід”, що забезпечує безпечний вихід з програми.

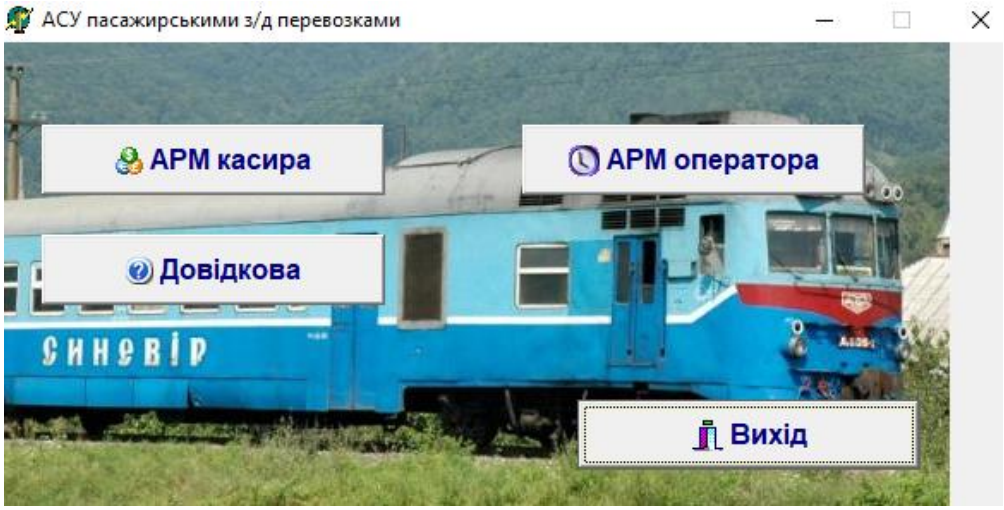


Рис 3.4. Головне меню програми

Натиснувши кнопку “АРМ Касира”, запускається меню, у якому користувач може дізнатися інформацію про рейси, вільні місця, а також продати квиток.

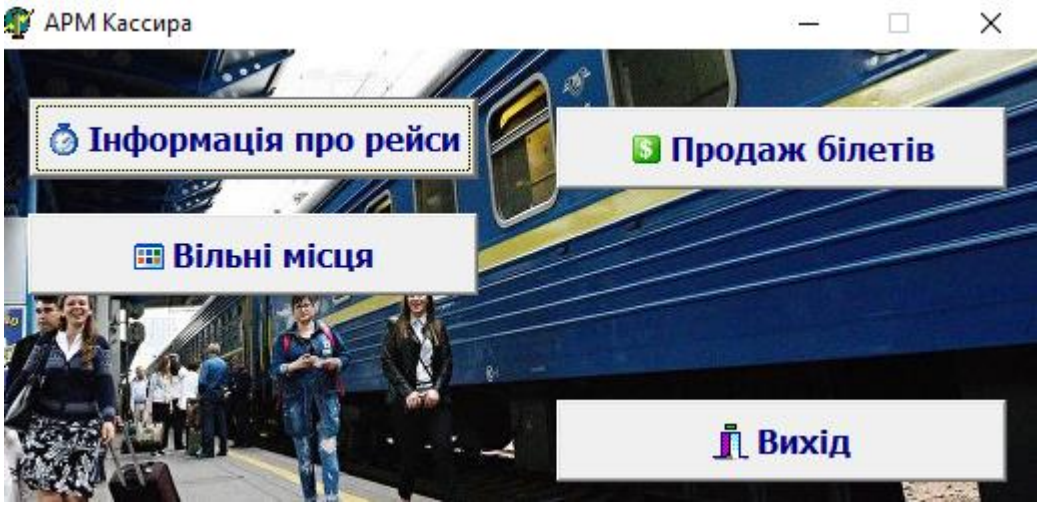


Рис. 3.5 Меню “АРМ Касира”

Натиснувши кнопку “Інформація про рейси”, перед користувачем з’являється вікно, у якому він може здійснити пошук потрібного рейсу. Для цього йому необхідно ввести станцію відправлення потягу та станцію прибуття. Також у цьому вікні доступні функції, завдяки яким можна дізнатися інформацію про потяг, наявність квитків.

