Національний університет «Львівська політехніка»

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Бази даних»**

на тему:

«База даних для обліку товарів продуктового магазину»

Виконав: студент

напряму 6.050103 «Програмна інженерія»

групи ПІ - 32

Андрущакевич О. Т.

Керівник:

старший викладач кафедри програмного забезпечення, к.т.н. Квятковський Б.О.

Оцінка:

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

Кількість балів\_\_\_\_\_\_\_ Оцінка ECTS\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_ Павич Н.Я.

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_ Квятковський Б.О.

(підпис)

Львів – 2017 рік

**ЗМІСТ**

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ОБЛІКУ ДІЯЛЬНОСТІ ПРОДУКТОВОГО МАГАЗИНУ

1.1. Опис предметної області …………………………………………………3

1.2. Постановка завдання ……………………………………………………..5

1.3. Специфікація вимог до системи обліку діяльності продуктового

магазину ……………………………………………………………………….8

1.3.1. Вступ ………………………………………………………….….…8

1.3.2. Загальний опис ….…………………………………………….……8

1.3.3. Характеристики системи ……………..……………………….…11

1.3.4. Вимоги зовнішніх інтерфейсів ……….…….……………………15

1.3.5. Інші не функціональні вимоги ……..…..…………………….…15

РОЗДІЛ 2.ПРОЕКТУВАННЯ БАЗИ ДАНИХ ……….……………………..17

2.1. Концептуальне моделювання предметної області ...…………………..17

2.2. Логічне проектування бази даних .…………………..………….……...24

2.3. Проектування типових запитів і транзакцій …………..………………28

РОЗДІЛ 3.РЕАЛІЗАЦІЯ ПРОГРАМИ РОБОТИ З БАЗОЮ ДАНИХ …….35

3.1. Реалізація доступу до даних ..………………………………………..…35

3.2. Реалізація функціональних характеристик системи …….…………..36

3.3. Опис роботи програми …………………………………………….….36

ВИСНОВКИ ………………………………………………………..…………48

СПИСОК ЛІТЕРАТУРИ ……………………………………………...……49

Додаток А. Скрипт створення БД …………………………………………50

**РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ОБЛІКУ ДІЯЛЬНОСТІ ПРОДУКТОВОГО МАГАЗИНУ (збут)**

* 1. **Опис предметної області «Продуктовий магазин»**

У якості предметної області мною був обраний «Продуктовий магазин», а саме реалізація продажу продовольчих товарів.

Продуктовий магазин – це підприємство чи заклад [роздрібної торгівлі](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%B4%D1%80%D1%96%D0%B1%D0%BD%D0%B0_%D1%82%D0%BE%D1%80%D0%B3%D1%96%D0%B2%D0%BB%D1%8F); це підприємство, що виступає в якості посередника між виробниками продовольчих товарів та бажаючими придбати цей товар юридичними і фізичними особами.

Завдання мережевих магазинів полягає у своєчасній доставці товарів, потрібних для жителів. Для цього їм необхідна єдина інформаційна система, в якій буде відслідковуватися надходження і подальший продаж товарів. Достатня кількість товарів за якісними характеристиками та термінами зберігання буде залежати від того, наскільки продумані канали взаємодії між постачальником (складом) та замовником (магазином). А насамперед доставка товарів продовольчої групи на склади залежить від маркетингу та менеджменту між постачальниками (складами) та підприємствами по виробництву харчової продукції (молокозаводи, м’ясокомбінати, хлібозаводи, фермерські господарства по вирощуванню сільськогосподарської продукції, тощо) а також фірми, що імпортують харчові продукти.

Доставлений постачальником товар у відповідності до товарно-транспортної накладної обліковується, частина товару залишається у складському приміщені магазину, а частина відправляється в торгівельний зал для реалізації покупцю. Кожен товар постачальник відпускає за ціною, яка вказана у накладній. Зазвичай, ця ціна нижче, за ціну яку запропоновано споживачам. Щодня товарознавцем вираховується прибуток від продажу товарів, кількість проданого товару протягом дня, і навіть залишки на складі, що дозволяє ухвалити рішення про доцільність замовлення нової партії товару в постачальника.

Продуктовий магазин є кінцевим пунктом збуту товару. Кожний продуктовий магазин має свій асортимент товару для роздрібної торгівлі (у відповідності до ліцензії). В магазині присутні продукти загалом різного виду.

До основних належать:

* молочні продукти;
* хлібобулочні та кондитерські вироби;
* м’ясна та рибна продукція;
* овочі та фрукти;
* сипучі продукти (крупи);
* напої;
* інші продукти широкого вжитку.

Продукт циркулює наступним чином:

1. Доставка товару в магазин – продукти поступають на [склад](http://ua-referat.com/%D0%A1%D0%BA%D0%BB%D0%B0%D0%B4) магазину за

накладними по приходу;

2. Продаж з магазину – відбувається за видатковими чеками.

Продукти зберігаються до їх продажу, чи списання (останнє трапляється відносно періодично, у зв’язку з закінченням терміну вжитку товару або його пошкодженням).

Сфера діяльності мережі магазинів характеризується великими масивами інформаційних даних та обсягом виконуваних робіт. Функціонування продуктового магазину пов’язане із злагодженим робочим середовищем працівників та впорядкованим набором даних щодо кількісного асортименту запропонованих товарів та становищем продуктового магазину загалом. Для налагодження зв'язку між цими сутностями необхідно створити єдине сховище даних - базу даних для зберігання необхідної інформації та програмні засоби для роботи з цією базою, зокрема для отримання з неї певних даних, внесення нових та редагування існуючих. Створювана база даних повинна відповідати обладнанню продуктового магазину, в тому числі і програмному забезпеченню, що забезпечує виконання встановлених дій.

Розробка додатку для продажу продовольчих товарів в продуктових магазинах є актуальним через те, що полегшує роботу працівникам, організовує роботу персоналу та пришвидшує оформлення документів. Перевагою створеної бази даних є те, що всі дані будуть зберігатись в одному місці без втрат інформації та будуть відслідковуватись будь-які зміни в діяльності магазину. Зручність полягає в тому, що будь-який працівник магазину може звернутись до бази даних та вивести з неї потрібну йому інформацію по продажу продуктів.

Також перевагою є те, що до даної БД можна звернутись з будь якого комп’ютера в продуктовому магазині, а адміністратор може легко контролювати будь-які зміни які вносились в базу даних протягом всього періоду часу.

* 1. **Постановка завдання**

Оскільки додаток призначений для полегшення процесу продажу продовольчих товарів в продуктових магазинах, у ньому має бути реалізований перелік функцій, таких як вибір певного продукту, вивід необхідної інформації про продукт, перелік наявних продуктів, ведення обліку продуктів тощо. Функціонал програмного продукту не повинен закінчуватися на вищесказаних задачах. Крім цього повинен вестись облік продажу продовольчих товарів, список клієнтів, працівників, таблиця продажів продуктів та таблиця прибутків. Для зручної роботи і використання всіх переваг створеного програмного продукту використовуються тригери для обліку дій які проводяться з базою даних працівниками продуктового магазину. Відповідно до цього у базі даних, яка є невід’ємною частиною подібних систем, повинен бути сформований відповідний набір таблиць.

Кожен продуктовий магазин має бізнес правила. Наступні бізнес-правила описані нижче:

* У разі продажу неякісного товару, покупець має право на відшкодування коштів за ціною товару або його заміну, згідно при наявності чеку купівлі-продажу даного товару;
* У разі придбання покупцем товару на певну суму, йому можуть нараховуватися бонуси, які він може використати на придбання нового товару.

В системі є власні бізнес-процеси:

* В продуктовому магазині є перелік необхідного асортименту товарів, які клієнт може купити одразу, сплативши вартість товарів на касі у готівковий або безготівковий спосіб;
* Вартість деяких видів товарів може бути знижена в разі приналежності їх до акційної групи товарів;
* У разі придбання товарів з протермінованим терміном реалізації або при наявності ознак погіршення органолептичних якостей товару, покупець має право обміну товару або відшкодування коштів за його вартістю.

В системі є два типи користувачів, їхні характеристики наступні:

Перший тип – це адміністратор торгівельного залу (старший продавець, директор). Це тип користувача, до обов’язків якого входить, перш за все, контроль і керівництво над роботою групи продавців, продавців-консультантів, касирів. У цього типу користувача є всі права та доступ до бази даних. Він налаштовує базу даних та надає права для інших типів користувачів. Може видаляти, додавати та редагувати інформацію записану в базу даних, оформляти необхідні звіти. Основне завдання користувача – перевіряти коректність ведення бази, очищувати базу від застарілих / некоректних даних. Окрім цього, він займається списанням, переоцінкою, резервуванням товару, бере участь в інвентаризації, веде класифікатор товару. На ньому лежить відповідальність за дотримання строків реалізації, за поповнення асортименту торгового залу, за своєчасні замовлення товару.

Другий тип – це працівник магазину (касир). Це тип користувача, що безпосередньо виконує касові операції та здійснює роботу по ефективному обслуговуванню покупців у сферах торгівлі. Має обмежені права доступу до даних, має лише частковий доступ до функцій системи. Йому надається змога додавати нових клієнтів, оформлювати замовлення, звіти та переглядати усю необхідну інформацію.



Рис.1.1 Діаграма прецедентів

* 1. **Специфікація вимог до системи обліку та продажу продуктів в продуктовому магазині**

**1.3.1. Вступ**

***1.3.1.1. Призначення і мета***

В даному підрозділі розглядаються вимоги до програмного продукту, призначеного для забезпечення успішної роботи та функціонування продуктового магазину. Метою даного продукту є забезпечення можливості створення, збереження, обробки та аналізу даних, що використовуються у роботі продуктового магазину, зокрема:

* Дані про клієнтів та працівників продуктового магазину;
* Дані про операції продажів та поставок;
* Дані про асортименти продуктів.

***1.3.1.1 Межі продукту***

Програмний продукт включає в себе базу даних для зберігання необхідної інформації та програмні засоби для роботи з цією базою, зокрема для отримання з неї певних даних, внесення нових та редагування існуючих.

**1.3.2. Загальний опис**

***1.3.2.1.Перспективи продукту***

Визначений у цій специфікації програмний продукт є самодостатнім і незалежним від інших систем, покликаний систематизувати роботу продуктових магазинів які займаються роздрібною торгівлею. Ідея виникла на основі систем для керування продажів та доставок необхідних продуктів в продуктовий магазин. Даний програмний продукт складається з двох підсистем: бази даних та програмного додатку для роботи з нею. Програмний додаток повинен містити всі функції для повноцінної роботи продуктового магазину та бути простим в використанні для недосвідчених користувачів.

***1.3.2.2. Характеристика продукту***

Завдання програмного продукту – надати працівникам продуктового магазину можливість зберігати дані про своїх клієнтів, про наявний асортимент продукції, про операції купівлі-продажу та мати зручний доступ до цих даних та інструменти для їх обробки. Основні функції продукту:

А) Збереження необхідних даних про клієнтів та перелік продуктів;

Б) Надання можливості отримати ці дані у формі, що відповідає конкретному випадку використання (читання даних);

В) Надання можливості редагувати ці записи (запис даних).

**Система для продажу та доставки продуктів в продуктовий магазин**

Керування асортиментом товарів

Керування працівниками

Керування послугами

Керування фінансами

Керування клієнтами

Рис.1.2. Характеристика продукту

***1.3.2.3. Класи користувачів та їх характеристики***

Дана система використовується всередині продуктового магазину, тому користувачами будуть працівники цього магазину та клієнти які здійснюють купівлю товарів. Поділ користувачів буде відбуватись за рівнями доступу відповідно до посади. Як приклад наведемо таку систему користувачів:

Таблиця 1.1. Типи користувачів системи

|  |  |
| --- | --- |
| Тип користувача | Права доступу |
| Адміністратор торгівельного залу  (старший продавець, директор) | Повний доступ до бази даних; додавання, видалення та редагування будь-яких записів в базі даних. |
| Касир продуктового магазину | Має обмежені права доступу до даних; додавання нових клієнтів, оформлення замовлень, звітів та перегляд необхідної інформації. |

***1.3.2.4. Середовище функціонування***

Для повноцінного функціонування системи необхідна платформа повинна відповідати наступним вимогам:

Таблиця 1.2. Системні вимоги ПК

|  |  |
| --- | --- |
| Процесор | x64 cумісний процесор з частотою 1.4 Ghz  (рекомендовано 2Ghz і вище) |
| Пам’ять | 1 Gb (рекомендовано не менше 2Gb із можливістю подальшого збільшення об’єму відповідно до збільшення бази даних. |
| Операційна система | WindowsXPSP3 (рекомендовано Windows 7 і вище) |
| Додаткові програмні компоненти | Microsoft .NETFramework 3.5, а також MicrosoftSQLServer 2012 і вище для організації бази даних, а також роботи  з системою керування базою |
| Інтернет | Наявність доступу до мережі інтернет є бажаною  для своєчасного оновлення компонентів системи |

Системні вимоги до продукту співпадають з системними вимогами до вищезгаданих компонентів та залежать від версій компонент (детальнішу інформацію можна отримати на сайті розробників цих компонент).

***1.3.2.5. Обмеження проектування та реалізації***

Програмний продукт немає апаратних обмежень, окрім тих, що згадані в попередньому пункті а також будь-яких обмежень, пов’язаних із сумісністю з іншими програмами. Система не повинна вимагати наявності додаткових програмних компонентів окрім Microsoft .NET Framework, а також Microsoft SQL Server.

Часові обмеження виконання проекту – 4 місяці.

Грошові рамки фінансування – 42 000$. З урахуванням того, що даний проект буде виконуватися двома особами із 8-ми годинним робочим днем та погодинною оплатою – 25$ на кожного.

***1.3.2.6. Документація користувача***

У комплекті з системою надаватиметься набір користувацьких інструкцій та рекомендацій по роботі з системою згідно із стандартом IEEE Std 1063-2001 «IEEE Standard for Software User Documentation».

**1.3.3. Характеристики системи**

***1.3.3.1. Авторизація користувачів***

Пріоритет **високий**. Авторизація користувача відповідно до його типу.

***1.3.3.1.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.3. Послідовність дій/відгук.

***1.3.3.1.2. Функціональні вимоги***

REQ-1: Вибір типу користувача при вході в систему.

REQ-2: Введення даних входу (логін/пароль) та перевірка їх коректності.

REQ-3: Приховування даних входу.

***1.3.3.2. Керування даними по проекту***

Пріоритет **високий**. Додавання, редагування та видалення даних по купівлі-продажі продуктів.

***1.3.3.2.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.4. Послідовність дій/відгук.

***1.3.3.2.2. Функціональні вимоги***

REQ-1: Додавання замовлення (Штрих-код, Назва продукту, Виробник, Кількість продукту, Акція на продукт, Ціна продукту, Дата здійснення купівлі, П.І.Б працівника, П.І.Б клієнта, Інформація про магазин, Загальна сума замовлення). Передбачити неправильне введення даних у наведені вище поля.

