Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Кафедра ПІ

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КУРСОВОЇ РОБОТИ

з дисципліни “Бази даних”

Тема роботи: «Інформаційна система «Облік на автотранспортному підприємстві»

Виконав

ст. гр. ПЗПІ-21-11 Коханевич П.Г.

Керівник:

ст. викл. каф. ПІ Широкопетлєва М.С.

Робота захищена на оцінку \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Комісія:

доц. каф. ПІ Русакова Н.Є

ст. викл. каф. ПІ Черепанова Ю.Ю.

ст. викл. каф. ПІ Широкопетлєва М.С.

Харків, 2022 р.

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

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

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

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

**Курс** \_\_\_2\_\_\_\_ **Група** \_\_\_ПЗПІ-21-11\_\_\_\_\_\_\_\_ **Семестр**\_\_\_\_ 3\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

\_\_\_\_\_\_\_\_\_\_\_*Коханевича Павла Григоровича* \_\_\_\_\_\_\_\_\_\_\_\_

**1. Тема роботи:** \_\_\_\_«Інформаційна система «*Облік на автотранспортному*

\_*підприємстві*\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Строк здачі закінченої роботи** \_\_\_13.01.2023\_\_\_\_\_\_\_\_

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

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

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

**6. Дата видачі завдання** \_\_\_15.09.22 р.\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Аналіз предметної області | 15.09.22 – 20.09.22 | Виконано |
| 2 | Постановка задачі | 21.09.22 – 27.09.22 | Виконано |
| 3 | Побудова ER-діаграми бази даних | 28.09.22 – 10.10.22 | Виконано |
| 4 | Оформлення розділів 1, 2 та 3.1, 3.2 пояснювальної записки | 11.10.22 - 23.10.22 | Виконано |
| 5 | Перша контрольна точка з курсового проекту | 24.10.22 – 30.10.22 | Виконано |
| 6 | Нормалізація бази даних | 30.10.22 - 18.11.22 | Виконано |
| 7 | Створення демо-версії програми | 30.10.22 – 20.11.22 | Виконано |
| 8 | Тестування програми, наповнення бази даних | 21.11.22 - 24.11.22 | Виконано |
| 9 | Друга контрольна точка з курсового проекту | 25.11.22– 03.12.22 | Виконано |
| 10 | Реалізація остаточної версії програми | 03.12.22-13.12.22 | Виконано |
| 11 | Оформлення інших розділів пояснювальної записки | 03.12.22 – 15.12.22 | Виконано |
| 12 | Захист курсового проекту (третя контрольна точка) | 16.12.22- 13.01.23 | Виконано |

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

Керівник \_\_\_\_ ст. викл*. Широкопєтлева М.С.*

15.09.2022 р.

РЕФЕРАТ

Пояснювальна записка містить у собі: 68 сторінок, 4 основні частини, які поділяють на підпункти, 27 ілюстрацій, 1 додаток, 5 джерел по переліку посилань.

АВТОТРАНСПОРТНЕ ПІДПРИЄМСТВО, АВТО, АТРИБУТ, БАЗА ДАНИХ, ВОДІЙ, ПРОГРАМА, ФУНКЦІЇ.

Метою роботи являється реалізація і детальне проектування бази даних для обліку автотранспортного підприємства, а також формування програмного забезпечення, яке дозволить зручно працювати із інформаційною системою підприємства.

Під час розробки бази даних потрібно використовувати такі методи рішення задач, як аналіз предметної області та існуючих аналогів, які дадуть краще розуміння майбутньої бази даних, концептуальне проектування, формування ER-діаграми, нормалізація бази, створення логічної моделі.

Об’єкт дослідження – розробка інформаційної системи, реалізована бази даних і програма, яка дозволить керувати базою та буде задовольняти основні потреби співробітника працюючого на автотранспортному підприємстві .

Результатом виконання курсової роботи є інформаційна система, яка була сформована під вимоги предметної області. Вона дозволяє проводити пошук, сортування і фільтрацію по обраних критеріях певного об’єкта предметної області, а також редагувати, додавати, видаляти певні об’єкти. Крім того, результатом виконання можна вважати спроектовану та реалізовану базу даних, яка відповідає основним вимогам третьої нормальної форми, яка дозволяє оптимізувати роботу бази даних.

**ЗМІСТ**

[Вступ 6](#_Toc123550260)

[1Аналізтаконцептуальнемоделюванняпредметної області 7](#_Toc123550261)

[1.1 Аналіз предметної області 7](#_Toc123550262)

[1.2 Аналіз існуючих аналогів 8](#_Toc123550263)

[1.3 Концептуальне моделювання предметної області 10](#_Toc123550264)

[2 Постановка задачі 19](#_Toc123550265)

[3Проектування бази даних 21](#_Toc123550266)

[3.1 Побудова ER-діаграми 21](#_Toc123550267)

[3.2 Побудова логічної моделі бази даних на основі ER-діаграми 24](#_Toc123550268)

[3.3 Побудова логічної моделі бази даних шляхом нормалізації 30](#_Toc123550269)

[4Опис програми 41](#_Toc123550270)

[4.1 Загальні відомості 41](#_Toc123550271)

[4.2 Виклик і завантаження 42](#_Toc123550272)

[4.3 Призначення і логічна структура 43](#_Toc123550273)

[4.4 Описання фізичної моделі бази даних 45](#_Toc123550274)

[4.5 Опис програмної реалізації 51](#_Toc123550275)

[4.6 Описання задачі автоматизації 61](#_Toc123550276)

[Висновки 64](#_Toc123550277)

[Перелік джерел посилання 65](#_Toc123550278)

[Додаток А Код реалізації задачі автоматизації 66](#_Toc123550279)

**ВСТУП**

Розвиток сучасних технологій стрімко зростає з кожним днем. Багато технологій, які пов’язані із глобальною мережею Інтернет так чи інакше працюють із певною інформацією, яка з часом збільшується. На даний момент кількість інформації, яка знаходиться в мережах і фізичних носіях вимірюється у мільярдах терабайтів. Сучасні комп’ютери мають доволі потужні характеристики, але навіть вони не зможуть впоратися із такою кількістю даних, якщо вона буде хаотичною розподілена і не матиме логічного зв’язку.

Безумовно можна погодитися, що зберігати дані в електронному форматі в кілька разів краще ніж в друкованому. Багато великих підприємств зберігають свої облікові дані в електронному форматі, але кількість даних, що там знаходиться також має великий обсяг і виникає гостра потреба мати зручний доступ до них, який не вимагав велику кількість часу. Враховуючи ці фактори можна сказати, що інформацію потрібно якимось чином розбивати, щоб мати більш легкий доступ до них. Особливо такі проблеми важливі для підприємств, які мають велику кількість даних, яку потрібно максимально швидко змінювати, наприклад автотранспортне підприємство, яке займається перевезенням товарів і містить велику кількість авто і водіїв. Проблемами внесення нових авто і водіїв мають займатися певні співробітники на цьому підприємстві, але вони повинні мати зручний доступ до цієї інформації і мати можливість з нею працювати.

Отже, мета даної курсової роботи полягає у проектуванні бази даних та створені програмного забезпечення, яке надає співробітникам автотранспортного підприємства можливість зручної роботи із обліковими даними на цьому підприємстві, а головне швидко виконувати маніпуляції із інформацією, а саме додавати, видаляти, редагувати і фільтрувати дані.

# **1 АНАЛІЗ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ**

## Аналіз предметної області

Дана курсова робота розглядає предметну область <Облік на автотранспортному підприємстві>. Можна сказати, що автотранспортні підприємства мають велику кількість особливостей, але можна виділити основні, які дозволять сформувати загальне розуміння про підприємство. Відслідковування великої кількості інформації на підприємстві і формування документації потребує злагодженої системи, яка дозволяє диспетчерам і адміністраторам впевнено виконувати свої обов’язки і кожне підприємство має таку систему.

Загалом будь-яке автотранспортне підприємство мусить мати автомобілі, трудові договори, водіїв, різні деталі до авто, автопарковки, перелік наявних рейсів, тому що саме з даними об’єктами працює будь-яке автотранспортне підприємство.

Візьмемо за основного користувача диспетчера автотранспортного підприємства. Загалом диспетчера мають формувати та відслідковувати рейси, які належать водіям на цей час, також вони мають змогу переглянути інформації про всіх водіїв, автомобілів, автопарків для кращої оцінки сформованих рейсів. Результатом діяльності диспетчера являється правильно сформовані рейси із врахуванням факторів, щодо складності певного рейса й аналізу доступних водіїв.

Підприємства можуть мати адміністраторів, які мають справу із аналізом інформації про певних водіїв і автомобілів, а також редагують і змінюють інформацію про всі основні об’єкти, щоб система підприємства функціонувала краще та не мала зайвої чи застарілої інформації. Результатом діяльності адміністратора являється злагоджена робота підприємства і можливість швидко надати інформацію про об’єкт власникам підприємств або іншим особам , які вимагають наявність окремої інформації.

Можна виділити процес, а саме прийняття водія на роботу, відділ кадрів має прийняти нового водія на роботу, скласти трудовий договір, потім адміністратор згідно наданої інформації про водія заносить його до системи вручну, а також його трудовий договір. Диспетчер, переглядаючи інформацію про водіїв, залучає його до роботи, тобто формує рейси за участі водія також вручну.

## Аналіз існуючих аналогів

Системи обліку автотранспортного підприємства у вільному доступі знайти доволі складно, особливо в безкоштовному доступі, тому для аналізу було обрано певну платну програму.

Дана програма називається “BAS Управління автотранспортом Стандарт” [1]. З усіх доступних аналогів, саме дана програма описана дуже детально і можна провести повний аналіз її функціоналу. (див.рис. 1.1).

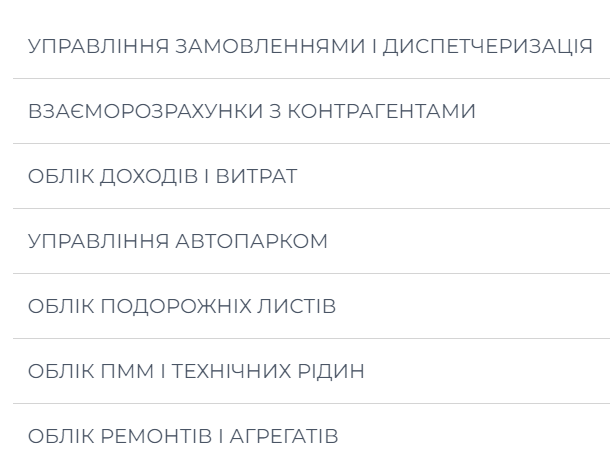


Рисунок 1.1 – Функції програми

Можна сказати, що програма містить безліч цікавих і корисних функцій, які справді можуть допомогти підприємству слідкувати за роботою.

Програма має багатокористувацькі можливості, тобто різні працівники можуть мати різні доступи до інформації і паралельно виконувати свої задачі. Вона охоплює багато різних предметних областей автотранспортного підприємства, що робить дану програму універсальною і дуже складною у реалізації.

Звісно кожна програма може мати свої недоліки і дана програма не виключення. Звернемо увагу на облік водіїв, дана програма може виконувати багато задач автоматизації, але на мою думку тут пропущена ще одна, не менш важлива задача автоматизації, а саме перевірка термінів трудового договору, оскільки водій, який виконує свою роботу із закінченим терміном договору може не отримати стажу або навіть мати проблеми із зарплатою за цей час. Можна знайти ще один цікавий момент у призначені програми, тому що вона орієнтована на доволі розвинену структуру підприємства та різними видами доступу до даних, але для середніх і маленьких автотранспортних підприємств буде краще, якщо диспетчер й адміністратор матимуть однаковий рівень доступу, оскільки у випадках непередбачуваної ситуації вони завжди можуть замінити один одного або деякий час працювати із іншими даними для покращення якості роботи.

Для кращого розуміння інформаційних систем, які базуються на автотранспортному підприємстві потрібно розглянути програму “jSolutions” [2]. Дана програма містить велику кількість різних функцій, які можуть знадобитися більшості співробітникам на підприємстві. Звісно програма має багатокористувацький режим, але загалом система створювалася для диспетчерів та заправників.

Основною перевагою даного продукту можна вважати складні та якісні аналізи роботи підприємства, оскільки програма навіть враховує збитки та кількість дорожньо-транспортних пригод, а також кількість можливим збитків, які завданні через невчасну доставку певного вантажа.

Для вирішення оперативних завдань роботи транспортних підрозділів підприємств і компаній в системі реалізовані функції, що дозволяють поставити на потік обробку і фіксацію цілий ряд параметрів, які є ключовими для оперативного обліку. Аналіз діяльності компанії завжди пов'язаний з формуванням цілого ряду звітів і зведень. На підставі первинної звітності зручно консолідувати інформацію і формувати управлінські показники. Так і в програмі “jSolutions” для автотранспорту, використовуючи графічний редактор, можна побудувати цілий ряд звітів такі як: відомість транспортних послуг, реєстр руху подорожніх листів, подорожній лист, відомість витрат палива, контроль подорожніх листів, звіт про роботу автотранспорту (див.рис. 1.2).