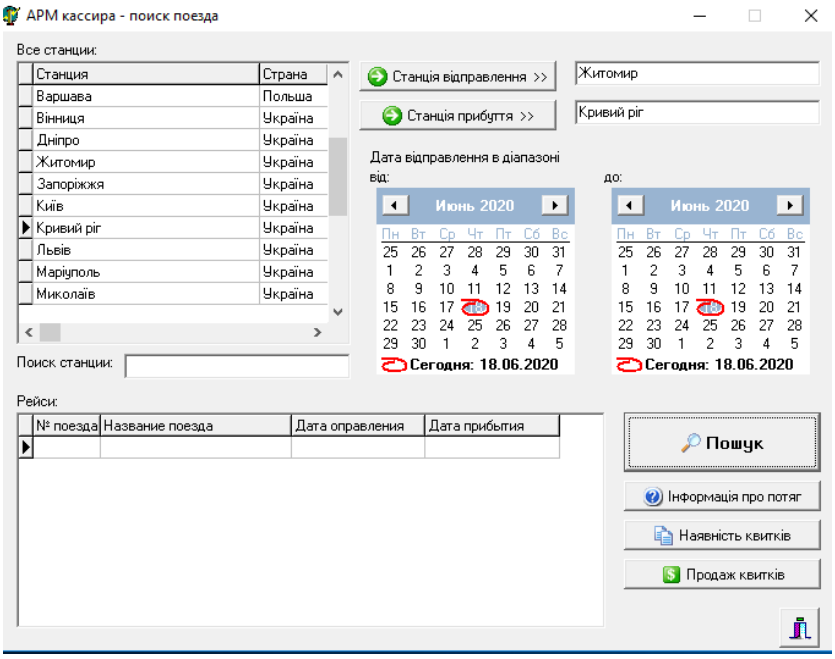


Рис. 3.6 Вікно “Інформація про рейси”

У меню “АРМ оператора” користувач може дізнатися інформацію про станції, склади потягів, шляхи слідування, продані квитки, вагони і місця, ознайомитися з розкладом.

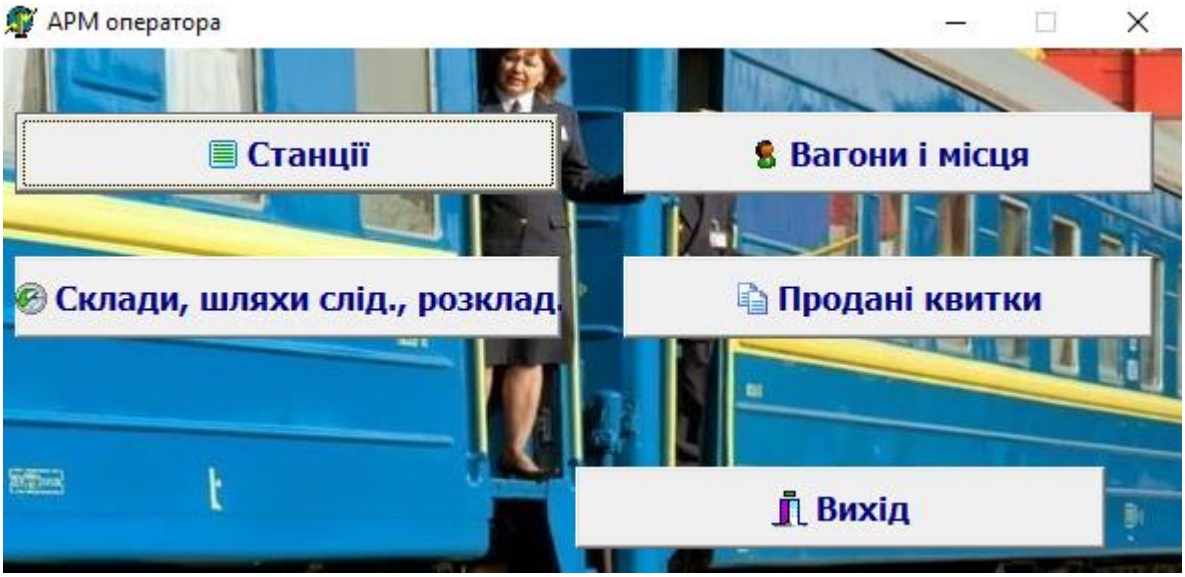


Рис. 3.7 Меню “АРМ оператора”