REQ-2: Редагування даних по купівлі-продажі продукту (Штрих-код, Назва продукту, Виробник, Кількість продукту, Акція на продукт, Ціна продукту, Дата здійснення купівлі, П.І.Б працівника, П.І.Б клієнта, Інформація про магазин, Загальна сума замовлення). Передбачити неправильне введення даних у наведені вище поля.

REQ-3: Видалення замовлення. Передбачити запит на видалення із повторним підтвердженням операції.

***1.3.3.3. Керування інформаційними даними продукту***

Пріоритет **високий**. Додавання, редагування та видалення даних про асортимент продуктів.

***1.3.3.3.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.5. Послідовність дій/відгук.

***1.3.3.3.2. Функціональні вимоги***

REQ-1: Додавання продукту (Назва продукту, Штрих-код, Виробник, Тип продукту, Кількість продукту, Розмір продукту, Дата виробництва продукту, Термін придатності продукту, Акція на продукт, Ціна продукту). Передбачити неправильне введення даних у наведені вище поля.

REQ-2: Редагування даних продукту (Назва продукту, Штрих-код, Виробник, Тип продукту, Кількість продукту, Розмір продукту, Дата виробництва продукту, Термін придатності продукту, Акція на продукт, Ціна продукту). Передбачити неправильне введення даних у наведені вище поля.

REQ-3: Видалення замовлення. Передбачити запит на видалення із повторним підтвердженням операції.

***1.3.3.4. Керування працівниками***

Пріоритет **високий**. Додавання, редагування та видалення даних про працівників.

***1.3.3.4.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.6. Послідовність дій/відгук.

***1.3.3.4.2. Функціональні вимоги***

REQ-1: Додавання працівника (П.І.Б, Дата народження, Робочий графік, Зарплата). Передбачити неправильне введення даних у наведені вище поля.

REQ-2: Редагування даних по працівнику (П.І.Б, Дата народження, Робочий графік, Зарплата). Передбачити неправильне введення даних у наведені вище поля.

REQ-3: Видалення працівника. Передбачити запит на видалення із повторним підтвердженням операції.

***1.3.3.5. Керування клієнтами***

Пріоритет **високий**. Додавання, редагування та видалення даних по клієнтам.

***1.3.3.5.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.7. Послідовність дій/відгук.

***1.3.3.5.2. Функціональні вимоги***

REQ-1: Додавання клієнта (П.І.Б, Дата народження, Моб.телефон, Номер дисконтної картки, Відсоток знижки). Передбачити неправильне введення даних у наведені вище поля.

REQ-2: Редагування даних по клієнту (П.І.Б, Дата народження, Моб.телефон, Номер дисконтної картки, Відсоток знижки). Передбачити неправильне введення даних у наведені вище поля.

REQ-3: Видалення клієнта. Передбачити запит на видалення із повторним підтвердженням операції.

***1.3.3.6. Керування акціями***

Пріоритет **середній**. Додавання, редагування та видалення даних про додаткові акції на певний продукт.

***1.3.3.6.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.8. Послідовність дій/відгук.

***1.3.3.6.2. Функціональні вимоги***

REQ-1: Додавання акцій (Назва акції, Відсоток акції, Дата початку акції, Дата кінця акції). Передбачити неправильне введення даних у наведені вище поля.

REQ-2: Редагування акцій (Назва акції, Відсоток акції, Дата початку акції, Дата кінця акції). Передбачити неправильне введення даних у наведені вище поля.

REQ-3: Видалення акцій. Передбачити запит на видалення із повторним підтвердженням операції.

***1.3.3.7. Керування мережами магазинів / складських приміщень***

Пріоритет **високий**. Додавання, редагування та видалення даних про наявну мережу магазинів / складських приміщень.

***1.3.3.7.1. Послідовність дія/відгук***

**База даних**

**Форма редагування даних**

**Користувач**

Рис. 1.9. Послідовність дій/відгук.

***1.3.3.7.2. Функціональні вимоги***

REQ-1: Додавання нового магазину/складу (Склад чи магазин, Назва магазину/складу, Адреса магазину/складу). Передбачити неправильне введення даних у наведені вище поля.

REQ-2: Редагування інформації про магазин/склад (Склад чи магазин, Назва магазину/складу, Адреса магазину/складу). Передбачити неправильне введення даних у наведені вище поля.

REQ-3: Видалення магазину/складу. Передбачити запит на видалення із повторним підтвердженням операції.

**1.3.4. Вимоги зовнішніх інтерфейсів**

***1.3.4.1. Користувацькі інтерфейси***

Інтерфейс повинен умовно складатися з двох частин: панель управління і таблиця для виведення даних. У першій частині є доступ до всіх функцій додатку, незалежно від даних у таблиці, з якими працює користувач. Дані в таблицях можна впорядковувати за їх вмістом. Адміністратор отримує доступ до всіх функцій програми, касир – тільки до тих, які відносяться до його безпосередніх обов’язків. При оновленні даних система сама заповнює комірки з відомою інформацією, полегшуючи роботу користувача. Дати не обов’язково вносити вручну: для цього передбачений графічний інтерфейс календаря.

В програмному продукті повинен бути присутній єдиний стандарт вигляду для всіх компонентів та вікон програми; відображатися поточний статус; індикація полів, обов’якових для заповнення; перевірка коректності введених даних.

***1.3.4.2. Апаратні інтерфейси***

Для роботи із системою рекомендується використовувати такі пристрої введення як клавіатура та мишка.

***1.3.4.3. Програмні інтерфейси***

Інтерфейс взаємодії з базою даних у якій вони будуть зберігатись. Програмний продукт не здійснює взаємодії з будь-яким іншим програмним забезпеченням, окрім того, яке описане у пункті 1.3.2. цієї специфікації.

**1.3.5. Нефункціональні вимоги до проекту.**

***1.3.5.1. Вимоги продуктивності***

Єдиною вимогою продуктивності є забезпечення існування прямої пропорційності між якістю апаратного забезпечення та продуктивністю, тобто краще апаратне забезпечення повинне забезпечувати кращу швидкість виконання операцій.

Система повинна максимально ефективно використовувати апаратні ресурси платформи на якій вона запущена. При виконанні запитів до бази даних не повинна виникати затримка.

***1.3.5.2. Вимоги надійності***

Система повинна виконувати періодичне збереження даних для того, щоб запобігти великим втратам у разі непередбачуваних ситуацій (перебої електроенергії тощо). Це завадить повній втраті важливих даних.

***1.3.5.3. Вимоги безпеки***

Програма повинна підтримувати автентифікацію користувача. Для автентифікації повинні використовуватися логін облікового запису та пароль. Пароль повинен бути прихований під час введення. Програма повинна надавати різні права доступу для різних класів користувачів.

***1.3.5.4. Атрибути якості програмного продукту***

Навчання роботі із системою повинно займати мінімальну кількість часу. Програма має бути зрозумілою та легкою у сприйнятті та запам’ятовуванні для користувача. Система повинна бути стабільною та ефективно використовувати апаратні ресурси та здійснювати свою роботу без перебоїв та втрат.

**РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ**

**2.1. Концептуальне моделювання предметної області**

**2.1.1. Опис сутностей**

База даних “DB\_NetworkOfGroceryStores” містить наступні сутності:

«Продукти» (Products), «Типи продуктів» (Subtypes), «Знижки/акції» (Sale),

«Магазин» (Store), «Резерв» (Reserve), «Клієнти» (Customers), «Працівники»

(Workers), «Графік роботи» (Timetable), «Замовлення» (Orders), «Чек» (Cheque).

*Продукти:*

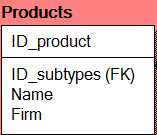


Рис. 2.1. Таблиця «Продукти».

Дана сутність містить загальну інформацію про каталог товарів продуктового

магазину:

Атрибути:

* *ID\_product* (Первинний ключ, INT) – поле ідентифікації продукту;
* *ID\_subtypes* (Зовнішній ключ, INT) – ідентифікатор типу продукту;
* *Name* (VARCHAR(80)) – поле, яке відображає назву продукту;
* *Firm*(VARCHAR(50)) – поле, яке відображає назву виробника (фірму) продукту;

*Типи продуктів:*

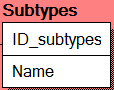


Рис. 2.2. Таблиця «Типи продуктів».

Сутність, яка несе в собі інформацію про тип кожного продукту:

Атрибути:

* *ID\_subtypes* (Первинний ключ, INT) – поле ідентифікації типу продукту;
* *Name* (VARCHAR(35)) – поле, яке відображає назву типу;

*Знижки / акції:*

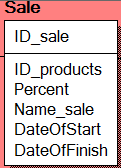


Рис. 2.3. Таблиця «Знижки / акції».

Сутність, яка несе в собі інформацію про знижки та акції на певні види товарів:

Атрибути:

* *ID\_sale* (Первинний ключ, INT) – поле ідентифікації знижки;
* *ID\_products* (Зовнішній ключ, INT) – ідентифікатор товару;
* *Percent* (INT) – поле, яке відображає відсоток знижки;
* *Name\_sale* (VARCHAR(50)) – поле, яке відображає назву акційної пропозиції;
* *DateOfStart* (DATE) – поле, яке відображає дату початку активації знижки;
* *DateOfFinish* (DATE) – поле, яке відображає дату завершення акційної пропозиції;

*Магазин:*

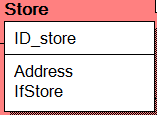


Рис. 2.4. Таблиця «Магазин».

Сутність, яка відображає інформацію про продуктовий магазин чи склад:

Атрибути:

* *ID\_store* (Первинний ключ, INT) – поле ідентифікації магазину / складу.
* *Address* (VARCHAR(80)) – поле, яке відображає інформацію про розташування магазину / складу;
* *IfStore* (INT) – поле, яке містить інформацію про те, чи це є склад чи магазин;

*Резерв:*

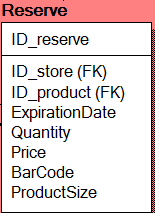


Рис. 2.5. Таблиця «Резерв».

Сутність, яка відображає інформацію про асортимент товарів, наявних в продуктовому магазині чи складі:

Атрибути:

* *ID\_reserve* (Первинний ключ, INT) – поле ідентифікатора.
* *ID\_store* (Зовнішній ключ, INT) – поле ідентифікації магазину / складу;
* *ID\_product* (Зовнішній ключ, INT) - поле ідентифікації продукту;
* *ExpirationDate* (DATE) – поле для відображення інформації про термін придатності;
* *Quantity* (INT) – поле для відображення кількості товару;
* *Prise* (FLOAT) – поле для відображення ціни продукту;
* *BarCode* (VARCHAR(13)) – поле для відображення штрих-коду продукту;
* *ProductSize* (FLOAT) – поле, яке відображає об’єм (вагу) продукту;

*Клієнти:*

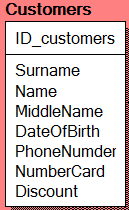


Рис. 2.6. Таблиця «Клієнти».

Сутність, яка відображає інформаційні дані про усіх наявних клієнтів продуктового магазину:

Атрибути:

* *ID\_customer* (Первинний ключ, INT) – поле ідентифікації клієнта;
* *Surname* (VARCHAR(50)) – поле, яке відображає прізвища клієнта;
* *Name* (VARCHAR(50)) – поле, яке відображає імені клієнта;
* *MiddleName* (VARCHAR(50)) – поле, яке відображає по-батькові клієнта;
* *PhoneNumber* (VARCHAR(20)) – поле, яке відображає номер телефону клієнта;
* *NumberCard* (VARCHAR(16)) – поле, яке відображає номер дисконтної картки клієнта;
* *DateOfBirth* (DATE) – поле, яке відображає інформацію про дату народження клієнта;
* *Discount* (INT) - поле, яке відображає інформацію про відсоток знижки клієнта;

*Працівники:*

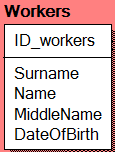


Рис. 2.7. Таблиця «Працівники».

Сутність, яка відображає інформаційні дані про усіх наявних працівників продуктового магазину:

Атрибути:

* *ID\_workers* (Первинний ключ, INT) – поле ідентифікації працівника;
* *Surname* (VARCHAR(50)) – поле, яке відображає прізвище працівника;
* *Name* (VARCHAR(50)) – поле, яке відображає ім’я працівника;
* *MiddleName* (VARCHAR(50)) – поле відображення по-батькові працівника;
* *DateOfBirth* (DATE) – поле, яке відображає інформацію про дату народження працівника;

*Графік роботи:*

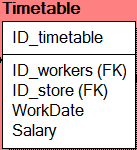


Рис. 2.8. Таблиця «Графік роботи».

Сутність, що містить інформацію про графік роботи працівників магазину / складу:

Атрибути:

* *ID\_timetable* (Первинний ключ, INT) – поле ідентифікації робочого дня працівника;
* *ID\_workers* (Зовнішній ключ, INT) –поле ідентифікації працівника;
* *ID\_store* (Зовнішній ключ, INT) – поле ідентифікації магазину / складу;
* *WorkDate* (DATE) – поле, що відображає дату робочого дня працівника;
* *Salary* (FLOAT) – поле для відображення зарплати працівника;

*Замовлення:*

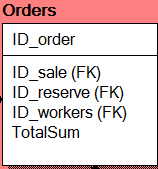


Рис. 2.9. Таблиця «Замовлення».

Сутність, яка надає інформацію про всі замовлення здійснені клієнтом:

Атрибути:

* *ID\_order* (Первинний ключ, INT) – ідентифікатор номеру замовлення клієнтом;
* *ID\_sale* (Зовнішній ключ, INT) – поле ідентифікації знижки;
* *ID\_reserve* (Зовнішній ключ, INT) – поле ідентифікації резерву;
* *ID\_workers* (Зовнішній ключ, INT) – поле ідентифікації працівника;
* *TotalSum* (FLOAT) – поле, яке відображає загальну суму замовлення:

*Чек:*

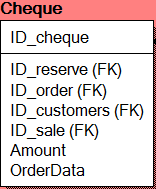


Рис. 2.10. Таблиця «Чек».

