Національний університет «Львівська політехніка»

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Бази даних»**

на тему:

«База даних для обліку діяльності мережі магазинів сантехніки»

Виконав: студент

спеціальності 121

«Інженерія програмного забезпечення»

групи П3-32

Ханас Л.І.

Керівник:

доцент кафедри програмного забезпечення, к.т.н., доцент Павич Н.Я.

Оцінка:

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

Кількість балів\_\_\_\_\_\_\_Оцінка ECTS\_\_\_\_

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

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_ Цимбалюк Т.М.

(підпис)

**ЗМІСТ**

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ОБЛІКУ ДІЯЛЬНОСТІ МЕРЕЖІ МАГАЗИНІВ САНТЕХНІКИ…………………………………………………………………………………...3

* 1. Опис предметної області ………………….. ………………………………………3
  2. Постановка завдання………………………………………………………………..5
  3. Специфікація вимог до системи мережі магазинів сантехніки……………………………………………...……………………………..6

1.3.1. Вступ…………………………………………………………………………...6

1.3.2. Загальний опис………………………………………………………………...7

РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ………………………………………………..12

2.1. Концептуальне моделювання предметної області ………………………………12

2.2. Логічне проектування бази даних………………………………………………...20

2.3. Проектування типових запитів і транзакцій……………………………………..22

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ РОБОТИ З БАЗОЮ ДАНИХ…..23

3.1. Реалізація доступу до даних………………………………………………………23

3.2. Реалізація функціональних характеристик системи…………………………….23

3.2.1. Авторизація…………………………………………………………………...23

3.2.2. Вибірка даних…………………………………………………………………26

3.2.3. Додавання даних……………………………………………………………...28

3.2.4. Видалення даних……………………………………………………………...28

3.2.5. Редагування даних……………………………………………………………28

ВИСНОВКИ…………………………………………………………………………………….30

СПИСОК ЛІТЕРАТУРИ…………………………………………………………………….....31

Додаток А. Скрипт створення БД 32

**РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ОБЛІКУ ДІЯЛЬНОСТІ МЕРЕЖІ МАГАЗИНІВ САНТЕХНІКИ**

* 1. **Опис предметної області “Мережа** **магазинів сантехніки”**

Магазин сантехнічних товарів призначений для продажу товарів покупцям. База даних сантехнічного магазину - багатофункціональна і легка у використанні система для управління магазином. Основна мета створення бази даних для мережі магазинів сантехніки:

* облік працівників;
* облік клієнтів;
* облік продажу сантехніки.

У націнку товару входять : комунальні послуги магазину, орендна плата приміщення, в якому знаходиться магазин, заробітна плата адміністративного та обслуговуючого персоналу, дохід підприємства, сплата податків, витрати на розхідні матеріали.

Згідно з витратами, описаними вище, націнка на товар в середньому становить 30%.

Ревізію магазину потрібно проводити раз в місяць, в перший понеділок місяця, для того, щоб порахувати залишок товарів. Ревізія проводиться для того, щоб знати, які товари варто замовити, підрахувати скільки товару було продано з часу минулої ревізії та їхню суму, перевірити чи збігається це зі залишком товару з часу минулої ревізії.

Відповідно до ревізії, формується звітність. У звітність входять такі відомості : надходження товарів, списання товарів, оприбуткування товарів.

Коли термін зберігання товару закінчується, їх забирають з полиць магазину та віддають на утилізацію. Під час ревізії товари, які віддали на утилізацію, не враховуються.

В магазині діють акції для придбання сантехніки за зниженою ціною. Коли до кінця терміну зберігання товару залишилось три дні, цей товар продається зі знижкою. Знижки формуються згідно з кількістю залишку цього товару. Під час ревізії товар, який продаються чи продавались зі знижкою, обраховуються окремо.

Клієнтам надаються бонусні картки для накопичення процентів під час кожної покупки залежно віл загальної суми, потім покупець може використати карту, для отримання знижки. На касі, при купівлі товару, покупець отримує чек з переліком товарів, яких він купив та з їх ціною.