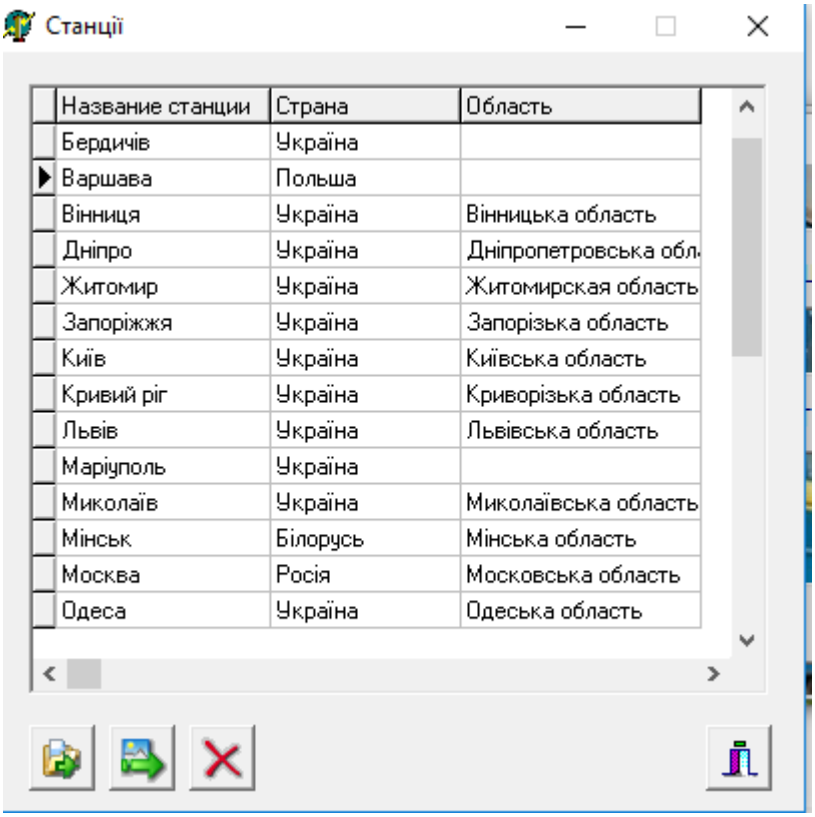


Рис. 3.8 Вікно “Станції”

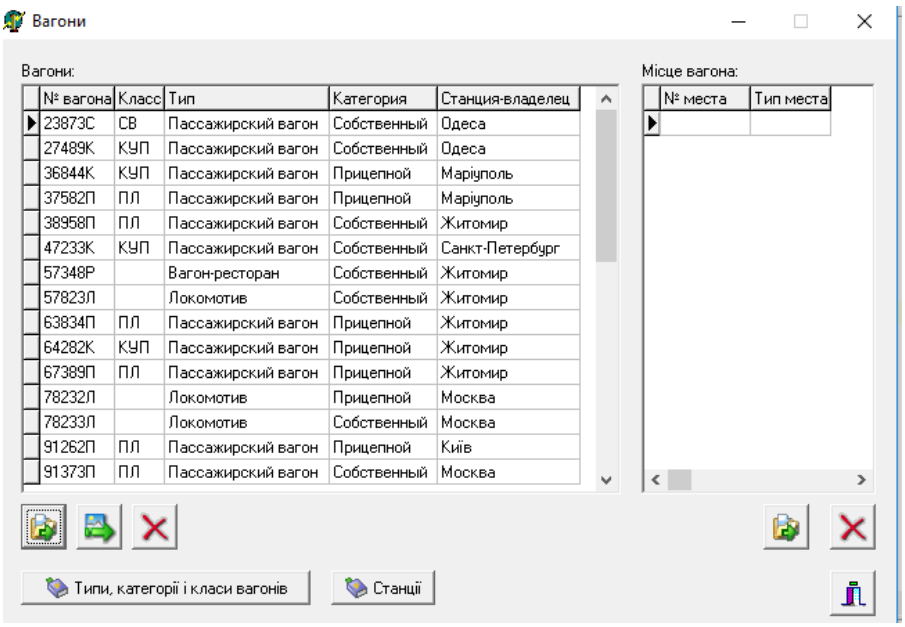


Рис. 3.9 Вікно “Вагони і місця”

Також користувач за необхідністю може змінити інформацію.