Сутність, для виводу кінцевої інформації про здійснену покупку клієнтом:

Атрибути:

* *ID\_cheque* (Первинний ключ, INT) – поле ідентифікації вихідного рахунку

клієнта (чеку);

* *ID\_order* (Зовнішній ключ, INT) - поле ідентифікації замовлення;
* *ID\_reserve* (Зовнішній ключ, INT) – поле ідентифікації резерву;
* *ID\_customer* (Зовнішній ключ, INT) – поле ідентифікації клієнта;
* *ID\_sale* (Зовнішній ключ, INT) – поле ідентифікації знижки;
* *Amount* (INT) – поле, яке відображає інформацію про кількість продуктів;
* *OrderDate* (DATE) – поле, яке відображає дату здійснення купівлі клієнтом;

**2.1.2. Опис зв’язків між сутностями**

* Між сутностями «Subtypes» і «Products» встановлено зв’язок один до багатьох, через те, що є великий вибір продовольчих товарів, які належать до одного типу.
* Сутності «Products» і «Reserve» з’єднані зв’язком один до багатьох, через те, що є в багатьох резервах магазину чи складу міститься певний продукт.
* Сутності «Reserve» і «Cheque» з’єднані зв’язком один до багатьох, через те, що можна купляти багато продуктів, які знаходяться в даний момент в конкретному магазині чи складі.
* Сутності «Store» і «Reserve» з’єднані зв’язком один до багатьох, через те, що в одному магазині чи складі є багато асортиментів продуктів.
* Сутності «Reserve» і «Orders» з’єднані зв’язком один до багатьох. Тобто в багатьох замовленнях клієнта може міститися певний продукт, який є на складі чи магазині (тобто, з одного резерву клієнт може придбати багато одиниць певного товару).
* Сутності «Orders» і «Cheque» з’єднані зв’язком один до багатьох, через те, що одне замовлення товару може містити багато вихідних чеків.
* Сутності «Customers» і «Cheque» з’єднані зв’язком один до багатьох, так як один клієнт може здійснити багато замовлень.
* Сутності «Sale» і «Orders» з’єднані зв’язком один до багатьох, через те, що багато замовлень можуть містити одну акційну знижку.
* Сутності «Sale» і «Cheque» з’єднані зв’язком один до багатьох. через те, що на велику кількість чеків є певна знижка.
* Сутності «Workers» і «Orders» з’єднані зв’язком один до багатьох, через те, що один працівник магазину може виконувати багато замовлень клієнта (в багатьох замовленнях клієнта приймає участь один працівник).
* Сутності «Workers» і «Timetable» з’єднані зв’язком один до багатьох, через те, що один працівник магазину може працювати за певним графіком роботи в різні дати та зміни.
* Сутності «Store» і «Timetable» з’єднані зв’язком один до багатьох, через те, що в одному магазині чи складі є різні розпорядки робочих днів та годин (змін).

**2.2. Логічне проектування бази даних**

Після реалізації сутностей було створено зв’язки між ними за допомогою існуючих в середовищі CAErwinDataModeler зв’язків: «один-до-багатьох» (Non-IdentifyingRelationship), «багато-до-багатьох» (Many-to-manyRelationship). При цьому середовище автоматично згенерує зовнішні ключі та доповнить ними сутності. Для кожної таблиці створюється кластерний індекс за первинним ключем. Первинні ключі автоінкрементні. Атрибути допускають значення NULL, за винятком ключів.

Після створення логічної моделі до переходу у фізичну слід обрати пункт меню :”Model”->”PhysicalModel”. Після цього середовище автоматично доповнює модель транзитними сутностями для нормалізації відношень «багато-до-багатьох». Слід перевірити правильність зв’язків та виконати перевірку схеми на можливість конвертації в SQL-код за допомогою пункту меню «Tools»-“Forward Engineer”-“Check Model”.

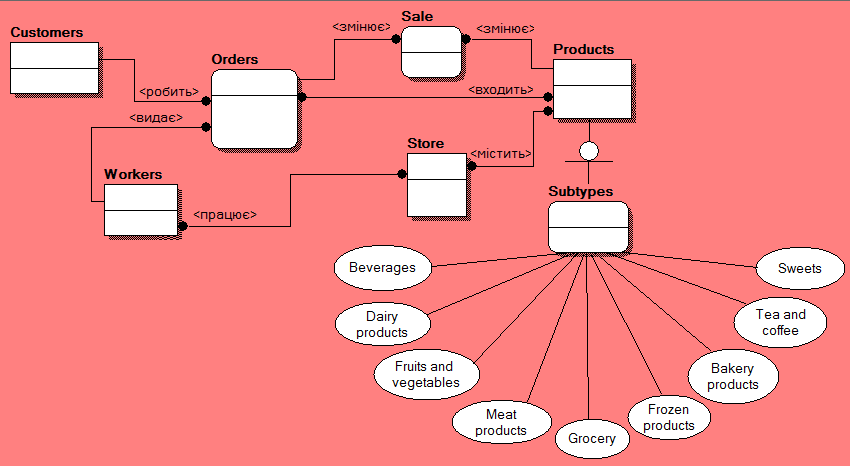


Рис.2.11. Концептуальна модель БД.

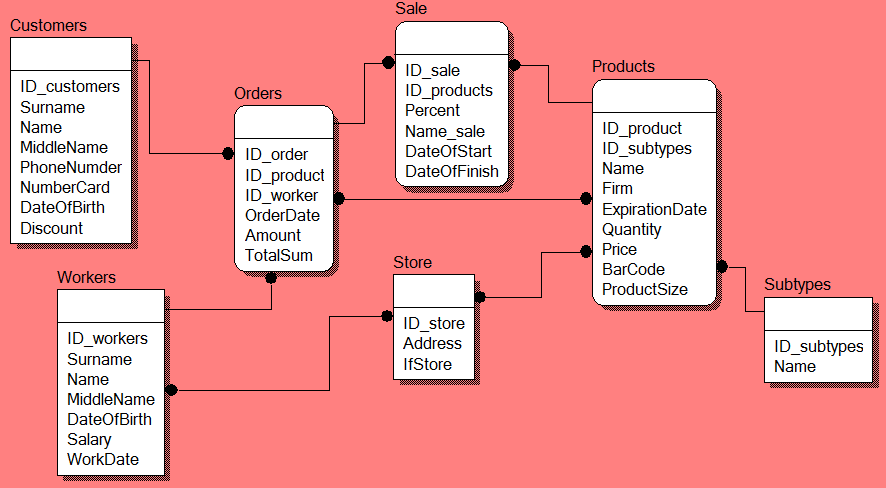


Рис.2.12. Логічна модель БД.

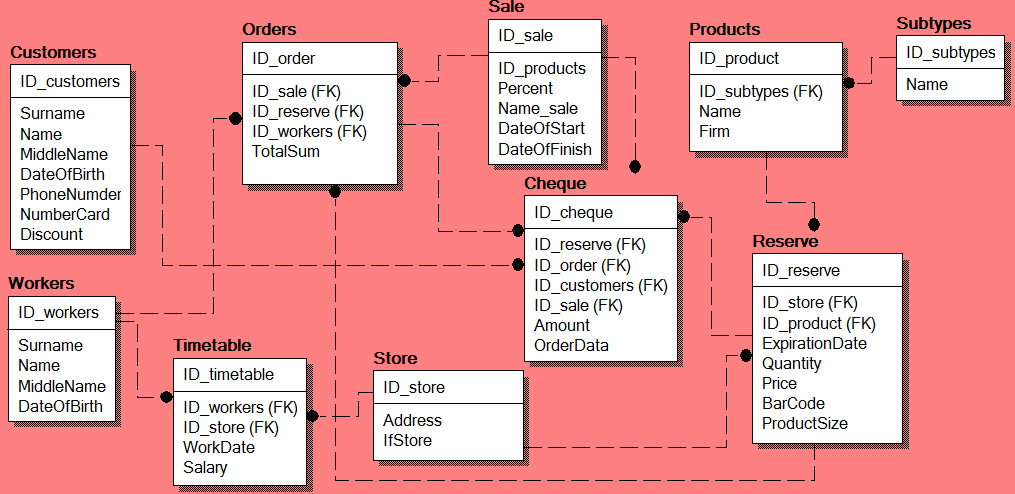


Рис.2.13. Фізична модель БД.

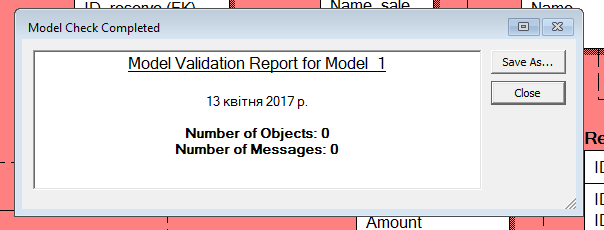


Рис.2.14. Перевірка фізичної моделі

Наступним кроком є генерація скрипта бази даних. На цьому кроці відбувається злиття двох БД відповідних за постачання товарів та збут (безпосередній продаж). Для цього потрібно переключитися на фізичну модель і на панелі інстументів вибрати "Forward Engineer ->Schema Generation". Натиснувши кнопку "Preview" ми отримаємо код скрипта (*Скрипт створення БД внесено у додаток А)*.

Після генерації скрипта, я перенесла його у середовище SQL Server Management Studio 2014 та з його допомогою створила базу даних “ DB\_NetworkOfGroceryStores ” та заповнила початковими вхідними даними.

 Рис.2.15. Діаграма бази даних в SQL Server 2014.

Для полегшення роботи з даними створено такі віртуальні таблиці:

* vTimetableWorkers – віртуальна таблиця для відображення всієї інформації про працівника продуктового магазину (Прізвище, Ім’я, Побатькові, Дата роботи, Зарплата).
* vInfoProducts – віртуальна таблиця для відображення всієї інформації про наявні продукти в магазині (Назва продукту, Штрих-код, Фірма, Тип продукту, Ціна).
* vCheque – віртуальна таблиця для відображення всієї чекової інформації (інформації про замовлення клієнтів) (Назва продукту, Фірма, Штрих-код продукту, Ціна, Кількість, Загальна сума, Знижка на товар, Прізвище клієнта, Ім’я клієнта, По батькові клієнта, Знижка клієнта, Прізвище продавця, Ім’я продавця, Дата здійснення купівлі, Адреса магазину).

Створено такі обмеження:

Для таблиці *Customers*:

* PhoneNumber – обмеження на ввід більшої кількості символів за допустиму кількість цифер в мобільному номері клієнта.
* NumberCard - обмеження на ввід більшої кількості символів за допустиму кількість цифер в номері дисконтної карти клієнта.

Для таблиці *Reserve*:

* BarCode – обмеження на штрих-код товару.
* Quantity – обмеження на вихід за нульові межі.

**2.3. Проектування типових запитів і транзакцій**

Я вирішила відмовитись від використання індексів у своїй базі даних оскільки середовищем функціонування є невелика організація. Відповідно до цього база буде містити невелику кількість записів і функціонуватиме достатньо швидко.

Код основних операцій взаємодії з базою даних подано нижче:

**2.3.1. Додавання продукту в таблицю Products**

Для того, щоб продавати продукти, потрібно спочатку додати продукти в таблицю Products, що показуватиме весь можливий асортимент продуктів, що будуть можливими для купівлі-продажу:

INSERT INTO Products

VALUES (10, 'Окунь', 'Західний Буг');

**2.3.2. Вибірка продуктів**

Даний запит буде використовуватись для вибору продуктів за заданими критеріями (в даному випадку, вивід всіх продуктів, в яких термін придатності протерміновано):

SELECT pr.Name, pr.Firm, res.BarCode, res.Quantity, res.Price, res.ExpirationDate

FROM Products pr, Reserve res

WHERE res.ID\_product=pr.ID\_product AND

res.ExpirationDate < GETDATE();

**2.3.3. Додавання знижок**

Для додавання знижок в таблицю Sale, можна використовувати такий запит:

INSERT INTO Sale

VALUES(10116, 2, 'Акція дня', '2017-04-29', '2017-04-30');

**2.3.4. Зчитування наявних знижок на товари**

Зчитування наявних знижок на товари та сортування за відсотком знижки в порядку спадання можна виконати з допомогою наступного запиту:

SELECT prd.Name, prd.Firm, sal.Percents

FROM Products prd, Sale sal

WHERE sal.ID\_products = prd.ID\_product

ORDER BY sal.Percents DESC;

**2.3.5. Додавання нового клієнта**

Кожного клієнта потрібно додати в таблицю, після чого можна працювати з ним. Код запиту наведено нижче:

INSERT INTO Customers

VALUES ('Урбанська', 'Надія', 'Романівна', '1997-10-19', '+38067488908', 2500009754456, 0);

**2.3.6. Зчитування інформації про продукт за штрих-кодом**

Вивід всієї інформації про продукт за штрих-кодом здійснюється за допомогою коду:

SELECT pr.Name, pr.Firm, res.BarCode, res.Quantity, res.Price, res.ExpirationDate

FROM Products pr, Reserve res

WHERE res.ID\_product=pr.ID\_product AND

res.BarCode = 5997207710014;

**2.3.7. Додавання нового замовлення**

Щоб додати нове замовлення за відомим штрих-кодом товару та прізвищем продавця і загальної суми, використовуємо такий код:

INSERT INTO Orders (ID\_sale,ID\_reserve,ID\_workers,TotalSum)

VALUES(

(SELECT sal.ID\_sale FROM Sale sal, Reserve res

WHERE res.BarCode = 5997207710014 AND sal.ID\_products=res.ID\_product),

(SELECT ID\_reserve FROM Reserve WHERE BarCode = 5997207710014),

(SELECT ID\_workers FROM Workers WHERE Surname = 'Носаль'), 123.98)

**2.3.8. Оновлення інформації про клієнта**

Часто необхідно оновити інформацію про клієнта. Для цієї операції використовується код:

UPDATE Customers

SET PhoneNumder ='+38067488908' WHERE NumberCard = 2500009754456

**2.3.9. Видалення інформації про клієнта**

Клієнта можна видаляти з БД для цього використовується наступний код:

DELETE \* FROM Customers cus

WHERE ID\_customers = (SELECT ID\_customers FROM Cheque WHERE ID\_customers = cus.ID\_customers);

**2.3.10. Видалення інформації про знижку**

Щоб видалити інформацію про знижки, які на даний момент не є дійсними (застарілими) виконаємо код:

DELETE

FROM Sale

WHERE DateOfFinish < GETDATE();

**2.3.11. Додавання інформації про магазин**

Код запиту наступний:

INSERT INTO Store

VALUES ('ТзОВ ТВК "Львівхолод" Магазин "Рукавичка" м.Львів, вул.Шота Руставелі, 40', 1)

**2.3.12. Редагування інформації про продукт**

При зміні будь-якої інформації (в даному випадку ціни) про продукт в резерві за певним параметром (штрих-кодом) використовується наступний код:

UPDATE Reserve

SET Price = 89.95

WHERE

BarCode = 5997289942214

**2.3.13. Редагування інформації про працівника**

Щоб змінити заробітню плату працівника за прізвищем, можна використати наступний код щоб модифікувати дані:

UPDATE Timetable

SET Salary = 600

WHERE ID\_workers = (SELECT ID\_workers FROM Workers WHERE Surname = 'Марцінко' AND Name = 'Надія')

**2.3.14. Редагування кінцевої суми в замовленні**

При редагуванні кінцевої суми в замовленні, це можна зробити наступним запитом:

UPDATE Orders

SET TotalSum = (SELECT Price FROM Reserve WHERE Orders.ID\_reserve = Reserve.ID\_reserve) / (SELECT Percents FROM Sale WHERE Orders.ID\_sale = Sale.ID\_sale)

**2.3.15. Вивід списку продуктів по торгових відділах**

Для того, щоб вивести список продуктів (інформація про назву та штрих-код продукту) по торгових відділах виконаємо код:

DECLARE @NAME\_SUBTYPE VARCHAR(35),

@ID INT,

@NAME\_PRODUCT VARCHAR(80),

@FIRM VARCHAR(50),

@PRICES FLOAT,

@BARCODE VARCHAR(13),

@message VARCHAR(100)

PRINT 'СПИСОК ПРОДУКТІВ ПО ТОРГОВИХ ВІДДІЛАХ: '

DECLARE cursorSubtype CURSOR LOCAL FOR

SELECT Name, ID\_subtypes

FROM Subtypes

-- ORDER BY ID\_groupProduct

OPEN cursorSubtype

FETCH NEXT FROM cursorSubtype

INTO @NAME\_SUBTYPE, @ID

WHILE @@FETCH\_STATUS=0

BEGIN

PRINT ' '

PRINT ' '

PRINT ' '

SELECT @message='Група \*\*\*\*\*\* '+ @NAME\_SUBTYPE

PRINT @message

PRINT '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

SELECT @message='Назва продукту Виробник Ціна Штрихкод'

PRINT @message

PRINT '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

DECLARE cursorProduct CURSOR FOR

SELECT prd.Name, prd.Firm, res.Price, res.BarCode

FROM Reserve res, Products prd

WHERE prd.ID\_subtypes = @ID

AND prd.ID\_product = res.ID\_product

OPEN cursorProduct

FETCH NEXT FROM cursorProduct

INTO @NAME\_PRODUCT, @FIRM, @PRICES, @BARCODE

IF @@FETCH\_STATUS<>0

PRINT ' Немає продуктів'

WHILE @@FETCH\_STATUS=0

BEGIN

SELECT @message= CONVERT(VARCHAR(80),@NAME\_PRODUCT,103)+' '+ CONVERT(VARCHAR(50),@FIRM,103)+' '+ CONVERT(VARCHAR(10),@PRICES,103)+' '+ CONVERT(VARCHAR(50),@BARCODE,103)

PRINT @message

FETCH NEXT FROM cursorProduct

INTO @NAME\_PRODUCT, @FIRM, @PRICES, @BARCODE

END

CLOSE cursorProduct

DEALLOCATE cursorProduct

FETCH NEXT FROM cursorSubtype

INTO @NAME\_SUBTYPE, @ID

END

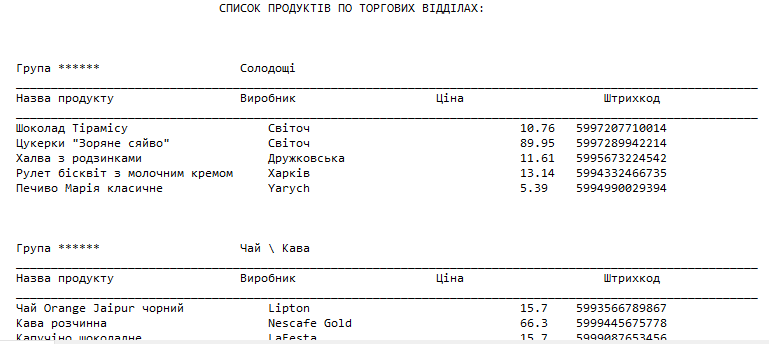


Рис.2.16 Результат виконання коду.

**2.3.16. Транзакція оновлення інформації про товар на складі**

DECLARE

@tempID\_reserve INT,

@tempID\_store INT = 1,

@tempID\_product INT = 10104,

@tempID\_sal\_res\_inv INT = 90003,

@tempExpirationDate DATE = '2017-11-28',

@tempQuantity INT = 14,

@tempPrice FLOAT = 13.14,

@tempBarCode VARCHAR(13) = 5994332466735,

@temProductSize FLOAT = 290

--------------------TRASACTION------------------------------------

BEGIN TRY

BEGIN TRAN

SET @tempID\_reserve = ISNULL((SELECT Reserve.ID\_reserve FROM Reserve WHERE Reserve.ID\_product = @tempID\_Product), 0)

IF @tempID\_reserve = 0

BEGIN

INSERT INTO Reserve(ID\_reserve, ID\_store, ID\_product, ID\_sal\_res\_inv, ExpirationDate, Quantity, Price, BarCode, ProductSize)

VALUES (0 , @tempID\_store, @tempID\_product, @tempID\_sal\_res\_inv, @tempExpirationDate, @tempQuantity, @tempPrice, @tempBarCode, @temProductSize)

END

ELSE

BEGIN

UPDATE Reserve

SET Reserve.ExpirationDate = @tempExpirationDate, Reserve.Quantity = @tempQuantity, Reserve.Price = @tempPrice, Reserve.BarCode = @tempBarCode, Reserve.ProductSize = @temProductSize

WHERE Reserve.ID\_reserve = @tempID\_reserve

END

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

END CATCH;

GO

**2.3.17. Видалення запису в таблиці за відомим полем (збережена процедура, транзакція)**

CREATE PROCEDURE proc\_deleteWorkers AS

SELECT

ERROR\_NUMBER()AS ErrorNumber,

ERROR\_SEVERITY()AS ErrorSeverity,

ERROR\_STATE()AS ErrorState,

ERROR\_LINE()AS ErrorLine,

ERROR\_PROCEDURE()AS ErrorProcedure,

ERROR\_MESSAGE()AS ErrorMessage;

SET XACT\_ABORT ON;

BEGIN TRY

BEGIN TRANSACTION;

DELETE FROM Workers

WHERE Surname = 'Заліско'

COMMIT TRANSACTION;

END TRY BEGIN CATCH

EXECUTE proc\_deleteWorkers;

IF (XACT\_STATE())=-1

BEGIN PRINT'Транзація нефіксована. Відкат'

ROLLBACK TRANSACTION; END;

IF (XACT\_STATE())= 1 BEGIN

PRINT'Транзакція фіксована. COMMIT'

COMMIT TRANSACTION;

END;

END CATCH;

**2.3.18. Вставка нового запису в таблицю (збережена процедура, транзакція)**

CREATE PROCEDURE proc\_InsertWorkers AS

SELECT

ERROR\_NUMBER()AS ErrorNumber,

ERROR\_SEVERITY()AS ErrorSeverity,

ERROR\_STATE()AS ErrorState,

ERROR\_LINE()AS ErrorLine,

ERROR\_PROCEDURE()AS ErrorProcedure,

ERROR\_MESSAGE()AS ErrorMessage;

SET XACT\_ABORT ON;

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO Workers

(Surname, Name, MiddleName, DateOfBirth)

VALUES('Урбанська','Надія', 'Ярославівна', '1980-09-20')

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

EXECUTE proc\_InsertWorkers;

IF (XACT\_STATE())=-1

BEGIN PRINT'Транзація нефіксована. Відкат'

ROLLBACK TRANSACTION;END;

IF (XACT\_STATE())= 1 BEGIN

PRINT'Транзакціяфіксована. COMMIT'

COMMIT TRANSACTION;

END;

END CATCH;

**2.3.19. Оптимізація виконання запитів**

CREATE UNIQUE INDEX Searsh\_customer\_NumberCard ON Customers(NumberCard)

CREATE UNIQUE INDEX Searsh\_workers\_Surname ON Workers(Surname)

CREATE UNIQUE INDEX Searsh\_products\_Name ON Products(Name)

CREATE UNIQUE INDEX Searsh\_store\_Name ON Store(Name)

CREATE UNIQUE INDEX Searsh\_reserve\_BarCode ON Reserve(BarCode)

CREATE UNIQUE INDEX Searsh\_subtypes\_Name ON Subtypes(Name)

CREATE UNIQUE INDEX NCLIX\_Products\_Idproduct ON Products(ID\_product)

CREATE UNIQUE INDEX NCLIX\_Reserve\_BarCode ON Reserve(BarCode)

**РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ РОБОТИ З БАЗОЮ ДАНИХ**

**3.1. Реалізація доступу до даних**

Для реалізації доступу до даних мною було обрана технологія ADO.NET. ADO.NET (ActiveX Data Objects .NET) — це набір бібліотек, що поставляється з Microsoft .NET Framework і призначений для взаємодії з різними сховищами даних. Бібліотеки ADO.NET включають класи для приєднання до джерела даних, виконання запитів і обробки їхніх результатів. Крім того, ADO.NET можна використовувати в якості надійного, ієрархічно організованого, відокремленого кешу даних для автономної роботи з даними.

Реалізацію програмного додатку взаємодії з базою даних було вирішено здійснити на основі платформи .NETFramework 4.6.1 за допомогою мови програмування C# 6.0.

Перевагою платформи ADO.NET є те, що вона дозволяє розробникам створювати додатки для доступу до даних, що працюють з концептуальною моделлю, а не безпосередньо з реляційною схемою зберігання.

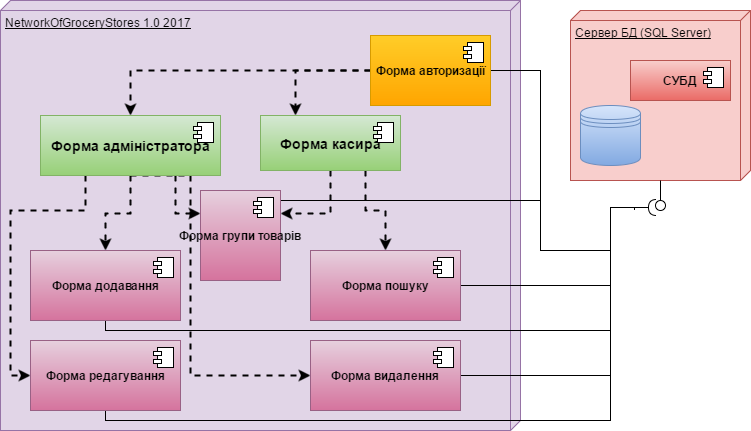


Рис. 3.1. Діаграма компонентів.

Код встановлення з’єднання з базою даних:

SqlConnection conn;

conn = new SqlConnection(@"Server = (local); Database = DB\_NetworkOfGroceryStores;

Uid = sa; Pwd = 11111111;");

conn.Open();

* 1. **Реалізація функціональних характеристик системи**
     1. **Авторизація**

Авторизація реалізована наступним чином:

За допомогою форми AcountWindow користувач вводить свій логін та пароль (заздалегідь внесений в реєстр паролей). Приховування пароля реалізовано за допомогою компоненти PasswordBox. На основі введеного логіна користувача відбувається підключення до бази даних. Обробник входу містить наступний код:

public partial class AcountWindow : Window

{

public static bool AdminOrNot = false;

public static string NAMEUSER;

public static string NAMESTORE;

public static string ADRESSSTORE;

public AcountWindow()

{

InitializeComponent();

}

private void AvtorisationButton\_Click(object sender, RoutedEventArgs e)

{

try

{

if (Login.Text == "dbadmin" && Password.Password == "12345")

{

this.Visibility = Visibility.Visible;

NAMEUSER = "Адміністратор Урбанська Надія";

NAMESTORE = "ТзОВ ТВК Львівхолод Магазин Рукавичка";

ADRESSSTORE = "м.Львів, вул.Золота, 17";

AdminOrNot = true;

this.Hide();

ShopMainWindow adminPage = new ShopMainWindow();

adminPage.Show();

}

else if (Login.Text == "cashier" && Password.Password == "02042017")

{

AdminOrNot = false;

NAMEUSER = "Касир Урбанська Надія";

NAMESTORE = "ТзОВ ТВК Львівхолод Магазин Рукавичка";

ADRESSSTORE = "м.Львів, вул.Золота, 17";

this.Hide();

ShopMainWindow cashierPage = new ShopMainWindow();

cashierPage.Show();

}

else

{

MessageBox.Show("Не правильно введено логін або пароль! Будьте уважні!", "Помилка: ", MessageBoxButton.OK, MessageBoxImage.Error);

Login.Clear();

Password.Clear();

label1.Foreground = new SolidColorBrush(Colors.Red);

label1\_Copy.Foreground = new SolidColorBrush(Colors.Red);

}

}

catch (System.Exception ex)

{

MessageBox.Show(ex.Message);

}

}}

### **Вибірка з бази даних відповідних стовбців для певного типу користувача**

Вибір даних відбувається в залежності від користувача, який ввійшов в систему. Кожен користувач має свої права на доступ до БД та роботу з даними.

public partial class SubtypesAllPoductsWindow : Window

{

SqlConnection conn;

public SubtypesAllPoductsWindow()

{

InitializeComponent();

if (!AcountWindow.AdminOrNot)

{

UpdateGoods\_button.Visibility = Visibility.Hidden;

UpdateSale\_button.Visibility = Visibility.Hidden;

}

}

private void Milk\_button\_Click(object sender, RoutedEventArgs e)

{

try

{

conn = new SqlConnection(@"Server = (local); Database = DB\_NetworkOfGroceryStores; Uid = sa; Pwd = 11111111;");

conn.Open();

SqlDataAdapter adapt = new SqlDataAdapter("SELECT Name, Firm, Price, Quantity, ProductSize FROM vProductOfSubtypes Where NameSubtypes = 'Молочні продукти'", conn);

DataSet ds = new DataSet();

DataTable dt = new DataTable();

adapt.Fill(dt);

dt.Columns["Name"].ColumnName = "Назва продукту";

dt.Columns["Firm"].ColumnName = "Назва фірми";

dt.Columns["Price"].ColumnName = "Ціна";

dt.Columns["Quantity"].ColumnName = "Кількість";

dt.Columns["ProductSize"].ColumnName = "Розмір";

ProductOfSubtypes\_dataGrid.ItemsSource = dt.DefaultView;

conn.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

conn.Close();

}

}

Даний метод вибору даних використовується у всіх вибірках з БД.

* + 1. **Додавання нових даних в БД після заповнення користувачем форми**

Методи додавання даних, так само як і методи вибірки, реалізовані за одним шаблоном.

private void AddCustomer\_button\_Click(object sender, RoutedEventArgs e)

{

try

{

conn = new SqlConnection(@"Server = (local); Database = DB\_NetworkOfGroceryStores; Uid = sa; Pwd = 11111111;");

conn.Open();

SqlCommand query = new SqlCommand("INSERT INTO Customers VALUES('" +Surname.Text+"','"+Name.Text+"', '"+MiddleName.Text+"', '"+DateTime.Parse(DataBirth.Text)+"','"+Phone.Text+"', '"+NumberCard.Text+"', "+Int16.Parse(Discount.Text)+")", conn);

query.ExecuteNonQuery();

MessageBox.Show("Клієнта записано в базу даних!", "Додавання: ", MessageBoxButton.OK, MessageBoxImage.Information);

Surname.Text = "";

Name.Text = "";

MiddleName.Text = "";

DataBirth.Text = "";

Phone.Text = "";

NumberCard.Text = "";

Discount.Text = "";

conn.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

conn.Close();

}

}

* + 1. **Редагування даних продуктового магазину після заповнення користувачем форми редагування даних**

Користувач вибирає комірку яку він хоче відредагувати після чого він повинен відредагувати дані.