Розробка і реалізація бази даних для магазину сантехніки є досить актуальною, адже це зменшить кількість роботи з паперами, дозволить витрачати менше часу для пошуку необхідної інформації та пришвидшить виконання завдань пов'язаних з обліком товарів, працівників та покупців.

* 1. **Постановка завдання**

Загальні вимоги до БД і додатку:

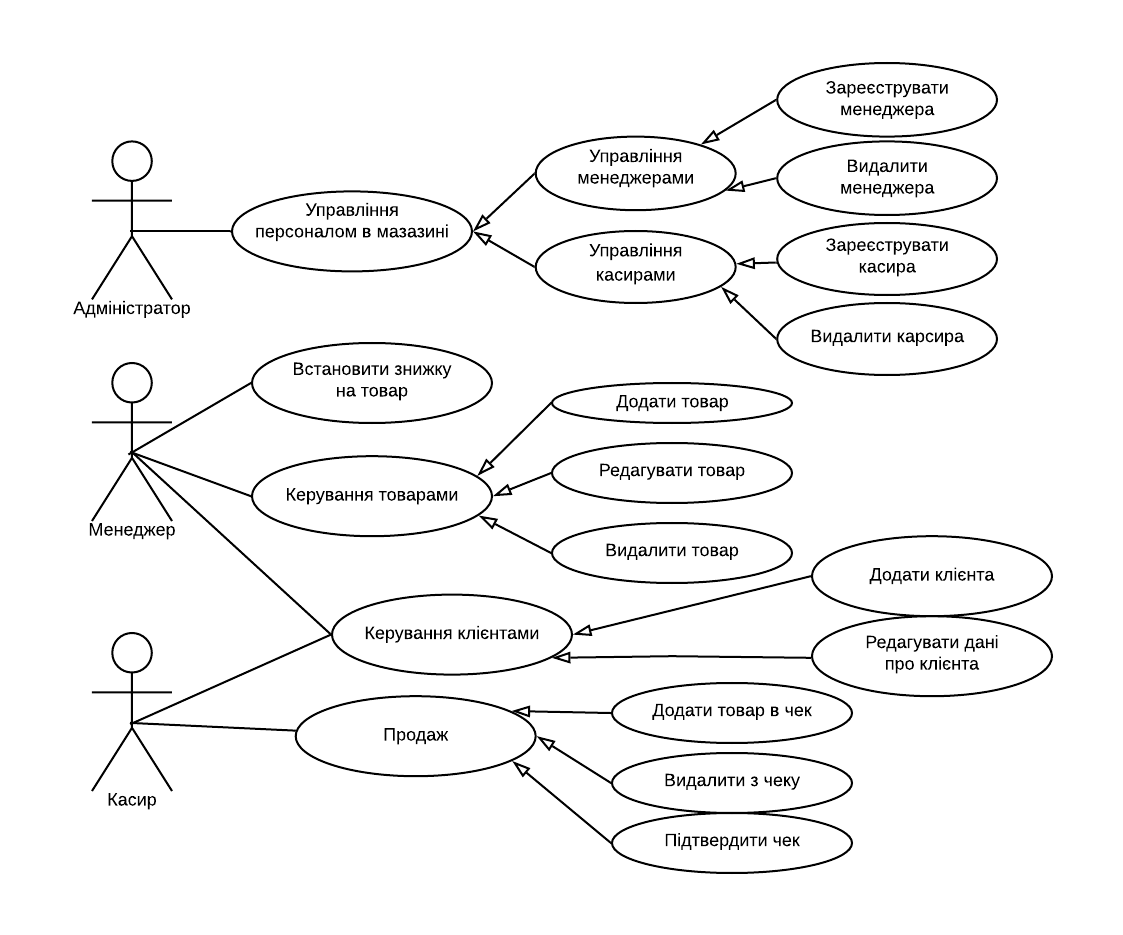
• ПЗ повинне мати механізм реєстрації працівника в системі

