ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра \_\_\_\_\_\_\_\_\_\_\_*Штучного інтелекту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**КУРСОВИЙ ПРОЕКТ З ДИСЦИПЛІНИ**

**Організація баз даних і знань**

База даних «Залізна дорога»

(тема проекту)

Студентки 1 курсу групи ІТКНу-18-1

Спеціальності 122 – Комп’ютерні науки

Щетініної А.В. .

Керівник доц. каф.ШІ, к.т.н. Магдаліна І.В.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_. Оцінка: ECTS\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_ Волощук О.Б.

\_\_\_\_\_\_\_\_\_\_ Магдаліна І.В.

\_\_\_\_\_\_\_\_\_\_ Прийма А.К.

м. Харків – 2019 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет  **\_\_\_\_\_\_\_\_\_\_\_\_**ком’ютерних наук\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_штучного інтелекту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітній ступінь \_\_\_\_бакалавр

Спеціальність\_\_\_\_122 – «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри \_\_\_\_\_\_**ШІ**\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**(В.О. Філатов)**\_\_\_**

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_року

**З А В Д А Н Н Я**

**НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ**

Щетініної Анастасії Володимирівни .

1.Тема проекту База даних «Залізна дорога»\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

керівник проекту доцент кафедри ШІ, кандидат технічних наук Магдаліна І.В.

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

2. Строк подання студентом проекту (роботи) 11.01.2019 .

3. Вихідні дані до проекту (роботи). вимоги до апаратної платформи — стандартні; середовище: Oracle Database; мова програмування: SQL; ОС Windows 10 .

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, аналіз предметної області, розробка інфологічної моделі, кервіництво користувача .

5. Перелік графічного матеріалу (рисунки, екранні форми)

схема фізичної бази даних, схема роботи задачі .

6. Дата видачі завдання 25 .«11» 2018р .

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів проекту | Строк виконання етапів проекту | Примітка |
| **1** | Узгодження та затвердження теми проекту | 25.11.2018 | виконано |
| **2** | Аналіз предметної області | 01.12.2018 - 14.12.2018 | виконано |
| **3** | Розробка схеми даних та логічної моделі БД | 14.12.2018 - 16.12.2018 | виконано |
| **4** | Реалізація візуального інтерфейсу програми, програмна реалізація проекту | 16.12.2018 - 29.12.2018 | Виконано |
| **5** | Оформлення пояснювальної записки | 30.12.2018 - 10.01.2019 | Виконано |
| **6** | Захист проекту | 11.01.2019 | Виконано |

Студент Щетініна А.В.

(підпис) (прізвище, ім´я, по-батькові)

Керівник проекту Магдаліна І.В.

(підпис) (прізвище, ім´я, по-батькові)

ЗМІСТ

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 5](#_Toc534918449)

[вступ 6](#_Toc534918450)

[1 Основні проблеми розробки сучасних баз даних, аналіз предметної області та постановка задачі курсової роботи 7](#_Toc534918451)

[1.1 Актуальність проблем розробки баз даних, основні поняття та визначення 8](#_Toc534918452)

[1.2 Аналіз предметної області 9](#_Toc534918453)

[1.3 Постановка задачі курсової роботи 10](#_Toc534918454)

[1.4 Обґрунтування вибору СУБД 10](#_Toc534918455)

[2. РОЗРОБКА ІНФОЛОГІЧНОЇ МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ 12](#_Toc534918456)

[3. РОЗРОБКА ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 14](#_Toc534918457)

[4. КЕРІВНИЦТВО КОРИСТУВАЧА 15](#_Toc534918458)

[ВИСНОВОК 20](#_Toc534918459)

[СПИСОК ДЖЕРЕЛ 21](#_Toc534918460)

[ДОДАТОК А 22](#_Toc534918461)

[ДОДАТОК Б 26](#_Toc534918462)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Oracle Database (часто просто Oracle) — об'єктно-реляційна [система керування базами даних](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) від [Oracle Corporation](https://uk.wikipedia.org/wiki/Oracle_Corporation)

SQL ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Structured query language — мова структурованих запитів) — [декларативна мова програмування](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) для взаємодії користувача з [базами даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), що застосовується для формування запитів, оновлення і керування [реляційними БД](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

ER-діаграма (Entity-Relationship) – графічна реалізація відношення «Сутність-зв´язок».

Primary key - первинний ключ, обмеження, що дозволяє однозначно ідентифікувати кожну запис в таблиці SQL.

Атрибут - впорядкована пара <A, V> де A - ім'я атрибута, V - безліч його значень (домен).

БД – база даних.

СУБД – Система управління базами даних.

Тип даних — характеристика, яку явно чи неявно надано об'єкту (змінній , функції, полю запису, константі , масив у тощо).

Графічний інтерфейс користувача (ГІК, англ. GUI, Graphical user interface) — інтерфейс між комп'ютером і його користувачем, що використовує піктограми, меню, і вказівний засіб для вибору функцій та виконання команд. Зазвичай, можливе відкриття більше, ніж одного вікна на одному екрані.

1:м – зв'язок «один до багатьох».

# вступ

Основою сучасної інформаційної технології становлять бази даних (БД) і системи управління базами даних (СУБД), роль яких як єдиного засобу зберігання, обробки й доступу до більших обсягів інформації постійно зростає. При цьому істотним є постійне підвищення обсягів інформації, збереженої в БД, що спричиняє вимогу збільшення продуктивності таких систем. Різко зростає попит у різноманітних застосуваннях на інтелектуальний доступ до інформації. Це особливо проявляється при організації логічної обробки інформації в системах баз знань, на основі яких створюються сучасні експертні системи.

Курсовий проект призначений для придбання практичних навичок у проектуванні баз даних для зберігання й обробки інформації, використання сучасних інформаційних технологій для організації збору, обробки й зберігання даних, виявленні можливих шляхів підвищення ефективності прийняття рішень у керуванні підприємством.

Темою даного курсового проекту є проектування бази даних для залізної дороги, який зберігає у собі інформацію про працівників, розклад залізної дороги, склад потягів, типи вагонів за допомогою СУБД Oracle, а також інформаційну систему для цієї предметної області, що має зручний та інтуїтивно-зрозумілий інтерфейс для будь-якого користувача.

# 1 Основні проблеми розробки сучасних баз даних, аналіз предметної області та постановка задачі курсової роботи

База даних ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *database*) – сукупність [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96_(%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%82%D0%B5%D1%85%D0%BD%D1%96%D0%BA%D0%B0)), організованих відповідно до предметної області, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96). Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

Відомі два підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується рядом недоліків.

Характерна риса файлового підходу - вузька спеціалізація як обробних програм, так і файлів даних, що служить причиною великої надлишковості, тому що ті самі елементи даних зберігаються в різних системах. Оскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше області.

Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів передання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зав’язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань.

Вказані причини обумовили появу баз даних. База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні "файл-програма", вона явним чином декларується в базі.

Головним завданням БД є гарантоване збереження значних обсягів інформації (так звані «записи даних») та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин : збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Кожна створена база даних має свою предметну область. Предметна область – це необхідний для розробки бази даних об’єкт, який має в собі дані, які будуть зберігатися в базі даних.

## 1.1 Актуальність проблем розробки баз даних, основні поняття та визначення

3 розвитком інформаційного забезпечення систем автоматизованої обробки інформації, прагненням забезпечити виконання нових режимів обробки даних у реальному часі і з мультидоступом до схованих даних позначилась нова тенденція до складення інформаційного забезпечення розподілених баз даних. В умовах використання таких баз створюються комплексні масиви нелінійної структури, які мають усі дані про ту чи іншу предметну область або про керований об'єкт як постійного, так і перемінного характеру.

Взагалі база даних є сукупність даних на машинних носіях, які використовуються при функціонуванні системи обробки інформації, організовані по визначеним правилам, які передбачають загальні принципи описування збереження і маніпулювання ними, а також які незалежні від прикладних програм. В основі організації бази даних є модель даних, яка визначає правила, у відповідності з якими структуруються дані. За допомогою моделі представляється велика кількість даних і описуються взаємно зв’язки між ними. Найбільш поширені такі моделі даних: ієрархічна, мережна, реляційна.