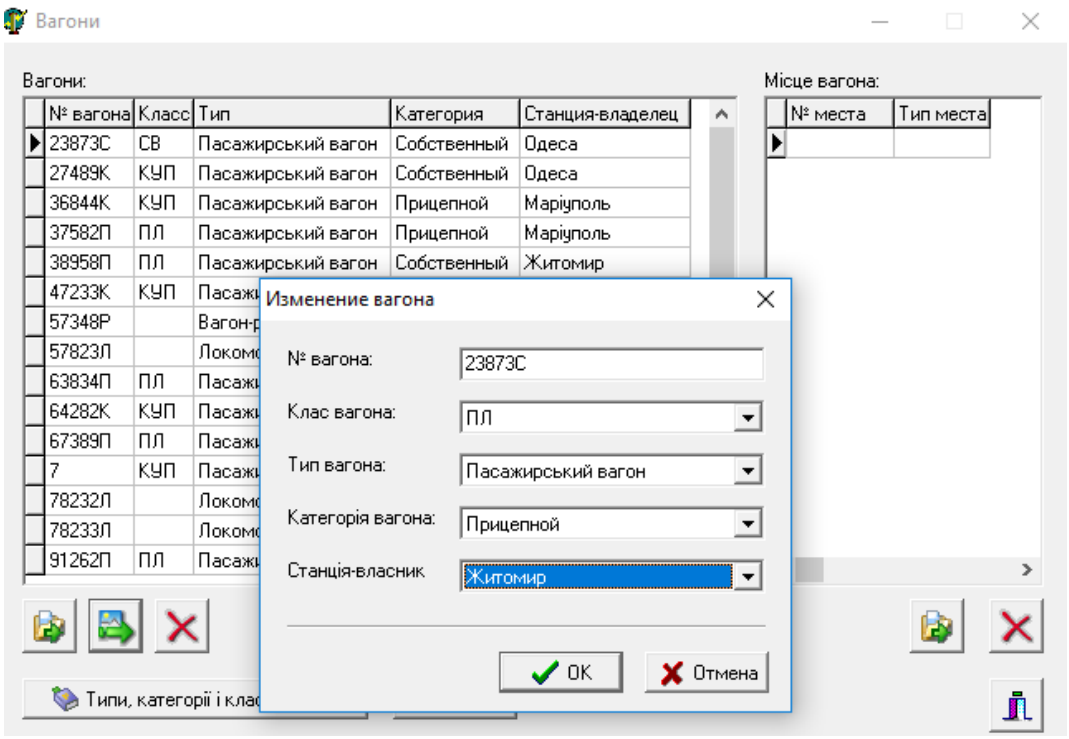


Рис. 3.10 Вікно “Зміна вагону”

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі пояснювальної записки сформульовано алгоритм дій користувача для здійснення операцій у програмному забезпеченні, наведено опис процесу функціонального проектування системи та наведена характеристика даталогічної моделі БД.

Також у цьому розділі здійснено тестування програмного забезпечення.

ВИСНОВКИ

У даній бакалаврській кваліфікаційні роботі було описано процес проектування інформаційної системи для моніторингу роботи залізничного транспорту.

Дана система дозволяє керувати розкладом залізничного транспорту, здійснити оформлення білетів, переглядати і вносити зміни до бази даних поїздів, вагонів, проданих білетів, формувати склад потягу. Також програмний додаток дозволяє створити і оформити квиток на заданий користувачем маршрут, оформити документ з інформацією про склад потягу по вагонах, шлях прямування потягу, продані квитки для заданого періоду.

Для реалізації програмної частини було використано мову програмування Delphi, декларативну мову програмування для взаємодії користувача з базами даних SQL.

Результатом атестаційної бакалаврської роботи є готове програмне рішення, яке можна застосовувати у сфері роботи залізничного вокзалу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Агальцов, В.П. Бази даних. У 2-х т. Т. 2. Розподілені і віддалені бази даних: Підручник / В.П. Агальцов. - М .: ИД ФОРУМ, НДЦ ИНФРА-М, 2013. - 272 с.

2. Агальцов, В.П. Бази даних. У 2-х т.т. 1. Локальні бази даних: Підручник / В.П. Агальцов. - М .: ИД ФОРУМ, НДЦ ИНФРА-М, 2013. - 352 с.

3. Голіцина, О.Л. Бази даних / О.Л. Голіцина, Н.В. Максимов, І.І. Попов. - М .: Форум, 2004. - 352 с.

4. Голіцина, О.Л. Бази даних: Навчальний посібник / О.Л. Голіцина, Н.В. Максимов, І.І. Попов. - М .: Форум, 2012. - 400 с.

5. Карпова, І.П. Бази даних: Навчальний посібник / І.П. Карпова. - СПб .: Пітер, 2013. - 240 с.

6. Пасічник В.В. Організація баз даних та знань: Підручник. / Пасічник В.В.,

Резніченко В.А. - К.: BHV, 2006. - 384с.

7. Ковальчук А.М. Принципи проектування баз даних: Навчальний посібник. /

[Ковальчук А.М., Левицький В.Г. та ін.]- Ж.: ЖДТУ, 2009. - 123с.

8. Методичні вказівки до виконання лабораторних робіт з нормативної дисципліни "Організація баз даних та знань"/ [Упоряд. Л.Г. Ахметшина.] -

Дніпропетровськ: НГА, 2001. - 80с.

9. Ицик Бен-Ган. Microsoft SQL Server 2008. Основы T-SQL/ Ицик Бен-Ган; пер.

с англ. — СПб.: БХВ-Петербург, 2009. — 432 е.: ил.

10. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение.

Теория и практика. / Коннолли Т., Бегг К., пер. с англ. – [3-е издание. :] —

М. : Издательский дом "Вильяме", 2003. — 1440 с.