• ПЗ повинне надавати різні робочі інтерфейси в залежності від ролі користувача

• ПЗ має надавати касирам механізм для додавання товарів у кошик і подальшого їх опрацьовування

• БД має зберігати ключову інформацію про касирів та клієнтів: дані про продані товари, кількість товарів, працівників, клієнтів, тощо

Сценарії взаємодії між користувачами системи та випадками використання зображено на діаграмі прецедентів:



Мал.1.1 Діаграма прецедентів

* 1. **Специфікація вимог до системи обліку “Мережа магазинів сантехніки ”**
     1. **Вступ**

Призначенням даного продукту є автоматизація роботи магазину сантехніки, можливість контролю продажу для працівників. Забезпечення всієї необхідної інформації про товари в магазині; про магазини, їх розташування та працівників.

* + 1. **Загальний опис**

**1.3.2.1. Перспективи продукту**

Даний продукт являється самодостатньою системою, покликаний систематизувати роботу магазину, що займаються продажом сантехніки. Ідея виникла на основі систем управління великими мережами магазинів, але на відміну від таких вона повинна бути простою і зрозумілою навіть недосвідченим користувачам.

**1.3.2.2. Характеристика продукту**

Продукт є повністю незалежним і повністю самодостатнім програмним забезпеченням. Основні обмеження на продукт накладає реалізація бази даних та зв'язків у ній. Реалізацією роботи програми під такими обмеженнями повинен займатися системний адміністратор.

Продукт повинен виконувати наступні функції:

• видавати інформацію про наявні товари на складах, їх ціну та кількість;

• зберігати загальний архів продажів.

**1.3.2.3. Класи користувачів та їх характеристики**

1. Адміністратор системи – користувач, що може додавати/видаляти менеджерів і касирів.
2. Менеджер – користувач, який має змогу порахувати витрати/прибуток, додати товар.
3. Касир – користувач, який приймає замовлення, опрацьовує та видає чеки.

**1.3.2.4. Середовище функціонування**

Для функціонування системи необхідна платформа повинна відповідати наступним вимогам:

|  |  |
| --- | --- |
| Операційна система | ОС windows XP і вище |
| Процесор | Intel Pentium 7 і вище |
| Оперативна пам’ять | 512 МБ |
| Обсяг вільної пам’яті | 315 МБ |

Таблиця 1.Мінімальні системі вимоги для коректної роботи продукту

**1.3.2.5. Обмеження проектування та реалізації**

Продукт повинен бути реалізований впродовж 6 місяців після старту розробки. Система не повинна вимагати наявності додаткових програмних компонентів крім Microsoft SQL Server.

**1.3.2.6. Документація користувача**

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

* + 1. **Характеристики системи**

**1.3.3.1. Авторизація користувачів**

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

**Послідовність дія/відгук**

**База даних**

**Форма**

**входу**

**Користувач**

**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 товарами**

Пріоритет *високий*. Додавання, редагування даних про товари.

**Послідовність дія/відгук**

**База даних**

**Форма редагування даних**

**Користувач**

**Функціональні вимоги**

REQ-1: Додавання товарів. Передбачити можливість неправильного введення даних.

REQ-2: Редагування даних про товари.

**1.3.3.3. Керування клієнтами**

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

**Послідовність дія/відгук**

**База даних**

**Форма редагування даних**

**Користувач**

**Функціональні вимоги**

REQ-1: Реєстрація нового клієнта. Передбачити можливість неправильного введення даних.

REQ-2: Редагування даних про клієнта.

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

**1.3.3.4. Керування працівниками**

Пріоритет *високий*. Редагування даних про працівника.

**Послідовність дія/відгук**

**База даних**

**Форма редагування даних**

**Користувач**

**Функціональні вимоги**

REQ-1: Реєстрація нового працівникачи. Передбачити можливість неправильного введення даних.