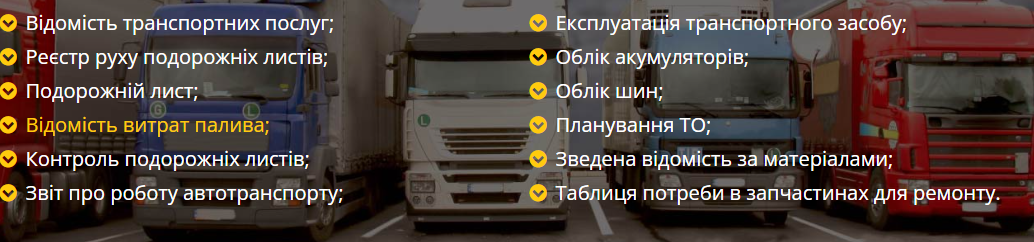


Рисунок 1.2 – Функції для звітів

Можна сказати, що дана програма містить потужний функціонал, який дозволяє контролювати стан автомобілів та оптимізувати витрати на підприємстві, але існує доволі суттєвий недолік. Основним недоліком можна вважати відсутність обліку водіїв, що свідчить про недостатній набір функціоналу для повноцінного контролю підприємства.

## Концептуальне моделювання предметної області

Почнемо з опису функціонального призначення інформаційної системи (ІС). Аналізуючи предметну область обліку автотранспортного підприємства ми можемо сказати, що основними користувачами даної інформаційної системи будуть диспетчера, а також певні адміністратори (див.рис. 1.2). Дана модель добре описує основні задачі працівників і ми можемо аналізувати певні моменти. На цій моделі бізнес-прецедентів ми можемо сказати, що адміністратор і диспетчер мають однаковий рівень доступу і можуть виконувати при необхідності спільні задачі разом або легко переглядати всю інформацію при необхідності. Даний підхід надає змогу маленьким і середнім підприємствам оптимізувати свою роботу і надати основним співробітникам всю необхідну інформацію, оскільки під час роботи не буде виникати питання доступу до різних типів інформації.

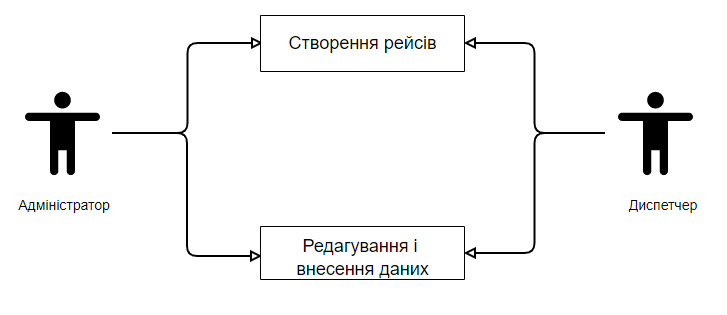


Рисунок 1.2 - Модель бізнес-прецедентів

Важливим моментом концептуального моделювання предметної області є опис концептів предметної області, їх властивостей та зв’язків між ними. При виборі предметної області ми можемо виділити елементи даної області (див.рис. 1.3).

Вони дають розуміння, що структура даної області доволі широка і вона дозволяє отри мати повноцінне уявлення про повну роботу автотранспортного підприємства. Звісно підприємство може мати багато інших об’єктів, але все залежить від специфікації підприємства і його основної задачі.

При аналізі загальної діаграми класів можна сказати, що там враховані всі ключові інформаційні об’єкти автотранспортного підприємства, які лише можуть існувати та використовуватися.

Оскільки діаграма має загальний характер, тому при її реалізації були опущенні атрибути, які характерні для даних об’єктів.

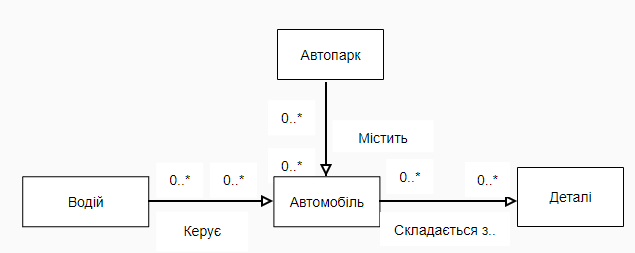


Рисунок 1.3 - Загальна діаграма класів

Можна сказати, що основними елементами предметної області є водії, автомобілі, автопарки, а також деталі.

Для повного розуміння суті об’єктів потрібно детально розібрати атрибути або їхні характеристики, які містяться в кожному об’єкті та стосуються лише їхніх інформаційних особливостей:

* водій повинен мати такі властивості: ім’я, прізвище, по-батькові, рік народження, номер трудового договору, дата початку і закінчення договору, зарплата, стать, електрону пошту, категорію прав;
* автопарк повинен мати: площу, кількість місць для авто, адрес, а саме область, район, місто, вулицю, номер будівлі;
* автомобіль містить у собі: рік випуску, колір, назву моделі, бренд, максимальну тягову здатність, назву палива, ціну за літр;
* деталь містить у собі: назву деталі, ціна деталі за однин екземпляр, ширину і висоту деталі, та кількість деталей такого типу в одній моделі.

Можемо сказати, що дані характеристики повністю підходять до об’єктів.

Далі потрібно трішки детальніше розглянути зв’язки, що існують між об’єктами:

* <водій - автомобіль> він має тип зв’язку “багато до багатьох”, оскільки водій може мати різні автомобілі при різних рейсах, так само і автомобіль може належати багатьом водіям залежно від рейсу;
* <автопарк - автомобіль> він має тип зв’язку “багато до багатьох”, оскільки автомобіль може мати різні парковки залежно від часу;
* <автомобіль - деталі> він має тип зв’язку “багато до багатьох”, оскільки однин автомобіль може складатися із багатьох деталей та деталі можуть належати лише до багатьох автомобілей;

Тепер потрібно розглянути опис інформаційних потреб користувачів. Можна почати із роботи адміністратора, якому потрібно швидко занести нових співробітників та авто в базу. Щоб полегшити роботу він повинен мати змогу сортувати, фільтрувати і мати функцію пошуку автомобіля та водія за їхніми назвами, кольором, максимальною тяговою здатністю, можливо датою народження або датою випуску залежно від об’єкта. Він повинен мати змогу проглядати всю доступну інформацію про автопарковки та інші об’єкти інформаційної системи, щоб аналізувати стан підприємства. Крім того адміністратор повинен мати змогу працювати із інформацією тобто редагувати, видаляти і додавати. Може виникнути потреба у формуванні певних статистик, особливо найкращого водія чи навіть автомобіля, щоб приписати певні нагороди чи послуги. Формування звітів по водію та деталей, які наявні на підприємстві також охоплює інформаційні потреби адміністратора. Щоб дізнатися певні об’єкти, які виділяються своїми елементи, наприклад найкращі водії чи автомобілі мають знаходитися швидко за допомогою одного запита.

Диспетчер також має потребу переглядати і працювати із доступною інформацією, яка наявна в системі. Основною задачею диспетчера це мати змогу формувати рейси і редагувати їх. Оскільки диспетчер має також аналізувати деталі, то формування звіту щодо деталей, які має підприємство також входить до його інформаційних потреб. Аналізу популярності палива серед автомобілів допомагає диспетчерові аналізувати, які найкраще авто обирати для перевезення. Всі інформаційні потреби ми можемо побачити на повній моделі бізнес-прецедентів (див.рис. 1.4).

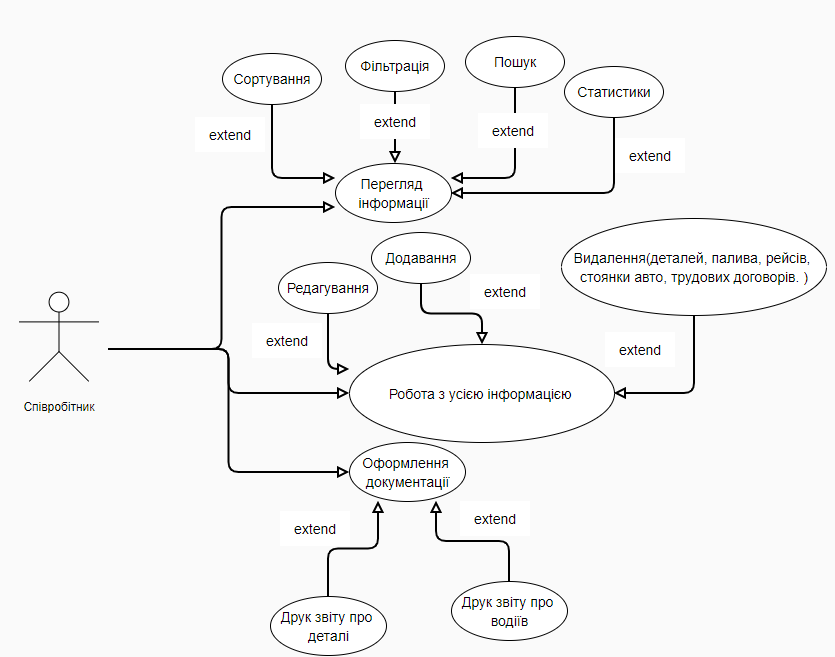


Рисунок 1.4 - Повна модель бізнес-прецедентів

Під час опису інформаційних потреб користувачів обов’язково необхідно виділити та описати задачу автоматизації.

Головна ідея автоматизація в даному курсовому проекту полягає в тому, що користувачі даної інформаційної системи зможуть не перейматися питанням дати закінчення трудового договору, оскільки програмне забезпечення буде автоматично визначати таких працівників та відправляти повідомлення на електронну пошту. Це зможе облегшити життя працівників компанії і зберегти багато часу. Потреба в автоматизації діяльності перевірки полягає в тому, що водіїв може бути багато і визначити вручну в кого сплив термін трудового договору може стати великою проблемою. Детально процеси автоматизації можна розглянути на діаграмі видів діяльності для задачі автоматизації (див.рис. 1.5).



Рисунок 1.5 - Діаграма видів діяльності для задачі автоматизації

Потрібно приділити увагу також документообігу на автотранспортному підприємстві, тому що інформація, яка може знадобитися працівникам і відіграє важливу роль у контролі інформаційних процесів, потребує окремого оформлення і зберігання. Це може бути корисно, коли потрібно швидко дізнатися інформацію про подорожній лист. Подорожній лист - первинний документ, призначений для обліку вантажних автомобільних перевезень, що всебічно характеризує роботу водія й автомобіля з моменту їхнього виїзду з автотранспортного підприємства і до повернення на підприємство. Потрібно виділити ще один важливий документ, який призначений для списання товарно-матеріальних цінностей, обліку на шляху їхнього проходження, оприбуткування, складського, оперативного і бухгалтерського обліку, а також для розрахунків за перевезення вантажів і обліку виконаної роботи, тобто це товарно-транспортна накладна.

На кожну поїздку автомобіля і для кожного вантажоодержувача виписується окрема товарно-транспортна накладна. Ця вимога поширюється на всі види вантажних перевезень автомобільним транспортом, незалежно від умов оплати за його роботу і видів перевезень.

Продовжимо з опису алгоритмічних залежностей. Візьмемо питання формування звіту деталі і визначення загальної вартості всіх наявних деталей цього екземпляру на підприємстві.

Обрахунок кількості наявних деталей одного екземпляру:

;

де: k – кількість деталей, які є в підприємстві;

m – кількість автомобілів, які належать до тієї моделі, яка використовує дану деталь;

d – кількість деталей одного виду, яка має одна модель.

Тепер потрібно проаналізувати область обмеження цілісності. Дана предметна області має деякі обмеження ідентифікації, а саме:

* водій може мати однакове повне ім’я і рік народження, тому водія можна легко ідентифікувати за унікальним номером;
* автомобіль також може мати спільний колір, модель, рік випуску, тому краще його ідентифікувати за унікальним номером;
* модель містить в собі унікальну назву моделі, тому ідентифікувати її можна лише за назвою;
* деталі можуть мати багато спільних характеристик такі як: розмір, назву, ціну, тому краще ідентифікувати деталі за унікальним номером;
* трудовий договір має свій номер, який не може повторюватися, тому він ідентифікується саме за ним;
* автопарковки має лише одного типу парковочних місць для авто;
* водій може мати лише категорію прав, які стосуються тягачів і вантажних автомобілів тобто це категорії: С1, С1Е, СЕ, С1. Якщо водій має дві категорії, які підходять під цей опис, то обирається найвища категорія.

Розглянемо обмеження стосовно відношення між поняттями:

* автопарковка має належати лише одному адресу і один адрес ніколи не має декілька автопарковок;
* водій може мати декілька трудових договорів, оскільки вони можуть продовжуватися;
* один екземпляр деталі може належати виключно одній моделі авто;
* рейси, які будуть формуватися у результаті поєднання водіїв і авто, мають в собі лише один напрям, тобто рейс повернення назад буде рахуватися, як окремий рейс.