База даних орієнтована на інтегровані запити, а не на одну програму і використовується для інформаційних потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

Системи управління базами даних – це програмні засоби, за допомогою яких можна створювати бази даних, заповнювати їх та працювати з ними. У світі існує багато різноманітних систем управління базами даних. Багато з них насправді є не закінченими продуктами, а спеціалізованими мовами програмування, за допомогою яких кожний, хто вивчить дану мову, може сам створювати такі структури, які йому потрібні, і вводити в них необхідні елементи управління.

## 1.2 Аналіз предметної області

Темою курсового проекту є «Проектування бази даних для залізної дороги». У даному випадку база даних призначена для внутрішнього користування самими працівниками для складання розкладу маршрутів.

База даних повинна зберігати у собі інформацію про типи вагонів та склад потягів, працівників та їх посади, станції та маршрути. Для цього буде створена база даних, яка містить у собі 11 таблиць, які мають зв’язок 1:м, за допомогою СУБД Oracle.

Досить великою проблемою в розробці додатку є високе навантаження, велика кількість даних та постійне оновлення інформації. Щоденно до бази даних має доступ більше тисячі працівників залізної дороги, що також додає проблем.

## 1.3 Постановка задачі курсової роботи

В курсовому проекті за мету взято створення бази даних, яка має надавати можливість зберігання дуже великої кількості інформації, забезпечувати швидкий пошук інформації для клієнта. База даних повинна реалізовувати змогу оновлення та додавання, або видалення даних, які в ній зберігаються, а також зручне та надійне користування, та доступ усіх необхідних програм до неї.

При розробці бази даних необхідно створити таблиці в SQL Navigator, заповнити їх необхідними полями(атрибутами) та потрібними даними. Але до реалізації таблиць необхідно розробити модель БД, в якій буде зображено зв’язки між таблицями та основну схему побудови таблиць.

Після цього створити запити завдяки мові SQL , та перевірити їх роботу на видання різної інформації. Створена база даних в SQL Navigator повинна мати підключення до створеного додатку, щоб безпосередньо її використовувати.

## 1.4 Обґрунтування вибору СУБД

Продуктивність. Oracle Database дозволяє автоматично управляти рівнями сервісу і тиражувати еталонні конфігурації в рамках всієї мережі.

Прості засоби розробки. Новий інструмент розробки додатків HTML DB дозволить простим користувачам створювати ефективні програми для роботи з базами даних в короткі терміни.

Самоврядування. Спеціальні механізми Oracle Database дозволяють самостійно перерозподіляти навантаження на систему, оптимізувати і коригувати SQL-запити, виявляти і прогнозувати помилки.

Великі бази даних. Тепер максимальний розмір екземпляра бази даних Oracle може досягати 8 екзабайт.

Недорогі серверні системи. Oracle Database може використовувати недорогі однопроцесорні комп'ютери

# 2. РОЗРОБКА ІНФОЛОГІЧНОЇ МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ

Інфологічна модель створюється за результатами проведення досліджень предметної області. Інфологічна модель – це опис майбутньої бази даних, поданий за допомогою природної мови, формул, графіків, діаграм, таблиць та інших засобів, зрозумілих як розробникам БД, так і звичайним користувачам. Призначення такої моделі полягає в адекватному описі процесів, інформаційних потоків, функцій системи за допомогою загальнодоступної і зрозумілої мови.

Створення інфологічної моделі є природним продовженням досліджень предметної області, але, на відміну від нього, є поданням БД з точки зору проектувальника/розробника.

Інфологічна модель предметної області після словесного опису найчастіше представляється у вигляді графічної схеми (ER-діаграми).

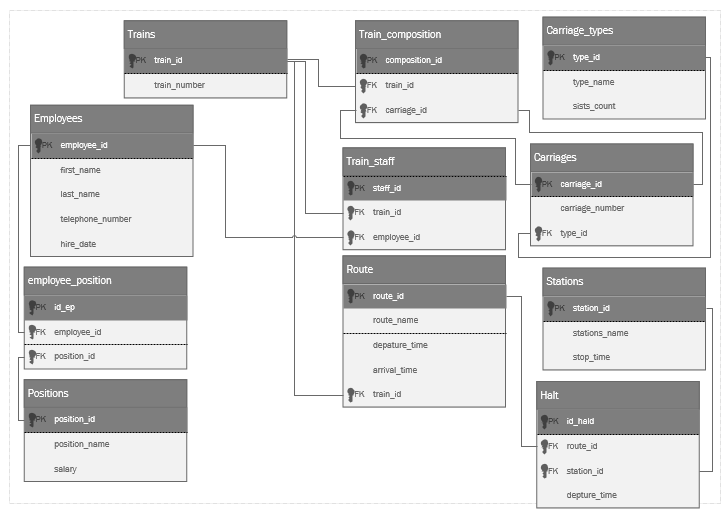


Рисунок 2.1. – ER-діаграма предметної області «Залізна дорога»

Отже потрібно побудувати базу даних, яка містить інформацію про потяги, типи вагонів, працівників, їх посади, час прибуття та відправлення потягів, станції та час зупинок на кожній з них.

Після аналізу предметної області було виділено 11 , а саме:

1. Типи вагонів – складається з 3 атрибутів – id вагону, назву типу та кількість місць.
2. Потяги - ця сутність утримує інформацію про потяги, маючи у собі 2 інформаційних поля – id потяга та його номер. Завдяки полю id потяга можна ідентифікувати будь-який потяг. Такий варіанти ідентифікації, або primary key будуть застосовуватися і в подальших сутностях даної бази даних.
3. Вагони – містить у собі поле id, номер вагону та тип вагону, який є зовнішнім ключем з таблиці «Типи вагонів»
4. Склад поїзду – проміжна таблиця, яка використовується для зв’язку двох попередніх таблиць «Потяги» та «Вагони», яка у подальшому використовується при складанні розкладу.
5. Працівники – мають атрибути фамілія, ім’я, номер телефону, дату прийому на роботу.
6. Посади – містить id, назву посади та відповідну зарплату.
7. Працівник-посада – проміжна таблиця, яка зв’язує таблиці «Працівники» та «Посада», так як один працівник може займати декілька посад.
8. Працівники потягу – таблиця, що містить і собі інформацію, про працівників, що обслуговують конкретний потяг. Має id, id потягу та id працівника.
9. Станції – складаються с атрибутів - id, назви станції та кількість хвилин, скільки триває зупинка за цій конкретній станції.
10. Маршрути – містить id, назву маршруту, час відправлення, час прибуття та id потяга, який їде за цим маршрутом.
11. Зупинки – таблиця, яка зв’язує «Маршрути» та «Станції», маюча id, id маршруту, id зупинки та час відправлення.

# 3. РОЗРОБКА ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Система управління базами даних Oracle дуже часто застосовується для зберігання важливої інформації у дуже великій кількості, яка може бути використана на веб-сайті або у спеціально розробленому додатку.

Якщо це звичайний сайт або форум – в базі можуть зберігатися повідомлення користувачів, дані для динамічних сторінок, дані про відвідини, якщо це який-небудь інтерактивний сервіс, то окрім даних про доступ (конфіденційних), там зберігається і інша інформація про користувача і його дії. Все це приводить до того, що загальна безпека сайту, вірніше за всю веб-сервера-системи, залежить від того, наскільки захищений саме сервер бази даних.

Але якщо це повноцінний додаток – в базі зберігаються данні, присвячені конкретній предметній області, та за допомогою цього додатку користувач має можливість редагувати, додавати та видаляти інформацію із бази даних.

У конкретному випадку для реалізації була обрана мова програмування C#. Для підключення до Oracle використовується спеціальний конектор, за допомогою якого ми маємо можливість працювати з даними не використовуючи спеціальну мову SQL, а за допомогою додатка.