11. Дейт К. Дж. Введение в системы баз данных / Дейт К. Дж.; пер. с англ.- [8-е издание] — М.: Издательский дом "Вильяме", 2005. — 1328 с: ил.

12. Глушаков, С.В. Бази даних / С.В. Глушаков, Д.В. Ломотько. - М .: Харків: Фоліо, 2000. - 504 с.

13. Дейт, К.Дж. Введення в системи баз даних / К.Дж. Дейт. - К .: Діалектика; Видання 6-е, 2004. - 784 с.

14. Ємельянов, Н.Є. Введення в СУБД ІНЕС / Н.Є. Ємельянов. - М .: Наука, 2012. - 256 с.

15. Мюллер, Р.Дж. Бази даних та UML. Проектування / Р.Дж. Мюллер. - М .: ЛОРИ, 2002. - 420 с.

16. Олле, Т.В. Пропозиції Кодас з управління базами даних / Т.В. Олле. - М .: Фінанси і статистика, 2013. - 286 с.

17. Постол Visual Studio .NET: розробка додатків баз даних / Постол, Анатолій. - М .: СПб: БХВ, 2003. - 544 с.

18. Розширене адміністрування та налаштування оптимальної продуктивності баз даних Oracle 9i. - М .: СПб: ЦКТіП Газпром, 2005. - 200 с.

19. Редько, В. Н. Бази даних та інформаційні системи / В.М. Редько, І.А. Бассараб. - М .: Знание, 2004. - 240 с.

20. Ульман, Дж. Основи систем баз даних / Дж. Ульман. - М .: Фінанси і статистика, 2017. - 292 с.

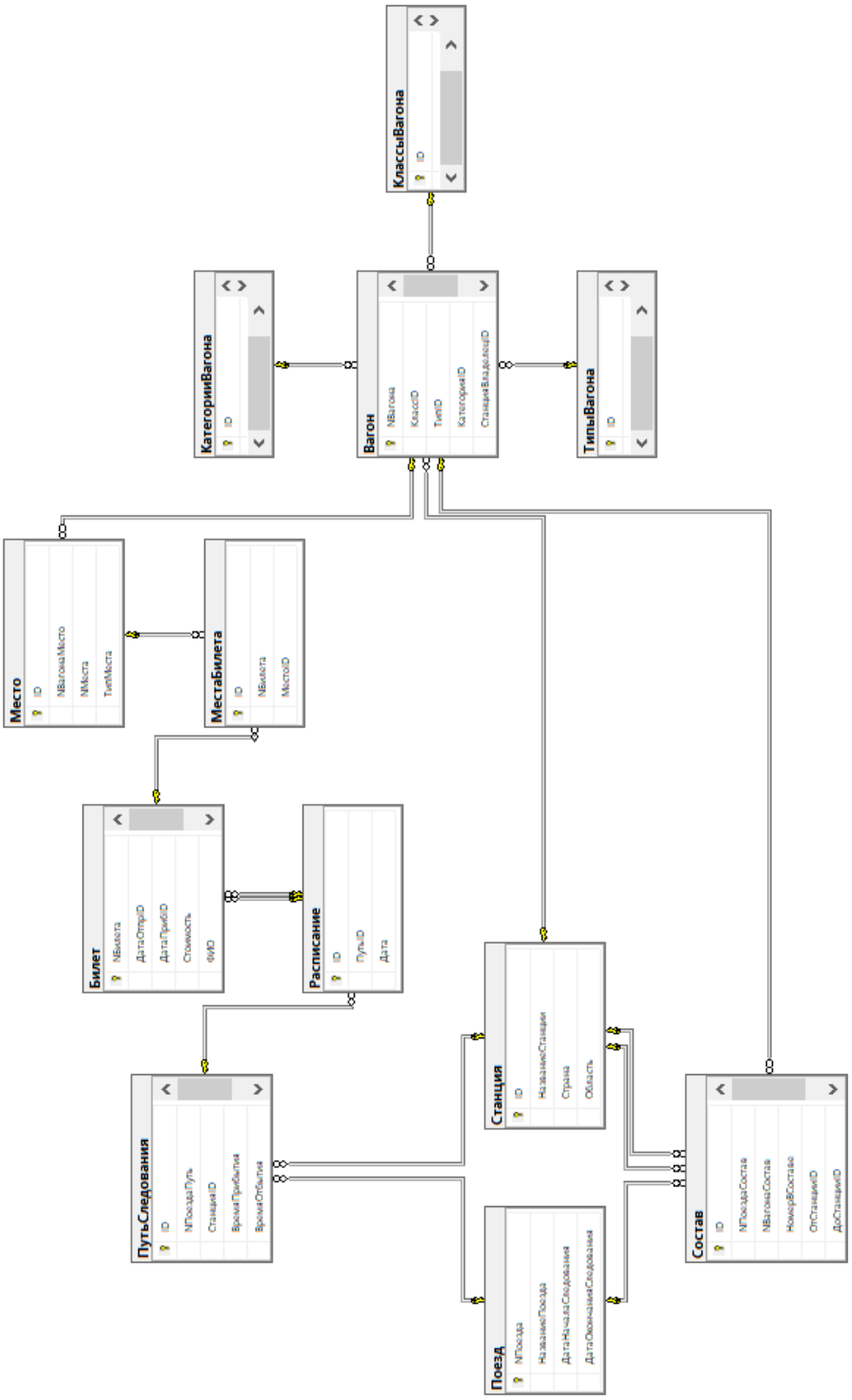
21. Хендерсон Керівництво розробника баз даних в Delphi 2 / Хендерсон, Кен. - М .: Діалектика, 2012. - 544 с.