//Select

public void SelectGoods()

{

try

{

conn = new SqlConnection(@"Server = (local); Database = DB\_NetworkOfGroceryStores; Uid = sa; Pwd = 11111111;");

conn.Open();

SqlDataAdapter adapt = new SqlDataAdapter("SELECT vpi.ID\_product, vpi.[Name], res.BarCode, vpi.Firm, sub.[Name], res.Price FROM Products vpi, Subtypes sub, Reserve res Where vpi.ID\_subtypes = sub.ID\_subtypes And vpi.ID\_product = res.ID\_product", conn);

DataSet ds = new DataSet();

DataTable dt = new DataTable();

adapt.Fill(dt);

dt.Columns["Name"].ColumnName = "Назва продукту";

dt.Columns["BarCode"].ColumnName = "Штрих-код";

dt.Columns["Firm"].ColumnName = "Фірма";

dt.Columns["name1"].ColumnName = "Назва типу";

dt.Columns["Price"].ColumnName = "Ціна";

GoodsShop\_dataGrid.ItemsSource = dt.DefaultView;

conn.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

conn.Close();

}

}

private void EditWorkers\_button\_Click(object sender, RoutedEventArgs e)

{

try

{

DataRowView rowSelected = GoodsShop\_dataGrid.SelectedItem as DataRowView;

if (rowSelected != null)

{

string id = rowSelected[0].ToString();

string name = rowSelected[1].ToString();

string barcode = rowSelected[2].ToString();

string firm = rowSelected[3].ToString();

string name\_gr = rowSelected[4].ToString();

string price = rowSelected[5].ToString();

Console.WriteLine(rowSelected[0].ToString());

conn = new SqlConnection(@"Server = (local); Database = DB\_NetworkOfGroceryStores; Uid = sa; Pwd = 11111111;");

conn.Open();

SqlCommand query = new SqlCommand("UPDATE Subtypes SET Name = '"+name\_gr+ "' WHERE ID\_subtypes = (SELECT ID\_subtypes From Products Where ID\_product = " + id + ")", conn);

query.ExecuteNonQuery();

SqlCommand query1 = new SqlCommand("UPDATE Products SET Name = '" + name + "', Firm = '" + firm + "' WHERE ID\_product = '" + id + "'", conn);

query1.ExecuteNonQuery();

SqlCommand query2 = new SqlCommand("UPDATE Reserve SET Price = "+float.Parse(price)+", BarCode = '"+barcode+"' WHERE ID\_product = '" + id + "'", conn);

query2.ExecuteNonQuery();

conn.Close();

MessageBox.Show("Товар відредаговано в базі даних!", "Редагування: ", MessageBoxButton.OK, MessageBoxImage.Information);

}

SelectGoods();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

conn.Close();

}

* 1. **Реалізація функціональних характеристик системи**

Після запуску програми з’являється форма входу. Користувачу, потрібно ввести свій логін і пароль та натиснути кнопку «Ввійти».

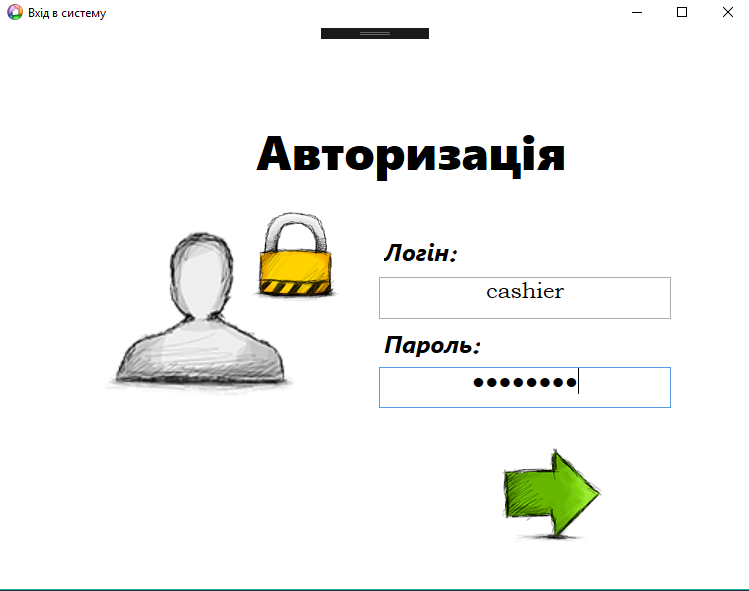


Рис.3.2. Форма авторизації.

Якщо в одне з полів не ввести дані, програма видасть повідомлення про помилку:

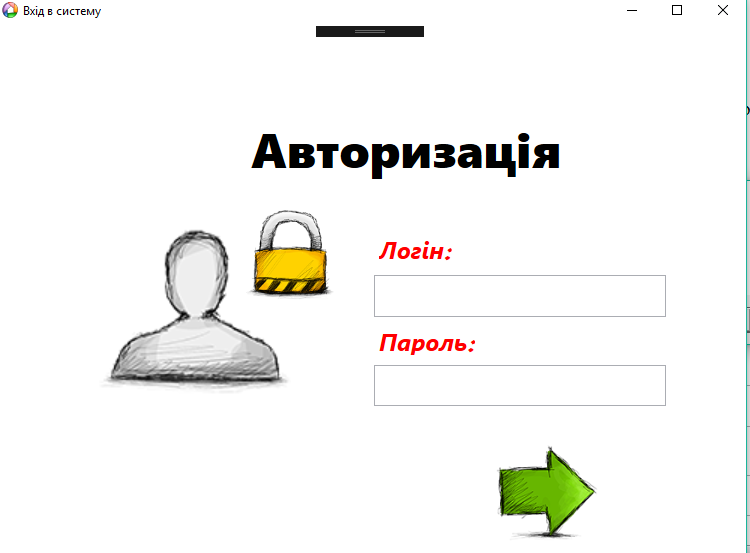
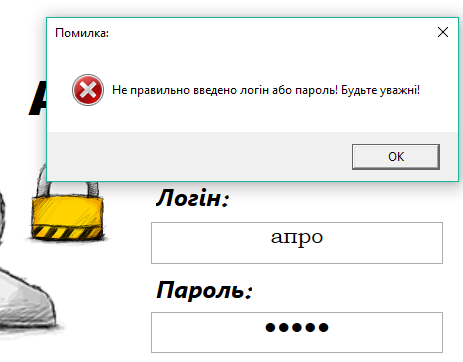


Рис.3.3. Повідомлення про помилку авторизації.

У разі вдалого входу відкриється відповідна до користувача форма. Якщо це буде користувач «Адміністратор», то відкриється форма яка містить в собі всі елементи доступні адміністратору продуктового магазину.

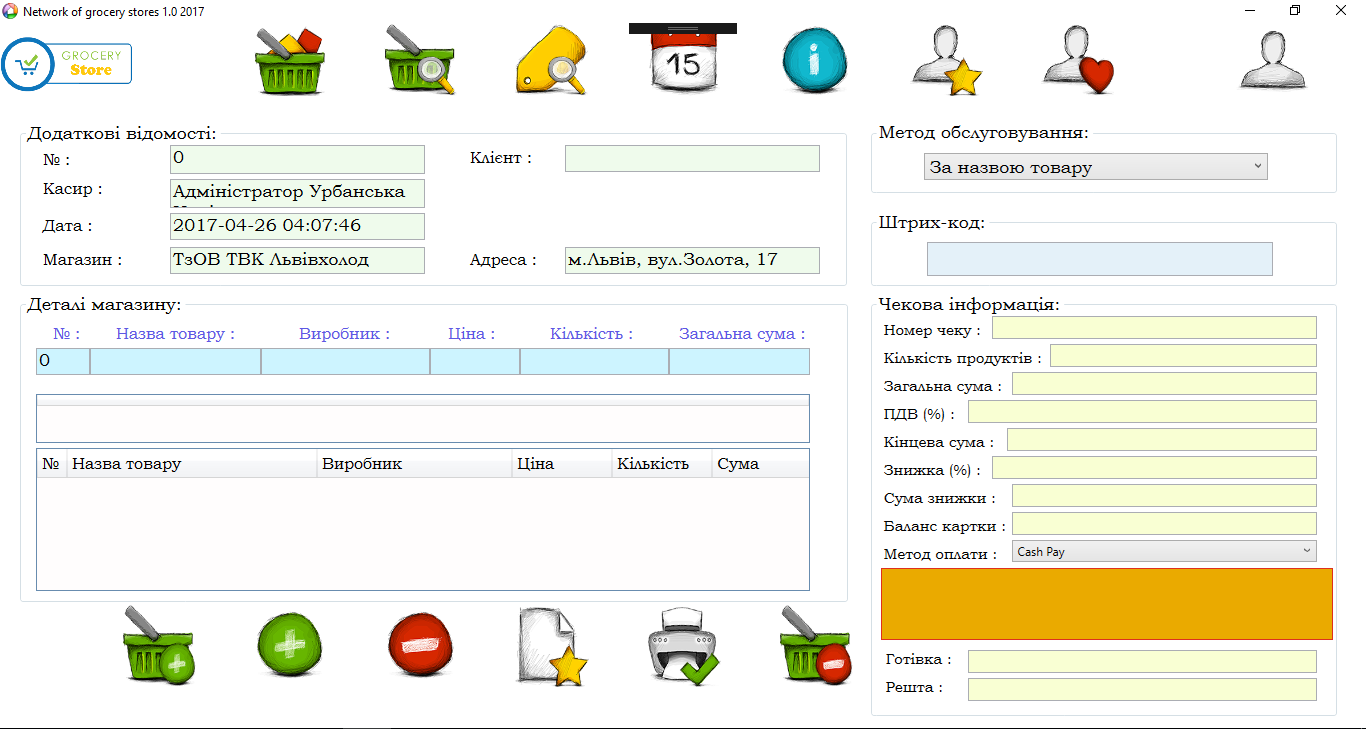


Рис.3.4. Форма користувача «Адміністратор».

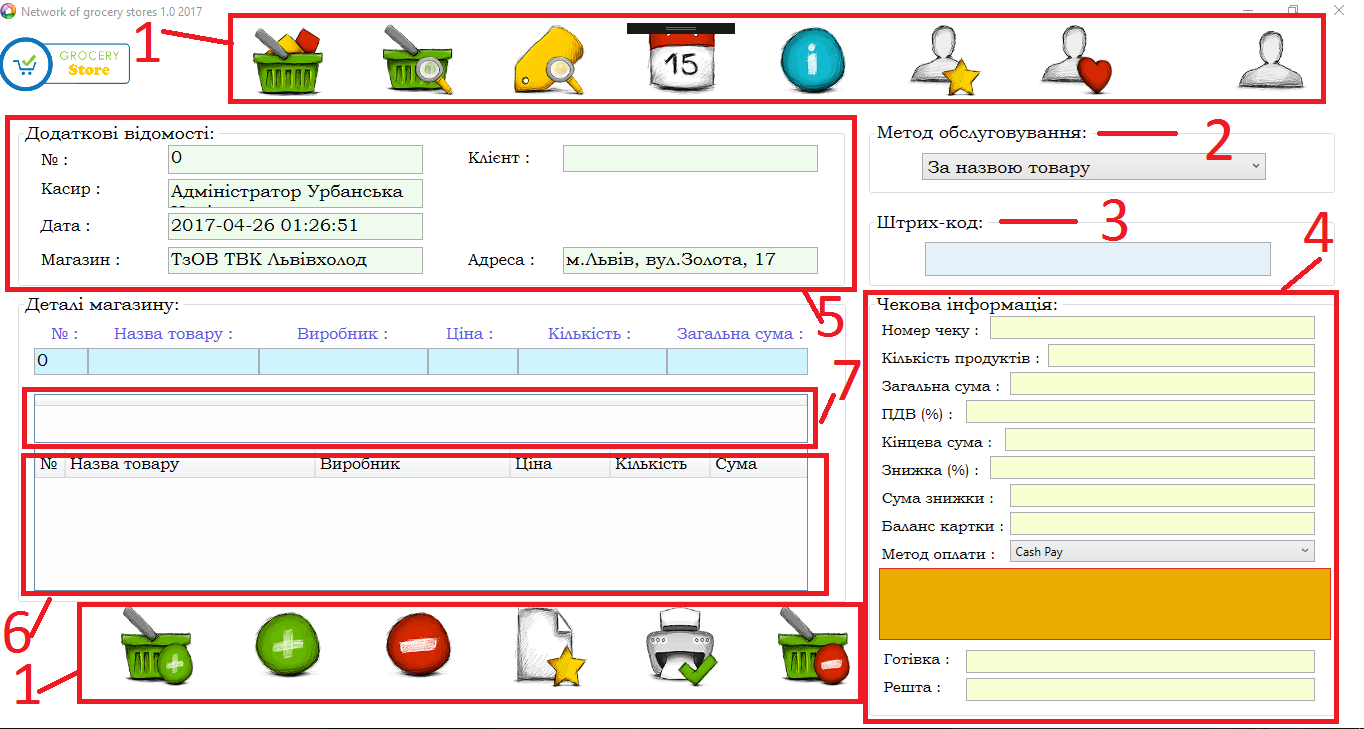


Рис.3.5. Призначення панелей.

1- панель основних функцій програми;

2- метод обслуговування (за назвою товару або штрих-кодом);

3- поле для вводу (виводу) штрих-коду;

4- відображення всієї чекової інформації;

5- загальні відомості;

6-таблиця замовлень клієнта (відображення купівлі товарів клієнта);

7-таблиця відомості про товар та його наявність.



Рис.3.6. Детальна інформація панелі основних функцій програми.

1- перехід до вікна пошуку товарів по групах (Рис.3.8);

2-пошук товарів (за назвою, виробником, ціною, штрих-кодом) (Рис.3.9);

3- пошук акцій (акція місяця, акція тижня, акція дня, майбутні акції)

(Рис.3.10);

4- робочий графік;

5-інформація про програму та інструкція;

6- інформація про працівників магазину (Рис.3.11);

7-інформація про клієнтів магазину (Рис.3.12);

8-інформація особистого кабінету;



Рис.3.7. Основні функції.