REQ-2: Редагування даних про працівника.

* + 1. **Вимоги зовнішніх інтерфейсів**

**1.3.4.1. Апаратні інтерфейси**

Комп'ютер на якому буде виконуватись дана програма повинен мати наступні апаратні характеристики:

* 32-х бітний або 64-х бітний процесор з тактовою частотою 1 ГГц, або більше;
* 512 МБ ОЗП та 315 МБ ЗСД

**1.3.4.2. Програмні інтерфейси**

Для повноцінної роботи необхідно встановити наступні програмні компоненти:

* ОС Windows XP / Windows 7 / Windows 8 / Windows 10

**1.3.4.3. Комунікаційні інтерфейси**

Для коректної роботи ПК повинен мати порт RJ-45 або ж модуль Wi-Fi, а локальна мережа підтримувати стандарт Fast Ethernet.

* + 1. **Нефункціональні вимоги до проекту**

**1.3.5.1. Вимоги продуктивності**

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, що запобігти великим втратам у разі непередбачуваних ситуацій (перебої електроенергії тощо).

**1.3.5.3. Вимоги безпеки**

Кожен користувач має певний тип доступу прив’язаний до облікового запису, доступ здійснюється за допомогою логіну та паролю. Пароль повинен бути прихований під час введення.

**1.3.5.4. Атрибути якості програмного продукту**

Навчання роботі із системою повинно займати мінімальну кількість часу, система повинна бути стабільною та ефективно використовувати апаратні ресурси.

**РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ**

## 2.1. Концептуальне моделювання предметної області

### 2.1.1. Опис сутностей

Задля вирішення задачі керування мережею магазинів сантехніки у БД було створено наступні сутності: «Зарплата», «Роль», «Робітник», «Покупець», «Товар», «Склад», «Тип товару», «Знижка», «Чек», «Знижка», «Продаж», «Магазин».

Сутність *Зарплата*

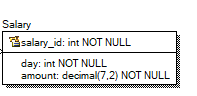


Рис.2.1 Таблиця «Зарплата»

Дана сутність містить інформацію про зарплату:

* salary\_id (PK) — ідентифікаційний номер зарплати;
* data — дата;
* amount — кількість;

Сутність *Роль:*

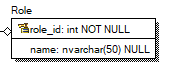


Рис.2.2 Таблиця «Роль»

Дана сутність містить інформацію про роль робітника:

* role\_id (PK) — ідентифікаційний номер ролі;
* name — назва ролі

Сутність *Робітник:*

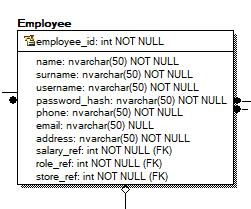


Рис.2.3 Таблиця «Робітник»

Дана сутність містить інформацію про робітників на підприємстві:

* employee\_id (PK) — ідентифікатор номеру робітника
* name — ім’я робітника;
* surname — прізвище робітника;
* username — логін
* ­password\_hash — хеш пароля
* phone — номер;
* email — електронна пошта;
* address — адреса;
* salary\_ref (FK) — ідентифікатор зарплати;
* store\_ref (FK) — ідентифікатор магазину;
* role\_ref (FK) — ідентифікатор ролі;

Сутність *Покупець:*

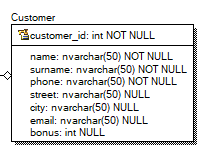


Рис.2.4 Таблиця «Покупець»

Дана сутність містить інформацію про зареєстрованих покупців магазину:

* customer\_id (PK) — ідентифікаційний номер зареєстрованого покупця;
* name — ім’я;
* surname — прізвище;
* phone — номер телефону;
* street — вулиця;
* city — місто;
* email — електронна пошта;
* bonus — бонус .