Потрібно звернути увагу на вимоги до ІС та існуючі обмеження. Почнемо розглядати можливі вимоги до ІС. До них можна віднести зручність інтерфейсу, достатню кількість функціоналу, щоб спокійно виконувати свою роботу, певні окремі функції, які можуть швидко зробити найбільш популярні алгоритми дій в ІС. Щодо існуючих обмежень, то до них можна віднести використання сучасних операційних систем, рекомендовано Windows 10 та Windows 11, які забезпечать в певній мірі надійну роботу ІС. Можна виділити системи управління базою даних (СУБД), які гарно поєднуються із даною інформаційною системою: Oracle, MySQL [3], SQL Server.

Можна зробити висновок, що вимоги, які стосуються технічних характеристик комп’ютерів не мають чітких обмежень і носять рекомендаційний характер.

Для кращого розуміння концептуального моделювання можна розглянути пару лінгвістичних відносин, які дозволять краще зрозуміти предметну область і швидше аналізувати інформацію:

* диспетчер автотранспорту — співробітник, який контролює рух автотранспорту, а також приймає повідомлення від водіїв та клієнтів, який відповідає за ведення документації. Також даний працівник спеціалізується на побудові оптимальних маршрутах, які мають найменші фінансові затрати;
* модель - це тип автомобіля (тип кузова), що випускається під певною маркою і має унікальну назву. Моделі можуть мати назви, які комбінують в собі набір літер та чисел;
* трудовий договір - це угода між працівником і власником підприємства, установи, організації або уповноваженим ним органом чи фізичною особою, за якою працівник зобов’язується виконувати роботу, визначену цією угодою та має дотримуватися цих вимог до кінця терміну договору.

# **ПОСТАНОВКА ЗАДАЧІ**

Потрібно створити інформаційну систему «Облік на автотранспортному підприємстві», використовуючи детальний аналіз предметної області, спроектувати та реалізувати ІС «Облік на автотранспортному підприємстві».

Перелік інформації, що повинна бути наявною в БД:

* інформація про водія (повне ім’я, рік народження, стать, номер трудового договору, категорія прав);
* інформація про автомобіль (інформація про модель, рік випуску, колір);
* інформація про трудовий договір ( номер, терміни дії);
* інформація про моделі (бренд, тягова здатність, паливо);
* інформація про автопарковки (адрес, площа, кількість місць);
* інформація про адреса автопарковок (місто, вулиця, будівля, район);
* інформація про деталі (назва, розмір, ціна);
* інформація про рейси (номер водія, автомобіля, місто і час прибуття і відбуття);
* інформація про паливо (назва і ціна).

Функції, що повинна підтримувати інформаційна система :

* додавання, редагування всієї інформації. Видалення (деталей, палива, рейсів, стоянки автомобіля, трудового договору), після видалення інші сутності, які місять інформацію про ці об’єкти будуть мати пусте поле замість первинних ключів видалених об’єктів;
* пошуку (Водія за ім’ям, прізвищем. Автомобіля за моделлю, кольором. Модель за брендом. Деталь за назвою. Парковка за адресою);
* фільтрація (Деталі: цін на деталі. Автомобіль: рік випуску автомобіля. Модель: максимальна тягова здатність. Водій: рік народження водія. Бензин: ціна на бензин. Автопарковка: кількість місць та площа);
* сортування (Водія за роком народження від найменшого до найбільшого. Моделі за максимальною тяговою здатністю від найбільшої до найменшої. Автопарковки за кількістю парковочних місць від найменшого до найбільшого).

Формулювання запитів на статистику:

* найкращий автомобіль в підприємстві (Основний критерій: максимальна тягова здатність. Додатковий: рік випуску);
* найкращий водій в підприємстві (Основний критерій: найбільша кількість рейсів. Додатковий: наймолодший вік);
* всі види палива, які є в підприємстві та кількість авто, що їх використовують;
* найбільша автопарковка в підприємстві (Основний критерій: площа. Додатковий: кількість місць).

Перелік звітів, що повинна формувати система:

* формування звіту по водію (ім’я, прізвище, по-батькові, рік народження, пошта, стать, категорія водійських прав);
* формування каталогу деталей (назва деталі, розмір, ціна, модель до якої вона підходить, кількість такого виду деталей в підприємстві, кількість деталей в одній моделі).

Задача автоматизації полягає в перевірці термінів трудового договору на термін кінця і якщо до кінця залишилося не більше трьох днів або він вже вичерпаний, то буде надсилатися повідомлення водію на пошту, який закріплений за певним трудовим договором поки водій не звернеться до відділу кадрів і не змінить договір, щоб адміністратор зміг спокійно занести новий договір в систему і видалити старий.

# **3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ**

## 3.1 Побудова ER-діаграми

Для того щоб спроектувати якісну базу даних потрібно побудувати ER-діаграму, яка допоможе більш точно сформувати розуміння майбутньої бази даних (див.рис. 3.1).

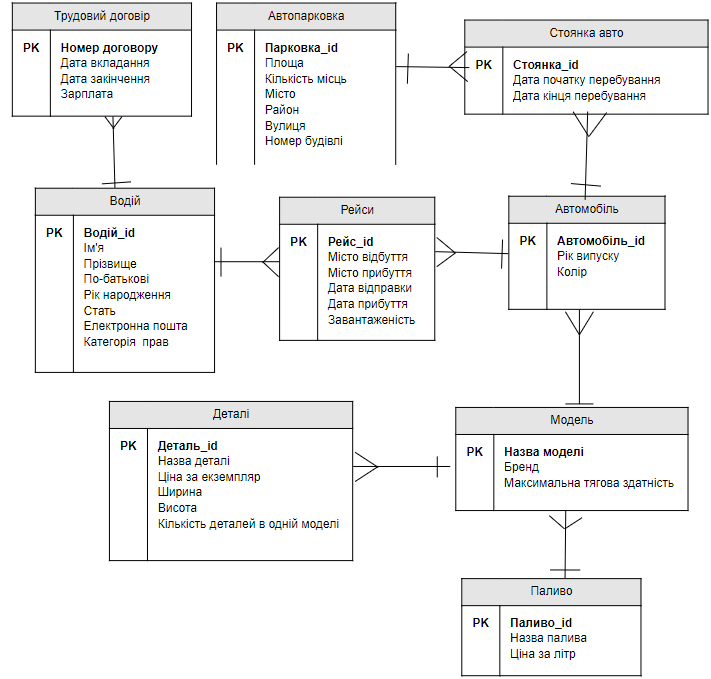


Рисунок 3.1 - ER-діаграма предметної області <Облік автотранспортного підприємства>

На підставі концептуальної моделі предметної області, ми можемо виділити стрижневі сутності: автомобіль, водій, автопарковки, деталі. Виділимо характеристичні сутності: трудовий договір, модель, паливо. Потрібно виділити асоціативні сутності також, а саме рейси та стоянки авто. Всі вони побудовані на базі основних об’єктів, які описані в концептуальній моделі. Крім основних об’єктів та їх атрибутів, які описані в концептуальному аналізі ми отримали нові сутності, які допомагають спроектувати ER – діаграму.

Перейдемо до переліку атрибутів, що містять дані сутності:

* трудовий договір має: номер договору, дата вкладання, дата закінчення, зарплата. Первинний ключ: номер договору;
* водій: водій\_id, ім’я, прізвище, по-батькові, рік народження, стать, категорія прав, електронна пошта. Первинний ключ: водій\_id;
* рейси: рейс\_id, дата відбуття, дата прибуття, місто відбуття, місто прибуття, завантаження. Первинний ключ: рейс\_id;
* адрес автопарковки: адрес\_id, місто, район, вулиця і номер будинку;
* автопарковка: автопарковка\_id, площа, кількість місць. Первинний ключ: автопарк\_id;
* стоянка авто: стоянка\_id, дата початку перебування, дата кінця перебування. Первинний ключ: стоянка\_id;
* автомобіль: автомобіль\_ id, рік випуску, колір. Первинний ключ: автомобіль\_id;
* модель: назва моделі, бренд, максимальна тягова здатність. Первинний ключ: назва моделі;
* паливо: паливо\_id, назва палива, ціна за літр. Первинний ключ: паливо\_id ;
* деталі: деталь\_id, назва деталі, ціна за екземпляр, ширина, висота. Первинний ключ: деталь\_id ;
* кількість в моделі: кількість\_id, кількість деталей в одній моделі. Первинний ключ: кількість\_id.

Враховуючи, що деякі зв’язки були описані в п.1.3, потрібно виділити увагу зв’язкам, які були розбиті або зміненні між сутностями для кращого проектування бази даних:

* <автопарк - стоянка> - він має тип зв’язку “один до багатьох”, оскільки стоянка це проміжна таблиця і один автопарк може мати багато стоянок в різний час для авто;
* <автомобіль - стоянка> - він має тип зв’язку “один до багатьох”, оскільки авто може мати різний час стоянки в різних парковках;
* <водій - рейси> - він має тип зв’язку “один до багатьох”, оскільки водій може мати багато рейсів в різний час, але рейс належить лише одному водію;
* <водій – трудовий договір> він має тип зв’язку “один до багатьох”, оскільки водій може декілька договорів договір, а договір може належати лише одному водію;
* <автомобіль - рейси> - він має тип зв’язку “один до багатьох”, оскільки авто може мати багато рейсів в різний час, але рейс належить лише одному авто;
* <деталь – модель> - він має тип зв’язку “багато до одного”, оскільки одна деталь може належати лише одній моделі, а модель може складатися із багатьох деталей;
* <паливо - модель> він має тип зв’язку “один до багатьох”, оскільки одне паливо може бути у багатьох автомобілів, але лише одне паливо може мати автомобіль;
* <модель - автомобіль> він має тип зв’язку “один до багатьох”, оскільки одна модель може належати багатьом автомобілям, але автомобіль має лише одну модель.

## 3.2 Побудова логічної моделі бази даних на основі ER-діаграми

У якості логічної моделі баз даних було обрано реляційні БД, так як реляційні бази даних відображають інформацію в найбільш простому для користувача вигляді.

Вони надають простоту і доступність для логіки користувача і систему. Суворі правила проектування допомагають зробити базу даних ефективною і оптимальною для будь-якої предметної області. Реляційні бази даних надають повну незалежність даних. Це гарно поєднується із обліком автотранспортного підприємства, оскільки користувачам потрібна проста логіка роботи і можливість легко працювати із інформацією. Для створення програмного забезпечення і на обслуговування її розробники витрачають мало часу, що дозволяє зберегти час підприємству.

Проаналізувавши доречність використання реляційної бази даних потрібно описати, яким чином з використанням первинних та зовнішніх ключів на основі ER-діаграми буде побудована схема реляційної бази даних. Отже потрібно почати із стрижневої сутності водій, яка об’єднана із трудовим договором зв’язком “один до багатьох”, оскільки водій може мати багато трудових договорів, тому первинний ключ сутності водія, тобто це Водій\_id, потрібно передати до трудового договору, як звичайний атрибут. Він буде слугувати зовнішнім ключем трудового договору. Водій також прив’язаний до асоціативної сутності рейси, які дозволять об’єднати інформацію про водія та автомобіль, тому первинні ключі водія , а також автомобіля потрібно передати, як атрибути до таблиці з рейсами. Як наслідок, первинні ключі Водій\_id та Автомобіль\_id стануть зовнішніми ключами в таблиці з рейсами.

Автопарковки та автомобілі мають зв’язок “один до багатьох” до асоціативної сутності Стоянка авто, що буде характеризувати авто та саму автопарковку, тому потрібно первинні ключі Парковка\_id та Автомобіль\_id передати до Стоянки авто, щоб отримувати інформації про них. Паливо має зв’язок “один до багатьох” із моделлю, тому потрібно надати сутності, яка характеризує модель атрибути, які дадуть змогу отримувати інформацію про паливо, оскільки моделі можуть мати різне паливо. Паливо\_id стає зовнішнім ключем у таблиці Модель. Можна зробити висновок, що автомобіль також має містити інформацію про модель, тому доречно передати сутності автомобіля атрибут, який стане зовнішнім ключем, а саме назву моделі.

Модель має зв’язок із деталями “один до багатьох”, оскільки деталі зазвичай створюють окремо для кожної моделі, тому сутності, яка характеризує деталь потрібно передати як атрибут назву моделі, яка буде слугувати зовнішнім ключем у таблиці Деталі.

Щоб правильно сформувати логічну модель перш за все потрібно перевести загальні назви атрибутів згідно іменування таблиць і атрибутів в бази даних.