1-додати замовлення;

2-додати товар;

3-відняти товар;

4-зберегти інформацію в БД;

5-друк чеку;

6-видалити замовлення;

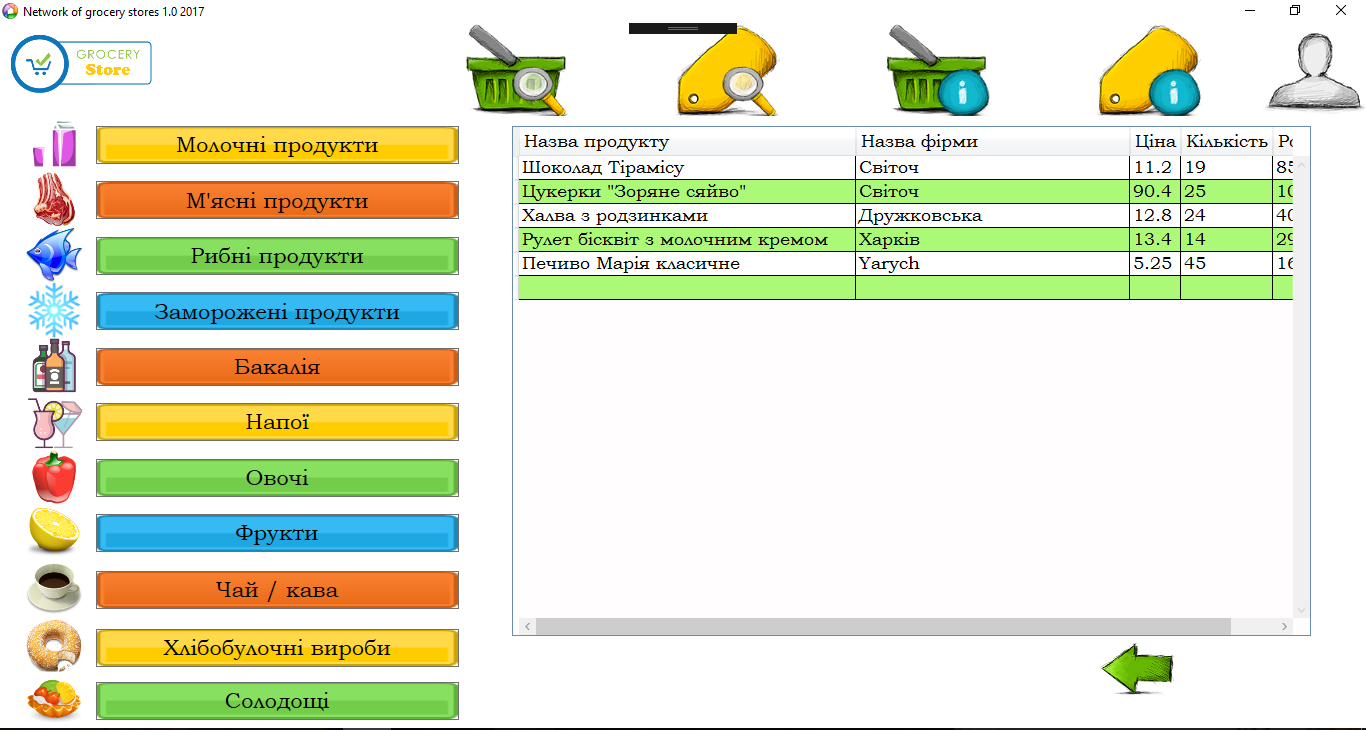


Рис.3.8. Вікно пошуку товарів по групах.

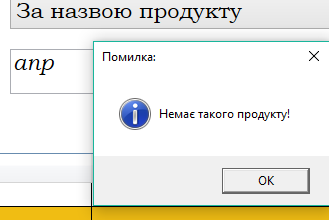
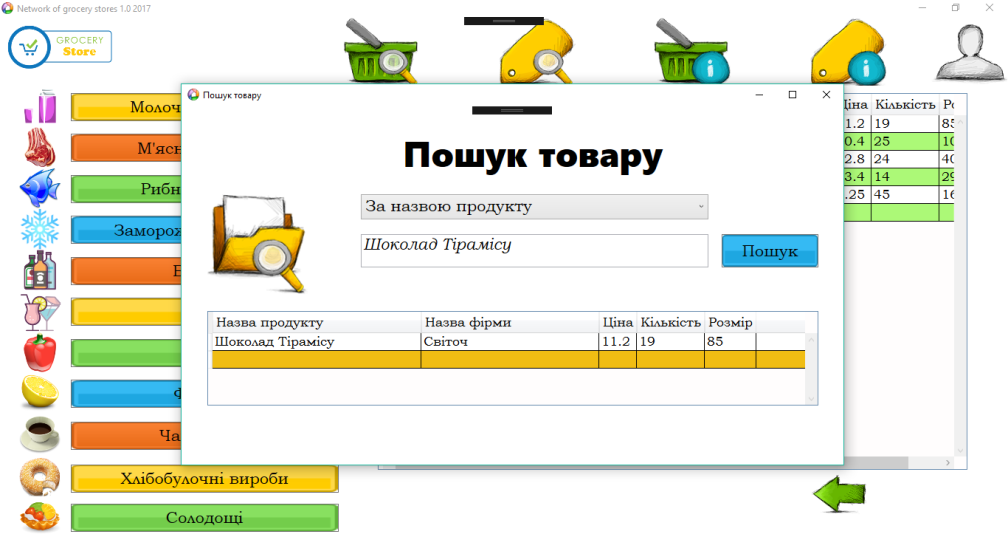


Рис.3.9. Пошук товару за назвою (вивід про помилку, в разі відсутності).

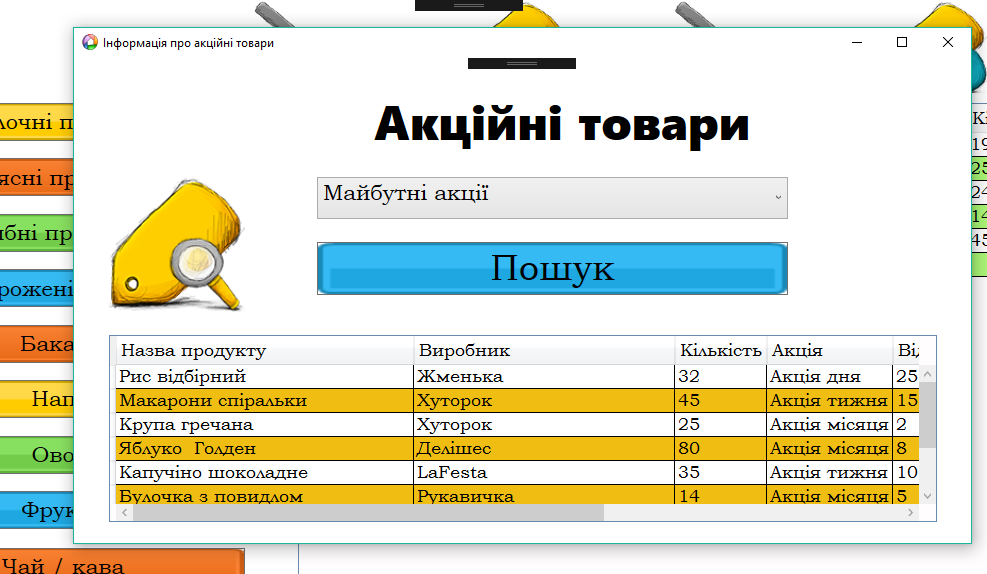


Рис.3.10. Вікно пошуку товару (майбутні акції).

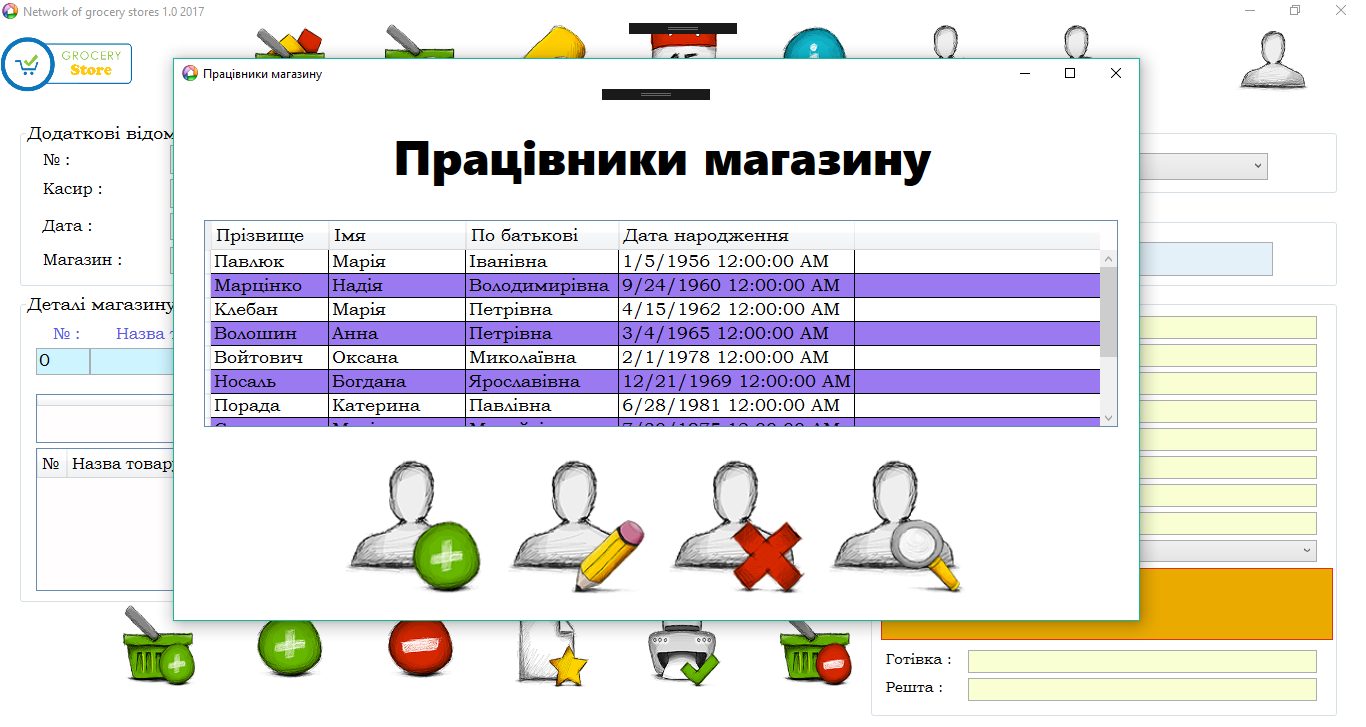


Рис.3.11. Вікно “Працівники магазину”.

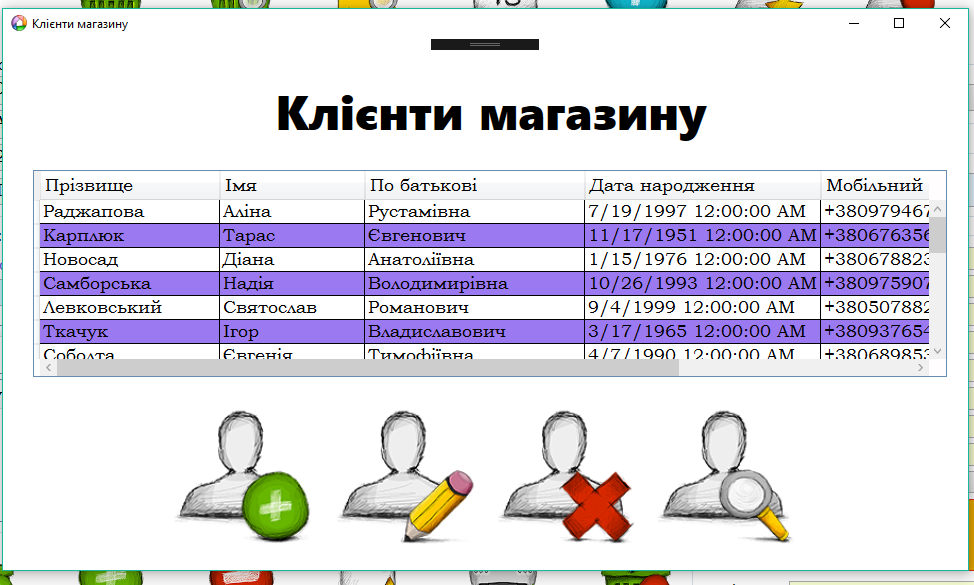


Рис.3.12. Вікно “Клієнти магазину”.

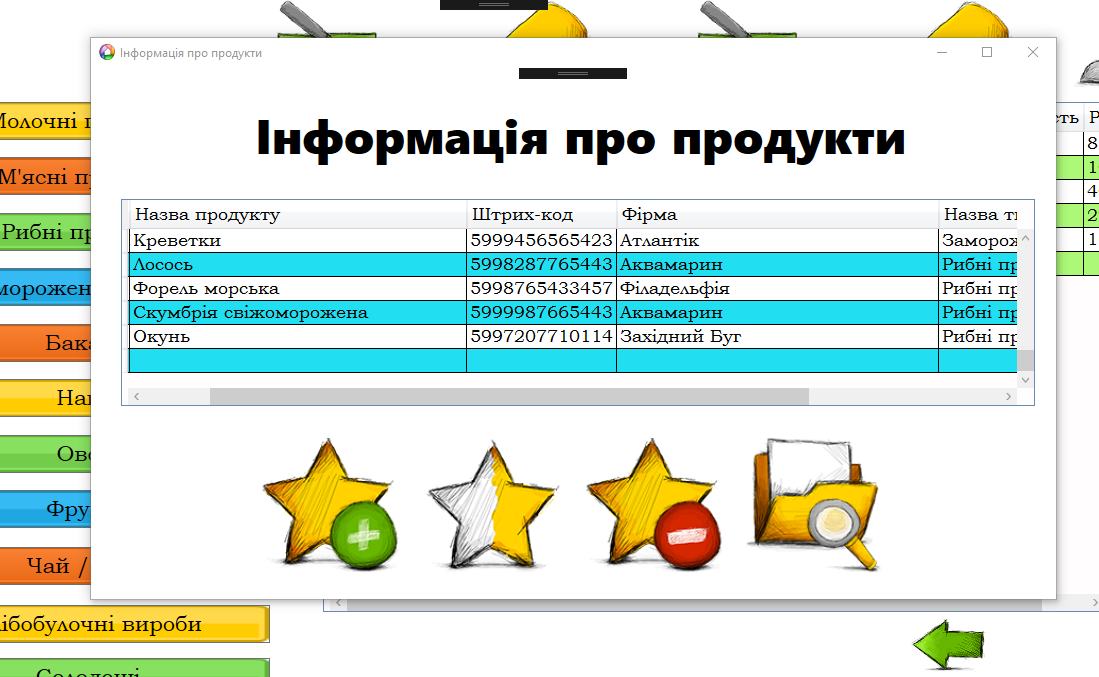


Рис.3.13. Вікно інформації про продукт.