# 4.КЕРІВНИЦТВО КОРИСТУВАЧА

Для завантаження програми на виконання необхідно вибрати файл Train-station.exe та двічі натиснути лівою кнопкою миші. Перед користувачем з’явиться головне вікно програми (рис 4.1)

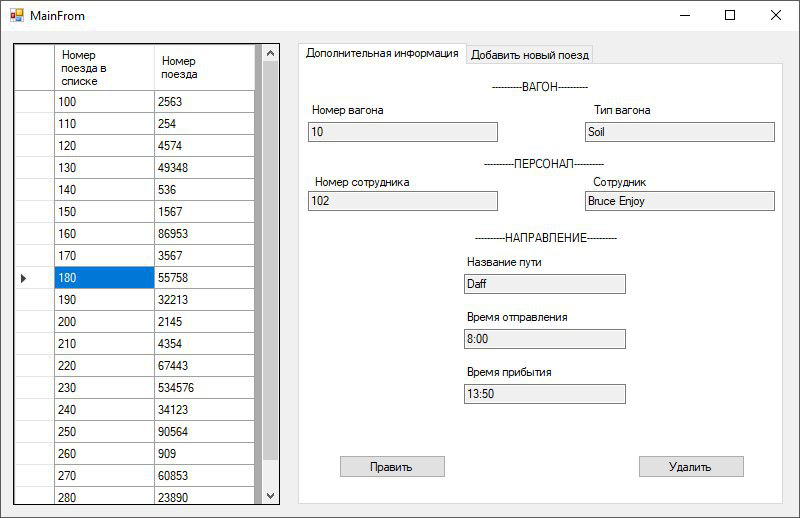


Рисунок 4.1 – Головне вікно програми «Train-station»

У робочій частині даного вікна знаходиться таблиця, з переліків усіх потягів, що містить в собі id потягу та його номер. На рисунку 4.1 виведена інформація про потяг, що має id “180”. При натисканні на id потягу у правій частині вікна розташована інформація про даний потяг. На рисунку 4.2 видно зміну інформації про потяг.

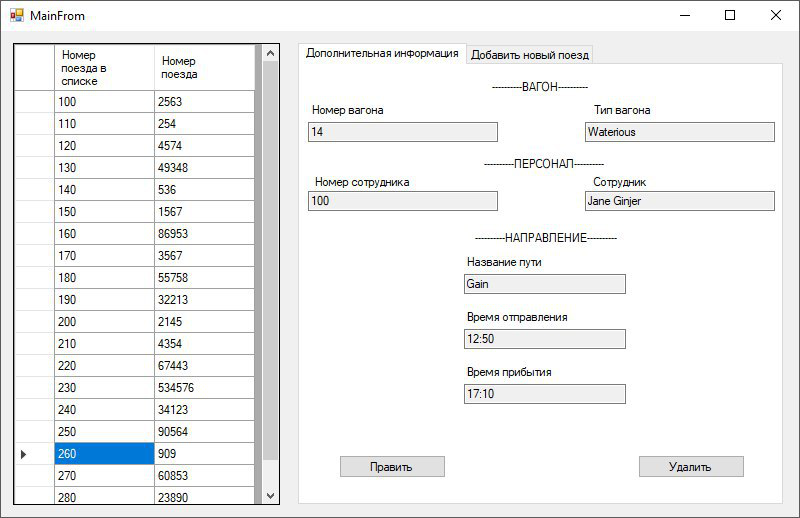


Рисунок 4.2 – поява інформації про обраний потяг, який має id “260”

Кожен потяг містить в собі дані про вагон(тип вагону та його номер), працівника (номер працівника та ім’я) та напрямок(назву маршруту, час прибуття та відправлення).

Користувач має змогу редагувати, видаляти та додавати новий потяг.

Для видалення потрібно обрати потрібний потяг та натиснути кнопку «Удалить»(рис.4.3), що розташована під основною інформацією про потяг.

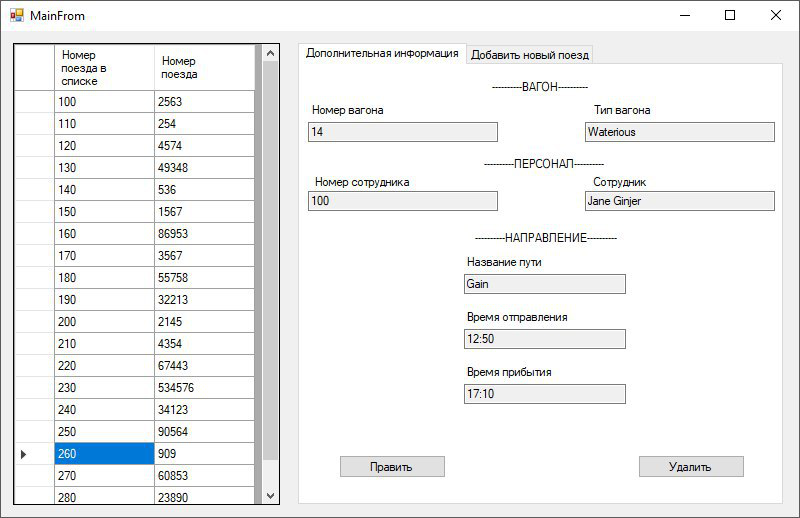


Рисунок 4.3 – кнопка «Удалить» для видалення інформації про обраний потяг

Після натискання на кнопку «Удалить» з’являється попередження для користувача, що застерігає його, від випадкового видалення даних(рис.4.4).

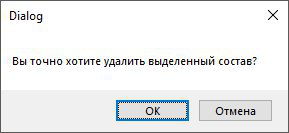


Рисунок 4.4 – попередження про видалення даних

Після підтвердження вибору користувача про видалення, відбувається видалення даних. Результат про успішне видалення даних можна побачити у діалоговому вікні(рис.4.5).

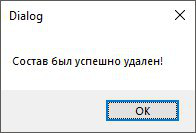


Рисунок 4.5 – підтвердження про успішне видалення даних

Для редагування інформації про потяг достатньо просто оновити дані і потрібному textbox. Дана функція має також додаткову перевірну на коректність даних, що вводяться користувачем(рис 4.6).

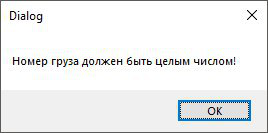


Рисунок 4.6 – попередження про некоректне введення даних

Для додавання нових даних(наприклад про новий потяг) створена спеціальна вкладка «Добавить новый поезд»(рис 4.7).

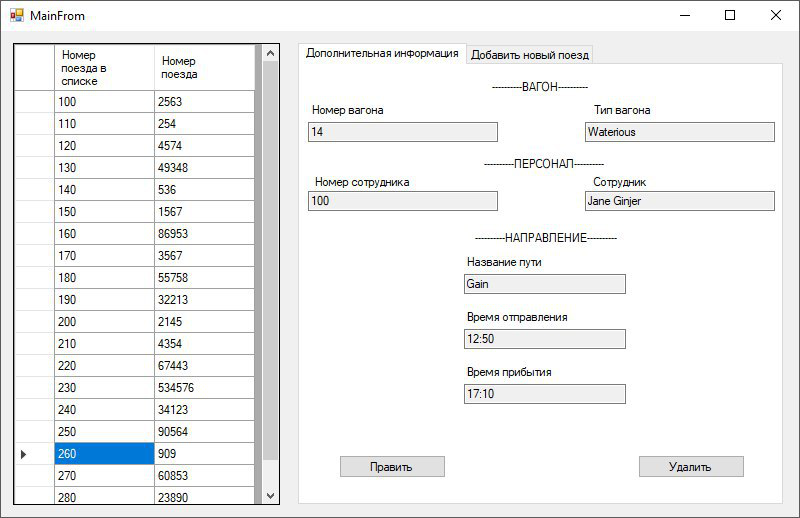


Рисунок 4.7 – вкладка для додавання нових даних

У цій вкладці користувач має можливість додати дані про новий потяг, а також про новий вагон, маршрут або працівника.(рис.4.8).

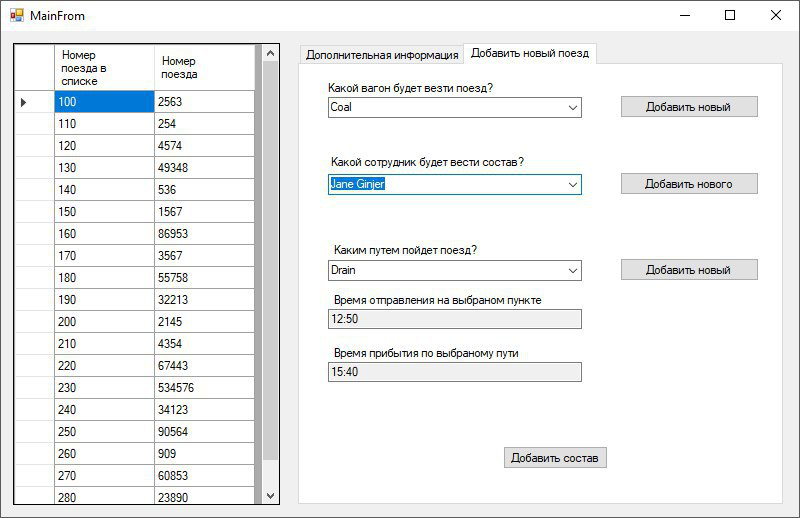


Рисунок 4.8 – вкладка для додання даних

Для доданні даних, наприклад, про новий вагон, з’являється нове вікно, для введення даних(рис. 4.9).