Створимо словник термінів для кращого розуміння логічної моделі:

* city – місто;
* district – район;
* street-вулиця;
* house\_number – номер будівлі;
* parking – автопарковка;
* parking\_id – унікальний номер парковки;
* area – площа;
* number\_places – кількість місць у парковкі;
* car\_place – стоянка авто;
* place\_id – ідентифікатор стоянки;
* begin\_date – дата початку перебування авто;
* end\_date –дата кінця перебування авто;
* car – автомобіль;
* car\_id – ідентифікатор автомобіля;
* release\_year – рік випуску;
* color – колір
* driver – водій
* driver\_id – ідентифікатор водія;
* name – ім’я;
* surname – прізвище;
* patronymic- по-батько;
* birth\_year- рік народження;
* gender – стать;
* email – електронна пошта;
* rights\_category – категорія водійських прав;
* trip - рейс
* trip\_id – ідентифікатор рейсі;
* begin\_city – місто відбуття;
* end\_city – місто прибуття;
* dispatch\_data – дата відбуття;
* arrival\_data – дата прибуття;
* loaded – завантаженість;
* driver\_contract – трудовий договір;
* approval\_date – дата вкладання;
* exhaustion\_date – дата закінчення;
* salary – зарплата;
* details – деталі;
* detail\_id – ідентифікатор деталі;
* price – ціна за деталь;
* width – ширина;
* height – висота;
* details\_model – кількість деталей одного виду в моделі;
* model – модель;
* model\_name – назва моделі;
* brand – бренд;
* max\_load – максимальна тягова здатність;
* fuel – паливо;
* fuel\_name – назва палива;
* price\_liter – ціна за літр.

Можемо розглянути саму логічну модель бази даних (див.рис. 3.2).

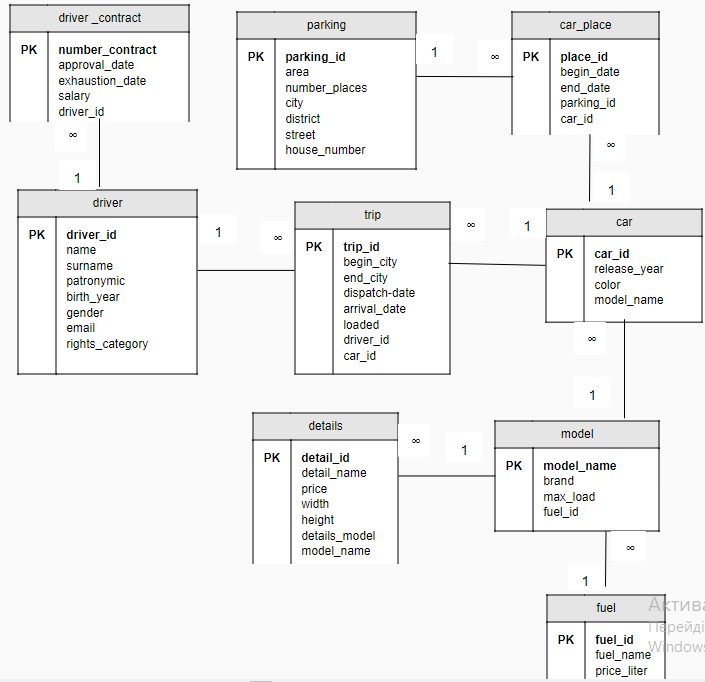


Рисунок 3.2 - Схема логічної моделі бази даних

Маючи словник термінів та саму логічну модель ми можемо проаналізувати кожну таблицю і довести, що вони знаходяться в третій нормальній формі (3НФ):

* таблиця parking містить первинний ключ parking\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже в другій нормальній формі (2НФ). Виділим функціональні залежності: атрибути area, number\_places, city, district, street, house\_number залежать від первинного ключа parking\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличці не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця car\_place містить первинний ключ place\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути begin\_date, end\_date, parking\_id, car\_id залежать від первинного ключа place\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має тому таблиця знаходиться в 3НФ;
* таблиця driver\_contract містить первинний ключ number\_contract, оскільки номер договору унікальний і атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути approval\_date, exhaustion\_date, salary, driver\_id залежать від первинного ключа number\_contract. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця driver містить первинний ключ driver\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути name, surname, patronymic, birth\_year, gender, email, right\_category залежать від первинного ключа driver\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця trip містить первинний ключ trip\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути begin\_city, end\_city, dispatch\_date, arrival\_date, loaded, driver\_id, car\_id залежать від первинного ключа trip\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця car містить первинний ключ car\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути release\_year, color залежать від первинного ключа car\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця details містить первинний ключ detail\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути detail\_name, price, width, height залежать від первинного ключа detail\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця model містить первинний ключ model\_name, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути brand, max\_load, fuel\_id залежать від внутрішнього ключа model\_name. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ;
* таблиця fuel містить первинний ключ fuel\_id, оскільки атрибути атомарні та внутрішній ключ не складовий, то можна сказати, що це вже 2НФ. Виділим функціональні залежності: атрибути fuel\_name, price\_liter залежать від первинного ключа fuel\_id. Інших залежностей таблиця не містить. При аналізі ми можемо побачити, що транзитивних залежностей в табличкі не має, оскільки всі не ключові атрибути залежать лише від первинного ключа, тому таблиця знаходиться в 3НФ.

Проаналізувавши кожну таблицю можна впевнено сказати, що логічна модель бази даних знаходиться в 3НФ.

## 3.3 Побудова логічної моделі бази даних шляхом нормалізації

Побудова логічної моделі бази даних шляхом нормалізації дуже важливий крок у розробці надійної і швидкої бази, яка дозволить вирішити проблему тривалого пошуку даних під час запитів.

Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей. Щоб досягти успішної нормалізації нашої бази, потрібно привести схему до 3НФ , яка і буде вважатися кінцевим етапом нормалізації. Для побудови логічної моделі бази даних шляхом нормалізації побудуємо універсальне відношення, промоделюємо залежності та оберемо первинні ключі. Для даної логічної моделі універсальне співвідношення складається із 5 сутностей, оскільки повна схема логічної моделі дуже об’ємна, тому для нормалізації її достатньо використати лише 5 основних сутностей: трудовий договір, водій, рейс, автомобіль, модель (див. рис. 3.3).

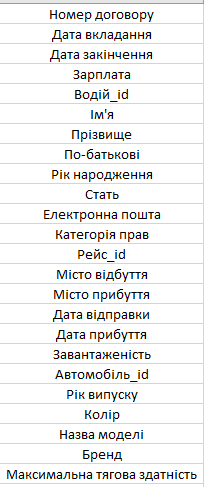


Рисунок 3.3 – Універсальне відношення

Можемо побачити, що універсальне відношення має безліч атрибутів, які потрібно пов’язати та розділити. Маючи універсальне відношення можемо почати перший етап нормалізації, а саме приведення до першої нормальної форми. Перша нормальна форма вимагає, щоб на перетині будь якого стовпця та будь якого запису знаходились атомарні значення.

Можна вважати, що перший етап виконано та унікальне відношення знаходиться в першій нормальній формі (1НФ), оскільки жодний атрибут не вимагає розподілу на ще конкретніші значення.

Наступним кроком буде приведення до 2НФ, яка вимагає вже детального розгляду. Відношення знаходиться у 2НФ в тому та тільки в тому випадку, коли воно знаходиться у 1НФ, та кожен неключовий атрибут повністю функціонально залежить від первинного ключа відношення. Першим чином потрібно виділити функціональні залежності, які можуть бути на універсальному відношенню та первинні ключі.

Залежності від номеру договору:

* номер договору → дата вкладання;
* номер договору → дата закінчення;
* номер договору → зарплата;
* номер договору → водій\_id;
* номер договору → ім’я;
* номер договору → прізвище;
* номер договору → по-батькові;
* номер договору → рік народження;
* номер договору → стать;
* номер договору → електронна пошта;
* номер договору → категорія прав.

Залежності від рейс\_id:

* рейс\_id → місто відбуття;
* рейс\_id → місто прибуття;
* рейс\_id → дата відправки;
* рейс\_id → дата прибуття;
* рейс\_id → завантаженість;
* рейс\_id → водій\_id;
* рейс\_id → ім’я;
* рейс\_id → прізвище;
* рейс\_id → по-батькові;
* рейс\_id → рік народження;
* рейс\_id → стать;
* рейс\_id → електронна пошта;
* рейс\_id → категорія прав;
* рейс\_id → автомобіль\_id;
* рейс\_id → рік випуску;
* рейс\_id → колір;
* рейс\_id → назва моделі;
* рейс\_id → бренд;
* рейс\_id → максимальна тягова здатність.

Залежність від водій\_id:

* водій\_id → ім’я;
* водій\_id → прізвище;
* водій\_id → по-батькові;
* водій\_id → рік народження;
* водій\_id → стать;
* водій\_id → електронна пошта;
* водій\_id → категорія прав.

Залежність від автомобіль\_id:

* автомобіль\_id → рік випуску;
* автомобіль\_id → колір;
* автомобіль\_id → назва моделі;
* автомобіль\_id → бренд;
* автомобіль\_id → максимальна тягова здатність.

Залежність від назви моделі:

* назва моделі → бренд;
* назва моделі → максимальна тягова здатність.

Залежність від номер договору та рейс\_id:

* номер договору та рейс\_id → дата вкладання;
* номер договору та рейс\_id → дата закінчення;
* номер договору та рейс\_id → зарплата;
* номер договору та рейс\_id → водій\_id;
* номер договору та рейс\_id → ім'я;
* номер договору та рейс\_id → прізвище;
* номер договору та рейс\_id → по-батькові;
* номер договору та рейс\_id → рік народження;
* номер договору та рейс\_id → стать;
* номер договору та рейс\_id → електронна пошта;
* номер договору та рейс\_id → категорія прав;
* номер договору та рейс\_id → місто відбуття;
* номер договору та рейс\_id → місто прибуття;
* номер договору та рейс\_id → дата відправки;
* номер договору та рейс\_id → дата прибуття;
* номер договору та рейс\_id → завантаженість;
* номер договору та рейс\_id → автомобіль\_id;
* номер договору та рейс\_id → рік випуску;
* номер договору та рейс\_id → колір;
* номер договору та рейс\_id → назва моделі;
* номер договору та рейс\_id → бренд;
* номер договору та рейс\_id → максимальна тягова здатність.

Виділивши функціональні залежності, можна сказати, що ключами універсального відношення будуть слугувати рейс\_id та номер договору (див. рис. 3.4)



Рисунок 3.4 – Функціональні залежності

Оскільки деякі неключові атрибути в універсальному відношенні залежать лише від частин складового ключа, то це не друга нормальна форма.

Отже потрібно окремо виділити функціональні залежності, які не повністю функціонально залежать від складеного ключа. Для початку потрібно виділити окрему функціональну залежність позначену красним кольором (див.рис. 3.4), тобто залежність від рейс\_id.

Виділивши окремо функціональну залежність від рейс\_ id ми отримали дві таблиці, які мають повні функціональні залежності, оскільки первинний ключ простий і кожен неключовий атрибут повністю залежить від ключа (див.рис. 3.5).

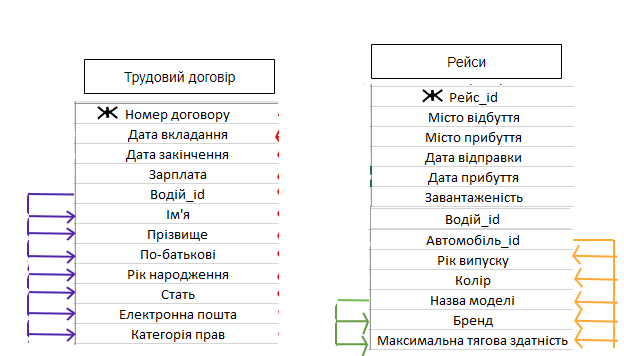


Рисунок 3.5 – Отримані відношення «Рейси» та «Трудовий договір»

Потрібно почати аналіз із відношення, яке має первинний ключ номер договору. Можемо побачити, що всі неключові атрибути повністю функціонально залежать від ключа та первинний ключ простий, тому дане відношення знаходиться в 2НФ. Аналогічно можемо побачити, що відношення із первинним ключем рейс\_id має простий ключ, тому всі неключові атрибути також повністю залежать від первинного ключа.

Приведення до другої нормальної форми можна вважати завершеною, оскільки утворені відношення не мають інших не повних функціональних залежностей та мають простий первинний ключ.

Наступним етапом нормалізації являється приведення даних таблиць до третьої нормальної форми. Відношення знаходиться в 3НФ в тому та тільки в тому випадку, якщо воно знаходиться у 2НФ та між неключовими атрибутами не має транзитивних залежностей.

Потрібно почати з відношення, яке має первинний ключ номер договору, оскільки ми можемо побачити, що атрибути: ім’я, прізвище, по-батькові, рік народження, стать, електронна пошта, категорія прав залежать від водій\_id, то спираючись на визначення 3НФ потрібно відокремити дану залежність, як окреме відношення, оскільки воно транзитивне (див.рис. 3.6).