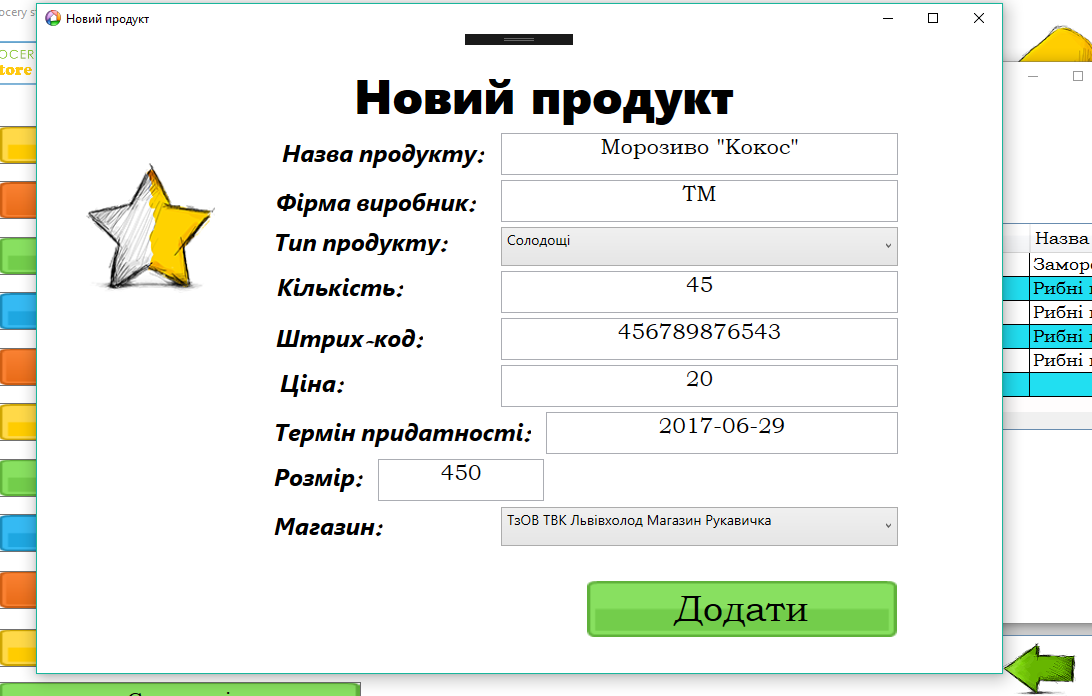


Рис.3.14. Вікно додавання нового продукту.

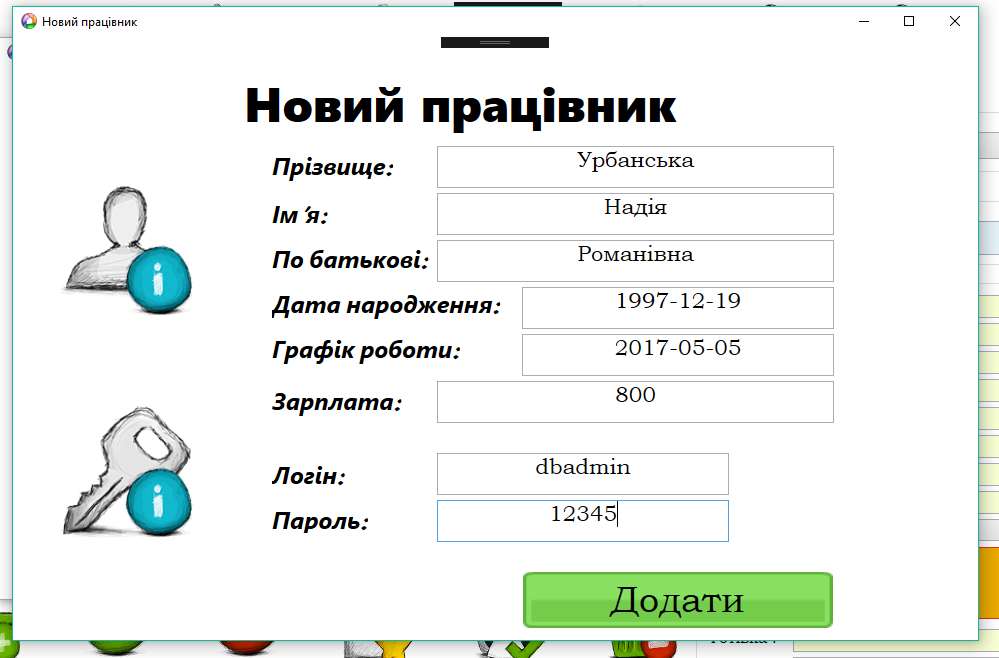


Рис.3.15. Вікно додавання нового працівника.

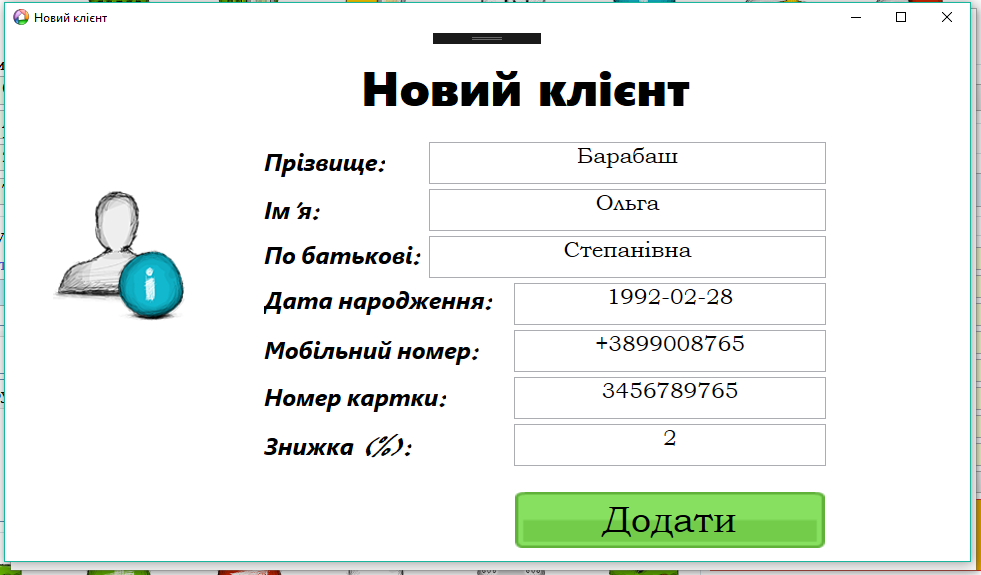


Рис.3.16. Вікно додавання нового клієнта.

При зміні користувача на «Касир», змінюються права доступу до бази даних. Касиру наявні функції: пошуку товару за групами, за назвою, за штрих-кодом, за ціною, за виробником; пошуку знижок за назвами (акція місяця, акція тижня, акція дня, майбутні акції); можна переглянути свій робочий графік та особисті дані профілю. Оскільки, касир здійснює основні операції по купівлі-продажі то, йому надається можливість лише пошуку товарів та знижок, оформлення операції купівлі-продажу.

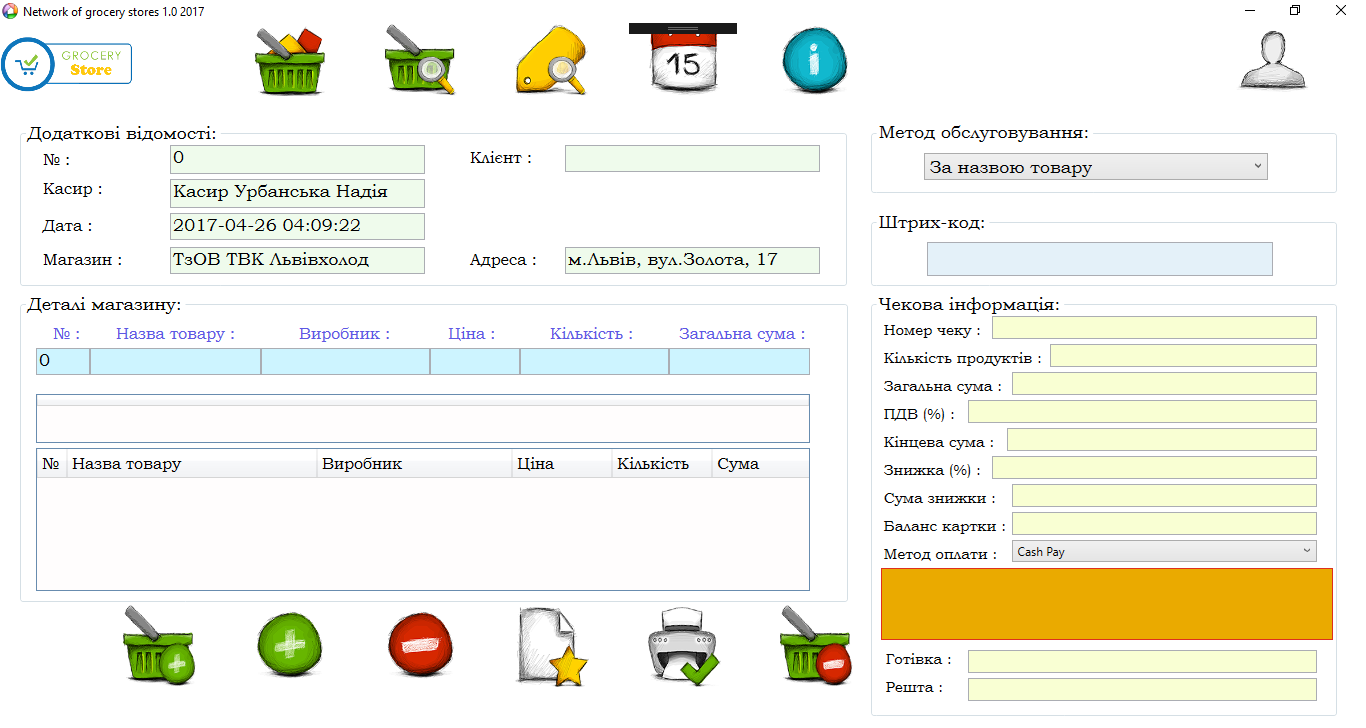


Рис.3.17. Форма користувача «Касир».

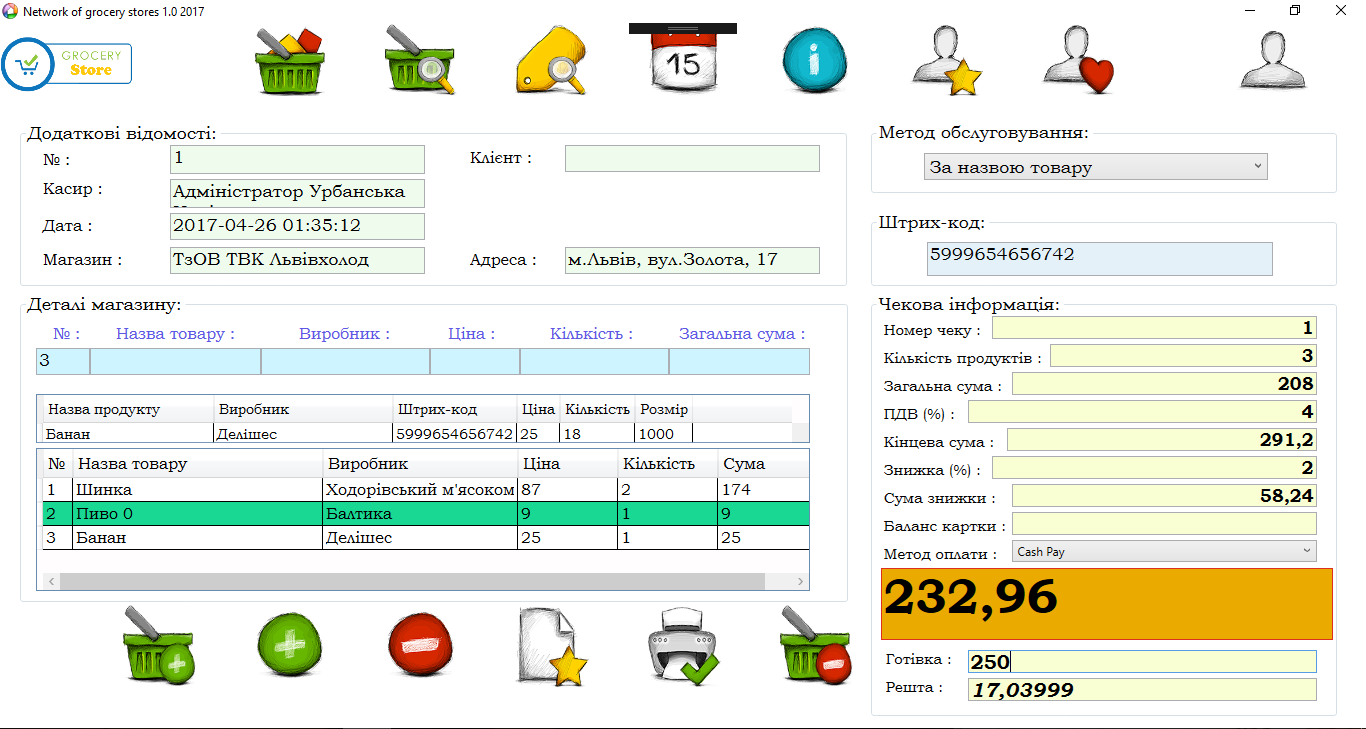


Рис.3.18. Робочий процес програми. Вигляд операції купівлі-продажу.

**ВИСНОВКИ**

На даній курсовій роботі мною була реалізована база даних керування купівлі-продажу товарів в продуктовому магазині, а також програмний додаток для взаємодії з цією базою.

В ході роботи над проектом було використано такі програмні продукти як Microsoft Visual Studio 2017 Сommunity Edition, CA Erwin Data Modeler R7, Microsoft SQL Server Management Studio 2014. Visual Studio являє собою середовище розробки даного проекту. Erwin Data Modeler використовувався для проектування концептуальної схеми бази даних. Сама ж база даних була реалізована за допомогою MS SQL Server. В якості технології зв’язку між додатком і базою даних використано технологію ADO.NET. Користувацький інтерфейс побудований на основі WPF.

Під час виконання роботи, я оволоділа навичками концептуального проектування баз даних, а також організації взаємодії бази з додатком за допомогою технології ADO.NET, навчилася проектувати типові запити та транзакції.

Основним недоліком розробленого додатку є неможливість його взаємодії з іншим типом бази даних без внесення необхідних поправок.

В програмі реалізовано всі функціональні вимоги, описані в специфікації вимог до ПЗ.