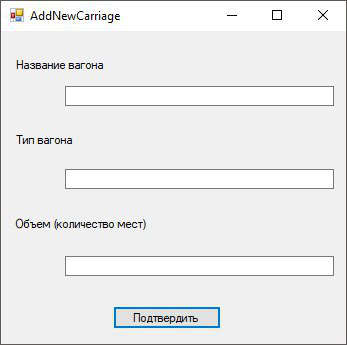


Рисунок 4.9 – форма для додання нового вагону

Усі поля для нових даних обов’язково мають перевірку для коректного вводу даних в базу даних. Поля не можуть бути пустими, числа - від’ємними і цілими.

# ВИСНОВОК

Метою даної курсової роботи була розробка бази даних в СУБД Oracle, що буде виводити згідно заданої теми потрібну для користувача інформацію про розклад роботи залізної дороги.

У ході роботи проведено аналіз предметної області, побудована інфологічна модель ПО, спроектована та реалізована в СУБД ORACLE. Був створений інтерфейс користувача для покращення візуального сприйняття інформації в середовищі «Visual Studio 2017» за допомогою мови програмування C#.

Технічні характеристики сучасних засобів керування базами даних постійно зростають. Для забезпечення високих показників роботи необхідно правильно обрати СУБД відповідно задачі. Задачею цього курсового проекту було створення бази даних для додатку “Train-station”. Для створення бази даних було обрано СУБД SQL Navigator для Oracle.

В базі даних були реалізовані запити на додавання даних у таблицю, обновлення даних, видалення та вибір інформації з таблиць. Всі запити працюють надійно та видають чітку інформацію.

# СПИСОК ДЖЕРЕЛ

1. ДСТУ 3008–95. Документація. Звіти в сфері науки і техніки. Структура і правила оформлення. – Державний стандарт України, 1995. – 36с.
2. Методические указание к курсовой работе по дисциплине «Организация баз данных и знаний» для студентов специальности 122 – Компьютерные науки / Учеред. Волощук О.Б. – Харьков: ХНУРЕ, 2018. – 22с.
3. Пасічник, В.В. Організація баз даних та знань. Підручник [Текст] / В.В. Пасічник, В.А. Резніченко. – К.: Видавнича група BHV, 2006. – 384 с.
4. Хомоненко, А.Д. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко [Текст] / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – СПб.: КОРОНА принт, 2004. – 736 с.
5. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: (ГОСТ 7.1–2003, IDT): ДСТУ ГОСТ 7.1:2006. – Чинний з   
   2007–07–01. – К.: Держспоживстандарт України, 2007. – ІІІ, 47 с.

# ДОДАТОК А

Лістинг створення бази даних

Створення таблиці «Тип вагонів»

**create** **table** Carriage\_types(

type\_id **NUMBER** **not** **null**,

type\_name **VARCHAR**(35),

sits\_count **NUMBER** **not** **null**,

**constraint** types\_pk **primary** **key** (type\_id)

);

Створення таблиці «Вагони»

**create** **table** Carriages(

carriage\_id **NUMBER** **not** **null**,

carriage\_number **VARCHAR**(10),

type\_id **NUMBER** **not** **null**,

**constraint** carriages\_pk **primary** **key** (carriage\_id),

**constraint** type\_fk

**foreign** **key** (type\_id)

**references** Carriage\_types(type\_id)

);

Створення таблиці «Потяги»

**create** **table** Trains(

train\_id **NUMBER** **not** **null**,

train\_number **NUMBER** **not** **null**,

**constraint** trains\_pk **primary** **key** (train\_id)

);

Створення таблиці «Склад потягу»

**create** **table** Train\_composition(

composition\_id **NUMBER** **not** **null**,

train\_id **NUMBER** **not** **null**,

carriage\_id **NUMBER** **not** **null**,

**constraint** composition\_pk **primary** **key** (composition\_id),

**constraint** train\_fk

**foreign** **key** (train\_id)

**references** Trains(train\_id),

**constraint** carriage\_fk

**foreign** **key** (carriage\_id)

**references** Carriages(carriage\_id)

);

Створення таблиці «Посади»

**create** **table** Positions(

position\_id **NUMBER** **not** **null**,

position\_name **VARCHAR**(20),

salary **NUMBER** **not** **null**,

**constraint** position\_pk **primary** **key** (position\_id),

**constraint** salary **check** (salary>0)

);

Створення таблиці «Працівники»

**create** **table** Employee(

employee\_id **NUMBER** **not** **null**,

first\_name **varchar**(30) **not** **null**,

last\_name **varchar**(30) **not** **null**,

telephone **varchar**(10) **not** **null**,

hire\_date **VARCHAR**(10) **not** **null**,

**constraint** employee\_pk **primary** **key** (employee\_id)

);

Створення таблиці «Працівник-посада»

**create** **table** employee\_positions(

id\_ep **NUMBER** **not** **null**,

employee\_id **NUMBER** **not** **null**,

position\_id **NUMBER** **not** **null**,

**constraint** ep\_fk **primary** **key** (id\_ep),

**constraint** employee\_fk

**foreign** **key** (employee\_id)

**references** Employee(employee\_id),

**constraint** position\_id

**foreign** **key** (position\_id)

**references** Positions(position\_id)

);

Створення таблиці «Працівники потягу»

**create** **table** Train\_staff(

staff\_id **NUMBER** **not** **null**,

train\_id **NUMBER** **not** **null**,

employee\_id **NUMBER** **not** **null**,

**constraint** staff\_pk **primary** **key** (staff\_id),

**constraint** train\_fk2 **foreign** **key** (train\_id)

**references** Trains(train\_id),

**constraint** employee\_fk2 **foreign** **key** (employee\_id)

**references** Employee(employee\_id)

);

Створення таблиці «Станції»

**create** **table** Stations(

station\_id **NUMBER** **not** **null**,

station\_name **VARCHAR**(30) **not** **null**,

stop\_time **NUMBER**,

**constraint** station\_pk **primary** **key** (station\_id)

);

Створення таблиці «Маршрути»

**create** **table** Routes(

route\_id **NUMBER** **not** **null**,

route\_name **VARCHAR**(50) **not** **null**,

depature\_time **VARCHAR**(5) **not** **null**,

arrival\_time **VARCHAR**(5) **not** **null**,

train\_id **NUMBER** **not** **null**,

**constraint** route\_pk **primary** **key** (route\_id),

**constraint** train\_fk3 **foreign** **key** (train\_id)

**references** Trains(train\_id)

);

Створення таблиці «Зупинки»

**create** **table** Halt(

halt\_id **NUMBER** **not** **null**,

route\_id **NUMBER** **not** **null**,

train\_id **NUMBER** **not** **null**,

depture\_time **VARCHAR**(5) **not** **null**,

**constraint** halt\_pk **primary** **key** (halt\_id),

**constraint** route\_id **foreign** **key** (route\_id)

**references** Routes(route\_id),

**constraint** train\_id4 **foreign** **key** (train\_id)

**references** Trains(train\_id)

);

# ДОДАТОК Б

Лістинг програми

Клас «Carriage\_type»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Carriage\_type

{

public int type\_id;

public string type\_name;

public int sits\_count;

public Carriage\_type() {

type\_id = 0;

type\_name = "";

sits\_count = 0;

}

public Carriage\_type(string info) {

if (info != null && info != "") {

string[] val = info.Split('!');

type\_id = Convert.ToInt32(val[0]);

type\_name = val[1];

sits\_count = Convert.ToInt32(val[2]);

}

}

}

}

Клас «Setup\_Program»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace TimeToDoIt

{

public class ClassSetupProgram

{

public string connectionToDBString = "Data Source=XE;User ID=HR;Password=mersedess;";

public ClassSetupProgram() { }

}

}

Клас «Carriages»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Carriages

{

public int carriage\_id;

public string carriage\_number;

public int type\_id;

public Carriages() {

carriage\_id = 0;

carriage\_number = "";

type\_id = 0;