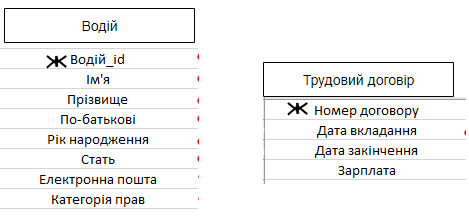


Рисунок 3.6 – Відношення «Водій» та «Трудовий договір»

Дані відношення не мають транзитивних залежностей і мають лише повну функціональну залежність від первинного ключа, тому утворені таблиці стоять у 3НФ. Дані відношення повинні мати назву «Водій» та «Трудовий договір» відповідно .

Далі візьмемо для аналізу відношення, яке має первинний ключ рейс\_id (див. рис. 3.5). Аналізуючи дане відношення можна побачити, що атрибути: рік випуску, колір, назва моделі, бренд та максимальна тягова здатність залежать від неключового атрибута автомобіль\_id, що свідчить про транзитивність. Коли ми виділимо дану залежність, як окреме відношення ми можемо помітити, що атрибути: бренд та максимальна тягова здатність залежать від неключового атрибута назва моделі, отже дану залежність потрібно також виділити, як окреме відношення. Як результат розбиття ми отримаємо три відношення (див.рис. 3.7).



Рисунок 3.7 - Відношення «Рейси» , «Автомобіль» та «Модель»

Дані відношення не мають транзитивних залежностей і мають лише повну функціональну залежність від первинного ключа, тому утворені таблиці стоять у 3НФ. Дані відношення повинні мати назву «Рейси» , «Автомобіль», «Модель» відповідно.

Оскільки всі відношення приведені до 3НФ, то можна вважати що всі ключові етапи нормалізації пройдені та залишається виставити зв’язки і оцінити отриманий результат (див. рис. 3.8).

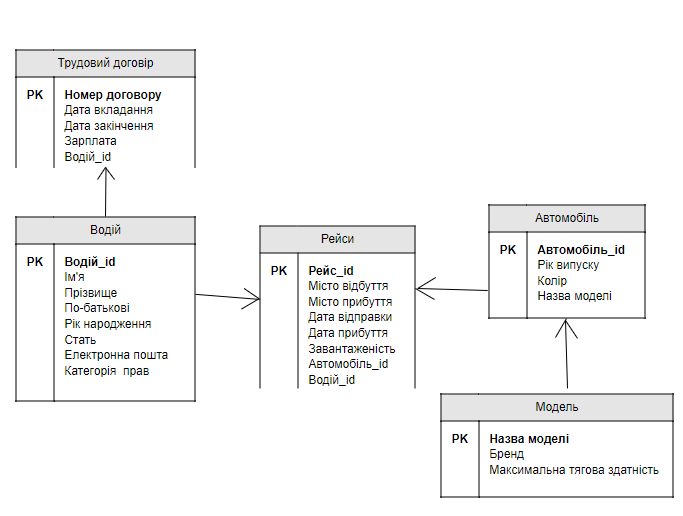


Рисунок 3.8 – Схема часткової реляційної бази даних, яка побудована на основі нормалізації

Аналогічним способом була проведена нормалізація і для інших атрибутів та відношень, які не були внесенні до універсального відношення (див.рис. 3.9).

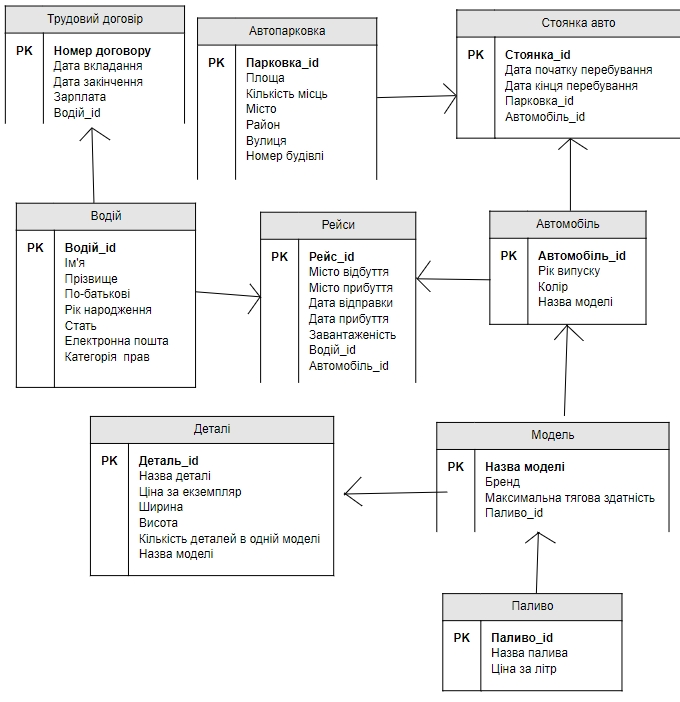


Рисунок 3.9 - Схема повної реляційної бази даних, яка побудована на основі нормалізації

Дивлячись на сформовану схему бази даних можна сказати, що вона повністю відповідає базі, яка була утворена у розділі 3.2. Оскільки для проведення нормалізації були обрані не всі сутності, які характеризували предметну область, тому отримана схема бази даних має значно менше таблиць та атрибутів.

Отже, фактично, ми взяли зв’язку з 5-ти сутностей з ER-діаграми, об’єднали їх атрибути в одне універсальне відношення та провели його покрокову нормалізацію. Оскільки основною метою застосування техніки нормалізації є зменшення надмірності та залежності даних, то можна вважати, що мета нормалізації даної схеми бази даних виконана. Потрібно згадати, що для нормалізації були використанні лише перші три нормальні форми, оскільки саме вони забезпечують баланс між фізичної реалізацією та логічною схемою. Звісно можна продовжити проводити нормалізацію до інших нормальних форм, але тоді швидкість роботи із даними може помітно погіршитись.

# **4 ОПИС ПРОГРАМИ**

## 4.1 Загальні відомості

Даний програмний продукт був створений в популярному середовищі розробки, яке використовується в багатьох проектах, а саме Intellij IDEA Community Edition 2022.1.3. Дана версія середовища знаходиться у вільному доступі в інтернеті, оскільки вона має лише основні інструменти для розробки програмного забезпечення. Існують також більш потужні версії даного середовища, але через не великі розміри нашого проекту використання більш функціональних аналогів було б недоцільно. Також Intellij IDEA гарно адаптована для використання мови програмування Java, тому програма була написана саме на цій мові.

Для розробки програмного забезпечення була обрана система управління базами даних (СУБД) відома як MySQL. MySQL - це система управління базами даних, яка використовується для підтримки реляційних баз даних. Це програмне забезпечення з відкритим кодом, що підтримується корпорацією Oracle. Дана СУБД має ряд переваг, тому що вона масштабована, швидка і надійна система управління базами даних, яка може працювати на будь-якій платформі, такі як Windows, Unix, Linux і т.д., і може бути встановлена ​​на робочому столі або будь-якій серверній машині.

Java Database Connectivity (JDBC)– це технологія , застосовувана для доступу програм на мові Java до реляційної бази даних . Використовуючи інтерфейс JDBC [4] java-прикладення, можна виконати SQL-запит [5] до СУБД, а також отримати і опрацювати результат цього запиту. Вона була обрана саме через адаптацію до мови Java, що дозволяє легко працювати з нею. Крім того дана технологія доступу має ряд переваг: легкість розробки, не треба встановлювати громіздку клієнтську програму, до будь-якої бази можна під'єднатись через легко описуваний URL.

Для нормальної роботи програми рекомендується використовувати:

* процесор: Intel(R) Core(TM) i5-4300M 2.60GHz;
* оперативну зовнішню пам’ять (ОЗП) 8 Гб;
* 64-розрядну операційну систему, процесор на базі архітектури x64
* операційна система Windows 10 або вище;
* монітор із роздільною здатністю 640 x 480 або вище.

## 4.2 Виклик і завантаження

Для того, щоб програма працювала правильно потрібно встановити плагін Database Navigator. Даний плагін допомагає встановити зв’язок між середовищем розробки та різними СУБД в тому числі і MySQL. Його можна завантажити відразу в самому середовищі для цього потрібно зайти в налаштування та обрати розділ Plugins, потім в пошуку знайти потрібний плагін та встановити його (див. рис. 4.1).

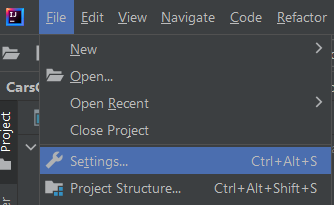


Рисунок 4.1 - Місце положення налаштувань

Після завантаження Database Navigator потрібно також встановити бібліотеку JDBC, яка дозволить мати всі необхідні функції для роботи з базами даних в середовищі. Крім того для формування звітів та надсилання повідомлення на пошту потрібно перевірити наявність бібліотек Apache POI та Jakarta Mail.

Для безпосереднього запуску програми потрібно відкрити середовище розробки та файл із проектом. Після перевірки всіх необхідних бібліотек потрібно активувати програму натиснувши на кнопку пуск, яка знаходиться у верхній частині екрана. (див.рис. 4.2).



Рисунок 4.2 - Кнопка пуск

Якщо певні елементи відсутні чи при установці бібліотек виникла помилка потрібно спробувати скористатися фреймворком для автоматизації складання проектів на основі опису їх структури – Maven. Він вимагає мінімальних зусиль при установці необхідних елементів.

## 4.3 Призначення і логічна структура

Програма має змогу виконувати багато різних функцій, які будуть корисні майбутньому користувачу. Вони були сформовані в результаті аналізу предметної області, який допоміг дізнатися основні завдання співробітника автотранспортного підприємства.

Після запуску програми користувач має побачити головне меню, яке матиме 9 кнопок для переходу до роботи з таблицями, а також 4 кнопки для отримання статистик по найкращому водію, автомобілю й автопаркові, а також перелік всіх видів палива та кількість авто, що їх використовують. Викликаються вони із файлів BestCar.java, BestDriver.java, BestParking.java, FullFuell.java відповідно. Кожна форма, яка містить у собі таблицю має змогу редагувати записи та додавати нові, щоб користувач не робив критичних помилок програма перевіряє записи, які користувач має бажання додати та надає повідомлення, коли введені дані мають некоректний зміст. Дані дві функції присутні в усіх таблицях без виключень, а таблиці у свою чергу реалізуються за допомогою файлів TCars.java, TContract.java, TDetails.java, TDriver.java, TFuel.java, TModel.java, TParking.java, TPlace.java, Ttrip.java. Видалення доступне лише у таблицях палива, рейсів, деталей, стоянок авто та трудового договору, які у свою чергу реалізовані у файлах TContract.java, TDetails.java, TPlace.java, TFuel.java, Ttrip.java. Видалити паливо можна лише тоді, коли жодна модель не використовує його на даний момент, в іншому випадку користувач отримає повідомлення про заборону видалення палива.

Звісно програмний продукт має змогу не тільки працювати з даними, а також і проводити пошуки. Для таблиць моделі, автомобіля, водія, деталей та парковки доступний пошук по декількох критеріях, які у свою чергу реалізовані у файлах TCars.java, TDetails.java, TDriver.java, TModel.java, TParking.java. В полі пошуку можна вводити і не повну назву певних характеристик, тому що система може провести пошук і за початком певного слова. Фільтрація даних присутня у таблицях водія, деталей, парковки, автомобіля, моделі, палива, які у свою чергу реалізовані у файлах TCars.java, TDetails.java, TDriver.java, TModel.java, TParking.java, TFuel.java. Якщо користувач пропустив введення мінімального значення, то система автоматично буде проводити пошук за мінімальним значенням у таблиці, яке наявне в базі, аналогічно із максимальним значенням. Сортування даних можна проводити у таблицях водія, парковки, моделі. які у свою чергу реалізовані у файлах TDriver.java, TModel.java, TParking.java. Також для кожної таблиці присутня кнопка «Display all», яка виводить всі елементи з бази даних та ігнорує фільтри.

Іншою функцією програмного продукту є формування звітів. Він може сформувати звіт по водію, який формується у файлі Driver.java, який буде містити всю інформацію про водія і звісно дату формування звіту та зберегти в бажаному для користувача місці на його комп’ютері. Також програма може сформувати каталог деталей, який формується у файлі Catalogue.java, який буде містити всю інформацію про деталі та навіть кількість деталей на підприємстві.

Не потрібно забувати також про можливість програмного продукту надсилати повідомлення на електронну пошту водіям в яких закінчився термін трудового договору або він закінчиться через три дня і менше. Дана функція не вимагає від користувача жодних дій та знаходиться у файлі Menu.java.

## 4.4 Описання фізичної моделі бази даних

Структури таблиць краще представити у вигляді відповідних команд CREATE

TABLE, які більш точно можна описати.