22. системи, Відкриті Відкриті системи. СУБД №04 / 2011 / Відкриті системи. - М .: Відкриті Системи, 2011. - 383 с

ДОДАТКИ

ДОДАТОК А

SQL Server Diagram



ДОДАТОК В

Код програми

uStation.pas

```
unit uStation;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, StdCtrls, Buttons;

type
  TfmStation = class(TForm)
    DBGrid1: TDBGrid;
    bbtEdit: TBitBtn;
    bbtDel: TBitBtn;
    bbtIns: TBitBtn;
    BitBtn2: TBitBtn;
    procedure bbtInsClick(Sender: TObject);
    procedure bbtEditClick(Sender: TObject);
    procedure bbtDelClick(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fmStation: TfmStation;

implementation

uses uDM, uStationEditIns;

{$R *.dfm}

procedure TfmStation.bbtInsClick(Sender: TObject);
begin
  fmStationEditIns.Caption := 'Äîäàâëåíèå ñòàíöèè';
```

```
  fmStationEditIns.edName.Clear;
  fmStationEditIns.edCountry.Clear;
  fmStationEditIns.edRegion.Clear;
  fmStationEditIns.ShowModal;
end;

procedure TfmStation.bbtEditClick(Sender: TObject);
begin
  fmStationEditIns.Caption := 'Èçìåíåíèå ñòàíöèè';
  fmStationEditIns.edName.Text := DM.qrStation.FieldByName('ÍàçâàíèåÑòàíöèè').AsString;
  fmStationEditIns.edCountry.Text := DM.qrStation.FieldByName('Ñòðàíà').AsString;
  fmStationEditIns.edRegion.Text := DM.qrStation.FieldByName('Îáëàñòü').AsString;
  fmStationEditIns.ShowModal;
end;

procedure TfmStation.bbtDelClick(Sender: TObject);
begin
  DM.spStationDel.Parameters.ParamByName('@ID').Value := DM.qrStation.FieldByName('ID').Value;
  DM.spStationDel.ExecProc;
  DM.qrStation.Requery();
end;

procedure TfmStation.BitBtn2Click(Sender: TObject);
begin
  Close;
end;

procedure TfmStation.FormCreate(Sender: TObject);
begin

end;

end.
```