}

public Carriages(string info) {

if (info != null && info != "")

{

string[] val = info.Split('!');

carriage\_id = Convert.ToInt32(val[0]);

carriage\_number = val[1];

type\_id = Convert.ToInt32(val[2]);

}

}

}

}

Клас «Employee\_Position»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Employee\_Position

{

public int empl\_pos;

public int employee\_id;

public int position\_id;

public Employee\_Position() {

empl\_pos = 0;

employee\_id = 0;

position\_id = 0;

}

public Employee\_Position(string info) {

if (info != null && info != "") {

string[] val = info.Split('!');

empl\_pos = Convert.ToInt32(val[0]);

employee\_id = Convert.ToInt32(val[1]);

position\_id = Convert.ToInt32(val[2]);

}

}

}

}

Клас «Employees»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Employees

{

public int employee\_id;

public string first\_name;

public string second\_name;

public string telephone;

public string hire\_date;

public Employees() {

employee\_id = 0;

first\_name = "";

second\_name = "";

telephone = "";

hire\_date = "";

}

public Employees(string info) {

if (info != null && info != "") {

string[] val = info.Split('!');

employee\_id = Convert.ToInt32(val[0]);

first\_name = val[1];

second\_name = val[2];

telephone = val[3];

hire\_date = val[4];

}

}

}

}

Клас «Halt»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Halt

{

public int halt\_id;

public int route\_id;

public int train\_id;

public string depture\_time;

public Halt() {

halt\_id = 0;

route\_id = 0;

train\_id = 0;

depture\_time = "";

public Halt(string info) {

if (info != null && info != "") {

string[] val = info.Split('!');

halt\_id = Convert.ToInt32(val[0]);

route\_id = Convert.ToInt32(val[1]);

depture\_time = val[2]; }

}

}

Клас «Positions»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Positions

{

public int position\_id;

public string position\_name;

public int salary;

public Positions() {

position\_id = 0;

position\_name = "";

salary = 0;

}

public Positions(string info) {

if (info != null && info != "") {

string[] val = info.Split('!');

position\_id = Convert.ToInt32(val[0]);

position\_name = val[1];

salary = Convert.ToInt32(val[2]);

}

}

}

}

Клас «Program»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeToDoIt

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

Клас «Routes»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Routes

{

public int route\_id;

public string route\_name;

public string depature\_time;

public string arrival\_time;

public int train\_id;

public Routes()

{

route\_id = 0;

route\_name = "";

depature\_time = "";

arrival\_time = "";

train\_id = 0;}

public Routes(string info)

{

if (info != null && info != "")

{

string[] val = info.Split('!');

route\_id = Convert.ToInt32(val[0]);

route\_name = val[1];

depature\_time = val[2];

arrival\_time = val[3];

train\_id = Convert.ToInt32(val[4]);}

}

}

}

Клас «Stations»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Stations

{

public int station\_id;

public string station\_name;

public string stop\_time;

public Stations() {

station\_id = 0;

station\_name = "";

stop\_time = "";}

public Stations(string info) {

if (info != null && info != "") {

string[] val = info.Split('!');

station\_id = Convert.ToInt32(val[0]);

station\_name = val[1];

stop\_time = val[2];}

}

}

}

Клас «Train»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Train

{

public int train\_id;

public int train\_number;

public Train() {

train\_id = 0;

train\_number = 0;

}

public Train(string info) {

if (info != null && info != "")

{

string[] val = info.Split('!');

train\_id = Convert.ToInt32(val[0]);

train\_number = Convert.ToInt32(val[1]);

}

}

}

}

Клас «Train\_Composition»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Train\_Composition

{

public int composition\_id;

public int train\_id;

public int carriage\_id;

public Train\_Composition() {

composition\_id = 0;

train\_id = 0;

carriage\_id = 0;

}

public Train\_Composition(string info) {

if (info != null && info != "")

{

string[] val = info.Split('!');

composition\_id = Convert.ToInt32(val[0]);

train\_id = Convert.ToInt32(val[1]);

carriage\_id = Convert.ToInt32(val[2]);

}

}

}

}

Клас «Train\_Staff»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeToDoIt

{

class Train\_Staff

{

public int staff\_id;

public int train\_id;

public int employee\_id;

public Train\_Staff()

{

staff\_id = 0;

train\_id = 0;

employee\_id = 0;

}

public Train\_Staff(string info)

{

if (info != null && info != "")

{

string[] val = info.Split('!');

staff\_id = Convert.ToInt32(val[0]);

train\_id = Convert.ToInt32(val[1]);

employee\_id = Convert.ToInt32(val[2]);

}

}

}

}

Код форми «Main»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace TimeToDoIt

{

public partial class MainForm : Form

{

ClassDataBase db = new ClassDataBase();

ClassSetupProgram stp = new ClassSetupProgram();

Random randNumber = new Random();

List<Train> trains = new List<Train>();

List<Carriage\_type> carriage\_types = new List<Carriage\_type>();

List<Carriages> carriages = new List<Carriages>();

List<Employee\_Position> employee\_positions = new List<Employee\_Position>();

List<Employees> employees = new List<Employees>();

List<Halt> halts = new List<Halt>();

List<Positions> positions = new List<Positions>();

List<Routes> routes = new List<Routes>();

List<Routes> routes\_distinct = new List<Routes>();

List<Stations> stations = new List<Stations>();

List<Train\_Composition> train\_compositions = new List<Train\_Composition>();

List<Train\_Staff> train\_staffs = new List<Train\_Staff>();

Regex numbers = new Regex(@"\b\d{1,9}\b");

Regex dateTime = new Regex(@"\b\d{1,2}\:\d{1,2}\b");

Regex words = new Regex(@"\b(\w+?)\b");

Regex names = new Regex(@"\b(\w+?)\s{1}(\w+?)\b");

Random rnd = new Random();

int current\_row = 0, current\_cell = 0;

void ShowFromDataGridView(int number)

{

int temp = 0;

current\_row = number;

for (int i = 0; i < train\_compositions.Count; i++)

{

if (train\_compositions[i].train\_id == number)

{

//if (Regex.IsMatch(train\_compositions[i].carriage\_id.ToString(), numbers.ToString(), RegexOptions.IgnoreCase)) {

tbCarriageNumber.Text = train\_compositions[i].carriage\_id.ToString();

//}

temp = Convert.ToInt32(train\_compositions[i].carriage\_id);

break;

}

}

for (int i = 0; i < carriages.Count; i++)

{

if (Convert.ToInt32(carriages[i].carriage\_id) == temp)

{

temp = Convert.ToInt32(carriages[i].type\_id);

break;

}

}

for (int i = 0; i < carriage\_types.Count; i++)

{

if (Convert.ToInt32(carriage\_types[i].type\_id) == temp)

{

tbCarriageType.Text = carriage\_types[i].type\_name.ToString();

break;

}

}

temp = 0;

for (int i = 0; i < train\_staffs.Count; i++)

{

if(train\_staffs[i].train\_id == number)

{

tbStaffNumber.Text = train\_staffs[i].employee\_id.ToString();

temp = Convert.ToInt32(train\_staffs[i].employee\_id);

break;

}

}

for (int i = 0; i < employees.Count; i++)

{

if (Convert.ToInt32(employees[i].employee\_id) == temp)

{

//if (Regex.IsMatch(employees[i].first\_name.ToString() + " " + employees[i].second\_name.ToString(), names.ToString(),RegexOptions.IgnoreCase)) {

tbStaffName.Text = employees[i].first\_name.ToString() + " " + employees[i].second\_name.ToString();

//}

break;

}

}

for (int i = 0; i < routes.Count; i++)

{

if (routes[i].train\_id == number)

{

//if (Regex.IsMatch(routes[i].arrival\_time.ToString(), dateTime.ToString(), RegexOptions.IgnoreCase)) {

tbRouteArrive.Text = routes[i].arrival\_time.ToString();

//}

tbRouteDeparture.Text = routes[i].depature\_time.ToString();

//if (Regex.IsMatch(routes[i].route\_name.ToString(), words.ToString(), RegexOptions.IgnoreCase)) {

tbRouteName.Text = routes[i].route\_name.ToString();