Потрібно почати із структури таблиці «CAR», яка має інформацію про автомобілі, якими володіє підприємство:

**CREATE TABLE `car` (**

**`car\_id` int NOT NULL AUTO\_INCREMENT,**

**`release\_year` year NOT NULL,**

**`color` varchar(30) NOT NULL,**

**`model\_name` varchar(30) NOT NULL,**

**PRIMARY KEY (`car\_id`),**

**UNIQUE KEY `car\_id\_UNIQUE` (`car\_id`),**

**KEY `model\_name\_idx` (`model\_name`),**

**CONSTRAINT `model\_name` FOREIGN KEY (`model\_name`) REFERENCES `model` (`model\_name`)**

**) ENGINE=InnoDB** **AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «car\_id». Наявний також зовнішній ключ «model\_name», оскільки на це вказує ключова команда при створенні зовнішнього ключа, а саме «FOREIGN KEY». Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «CAR\_PLACE», яка містить інформацію про дати перебування авто на парковках:

**CREATE TABLE `car\_place` (**

**`place\_id` int NOT NULL AUTO\_INCREMENT,**

**`begin\_date` datetime NOT NULL,**

**`end\_date` datetime NOT NULL,**

**`parking\_id` int NOT NULL,**

**`car\_id` int NOT NULL,**

**PRIMARY KEY (`place\_id`),**

**UNIQUE KEY `place\_id\_UNIQUE` (`place\_id`),**

**UNIQUE KEY `car\_id` (`car\_id`,`begin\_date`,`end\_date`),**

**KEY `parking\_id\_idx` (`parking\_id`),**

**CONSTRAINT `car\_id` FOREIGN KEY (`car\_id`) REFERENCES `car` (`car\_id`),**

**CONSTRAINT `parking\_id` FOREIGN KEY (`parking\_id`) REFERENCES `parking` (`parking\_id`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «place\_id». Наявні також зовнішні ключі «car\_id», «parking\_id», оскільки на це вказує ключова команда при створенні зовнішнього ключа, а саме «FOREIGN KEY». Існує також обмеження, що таблиця може мати лише унікальне поєднання автомобіля та часу його стоянки.

Структура таблиці «DETAILS», яка містить інформацію про всі деталі автомобілів, які має підприємство:

**CREATE TABLE `details` (**

**`detail\_id` int NOT NULL AUTO\_INCREMENT,**

**`detail\_name` varchar(30) NOT NULL,**

**`price` float NOT NULL,**

**`width` float DEFAULT NULL,**

**`height` float DEFAULT NULL,**

**`details\_model` int NOT NULL,**

**`model\_name` varchar(30) NOT NULL,**

**PRIMARY KEY (`detail\_id`),**

**UNIQUE KEY `detail\_id\_UNIQUE` (`detail\_id`),**

**KEY `model\_name\_idx` (`model\_name`),**

**CONSTRAINT `model\_name1` FOREIGN KEY (`model\_name`) REFERENCES `model` (`model\_name`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «detail\_id». Наявний також зовнішній ключ «model\_name», оскільки на це вказує ключова команда при створенні зовнішнього ключа, а саме «FOREIGN KEY».

Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «DRIVER», яка містить інформацію про всіх водіїв, працюють в підприємстві:

**CREATE TABLE `driver` (**

**`driver\_id` int NOT NULL AUTO\_INCREMENT,**

**`name` varchar(30) NOT NULL,**

**`surname` varchar(30) NOT NULL,**

**`patronymic` varchar(30) NOT NULL,**

**`birth\_year` year NOT NULL,**

**`gender` varchar(10) NOT NULL,**

**`email` varchar(45) NOT NULL,**

**`rights\_category` varchar(10) NOT NULL,**

**PRIMARY KEY (`driver\_id`),**

**UNIQUE KEY `driver\_id\_UNIQUE` (`driver\_id`),**

**UNIQUE KEY `email\_UNIQUE` (`email`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «driver\_id». Також створюється умова унікальності атрибута «email», який відповідає за назву поштового адресу, оскільки вона унікальна для кожного користувача. Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «DRIVER\_CONTRACT», яка містить інформацію трудові договори, які були підписані водіями:

**CREATE TABLE `driver\_contract` (**

**`number\_contract` varchar(15) NOT NULL,**

**`approval\_date` date NOT NULL,**

**`exhaustion\_date` date NOT NULL,**

**`salary` float NOT NULL,**

**`driver\_id` int NOT NULL,**

**PRIMARY KEY (`number\_contract`),**

**UNIQUE KEY `number\_contract\_UNIQUE` (`number\_contract`),**

**KEY `driver\_id\_idx` (`driver\_id`),**

**CONSTRAINT `driver\_id` FOREIGN KEY (`driver\_id`) REFERENCES `driver` (`driver\_id`) ON UPDATE RESTRICT**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «number\_contract». Наявний також зовнішній ключ «driver\_id», оскільки на це вказує ключова команда при створенні зовнішнього ключа, а саме «FOREIGN KEY». Існує також обмеження на заборону редагування, коли такий водій ще існує в іншій таблиці. Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «FUEL», яка містить інформацію про всі види палива, які використовуються підприємством:

**CREATE TABLE `fuel` (**

**`fuel\_id` int NOT NULL AUTO\_INCREMENT,**

**`fuel\_name` varchar(20) NOT NULL,**

**`price\_liter` float NOT NULL,**

**PRIMARY KEY (`fuel\_id`),**

**UNIQUE KEY `fuel\_id\_UNIQUE` (`fuel\_id`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «fuel\_id». Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «MODEL», яка містить інформацію про всі види моделей, які використовуються підприємством:

**CREATE TABLE `model` (**

**`model\_name` varchar(30) NOT NULL,**

**`brand` varchar(45) NOT NULL,**

**`max\_load` float NOT NULL,**

**`fuel\_id` int DEFAULT NULL,**

**PRIMARY KEY (`model\_name`),**

**UNIQUE KEY `model\_name\_UNIQUE` (`model\_name`),**

**KEY `fuel\_id\_idx` (`fuel\_id`),**

**CONSTRAINT `fuel\_id` FOREIGN KEY (`fuel\_id`) REFERENCES `fuel` (`fuel\_id`) ON DELETE SET RESTRICT ON UPDATE RESTRICT**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «model\_name». Наявний також зовнішній ключ «fuel\_id», оскільки на це вказує ключова команда при створенні зовнішнього ключа, а саме «FOREIGN KEY». Існує також обмеження на заборону видалення, коли таке паливо існує на підприємстві. Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «PARKING», яка містить інформацію про всі автопарки, які належать підприємству:

**CREATE TABLE `parking` (**

**`parking\_id` int NOT NULL AUTO\_INCREMENT,**

**`area` float NOT NULL,**

**`number\_places` int NOT NULL,**

**`city` varchar(30) NOT NULL,**

**`district` varchar(30) NOT NULL,**

**`street` varchar(30) NOT NULL,**

**`house\_number` varchar(10) NOT NULL,**

**PRIMARY KEY (`parking\_id`),**

**UNIQUE KEY `parking\_id\_UNIQUE` (`parking\_id`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний parking\_id. Крім того потрібно виділити властивість «AUTO\_INCREMENT» яка автоматично визначає для нового запису унікальний ключ.

Структура таблиці «TRIP», яка містить інформацію про всі рейси, які були та будуть здійснені:

**CREATE TABLE `trip` (**

**`trip\_id` int NOT NULL AUTO\_INCREMENT,**

**`begin\_city` varchar(45) NOT NULL,**

**`end\_city` varchar(45) NOT NULL,**

**`dispatch\_date` datetime NOT NULL,**

**`arrival\_date` datetime NOT NULL,**

**`loaded` varchar(20) NOT NULL,**

**`driver\_id` int NOT NULL,**

**`car\_id` int NOT NULL,**

**PRIMARY KEY (`trip\_id`),**

**UNIQUE KEY `trip\_id\_UNIQUE` (`trip\_id`),**

**UNIQUE KEY `driver\_id` (`driver\_id`,`dispatch\_date`,`arrival\_date`),**

**KEY `driver\_id\_idx` (`driver\_id`),**

**KEY `car\_id\_idx` (`car\_id`),**

**KEY `car\_id` (`car\_id`),**

**KEY `car\_id\_UNIQUE` (`car\_id`),**

**KEY `driver\_id\_UNIQUE` (`driver\_id`),**

**KEY `arrival\_date\_UNIQUE` (`arrival\_date`),**

**KEY `dispatch\_date\_UNIQUE` (`dispatch\_date`),**

**CONSTRAINT `car\_id1` FOREIGN KEY (`car\_id`) REFERENCES `car` (`car\_id`),**

**CONSTRAINT `driver\_id1` FOREIGN KEY (`driver\_id`) REFERENCES `driver` (`driver\_id`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci**

Згідно скрипту, який створює таблицю, можна сказати, що первинний ключ обраний «trip\_id». Наявні також зовнішні ключі «car\_id», «driver\_id», оскільки на це вказує ключова команда при створенні зовнішнього ключа, а саме «FOREIGN KEY». Існує також обмеження, що таблиця може мати лише унікальне поєднання водія та часу його рейсу. Крім того потрібно виділити властивість «AUTO\_INCREMENT», яка автоматично визначає для нового запису унікальний ключ.

Описавши фізичну модель бази даних можемо переходити до наступного розділу формування опису програмного продукту.

## 4.5 Опис програмної реалізації

Для початку потрібно відмітити, що поля додавання, видалення та редагування завжди знаходяться в лівій частині форми та мають одноманітний стиль.

Для прикладу можна обрати форму, яка дозволяє працювати із інформацією про паливо (див. рис. 4.3).

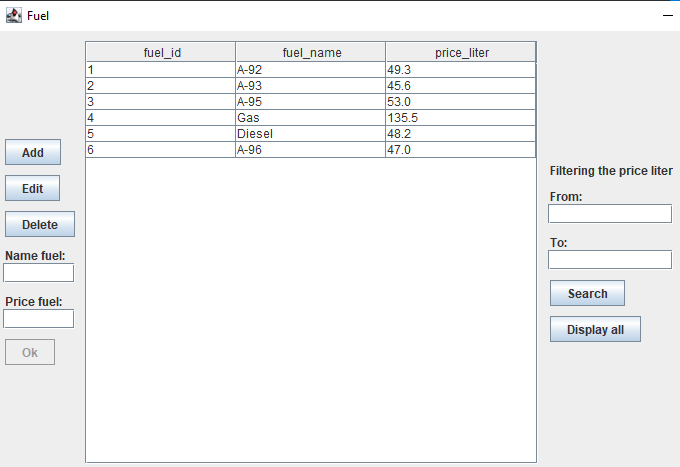


Рисунок 4.3 – Форма для інформації про паливо

Як вже було сказано нас цікавить саме можливість додавання, видалення та редагування на даній формі. Всі поля введення знаходяться в лівому куті екрану, там можна також помітити і три кнопки, які виконують відповідні дії.

Щоб додати нове паливо в таблицю потрібно заповнити всі поля, які запропоновані нижче кнопок і натиснути «ADD», якщо введені дані мають коректний зміст, то система має додати до таблиці нове паливо або користувач отримає повідомлення про те, що певні дані не є коректними.

Для редагування певної інформації потрібно натиснути на бажаний рядок в таблиці та натиснути кнопку «EDIT». Після натиснення всі поля для введення будуть заповнені інформацією із обраного рядка та користувач матиме змогу змінювати дану інформацію, але щоб підтвердити зміни потрібно натиснути кнопку «OK», яка розблокується відразу після вибору кнопки для редагування. Якщо всі поля будуть містити коректну інформацію, то зміни будуть збережені.

Щоб видалити певне паливо потрібно просто обрати бажаний рядок в таблиці і натиснути кнопку «DELETE». При видаленні палива існують певні обмеження, оскільки модель, яка використовує дане паливо не може залишитися без нього, тому при видаленні потрібно запевнитися, що паливо не використовується, оскільки програма просто не виконає дану команду і повідомить про це користувача.

Потрібно також виділити увагу важливим функціям програмного забезпечення, які дозволяють проводити пошук, фільтрацію та сортування по таблиці. Іноді поняття сортування та фільтрації можуть бути не зрозумілі для користувача, оскільки можна легко спутати дані визначення. Фільтрація - це процес пошуку і вибору записів відповідно до встановлених критеріїв. Фільтри також спрощують процес введення та видалення записів зі списку. Сортування – це впорядквання даних за зростанням або за спаданням. Воно дає можливість відсортувати список імен в алфавітному порядку, скласти список продуктів за рівнем запасів (від найбільшого до найменшого) або впорядкувати рядки за кольорами чи піктограмами. Отже з’ясувавши визначення даних понять можемо сказати, що вони суттєво відрізняються один від одного, тому можна продовжити розглядати поля, які призначенні для пошуку, фільтрації та сортування.

Поля для використання перелічених функцій завжди знаходяться в правій частині форми. Візьмемо до уваги форму, яка дозволяє працювати із інформацією про моделі авто (див. рис. 4.4).