uCarriage.pas

```
unit uCarriage;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
  Dialogs, Grids, DBGrids, StdCtrls, Buttons;

type
  TfmCarriage = class(TForm)
    DBGrid1: TDBGrid;
    bbtClassTypeCateg: TBitBtn;
    Label1: TLabel;
    DBGrid2: TDBGrid;
    Label2: TLabel;
    bbtStation: TBitBtn;
    bbtEdit: TBitBtn;
    bbtDel: TBitBtn;
    bbtIns: TBitBtn;
    BitBtn2: TBitBtn;
    bbtInsSeat: TBitBtn;
    bbtDelSeat: TBitBtn;
    procedure bbtInsCarriageClick(Sender: TObject);
    procedure bbtEditCarriageClick(Sender: TObject);
    procedure bbtDelCarriageClick(Sender: TObject);
    procedure bbtClassTypeCategClick(Sender: TObject);
    procedure bbtInsSeatClick(Sender: TObject);
    procedure bbtDelSeatClick(Sender: TObject);
    procedure bbtStationClick(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fmCarriage: TfmCarriage;

implementation

uses uDM, uCarriageEditIns, DB, uCarriageTypeCateg, uSeatEditIns, uStation;

{$R *.dfm}

procedure TfmCarriage.bbtInsCarriageClick(Sender: TObject);
begin
  fmCarriageEditIns.Caption := 'Äîáàâëåíèå âàãîíà';
```

```
  fmCarriageEditIns.edNCarriage.Clear;
  fmCarriageEditIns.dblcbClass.KeyValue := 0;
  fmCarriageEditIns.dblcbType.KeyValue := 0;
  fmCarriageEditIns.dblcbCateg.KeyValue := 0;
  fmCarriageEditIns.dblcbStation.KeyValue := 0;
  fmCarriageEditIns.ShowModal;
end;

procedure TfmCarriage.bbtEditCarriageClick(Sender: TObject);
begin
  fmCarriageEditIns.Caption := 'Èçìåíåíèå âàãîíà';
  fmCarriageEditIns.edNCarriage.Text := DM.qrCarriage.FieldByName('NÂàãîíà').AsString;
  fmCarriageEditIns.dblcbClass.KeyValue := DM.qrCarriage.FieldByName('ÊëàññID').AsInteger;
  fmCarriageEditIns.dblcbType.KeyValue := DM.qrCarriage.FieldByName('ÒèïID').AsInteger;
  fmCarriageEditIns.dblcbCateg.KeyValue := DM.qrCarriage.FieldByName('ÊàòåãîðèÿID').AsInteger;
  fmCarriageEditIns.dblcbStation.KeyValue := DM.qrCarriage.FieldByName('ÑòàíöèÿÂëàäåëåöID').AsInteger;
  fmCarriageEditIns.ShowModal;
end;

procedure TfmCarriage.bbtDelCarriageClick(Sender: TObject);
begin
  DM.spCarriageDel.Parameters.ParamByName('@NÂàãîíà').Value :=
DM.qrCarriage.FieldByName('NÂàãîíà').Value;
  DM.spCarriageDel.ExecProc;
  DM.qrCarriage.Requery();
end;

procedure TfmCarriage.bbtClassTypeCategClick(Sender: TObject);
begin
  fmCarriageTypeCateg.ShowModal;
end;

procedure TfmCarriage.bbtInsSeatClick(Sender: TObject);
begin
  fmSeatEditIns.edNCarriage.Text := DM.qrCarriage.FieldByName('NÂàãîíà').AsString;
  fmSeatEditIns.edNSeat.Clear;
  fmSeatEditIns.ShowModal;
end;

procedure TfmCarriage.bbtDelSeatClick(Sender: TObject);
begin
  DM.spSeatDel.Parameters.ParamByName('@ID').Value := DM.qrSeat.FieldByName('ID').Value;
```

```
  DM.spSeatDel.ExecProc;
  DM.qrSeat.Requery();
end;

procedure TfmCarriage.bbtStationClick(Sender: TObject);
begin
  fmStation.ShowModal;
end;

procedure TfmCarriage.BitBtn2Click(Sender: TObject);
begin
  Close;
end;

end.
```