//}

}

}

}

void LoadData() {

cbCarriage.Items.Clear();

cbStaff.Items.Clear();

cbTrain.Items.Clear();

trains.Clear();

carriages.Clear();

carriage\_types.Clear();

employees.Clear();

employee\_positions.Clear();

halts.Clear();

positions.Clear();

routes.Clear();

routes\_distinct.Clear();

stations.Clear();

train\_compositions.Clear();

train\_staffs.Clear();

db.Execute<Train>(ref stp, "Select \* from trains", ref trains);

db.Execute<Carriage\_type>(ref stp, "Select \* FROM carriage\_types", ref carriage\_types);

db.Execute<Carriages>(ref stp, "Select \* from carriages", ref carriages);

db.Execute<Employee\_Position>(ref stp, "Select \* from employee\_positions", ref employee\_positions);

db.Execute<Employees>(ref stp, "Select \* from employee", ref employees);

db.Execute<Halt>(ref stp, "Select \* from halt", ref halts);

db.Execute<Positions>(ref stp, "Select \* from positions", ref positions);

db.Execute<Routes>(ref stp, "Select \* from routes", ref routes);

db.Execute<Routes>(ref stp, "Select distinct \* from routes", ref routes\_distinct);

db.Execute<Stations>(ref stp, "Select \* from stations", ref stations);

db.Execute<Train\_Composition>(ref stp, "Select \* from train\_composition", ref train\_compositions);

db.Execute<Train\_Staff>(ref stp, "Select \* from train\_staff", ref train\_staffs);

}

void ShowData() {

for (int i = 0; i < carriage\_types.Count; i++)

{

cbCarriage.Items.Add(carriage\_types[i].type\_name);

}

for (int i = 0; i < employees.Count; i++)

{

cbStaff.Items.Add(employees[i].first\_name + " " + employees[i].second\_name);

}

for (int i = 0; i < routes\_distinct.Count; i++)

{

cbTrain.Items.Add(routes\_distinct[i].route\_name);

}

cbCarriage.SelectedIndex = 0;

cbStaff.SelectedIndex = 0;

cbTrain.SelectedIndex = 0;

dgvMain.Rows.Clear();

if (trains.Count != 0)

{

for (int i = 0; i < trains.Count; ++i)

{

dgvMain.Rows.Add(trains[i].train\_id, trains[i].train\_number);

}

}

else

MessageBox.Show("Что-то пошло не так. Возможно таблица пуста, либо не существует!", "Dialog", MessageBoxButtons.OK);

}

public MainForm()

{

InitializeComponent();

}

private void MainForm\_Load(object sender, EventArgs e)

{

LoadData();

ShowData();

}

private void dgvMain\_CellClick(object sender, DataGridViewCellEventArgs e)

{

tbCarriageNumber.Text = "";

tbCarriageType.Text = "";

tbRouteArrive.Text = "";

tbRouteDeparture.Text = "";

tbRouteName.Text = "";

tbStaffName.Text = "";

tbStaffNumber.Text = "";

current\_cell = dgvMain.CurrentRow.Index;

ShowFromDataGridView(Convert.ToInt32(dgvMain.CurrentRow.Cells[0].Value.ToString()));

}

private void btnAdditing\_Click(object sender, EventArgs e)

{

if (btnAdditing.Text == "Править")

{

tbCarriageNumber.ReadOnly = false;

tbRouteArrive.ReadOnly = false;

tbRouteDeparture.ReadOnly = false;

tbRouteName.ReadOnly = false;

tbStaffNumber.ReadOnly = false;

dgvMain.CurrentCell = dgvMain.Rows[current\_cell].Cells[0];

btnAdditing.Text = "Сохранить";

dgvMain.Enabled = false;

btnAdditing.Enabled = false;

//MessageBox.Show(current\_row.ToString(), "", MessageBoxButtons.OK);

//string[] val = tbRouteArrive.Text.Trim().ToString().Split(':');

//MessageBox.Show(int.Parse(val[0]).ToString() + "-" + int.Parse(val[1]).ToString(), "",MessageBoxButtons.OK);

}

else

{

tbCarriageNumber.ReadOnly = true;

tbRouteArrive.ReadOnly = true;

tbRouteDeparture.ReadOnly = true;

tbRouteName.ReadOnly = true;

tbStaffNumber.ReadOnly = true;

dgvMain.Enabled = true;

btnAdditing.Enabled = true;

dgvMain.CurrentCell = dgvMain.Rows[current\_cell].Cells[0];

btnAdditing.Text = "Править";

if (Regex.IsMatch(tbCarriageNumber.Text.Trim().ToString(), numbers.ToString(), RegexOptions.IgnoreCase) &&

Regex.IsMatch(tbRouteArrive.Text.Trim().ToString(), dateTime.ToString(), RegexOptions.IgnoreCase) &&

Regex.IsMatch(tbRouteDeparture.Text.Trim().ToString(), dateTime.ToString(), RegexOptions.IgnoreCase) &&

Regex.IsMatch(tbRouteName.Text.Trim().ToString(), words.ToString(), RegexOptions.IgnoreCase) &&

Regex.IsMatch(tbStaffNumber.Text.Trim().ToString(), numbers.ToString(), RegexOptions.IgnoreCase))

{

string command\_tc = "update train\_composition set train\_composition.carriage\_id = " + tbCarriageNumber.Text.Trim().ToString() + " where train\_composition.train\_id = " + current\_row.ToString();

string command\_staff = "update train\_staff set train\_staff.employee\_id = " + tbStaffNumber.Text.Trim().ToString() + " where train\_staff.train\_id = " + current\_row.ToString();

string command\_route\_name = "update routes set routes.route\_name = " + tbRouteName.Text.Trim().ToString() + " where routes.train\_id = " + current\_row.ToString();

string command\_route\_arrive = "update routes set routes.arrival\_time = " + tbRouteArrive.Text.Trim().ToString() + " where routes.train\_id = " + current\_row.ToString();

string command\_route\_departure = "update routes set routes.depature\_time = " + tbRouteDeparture.Text.Trim().ToString() + " where routes.train\_id = " + current\_row.ToString();

string[] val\_arival = tbRouteArrive.Text.Trim().ToString().Split(':');

string[] val\_departure = tbRouteDeparture.Text.Trim().ToString().Split(':');

if (int.Parse(val\_arival[0]) < 24 && int.Parse(val\_departure[0]) < 24 && int.Parse(val\_arival[1]) < 60 && int.Parse(val\_departure[1]) < 60)

{

if ((int.Parse(val\_arival[0]) > int.Parse(val\_departure[0])) || ((int.Parse(val\_arival[0]) == int.Parse(val\_departure[0])) && (int.Parse(val\_arival[1]) > int.Parse(val\_departure[1]))))

{

db.ExecuteNonQuery(ref stp, command\_tc);

db.ExecuteNonQuery(ref stp, command\_staff);

db.ExecuteNonQuery(ref stp, command\_route\_name);

db.ExecuteNonQuery(ref stp, command\_route\_arrive);

db.ExecuteNonQuery(ref stp, command\_route\_departure);

}

else if (MessageBox.Show("Поезд приходит в следующий день?", "Dialog", MessageBoxButtons.OKCancel) == DialogResult.OK)

{

db.ExecuteNonQuery(ref stp, command\_tc);

db.ExecuteNonQuery(ref stp, command\_staff);

db.ExecuteNonQuery(ref stp, command\_route\_name);

db.ExecuteNonQuery(ref stp, command\_route\_arrive);

db.ExecuteNonQuery(ref stp, command\_route\_departure);

}

else

{

MessageBox.Show("Данные не были обновлены!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Данные не были обновлены так как формат времени не сохранен!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Вы ввели что то не так!","Dialog",MessageBoxButtons.OK);

}

}

}

private void btnAdditing\_Enter(object sender, EventArgs e)

{

LoadData();

ShowData();

}

private void tbCarriageNumber\_Leave(object sender, EventArgs e)

{

int temp = 0, counter = 0;

if (int.TryParse(tbCarriageNumber.Text.Trim().ToString(), out temp))

{

for (int i = 0; i < carriages.Count; i++)

{

if (carriages[i].carriage\_id == Convert.ToInt32(tbCarriageNumber.Text.Trim().ToString()))

counter++;

}

if (counter == 0)