Сутність *Товар:*

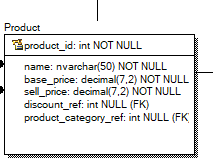


Рис.2.5 Таблиця «Товар»

Дана сутність містить інформацію про усі товари, що продаються у мережі магазинів:

* product\_id (PK) — ідентифікаційний номер товару;
* name — назва товару, що продається у магазині;
* base\_price — початкова ціна;
* sell\_price — кінцева ціна;
* product\_category\_ref (FK) — ідентифікатор категорії товару;
* discount\_ref (FK) — ідентифікатор знижки товару.

Сутність *Склад:*

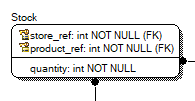


Рис.2.6 Таблиця «Склад»

Дана сутність містить інформацію про наявність товарів на складі у магазинах:

* product\_ref (PK) — ідентифікаційний номер товару;
* store\_ref (PK) — ідентифікаційний номер магазину
* quantity — кількість товару;

Сутність *Тип товару:*

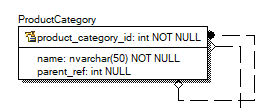


Рис.2.7 Таблиця «Тип товару»

Дана сутність описує типи товарів, що продаються у мережі магазинів:

* product\_category\_id (PK) — ідентифікатор типу товару;
* name — повна назва типу товару, що продається у магазині.
* parent\_ref (FK) — ідентифікатор для створення ієрархії категорій;

Сутність *Знижка:*

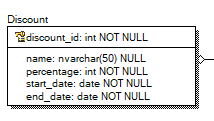


Рис.2.8 Таблиця «Знижка»

Дана сутність описує знижки знижки

* discount\_id (PK) — ідентифікаційний номер кольору;
* name— повна назва кольору або композиції кольорів.
* percentage — відсоток
* start\_date — початкова дата
* end\_date — кінцева дата

Сутність *Чек:*

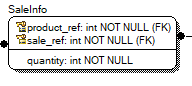


Рис.2.9 Таблиця «Чек»

Дана сутність формує чеки покупців

* product\_ref (PK) — ідентифікаційний номер товару;
* sale\_ref (PK) — ідентифікаційний номер покупки
* quantity — кількість даного товару в чеку

Сутність *Продаж:*

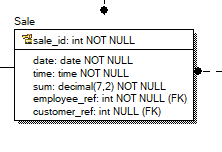


Рис.2.10 Таблиця «Продаж»

Сутність зберігає інформацію про один товар у чеку:

* sale\_id (PK) — ідентифікаційний номер продажі;
* date — дата ;
* time — час;
* sum - сума
* customer\_ref (FK) — ідентифікатор покупця;
* employee\_ref (FK) — ідентифікатор продавця;

Сутність *Магазин:*

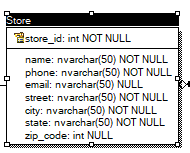


Рис.2.11 Таблиця «Магазин»

Сутність зберігає інформацію про один товар у чеку:

* store\_id (PK) — ідентифікаційний номер магазину;
* name — назва
* phone — телефон
* email — електронна адреса
* street — вулиця
* city — місто
* state — країна
* zip\_code — поштовий індекс

### 

### 2.1.2. Опис зв’язків між сутностями

Між сутностями було встановлено наступні зв’язки:

* Між сутностями *Зарплата* та *Робітник* було встановлено зв’язок один-до-багатьох. (Робітник може мати лише один тип зарплати, а зарплата може мати *багато* робітників.)
* Між сутностями *Роль* та *Робітник* було встановлено зв’язок один-до-багатьох. (Робітник може мати лише одну відповідну роль, в свою чергу роль може мати багато робітників.)
* Між сутностями *Магазин* та *Робітник* було встановлено зв’язок один-до-багатьох.
* Між сутностями *Покупець* та *Продаж* було встановлено зв’язок один-до-багатьох.
* Між сутностями *Робітник* та *Продаж* було встановлено зв’язок один-до-багатьох.
* Сутність Тип Товару має зв’язок один-до-одного на себе для створення ієрархії типів товару
* Між сутностями *Тип Товару* та *Товар* було встановлено зв’язок один-до-багатьох.
* Між сутностями *Знижка* та *Товар* було встановлено зв’язок один-до-багатьох.
* Між сутностями *Товар* та *Продаж* було встановлено зв’язок багато-до-багатьох. (Для цього створилась допоміжна таблиця *Чек.*)
* Між сутностями *Магазин* та *Товар* було встановлено зв’язок багато-до-багатьох. (Для цього створилась допоміжна таблиця *Склад*.)