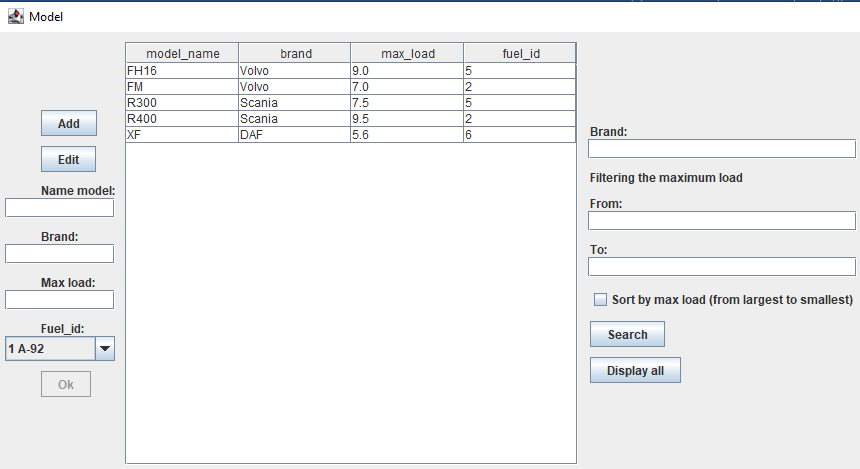


Рисунок 4.4 – Форма для інформації про модель

Можна помітити декілька полів в правій частині екрану, які відповідають за обробку інформації в таблиці. Поля, які мають назви звичайних атрибутів призначені для пошуку. Щоб здійснити пошук потрібно ввести повну назву бажаного параметра або часткову (див. рис. 4.5). Потрібно підмітити, що поля нечутливі до регістру.



Рисунок 4.5 – Приклад пошуку бренда

Після заповнення полів , які відповідають за пошук можна перейти до фільтрації певних параметрів. Фільтрація одного параметра має два поля для введення. Перше поле підписане як «FROM» в якому потрібно занести початкове значення фільтрації, а в поле «TO» потрібно занести кінцеве значення фільтрації, якщо одне із полів буде порожнім, то буде автоматично обране мінімальне або максимальне значення в таблиці відповідно. Також для зручності користувач зможе швидко переглянути всі записи в таблиці натиснувши кнопку «Display all».

Можна переглянути код, який формує запит залежно від обраних полів для редагування та пошуку:

**Statement state = Main.connection.createStatement();**

**String query = "select \* from model WHERE ";**

**int queryLength = query.length();**

**if (!brandSearchText.getText().equals("")) {**

**query = query + "LOWER(brand) LIKE LOWER('" + brandSearchText.getText() + "%') AND ";**

**}**

**If(!loadBefore.getText().equals("") || !loadAfter.getText().equals("")) {**

**if (loadBefore.getText().equals("")) {**

**query = query + "max\_load >= (SELECT min(max\_load) FROM model) AND max\_load <=" + loadAfter.getText() + " AND ";**

**} else if (loadAfter.getText().equals("")) {**

**query = query + "max\_load >=" + loadBefore.getText() + " AND max\_load <= (SELECT max(max\_load) FROM model) AND ";**

**} else {**

**query = query + "max\_load >=" + loadBefore.getText() + " AND max\_load <=" + loadAfter.getText() + " AND ";**

**}**

**}**

**if (query.length() == queryLength) {**

**query = "SELECT \* FROM model ";**

**} else if (hasAND(query)) {**

**query = query.substring(0, query.length() - 4);**

**}**

**if (sortLoad.isSelected()) {**

**query += " ORDER BY max\_load DESC";**

**}**

Наступним важливим функціоналом програми потрібно відмітити формування статистик, оскільки це може полегшити роботу користувача та надати програмному продукту більшої функціональної можливості. Загалом програма формує для користувача 4 статистики, які вважаються найнеобхіднішими для працівника автотранспортного підприємства. Перша статистика знаходить найкращого водія в підприємстві (див.рис.4.6).

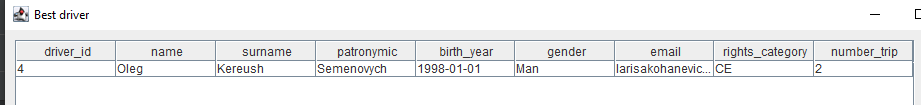


Рисунок 4.6 – Найкращий водій підприємства

При пошуку водія враховується кількість здійснених рейсів та його вік, якщо система знайшла двох і більше водіїв, які мають однакову найбільшу кількість рейсів та наймолодший вік, то користувач отримає всю інформацію про цих водіїв, оскільки приховувати наявність інших водіїв із однаковими високими результатами не є вирішенням проблеми користувача.

Пошук виконується за допомогою запиту до бази даних MySQL:

**SELECT driver.\*, count(trip\_id) AS number\_trip**

**FROM driver LEFT JOIN trip ON driver.driver\_id = trip.driver\_id**

**group by driver.driver\_id**

**HAVING count(trip\_id) = (SELECT distinct count(trip.trip\_id) FROM driver LEFT JOIN trip ON driver.driver\_id = trip.driver\_id**

**GROUP BY driver.driver\_id ORDER BY count(driver.driver\_id) DESC LIMIT 1)**

**AND driver.birth\_year = (SELECT distinct MAX(driver.birth\_year)**

**FROM driver LEFT JOIN trip ON driver.driver\_id = trip.driver\_id**

**group by driver.driver\_id**

**HAVING count(trip\_id) = (SELECT distinct count(trip.trip\_id) FROM driver LEFT JOIN trip ON driver.driver\_id = trip.driver\_id**

**GROUP BY driver.driver\_id ORDER BY count(driver.driver\_id) DESC LIMIT 1) ORDER BY birth\_year DESC );**

Наступний запит на статистику знаходить найкращий автомобіль в автотранспортному підприємстві (див.рис. 4.7).

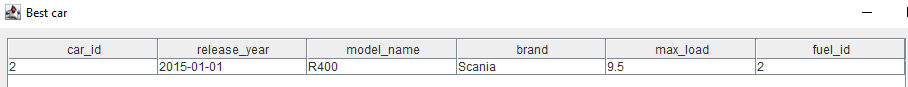


Рисунок 4.7 – Найкращий автомобіль на підприємстві

При пошуку водія враховується максимальна тягова здатність моделі, що належить цьому автомобілю та наймолодший рік випуску.

Користувач отримає лише основну інформацію про даний автомобіль, яка складається із специфікації моделі та самого автомобіля. При наявності двох і більше автомобілів, які задовільняють дані умови будуть відображені всі знайдені авто.

Пошук виконується за допомогою запиту до бази даних MySQL:

**SELECT car\_id, release\_year, model.\***

**FROM car LEFT JOIN model ON car.model\_name = model.model\_name**

**WHERE max\_load = (SELECT MAX(max\_load) FROM car LEFT JOIN model ON car.model\_name = model.model\_name) AND release\_year = (SELECT Max(release\_year)**

**FROM car LEFT JOIN model ON car.model\_name = model.model\_name**

**WHERE max\_load = (SELECT MAX(max\_load) FROM car LEFT JOIN model ON car.model\_name = model.model\_name));**

Наступний запит на статистику знаходить найкращу парковку в автотранспортному підприємстві (див.рис. 4.8). При пошуку парковки враховується його площа та кількість місць для автомобілів. При наявності більше однієї парковки, яка відповідає заданим критеріям користувач буде бачити всі знайдені об’єкти.

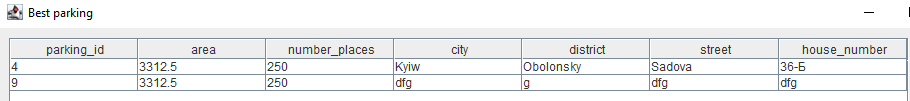


Рисунок 4.8 – Найкраща автопарковка в підприємстві

Пошук виконується за допомогою запиту до бази даних MySQL:

**SELECT \* FROM (SELECT \* FROM parking WHERE area = (SELECT MAX(area) FROM parking)) AS P WHERE number\_places = (SELECT MAX(D.number\_places) FROM (SELECT \* FROM parking WHERE area = (SELECT MAX(area) FROM parking)) as D );**

Останній запит на статистику відрізняється від інших, оскільки він надає перелік всіх наявних видів палива та кількість автомобілів, що використовують певний вид палива. Майже кожному підприємству у процесі діяльності доводиться купувати пальне у великих чи малих обсягах. Оскільки затрати на пальне доволі високі останніми роками, то даний запит доволі корисний для працівника автотранспортного підприємства, який зможе оцінити попит на пальне серед автомобілів та відмінити поставку палива, яке давно не використовується.

За допомогою нього можна легко визначити необхідність певного палива в підприємстві.

Пошук виконується за допомогою запиту до бази даних MySQL:

**SELECT fuel.\*, SUM(number\_car) as number\_cars FROM fuel LEFT JOIN (SELECT model.\*, count(car\_id) as number\_car FROM model LEFT JOIN car ON model.model\_name=car.model\_name GROUP BY model\_name) as M ON M.fuel\_id = fuel.fuel\_id GROUP BY fuel.fuel\_name;";**

Якщо паливо не використовується, то користувач буде бачити лише пусте поле, оскільки розрахунок кількості авто не дав результату.

Приклад формування даної статистики можна побачити на відповідному малюнку (див.рис. 4.9).

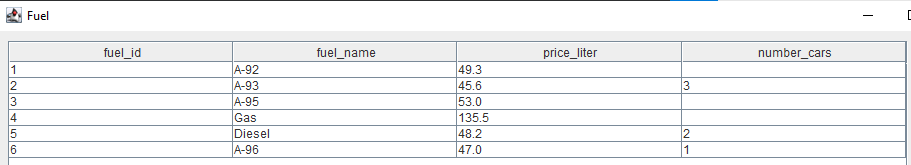


Рисунок 4.9 – Приклад статистики по паливу

Не можна забувати також про формування двох звітів, які програма може доволі швидко створити. Всі звіти формуються виключно у форматі DOCX, оскільки він зручний при створенні та його можна легко перевести в інші типи файлів, що надає працівнику трішки більше можливостей ніж з іншими файлами. Перший звіт належить формі, яка працює із інформацією про водія. На формі можна помітити кнопку «Report», яка відповідає за формування даного звіту. Потрібно обрати потрібний рядок із інформацією про водія та натиснути на неї, якщо програма не відкрила користувачеві вікно для вибору папки, то потрібно перевірити чи був виділений певний рядок із інформацією про водія. (див.рис. 4.10).

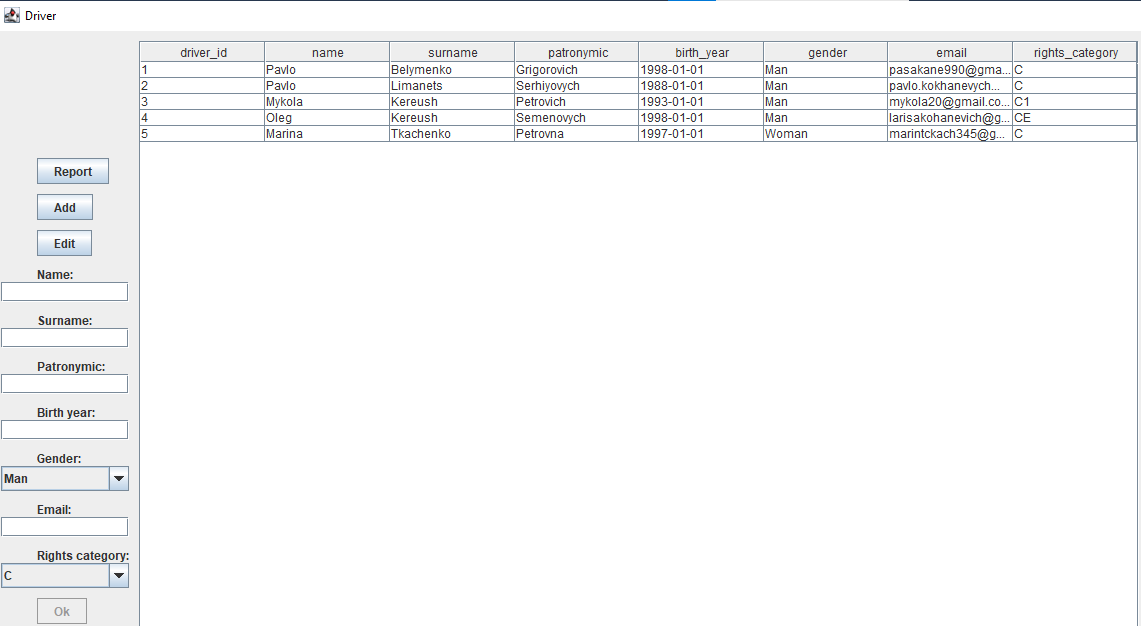


Рисунок 4.10 – Форма для роботи з водіями

Після натискання кнопки «Report», користувачу буде наданий доступ для вибору бажаної папки в яку буде збережений майбутній звіт. Дана форма має доступ до всіх відкритих дисків на робочому пристрої, що дозволяє користувачеві доволі детально розглянути запропоновані варіанти та обрати найкращий. Також вона має змогу змінювати стилі відображення папок. Сама форма має мінімалістичний вигляд, оскільки різноманітні кольори можуть погіршити швидкість пошуку та якість тексту (див. рис. 4.11).