{

if (MessageBox.Show("Груза с таким номером не существует! Добавить?", "Dialog", MessageBoxButtons.OKCancel) == DialogResult.OK)

{

AddNewCarriage d = new AddNewCarriage();

d.Show();

}

else

{

tbCarriageNumber.Focus();

}

}

}

else

{

MessageBox.Show("Номер груза должен быть целым числом!", "Dialog", MessageBoxButtons.OK);

tbCarriageNumber.Focus();

}

}

private void btnAddCarriage\_Click(object sender, EventArgs e)

{

AddNewCarriage d = new AddNewCarriage();

d.Show();

}

private void btnAddStaff\_Click(object sender, EventArgs e)

{

AddNewStaff d = new AddNewStaff();

d.Show();

}

private void btnAddRoute\_Click(object sender, EventArgs e)

{

AddNewRoute d = new AddNewRoute();

d.Show();

}

private void tbRouteDeparture\_KeyPress(object sender, KeyPressEventArgs e)

{

char ch = e.KeyChar;

if (!Char.IsDigit(ch) && ch != 8)

{

e.Handled = true;

}

}

private void tbRouteArrive\_KeyPress(object sender, KeyPressEventArgs e)

{

char ch = e.KeyChar;

if (!Char.IsDigit(ch) && ch != 8)

{

e.Handled = true;

}

}

private void tbAddDepartureTime\_KeyPress(object sender, KeyPressEventArgs e)

{

char ch = e.KeyChar;

if (!Char.IsDigit(ch) && ch != 8)

{

e.Handled = true;

}

}

private void tbAddArivalTime\_KeyPress(object sender, KeyPressEventArgs e)

{

char ch = e.KeyChar;

if (!Char.IsDigit(ch) && ch != 8)

{

e.Handled = true;

}

}

private void MainForm\_Activated(object sender, EventArgs e)

{

LoadData();

ShowData();

}

private void cbTrain\_SelectedIndexChanged(object sender, EventArgs e)

{

for (int i = 0; i < routes.Count; i++)

{

if (cbTrain.Text == routes[i].route\_name)

{

tbAddArivalTime.Text = routes[i].arrival\_time;

tbAddDepartureTime.Text = routes[i].depature\_time;

}

}

}

private void btnAddTrain\_Click(object sender, EventArgs e)

{

int train\_number = rnd.Next(99999);

int train\_id = trains[trains.Count - 1].train\_id + 10;

string command\_train = "insert into trains(train\_id, train\_number) values(" + train\_id.ToString() + "," + train\_number.ToString() + ")";

db.ExecuteNonQuery(ref stp, command\_train);

//string all\_commans = command\_train + '\n';

int composition\_id = (train\_compositions[train\_compositions.Count - 1].composition\_id) + 1;

int staff\_id = (train\_staffs[train\_staffs.Count - 1].staff\_id) + 10;

int route\_id = (routes[routes.Count - 1].route\_id) + 1;

for (int i = 0; i < carriage\_types.Count; i++)

{

if (carriage\_types[i].type\_name == cbCarriage.Text)

{

int carriage\_id = (carriages[carriages.Count - 1].carriage\_id) + 1;

string command\_cariage = "insert into carriages(carriage\_id, carriage\_number, type\_id) values(" + carriage\_id.ToString() + ",'Cariage" + carriage\_id.ToString() + "'," + carriage\_types[i].type\_id.ToString() + ")";

db.ExecuteNonQuery(ref stp, command\_cariage);

// all\_commans += command\_cariage + '\n';

string command\_composition = "insert into train\_composition(composition\_id, train\_id, carriage\_id) values("+ composition\_id.ToString() +","+ train\_id.ToString() +","+ carriage\_id.ToString() +")";

db.ExecuteNonQuery(ref stp, command\_composition);

//all\_commans += command\_composition + '\n';

break;

}

}

for (int i = 0; i < employees.Count; i++)

{

if ((employees[i].first\_name + " " + employees[i].second\_name) == cbStaff.Text)

{

string command\_staff = "insert into train\_staff(staff\_id, train\_id, employee\_id) values("+ staff\_id.ToString() +","+ train\_id.ToString() +","+ employees[i].employee\_id.ToString() +")";

db.ExecuteNonQuery(ref stp, command\_staff);

//all\_commans += command\_staff + '\n';

break;

}

}

for (int i = 0; i < routes.Count; i++)

{

if (routes[i].route\_name == cbTrain.Text)

{

string command\_route = "insert into routes(route\_id, route\_name, depature\_time, arrival\_time, train\_id) values("+ route\_id.ToString() +",'"+ routes[i].route\_name.ToString() +"','"+ routes[i].depature\_time.ToString() +"','"+ routes[i].arrival\_time.ToString() +"',"+ train\_id.ToString() +")";

db.ExecuteNonQuery(ref stp, command\_route);

// all\_commans += command\_route + '\n';

break;

}

}

//MessageBox.Show(all\_commans,"Dialog",MessageBoxButtons.OK);

}

private void tbStaffNumber\_Leave(object sender, EventArgs e)

{

int temp = 0, counter = 0;

if (int.TryParse(tbStaffNumber.Text.Trim().ToString(), out temp))

{

for (int i = 0; i < employees.Count; i++)

{

if (employees[i].employee\_id == Convert.ToInt32(tbStaffNumber.Text.Trim().ToString()))

{

counter++;

}

}

if (counter == 0)

{

if (MessageBox.Show("В базе нет таких сотрудников. Добавить?", "Dialog", MessageBoxButtons.OKCancel) == DialogResult.OK)

{

AddNewStaff d = new AddNewStaff();

d.Show();

}

else

{

tbStaffNumber.Focus();

}

}

}

else

{

MessageBox.Show("Номер сотрудника должен быть целым числом!", "Dialog", MessageBoxButtons.OK);

tbStaffNumber.Focus();

}

}

}

}

Код форми «AddNewCarriage»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace TimeToDoIt

{

public partial class AddNewCarriage : Form

{

ClassDataBase db = new ClassDataBase();

ClassSetupProgram stp = new ClassSetupProgram();

List<Carriage\_type> carriage\_types = new List<Carriage\_type>();

List<Carriages> carriages = new List<Carriages>();

Regex type\_name = new Regex(@"\b(\w+?)\b");

Regex value = new Regex(@"\d{1,}");

void LoadData()

{

carriages.Clear();

carriage\_types.Clear();

db.Execute<Carriage\_type>(ref stp, "Select \* FROM carriage\_types", ref carriage\_types);

db.Execute<Carriages>(ref stp, "Select \* from carriages", ref carriages);

}

public AddNewCarriage()

{

InitializeComponent();

}

private void AddNewCarriage\_Load(object sender, EventArgs e)

{

LoadData();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

int temp = 0;

if ((tbCarriageNumber.Text.Trim() != "") && (tbSitsCount.Text.Trim() != "") && (tbTypeName.Text.Trim() != ""))

{

if (int.TryParse(tbSitsCount.Text.ToString(), out temp))

{

if (Regex.IsMatch(tbCarriageNumber.Text.Trim().ToString(), type\_name.ToString()) &&

Regex.IsMatch(tbTypeName.Text.Trim().ToString(), type\_name.ToString()) &&

Regex.IsMatch(tbSitsCount.Text.Trim().ToString(), value.ToString()))

{

int type\_id = (carriage\_types[carriage\_types.Count - 1].type\_id) + 1;

int carriage\_id = (carriages[carriages.Count - 1].carriage\_id) + 1;

string command\_types = "insert into carriage\_types(type\_id, type\_name, sits\_count) values("+ type\_id.ToString() +",'"+ tbTypeName.Text.Trim().ToString() +"',"+ tbSitsCount.Text.Trim().ToString()+")";

db.ExecuteNonQuery(ref stp, command\_types);

string command\_carriages = "insert into carriages(carriage\_id, carriage\_number, type\_id) values("+ carriage\_id.ToString() +",'"+ tbCarriageNumber.Text.Trim().ToString() +"',"+ type\_id.ToString() + ")";

db.ExecuteNonQuery(ref stp, command\_carriages);

if (MessageBox.Show("Добавлен новый груз с номером" + carriage\_id.ToString() + "!", "Dialog", MessageBoxButtons.OK) == DialogResult.OK)

{

this.Close();