**СПИСОКЛІТЕРАТУРИ**

1. SQL Server Books Online. Web page: https://technet.microsoft.com/en-us/library/ms130214(v=sql.105).aspx
2. W3Schools Online Web Tutorials. Web page: <http://www.w3schools.com/>
3. Stack Overflow. Web page: stackoverflow.com
4. Beginning SQL Server for Developers:From Novice to Professional 2012 / Robin Dewson. –Appress,2012. – 823 ст.
5. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# /Дж.Рихтер. – Питер, 2016. – 896 ст. 6. Wikipedia. Webpage: <https://uk.wikipedia.org/>
6. <https://www.codeproject.com/Articles/30905/WPF-DataGrid-Practical-Examples>
7. <https://www.iconfinder.com/iconsets/Hand_Drawn_Web_Icon_Set>
8. Приклад ПЗ: <https://www.youtube.com/watch?v=Cawg9kWuCbM>
9. <http://stackoverflow.com/questions/13573380/inserting-values-into-a-sql-server-database-using-ado-net-via-c-sharp>
10. https://professorweb.ru/my/entity-framework/6/level3/3\_1.php

**Додаток А. Скрипт створення БД:**

CREATE TABLE Cheque

(

ID\_cheque INT NOT NULL ,

ID\_order INT NOT NULL ,

ID\_reserve INT NOT NULL ,

ID\_customers INT NOT NULL ,

ID\_sale INT NOT NULL ,

Amount INT NULL ,

OrderData DATE NULL

);

CREATE TABLE Customers

(

ID\_customers INT NOT NULL ,

Surname VARCHAR(50) NULL ,

Name VARCHAR(50) NULL ,

MiddleName VARCHAR(50) NULL ,

DateOfBirth DATE NULL ,

PhoneNumder VARCHAR(20) NULL ,

NumberCard VARCHAR(16) NULL ,

Discount INT NULL

);

CREATE TABLE Workers

(

ID\_workers INT NOT NULL ,

Surname VARCHAR(50) NULL ,

Name VARCHAR(50) NULL ,

MiddleName VARCHAR(50) NULL ,

DateOfBirth DATE NULL

);

CREATE TABLE Orders

(

ID\_order INT NOT NULL ,

ID\_workers INT NOT NULL ,

ID\_sale INT NOT NULL ,

ID\_reserve INT NOT NULL ,

TotalSum FLOAT NULL

);

CREATE TABLE Products

(

ID\_product INT NOT NULL ,

ID\_subtypes INT NOT NULL ,

Name VARCHAR(80) NULL ,

Firm VARCHAR(50) NULL

);

CREATE TABLE Reserve

(

ID\_reserve INT NOT NULL ,

ID\_store INT NOT NULL ,

ID\_product INT NOT NULL ,

ID\_sal\_res\_inv int NOT NULL ,

ExpirationDate DATE NULL ,

Quantity INT NULL ,

Price FLOAT NULL ,

BarCode VARCHAR(13) NULL ,

ProductSize FLOAT NULL

);

CREATE TABLE Sale

(

ID\_sale INT NOT NULL ,

ID\_products INT NOT NULL ,

Percents INT NULL ,

Name\_sale VARCHAR(50) NULL ,

DateOfStart DATE NULL ,

DateOfFinish DATE NULL

);

CREATE TABLE Timetable

(

ID\_timetable INT NOT NULL ,

ID\_workers INT NOT NULL ,

ID\_store INT NOT NULL ,

WorkDate DATE NULL ,

Salary FLOAT NULL

);

CREATE TABLE Subtypes

(

ID\_subtypes INT NOT NULL ,

Name VARCHAR(35) NULL

);

CREATE TABLE Store

(

ID\_store INT NOT NULL ,

Address VARCHAR(80) NULL ,

IfStore INT NULL

);

CREATE TABLE Invoice\_Products

(

ID\_invoice\_products int NOT NULL ,

ID\_invoice int NOT NULL ,

ID\_product int NOT NULL ,

Quantity int NULL

);

CREATE TABLE InvoiceForGoods

(

ID\_invoice INT NOT NULL ,

ID\_provisor INT NOT NULL ,

ID\_workers INT NOT NULL ,

OrderDate DATE NULL

);

CREATE TABLE Provisor

(

ID\_provisor int NOT NULL ,

LegalAddress VARCHAR(50) NULL ,

CompanyName VARCHAR(50) NULL ,

PhoneNumber VARCHAR(50) NULL ,

CheckingAccount VARCHAR(50) NULL

);

CREATE TABLE SalesInvoice

(

ID\_SalersInvoice int NOT NULL ,

ID\_invoice int NOT NULL ,

ID\_workers int NOT NULL ,

DeliveryDate DATE NULL

);

CREATE TABLE Sales\_Reserve\_Invoice

(

ID\_sal\_res\_inv int NOT NULL ,

ID\_invoice\_products int NOT NULL ,

Quantity int NULL ,

PriceUnit FLOAT NULL ,

ID\_SalersInvoice int NOT NULL

);

CREATE UNIQUE INDEX XPKCheck ON Cheque

(ID\_cheque ASC);

ALTER TABLE Cheque

ADD PRIMARY KEY (ID\_cheque);

CREATE INDEX XIF2Check ON Cheque

(ID\_reserve ASC);

CREATE INDEX XIF3Check ON Cheque

(ID\_order ASC);

CREATE INDEX XIF4Check ON Cheque

(ID\_customers ASC);

CREATE INDEX XIF5Check ON Cheque

(ID\_sale ASC);

CREATE UNIQUE INDEX XPKCustomers ON Customers

(ID\_customers ASC);

ALTER TABLE Customers

ADD PRIMARY KEY (ID\_customers);

CREATE UNIQUE INDEX XPKInvoice\_Product ON Invoice\_Products

(ID\_invoice\_products ASC);

ALTER TABLE Invoice\_Products

ADD PRIMARY KEY (ID\_invoice\_products);

CREATE INDEX XIF1Invoice\_Product ON Invoice\_Products

(ID\_product ASC);

CREATE INDEX XIF2Invoice\_Product ON Invoice\_Products

(ID\_invoice ASC);

CREATE UNIQUE INDEX XPKInvoiceForGoods ON InvoiceForGoods

(ID\_invoice ASC);

ALTER TABLE InvoiceForGoods

ADD PRIMARY KEY (ID\_invoice);

CREATE INDEX XIF1InvoiceForGoods ON InvoiceForGoods

(ID\_provisor ASC);

CREATE INDEX XIF2InvoiceForGoods ON InvoiceForGoods

(ID\_workers ASC);

CREATE UNIQUE INDEX XPKOrders ON Orders

(ID\_order ASC);

ALTER TABLE Orders

ADD PRIMARY KEY (ID\_order);

CREATE INDEX XIF4Orders ON Orders

(ID\_sale ASC);

CREATE INDEX XIF6Orders ON Orders

(ID\_workers ASC);

CREATE INDEX XIF7Orders ON Orders

(ID\_reserve ASC);

CREATE UNIQUE INDEX XPKProducts ON Products

(ID\_product ASC);

ALTER TABLE Products

ADD PRIMARY KEY (ID\_product);

CREATE INDEX XIF2Products ON Products

(ID\_subtypes ASC);

CREATE UNIQUE INDEX XPKProvisor ON Provisor

(ID\_provisor ASC);

ALTER TABLE Provisor

ADD PRIMARY KEY (ID\_provisor);

CREATE UNIQUE INDEX XPKReserve ON Reserve

(ID\_reserve ASC);

ALTER TABLE Reserve

ADD PRIMARY KEY (ID\_reserve);

CREATE INDEX XIF1Reserve ON Reserve

(ID\_store ASC);

CREATE INDEX XIF2Reserve ON Reserve

(ID\_product ASC);

CREATE INDEX XIF3Reserve ON Reserve

(ID\_sal\_res\_inv ASC);

CREATE UNIQUE INDEX XPKSale ON Sale

(ID\_sale ASC)

ALTER TABLE Sale

ADD PRIMARY KEY (ID\_sale);

CREATE UNIQUE INDEX XPKSales\_Reserve\_Invoice ON Sales\_Reserve\_Invoice

(ID\_sal\_res\_inv ASC);

ALTER TABLE Sales\_Reserve\_Invoice

ADD PRIMARY KEY (ID\_sal\_res\_inv);

CREATE INDEX XIF2Sales\_Reserve\_Invoice ON Sales\_Reserve\_Invoice

(ID\_invoice\_products ASC);

CREATE INDEX XIF3Sales\_Reserve\_Invoice ON Sales\_Reserve\_Invoice

(ID\_SalersInvoice ASC);

CREATE UNIQUE INDEX XPKSalesInvoice ON SalesInvoice

(ID\_SalersInvoice ASC);

ALTER TABLE SalesInvoice

ADD PRIMARY KEY (ID\_SalersInvoice);

CREATE INDEX XIF1SalesInvoice ON SalesInvoice

(ID\_workers ASC);

CREATE INDEX XIF2SalesInvoice ON SalesInvoice

(ID\_invoice ASC);

CREATE UNIQUE INDEX XPKStore ON Store

(ID\_store ASC);

ALTER TABLE Store

ADD PRIMARY KEY (ID\_store);

CREATE UNIQUE INDEX XPKSubtypes ON Subtypes

(ID\_subtypes ASC);

ALTER TABLE Subtypes

ADD PRIMARY KEY (ID\_subtypes);

CREATE UNIQUE INDEX XPKTimetable ON Timetable

(ID\_timetable ASC);

ALTER TABLE Timetable

ADD PRIMARY KEY (ID\_timetable);

CREATE INDEX XIF1Timetable ON Timetable

(ID\_workers ASC);

CREATE INDEX XIF2Timetable ON Timetable

(ID\_store ASC);

CREATE UNIQUE INDEX XPKWorkers ON Workers

(ID\_workers ASC);

ALTER TABLE Workers

ADD PRIMARY KEY (ID\_workers);

ALTER TABLE Cheque

ADD (FOREIGN KEY (ID\_reserve) REFERENCES Reserve (ID\_reserve));

ALTER TABLE Cheque

ADD (FOREIGN KEY (ID\_order) REFERENCES Orders (ID\_order));

ALTER TABLE Cheque

ADD (FOREIGN KEY (ID\_customers) REFERENCES Customers (ID\_customers));

ALTER TABLE Cheque

ADD (FOREIGN KEY (ID\_sale) REFERENCES Sale (ID\_sale));

ALTER TABLE Invoice\_Products

ADD (FOREIGN KEY (ID\_product) REFERENCES Products (ID\_product));

ALTER TABLE Invoice\_Products

ADD (FOREIGN KEY (ID\_invoice) REFERENCES InvoiceForGoods (ID\_invoice));

ALTER TABLE InvoiceForGoods

ADD (FOREIGN KEY (ID\_provisor) REFERENCES Provisor (ID\_provisor));

ALTER TABLE InvoiceForGoods

ADD (FOREIGN KEY (ID\_workers) REFERENCES Workers (ID\_workers));

ALTER TABLE Orders

ADD (FOREIGN KEY (ID\_sale) REFERENCES Sale (ID\_sale));

ALTER TABLE Orders

ADD (FOREIGN KEY (ID\_workers) REFERENCES Workers (ID\_workers));

ALTER TABLE Orders

ADD (FOREIGN KEY (ID\_reserve) REFERENCES Reserve (ID\_reserve));

ALTER TABLE Products

ADD (FOREIGN KEY (ID\_subtypes) REFERENCES Subtypes (ID\_subtypes));

ALTER TABLE Reserve

ADD (FOREIGN KEY (ID\_store) REFERENCES Store (ID\_store));

ALTER TABLE Reserve

ADD (FOREIGN KEY (ID\_product) REFERENCES Products (ID\_product));

ALTER TABLE Reserve

ADD (FOREIGN KEY (ID\_sal\_res\_inv) REFERENCES Sales\_Reserve\_Invoice (ID\_sal\_res\_inv));

ALTER TABLE Sales\_Reserve\_Invoice

ADD (FOREIGN KEY (ID\_invoice\_products) REFERENCES Invoice\_Products (ID\_invoice\_products));

ALTER TABLE Sales\_Reserve\_Invoice

ADD (FOREIGN KEY (ID\_SalersInvoice) REFERENCES SalesInvoice (ID\_SalersInvoice));

ALTER TABLE SalesInvoice

ADD (FOREIGN KEY (ID\_workers) REFERENCES Workers (ID\_workers));

ALTER TABLE SalesInvoice

ADD (FOREIGN KEY (ID\_invoice) REFERENCES InvoiceForGoods (ID\_invoice));

ALTER TABLE Timetable

ADD (FOREIGN KEY (ID\_workers) REFERENCES Workers (ID\_workers));

ALTER TABLE Timetable

ADD (FOREIGN KEY (ID\_store) REFERENCES Store (ID\_store));

CREATE VIEW vTimetableWorkers (

SELECT dbo.Workers.Surname, dbo.Workers.Name, dbo.Workers.MiddleName, dbo.Timetable.WorkDate, dbo.Timetable.Salary

FROM dbo.Timetable INNER JOIN

dbo.Workers ON dbo.Timetable.ID\_workers = dbo.Workers.ID\_workers

)

CREATE VIEW vInfoProducts(

SELECT dbo.Products.Name, dbo.Reserve.BarCode, dbo.Products.Firm, dbo.Subtypes.Name AS Type, dbo.Reserve.Price

FROM dbo.Products INNER JOIN

dbo.Reserve ON dbo.Products.ID\_product = dbo.Reserve.ID\_product INNER JOIN

dbo.Subtypes ON dbo.Products.ID\_subtypes = dbo.Subtypes.ID\_subtypes

)

CREATE VIEW vCheque(

SELECT dbo.Products.Name, dbo.Products.Firm, dbo.Reserve.BarCode, dbo.Reserve.Price, dbo.Cheque.Amount, dbo.Orders.TotalSum, dbo.Sale.Percents, dbo.Customers.Surname,

dbo.Customers.Name AS NameCustomer, dbo.Customers.MiddleName, dbo.Customers.Discount, dbo.Workers.Surname AS SurnameWorkers, dbo.Workers.Name AS NameWorkers, dbo.Cheque.OrderData,

dbo.Store.Address

FROM dbo.Products INNER JOIN

dbo.Reserve ON dbo.Products.ID\_product = dbo.Reserve.ID\_product INNER JOIN

dbo.Orders ON dbo.Reserve.ID\_reserve = dbo.Orders.ID\_reserve INNER JOIN

dbo.Cheque ON dbo.Orders.ID\_order = dbo.Cheque.ID\_order INNER JOIN

dbo.Sale ON dbo.Cheque.ID\_sale = dbo.Sale.ID\_sale INNER JOIN

dbo.Customers ON dbo.Cheque.ID\_customers = dbo.Customers.ID\_customers INNER JOIN

dbo.Workers ON dbo.Orders.ID\_workers = dbo.Workers.ID\_workers INNER JOIN

dbo.Store ON dbo.Reserve.ID\_store = dbo.Store.ID\_store

)