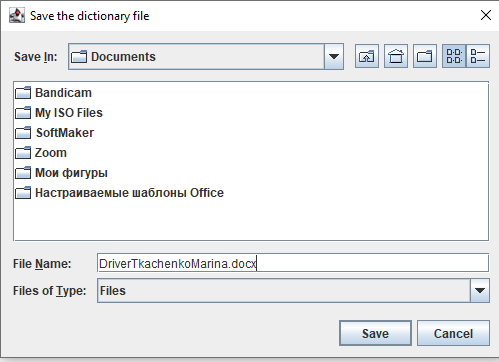


Рисунок 4.11 – Вікно зберігання звіту

Можна помітити, що вікно навіть пропонує рекомендовану назву файлу, яка буде складатися із ім’я та прізвища водія. Після зберігання файлу буде можливість переглянути сформований звіт та дізнатися всю інформацію про даного водія в окремому документі.

Наступний звіт формує каталог деталей, які наявні в автотранспортному підприємстві та вираховує кількість певної деталі в підприємстві. Функція для формування даного звіту знаходиться у формі для роботи з інформацією про деталі. Кнопка та форма мають однаковий функціонал як і попередня. Кнопка має назву «Report». Після натискання на неї не потрібно натискати на рядок із інформацією про деталі, оскільки програма і так буде працювати із усіма даними. Після натискання програма запропонує обрати бажане місце для зберігання та швидко сформує звіт.

Після відкриття сформованого звіту користувач зможе побачити табличку, яка містить всі дані про деталі. Також в кожному сформованому звіті буде доступна інформація про час та дату його формування (див.рис.4.12).

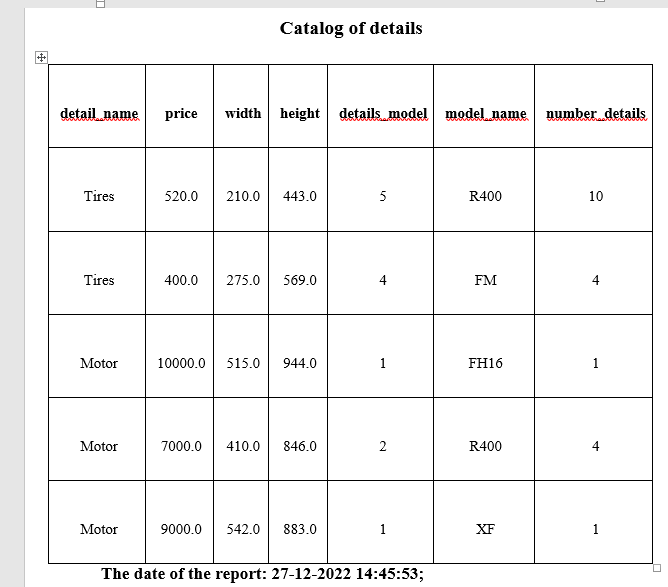


Рисунок 4.12 – Приклад сформованого каталогу деталей

Окремо потрібно виділити увагу останньому стовпцю даного каталогу деталей, оскільки саме він вказує на кількість певного виду деталей у підприємстві, що робить звіт більш корисним та складним при формуванні.

Обрахунок кількості деталей проводився за допомогою створеного запиту. Пошук виконується за допомогою запиту до бази даних MySQL:

**SELECT detail\_name, price, width, height, details\_model, details.model\_name, (number\_car \* details\_model) AS number\_details FROM details LEFT JOIN (SELECT model.model\_name, COUNT(car\_id) as number\_car FROM model LEFT JOIN car ON model.model\_name = car.model\_name GROUP BY car.model\_name) AS t ON details.model\_name = t.model\_name;**

Етап опису програмної реалізації даного програмного забезпечення можна вважати завершеним, оскільки описані всі ключові моменти роботи програми.

## 4.6 Описання задачі автоматизації

Задача автоматизації полягає в тому, що програмне забезпечення має самостійно знаходити трудові договори в яких закінчився термін придатності та повідомляти про це відповідного водія. Якщо термін трудового договору буде вичерпано через три дня і менше, то система також повідомить про це. Для того, щоб водій отримав повідомлення у зручній формі було прийнято рішення використовувати електронні пошти водіїв, оскільки це дуже зручний вид взаємодії. Повідомлення, яке отримає користувач, складається із прохання з’явитися до відділу кадрів та вирішити дану проблему з договором. Результатом виконання задачі автоматизації можна вважати отримане повідомлення на пошту відповідного водія (див.рис. 4.13).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4.13 – Результат виконання задачі автоматизації

Якщо розглянути конкретний приклад, то повідомлення має вигляд «Dear driver, your employment contract under number 07A ends 2022-12-19. Please come to the office to sign a new employment contract», але звісно номер договору та дата його кінця будуть різними для кожного водія.

Далі потрібно описати алгоритм вирішення задачі автоматизації. При запуску програми вона надсилає запит до бази даних із проханням надати всі трудові договори, які вичерпали свої терміни.

Запит до бази даних MySQL, який повертає вичерпані договори:

**SELECT \* FROM driver\_contract WHERE exhaustion\_date < '" + time + "' OR exhaustion\_date <= adddate('" + time + "', 3);**

Запит звісно враховує поточний час, який зазначений на робочому пристрої, щоб порівняти та знайти договори, які мають вичерпаний термін. Потім система за допомогою циклу проходить всі трудові договори та знаючи зовнішній ключ водія звертається до бази даних із проханням отримати електронні пошти даних водіїв та надсилає відповідні повідомлення.

Щоб детальніше розібратися в реалізації задачі автоматизації наведемо частину коду на мові програмування Java [6], який описує процес формування повідомлення та надсилання до відповідного водія:

**for (int i = 0; i < objects.length; i++) {  
 query1 = "SELECT email FROM driver WHERE driver\_id =" + objects[i][4].toString();  
 resultSet1 = state.executeQuery(query1);  
 resultSet1.next();  
 String emails = resultSet1.getString(1);  
 try {  
 // Create a default MimeMessage object.  
 MimeMessage message = new MimeMessage(session);  
  
 // Set From: header field of the header.  
 message.setFrom(new InternetAddress(from));  
  
 // Set To: header field of the header.  
 message.addRecipient(Message.RecipientType.*TO*, new InternetAddress(emails));  
  
 // Set Subject: header field  
 message.setSubject("Car company");  
  
 // Now set the actual message  
 message.setText("Dear driver, your employment contract under number " + objects[i][0] + " ends " + objects[i][2] + ". Please come to the office to sign a new employment contract.");  
  
 System.out.println("sending...");  
 // Send message  
 Transport.*send*(message);  
 System.out.println("Sent message successfully....");  
 } catch (MessagingException mex) {  
 mex.printStackTrace();  
 }  
}**

Утворене рішення задачі автоматизації було протестоване та оптимізоване під час тестування. Можна зробити висновок, що задача автоматизації повністю вирішена та задовольняє потреби майбутніх користувачів. Код наведений в додатку А.

# **ВИСНОВКИ**

У результаті виконання курсової роботи була створена інформаційна система «Облік на автотранспортному підприємстві». Вона містить всі необхідні функції для працівника автотранспортного підприємства. Програмне забезпечення дозволяє додавати, редагувати та навіть видаляти певні записи. Звісно система перевіряє введені значення на коректність та повідомляє, якщо вхідні дані некоректні. Наявні функції пошуку, сортування та фільтрації відразу по двом параметрам різної інформації, яка присутня в системі. Користувач може сформувати звіти по водію та каталог деталей та зберегти в потрібне йому місце на робочому пристрої. Були реалізовані чотири найпопулярніших запита в системі, які наводять найкращого водія, парковку та автомобіль в системі, а також статистику по пальному, яка дозволяє оцінити кількість автомобілів, які використовують даний вид палива. Реалізована функція пошуку не дійсних трудових договорів та надсилання повідомлення про це водіям на електрону пошту. Звісно система має недолік пов’язаний з тим, що надсилання проводиться лише тоді, коли програма вмикається, якщо працівник не буде вмикати програму кілька днів, то перевірка просто не буде проводитись.

Потрібно відмити, що дане програмне забезпечення використовує реляційну базу даних, яка була спроектована та нормалізована під час виконання курсової роботи. У ході реалізації проекту було проаналізовано предметну область та існуючі аналоги, а також проведено концептуальне моделювання предметної області. Отримано теоретичні знання та застосовано на практиці з ER-моделювання та побудови ER-діаграм, побудови логічних моделей баз даних на основі ER-діаграми та шляхом нормалізації універсального відношення.

Отримана система може бути корисна для роботи будь-яким працівникам підприємства, яке займається перевезенням товарів та має доволі розвинуту структуру, а саме різні парковки та різноманітні автомобілі.

# **ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. BAS Управління автотранспортом. Conto. URL: https://conto.com.ua/ua/products/bas/bas-upravlinnya-avtotransportom-standart/ (дата звернення: 02.12.2022).
2. Программа учета автотранспорта: модуль для автомобилей | jSolutions. Jsolutions.UA. URL: https://jsolutions.ua/sistema-upravleniya-avtotransportom (дата звернення: 05.12.2022).
3. MySQL documentation. Moved. URL: https://docs.oracle.com/cd/E17952\_01/index.html (дата звернення: 11.12.2022).
4. JDBC tutorial | what is java database connectivity(jdbc) - javatpoint. www.javatpoint.com. URL: https://www.javatpoint.com/java-jdbc (дата звернення: 23.11.2022).
5. Мартин Грабер. SQL. - К.: Ид-во ―ЛОРИ, 2003. - 644 с.
6. Кей Хорстманн, Гарі Корнелл. Java. Библиотека профессионала. Williams, 2016. Т. 2 : Расширенные средства программирования. 1008 с.

# **ДОДАТОК А**

Код реалізації задачі автоматизації

**private void getAndSendEmail() throws SQLException {**

**Calendar calendar = Calendar.getInstance();**

**SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");**

**String time = formatter.format(calendar.getTime());**

**Statement state = Main.connection.createStatement();**

**String query = "SELECT \* FROM driver\_contract WHERE exhaustion\_date < '" + time + "' OR exhaustion\_date <= adddate('" + time + "', 3);";**

**ResultSet resultSet = state.executeQuery(query);**

**ResultSetMetaData dataMeta = resultSet.getMetaData();**

**int colms = dataMeta.getColumnCount();**

**for (int i = 0; i < colms; i++) {**

**dflTable.addColumn(dataMeta.getColumnName(i + 1));**

**}**

**Vector<String> rows = new Vector<>();**

**while (resultSet.next()) {**

**for (int i = 0; i < colms; i++) {**

**rows.add(resultSet.getString(i + 1));**

**}**

**dflTable.addRow(rows);**

**rows = new Vector<>();**

**}**

**resultSet.close();**

**sendEmail();**

**}**

**private void sendEmail() throws SQLException {**

**Statement state = Main.connection.createStatement();**

**// Sender's email ID needs to be mentioned**

**String from = "ukrainigromadanin284@gmail.com";**

**// Assuming you are sending email from through gmails smtp**

**String host = "smtp.gmail.com";**

**// Get system properties**

**Properties properties = System.getProperties();**

**// Setup mail server**

**properties.put("mail.smtp.host", host);**

**properties.put("mail.smtp.port", "465");**

**properties.put("mail.smtp.ssl.enable", "true");**

**properties.put("mail.smtp.auth", "true");**

**// Get the Session object.// and pass username and password**

**Session session = Session.getInstance(properties, new javax.mail.Authenticator() {**

**protected PasswordAuthentication getPasswordAuthentication() {**

**return new PasswordAuthentication("ukrainigromadanin284@gmail.com", "wosbsugbhkdawlmy");**

**}**

**});**

**// Used to debug SMTP issues**

**session.setDebug(true);**

**String query1;**

**ResultSet resultSet1;**

**Object[][] objects = new Object[dflTable.getRowCount()][5];**

**for (int i = 0; i < objects.length; i++) {**

**for (int j = 0; j < 5; j++) {**

**objects[i][j] = dflTable.getValueAt(i,j);**

**}**

**}**

**for (int i = 0; i < objects.length; i++) {**

**query1 = "SELECT email FROM driver WHERE driver\_id =" + objects[i][4].toString();**

**resultSet1 = state.executeQuery(query1);**

**resultSet1.next();**

**String emails = resultSet1.getString(1);**

**try {**

**// Create a default MimeMessage object.**

**MimeMessage message = new MimeMessage(session);**

**// Set From: header field of the header.**

**message.setFrom(new InternetAddress(from));**

**// Set To: header field of the header.**

**message.addRecipient(Message.RecipientType.TO, new InternetAddress(emails));**

**// Set Subject: header field**

**message.setSubject("Car company");**

**// Now set the actual message**

**message.setText("Dear driver, your employment contract under number " + objects[i][0] + " ends " + objects[i][2] + ". Please come to the office to sign a new employment contract.");**

**System.out.println("sending...");**

**// Send message**

**Transport.send(message);**

**System.out.println("Sent message successfully....");**

**} catch (MessagingException mex) {**

**mex.printStackTrace();**

**}**

**}**

**state.close();**

**}**