uConnect.pas

```
unit uConnect;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TfmConnect = class(TForm)
    Label1: TLabel;
    edServer: TEdit;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    edUser: TEdit;
    Label2: TLabel;
    edPass: TEdit;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
    procedure Connect;
  end;

var
  fmConnect: TfmConnect;

implementation

uses uDM, IniFiles;
var
  CancelConnect: Boolean;
  ServerName, UserID, Password: string;
  PersistSec: Boolean;
{$R *.dfm}

procedure TfmConnect.Connect;
var
  IniFile: TIniFile;
  APchar: array[0..254] of Char;
begin
  if not Assigned(DM) then
    Application.CreateForm(TDM, DM);

  //Çàãðóçêà äàííûõ î ñåðâåðå áàçû äàííûõ íà óäàëåííîì êîìïüþòåðå èç
  //èíèöèàëèçàöèîííîãî ôàéëà, åñëè îí ñóùåñòâóåò:
  ServerName := '(local)';
  GetWindowsDirectory(APchar, 255);
  if FileExists(string(APchar)+'\Station.ini') then
  begin
    IniFile := TIniFile.Create(string(APchar)+'\Station.ini');
    try
      PersistSec := IniFile.ReadBool('Base', 'PersistSec', False);
      ServerName := IniFile.ReadString('Base', 'ServerName', '(local)');
      UserID := IniFile.ReadString('Base', 'UserID', 'sa');
      Password := IniFile.ReadString('Base', 'Password', '');
    finally
      IniFile.Free;
    end;
  end;
```

```
 CancelConnect := False;
 if not Assigned(fmConnect) then
   Application.CreateForm(TfmConnect, fmConnect);

 while (not DM.ADOConnection1.Connected) and (not CancelConnect) do
 begin
   //Çàäàíèå ïàðàìåòðîâ ïîäêëþ÷åíèÿ ê ñåðâåðó ÁÄ:
   DM.ADOConnection1.ConnectionString :=
     'Provider=SQLOLEDB.1;Persist Security Info=' + BoolToStr(PersistSec, True) + ';User ID=' +
     UserID + ';Initial Catalog=Âîêçàë;Data Source=' + ServerName +
     ';Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;' +
     'Use Encryption for Data=False;Tag with column collation when possible=False';
   if PersistSec then
     DM.ADOConnection1.ConnectionString :=
       DM.ADOConnection1.ConnectionString + ';Password=' + Password
   else
     DM.ADOConnection1.ConnectionString :=
       DM.ADOConnection1.ConnectionString + ';Integrated Security=SSPI';

   //Ïîïûòêà ïîäêëþ÷åíèÿ ê ÁÄ:
   try
     DM.ADOConnection1.Open;
   except
     on E: Exception do
     begin
       Application.MessageBox(PChar('Îøèáêà ïîäêëþ÷åíèÿ ê áàçå äàííûõ!' + Chr(13) +
         'Ñòðîêà îøèáêè:' + Chr(13) + E.Message), 'Îøèáêà', MB_OK+MB_ICONERROR);
       edServer.Text := ServerName;
       RadioButton1.Checked := not PersistSec;
       RadioButton2.Checked := PersistSec;
       edUser.Text := UserID;
       edPass.Text := Password;
       fmConnect.ShowModal;
     end;
   end;
 end;
 if CancelConnect then
   Application.Terminate;
end;

procedure TfmConnect.BitBtn1Click(Sender: TObject);
```

```
 //Çàïèñü â ini-ôàéë èìåíè ñåðâåðà:
 procedure SaveToFile;
 var
  IniFile: TIniFile;
  APchar: array[0..254] of Char;
  f: File;
 begin
  GetWindowsDirectory(APchar, 255);

  if not FileExists(string(APchar)+'\Station.ini') then
  begin
   //Ñîçäàíèå ôàéëà, åñëè îí íå ñóùåñòâóåò:
   AssignFile(f, string(APchar)+'\Station.ini');
   Rewrite(f);
   CloseFile(f);
  end;

  IniFile := TIniFile.Create(string(APchar)+'\Station.ini');
  try
   IniFile.WriteString('Base', 'ServerName', ServerName);
   IniFile.WriteBool('Base', 'PersistSec', PersistSec);
   if PersistSec then
   begin
    IniFile.WriteString('Base', 'UserID', UserID);
    IniFile.WriteString('Base', 'Password', Password);
   end;
  finally
   IniFile.Free;
  end;
 end;

begin
 ServerName := edServer.Text;
 PersistSec := RadioButton2.Checked;
 UserID := edUser.Text;
 Password := edPass.Text;
 SaveToFile;
 Close;
end;

procedure TfmConnect.BitBtn2Click(Sender: TObject);
```

```
begin
  CancelConnect := True;
end;

end.
```

uMain.pas

```
unit uMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, jpeg;

type
  TfmMain = class(TForm)
    BitBtn1: TBitBtn;
    bbtSellerARM: TBitBtn;
    bbtOperARM: TBitBtn;
    bbtInfo: TBitBtn;
    Image1: TImage;
    procedure bbtInfoClick(Sender: TObject);
    procedure bbtSellerARMClick(Sender: TObject);
    procedure bbtOperARMClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fmMain: TfmMain;

implementation

uses uTrainSearch, uSellerARM, uOperARM, uConnect, uDM;

{$R *.dfm}

procedure TfmMain.bbtInfoClick(Sender: TObject);
```

```
begin
  fmTrainSearch.Caption := 'Ñïðàâî÷íàÿ - ïîèñê ïîåçäà';
  fmTrainSearch.bbtTicketCell.Visible := False;
  fmTrainSearch.ShowModal;
end;

procedure TfmMain.bbtSellerARMClick(Sender: TObject);
begin
  fmSellerARM.ShowModal;
end;

procedure TfmMain.bbtOperARMClick(Sender: TObject);
begin
  fmOperARM.ShowModal;
end;

procedure TfmMain.FormShow(Sender: TObject);
begin
  fmConnect.Connect; //Ïîäêëþ÷åíèå ê ÁÄ

  with DM do
  begin
    qrStation.Open;
    qrTrain.Open;
    qrPath.Open;
    qrTimetable.Open;
    qrCarriage.Open;
    qrStock.Open;
    qrStock1.Open;
    qrSeat.Open;
    qrTicket.Open;
    qrCarriageSeat.Open;
    qrCarriageType.Open;
    qrCarriageCateg.Open;
    qrCarriageClass.Open;
    qrTicketSeats.Open;
    qrAllTickets.Open;
  end;
end;
end.
```