}

}

else

{

MessageBox.Show("Вы Ввели что то не так, проверьте информацию!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Объем должен быть указан целым числом!","Dialog",MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Заполните все поля!","Dialog",MessageBoxButtons.OK);

}

}

private void AddNewCarriage\_FormClosing(object sender, FormClosingEventArgs e)

{

}

}

}

Код форми «AddNewRoute»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace TimeToDoIt

{

public partial class AddNewRoute : Form

{

ClassDataBase db = new ClassDataBase();

ClassSetupProgram stp = new ClassSetupProgram();

List<Train> trains = new List<Train>();

List<Routes> routes = new List<Routes>();

Regex time = new Regex(@"\b\d{1,2}\:\d{1,2}\b");

Regex name = new Regex(@"\b(\w+?)\b");

Random rnd = new Random();

void LoadData()

{

trains.Clear();

routes.Clear();

db.Execute<Train>(ref stp, "Select \* from trains", ref trains);

db.Execute<Routes>(ref stp, "Select \* from routes", ref routes);

}

public AddNewRoute()

{

InitializeComponent();

LoadData();

}

private void btnAddRoute\_Click(object sender, EventArgs e)

{

int counter = 0;

for (int i = 0; i < routes.Count; i++)

{

if (routes[i].route\_name == tbRouteName.Text)

{

counter++;

}

}

if (tbArrivalTime.Text.Trim().ToString() != "" && tbDepatureTime.Text.Trim().ToString() != "" && tbRouteName.Text.Trim().ToString() != "")

{

if (counter == 0)

{

if (Regex.IsMatch(tbRouteName.Text.Trim().ToString(), name.ToString()) && Regex.IsMatch(tbArrivalTime.Text.Trim().ToString(), time.ToString()) && Regex.IsMatch(tbDepatureTime.Text.Trim().ToString(), time.ToString()))

{

string[] val\_arival = tbArrivalTime.Text.Trim().ToString().Split(':');

string[] val\_departure = tbDepatureTime.Text.Trim().ToString().Split(':');

if (int.Parse(val\_arival[0]) < 24 && int.Parse(val\_departure[0]) < 24 && int.Parse(val\_arival[1]) < 60 && int.Parse(val\_departure[1]) < 60)

{

if ((int.Parse(val\_arival[0]) > int.Parse(val\_departure[0])) || ((int.Parse(val\_arival[0]) == int.Parse(val\_departure[0])) && (int.Parse(val\_arival[1]) > int.Parse(val\_departure[1]))))

{

string command = "insert into routes(route\_id, route\_name, depature\_time, arrival\_time, train\_id) values(" + (((routes[routes.Count - 1].route\_id) + 1).ToString()) + ",'" + tbRouteName.Text.Trim().ToString() + "','" + tbDepatureTime.Text.Trim().ToString() + "','" + tbArrivalTime.Text.Trim().ToString() + "'," + trains[rnd.Next(trains.Count - 1)].train\_id.ToString() + ")";

db.ExecuteNonQuery(ref stp, command);

}

else

{

if (MessageBox.Show("Состав прибывает на следующий день?", "Dialog", MessageBoxButtons.OKCancel) == DialogResult.OK)

{

string command = "insert into routes(route\_id, route\_name, depature\_time, arrival\_time, train\_id) values(" + (((routes[routes.Count - 1].route\_id) + 1).ToString()) + ",'" + tbRouteName.Text.Trim().ToString() + "','" + tbDepatureTime.Text.Trim().ToString() + "','" + tbArrivalTime.Text.Trim().ToString() + "'," + trains[rnd.Next(trains.Count - 1)].train\_id.ToString() + ")";

db.ExecuteNonQuery(ref stp, command);

}

else

{

MessageBox.Show("Проверьте введенные данные!", "Dialog", MessageBoxButtons.OK);

}

}

}

else

{

MessageBox.Show("Проверьте введенные данные!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Введите все данные правильно!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Направление с таким именем уже существует!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Заполните все поля!","Dialog",MessageBoxButtons.OK);

}

}

private void tbDepatureTime\_KeyPress(object sender, KeyPressEventArgs e)

{

char ch = e.KeyChar;

if (!Char.IsDigit(ch) && ch != 8)

{

e.Handled = true;

}

}

private void tbArrivalTime\_KeyPress(object sender, KeyPressEventArgs e)

{

char ch = e.KeyChar;

if (!Char.IsDigit(ch) && ch != 8)

{

e.Handled = true;

}

}

private void AddNewRoute\_Load(object sender, EventArgs e)

{

}

}

}

Код форми «AddNewStaff»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace TimeToDoIt

{

public partial class AddNewStaff : Form

{

ClassDataBase db = new ClassDataBase();

ClassSetupProgram stp = new ClassSetupProgram();

List<Employee\_Position> employee\_positions = new List<Employee\_Position>();

List<Employees> employees = new List<Employees>();

List<Positions> positions = new List<Positions>();

Regex phone = new Regex(@"\b\d{10}\b");

Regex names = new Regex(@"\b(\w+?)\b");

Regex salary = new Regex(@"\b\d{3,}\b");

Regex date = new Regex(@"\b\d{1,2}\/\d{1,2}\/\d{4}\b");

void LoadData()

{

employees.Clear();

employee\_positions.Clear();

positions.Clear();

db.Execute<Employee\_Position>(ref stp, "Select \* from employee\_positions", ref employee\_positions);

db.Execute<Employees>(ref stp, "Select \* from employee", ref employees);

db.Execute<Positions>(ref stp, "Select \* from positions", ref positions);

}

public AddNewStaff()

{

InitializeComponent();

}

private void AddNewStaff\_Load(object sender, EventArgs e)

{

LoadData();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

if ((tbFirstName.Text.Trim().ToString() != "") &&

(tbSecondName.Text.Trim().ToString() != "") &&

(tbTelephone.Text.Trim().ToString() != "") &&

(tbSalary.Text.Trim().ToString() != "") &&

(tbPositionName.Text.Trim().ToString() != ""))

{

if (Regex.IsMatch(tbFirstName.Text.Trim().ToString(), names.ToString()) &&

Regex.IsMatch(tbSecondName.Text.Trim().ToString(), names.ToString()) &&

Regex.IsMatch(tbTelephone.Text.Trim().ToString(), phone.ToString()) &&

Regex.IsMatch(tbSalary.Text.Trim().ToString(), salary.ToString()) &&

Regex.IsMatch(tbPositionName.Text.Trim().ToString(), names.ToString()))

{

if (dtpHireDate.Value.Year < 2019)

{

int position\_id = (positions[positions.Count - 1].position\_id) + 1;

int employee\_id = (employees[employees.Count - 1].employee\_id) + 1;

int emp\_pos\_id = (employee\_positions[employee\_positions.Count - 1].empl\_pos) + 1;

string command\_position = "insert into positions(position\_id, position\_name, salary) values(" + position\_id.ToString() + ",'" + tbPositionName.Text.Trim().ToString() + "'," + tbSalary.Text.Trim().ToString() + ")";

db.ExecuteNonQuery(ref stp, command\_position);

string command\_employee = "insert into employee(employee\_id, first\_name, last\_name, telephone, hire\_date) values(" + employee\_id.ToString() + ",'" + tbFirstName.Text.Trim().ToString() + "','" + tbSecondName.Text.Trim().ToString() + "','" + tbTelephone.Text.Trim().ToString() + "','" + dtpHireDate.Value.Day.ToString() + '-' + dtpHireDate.Value.Month.ToString() + '-' + dtpHireDate.Value.Year.ToString() + "')";

db.ExecuteNonQuery(ref stp, command\_employee);

string command\_employee\_position = "insert into employee\_positions(id\_ep, employee\_id, position\_id) values(" + emp\_pos\_id.ToString() + "," + employee\_id.ToString() + "," + position\_id.ToString() + ")";

db.ExecuteNonQuery(ref stp, command\_employee\_position);

if (MessageBox.Show("Запись добавлена. Номер сотрудника " + employee\_id.ToString() + "!", "Dialog", MessageBoxButtons.OK) == DialogResult.OK)

{

this.Close();

}

}

else

{

MessageBox.Show("Введите дату верно!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Заполните все поля правильно!", "Dialog", MessageBoxButtons.OK);

}

}

else

{

MessageBox.Show("Заполните все поля!","Dialog",MessageBoxButtons.OK);

}

}

}

}