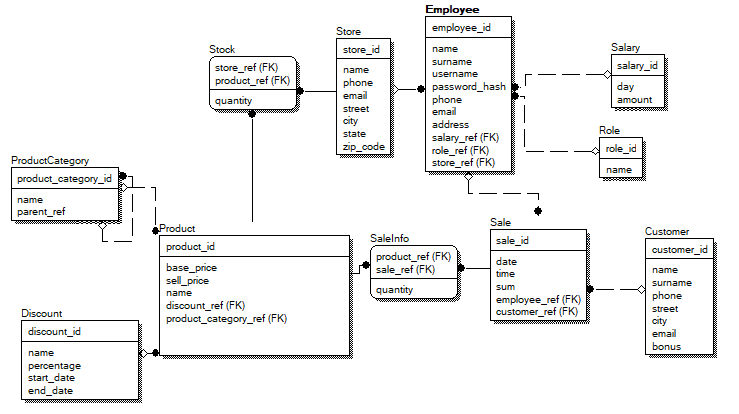
## 

## 2.2. Логічне проектування БД

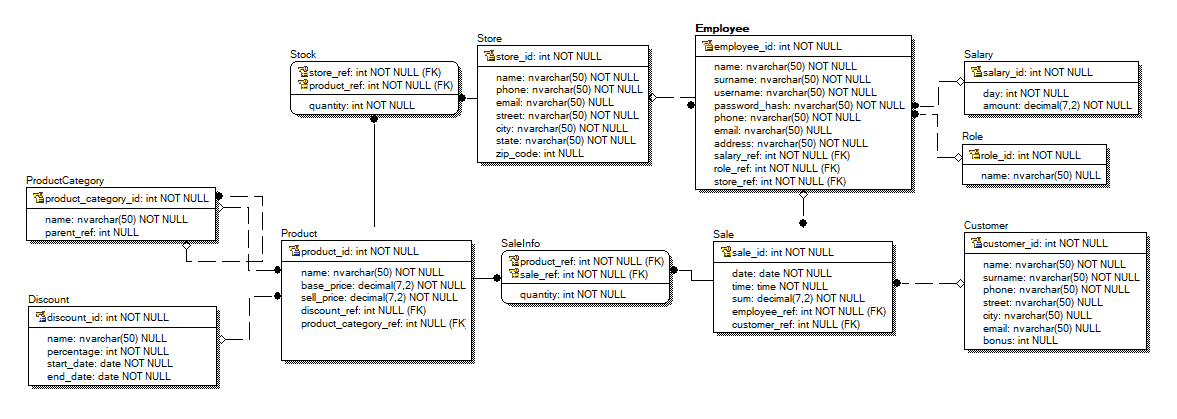
Логічна модель — це загальний погляд на дані, модель даних в пе­вній предметній області. Інколи її називають концептуальною моделлю даних. Такою моделлю і є модель типу сутність-зв'язок.

Фізична модель — це модель даних, реалізована у вигляді опису бази даних. Вона може бути реалізована на двох рівнях, а саме у вигляді трансформаційної моделі і моделі в системі керування базами даних.

В рамках даної роботи було створено логічну та фізичну моделі БД для обліку продажу комплектуючих ПК.



Мал.2.12. Логічна модель БД



Мал.2.13. Фізична модель БД

Після перенесення моделі в середовище SQL Server, схема якої представлена на рисунку 2.14

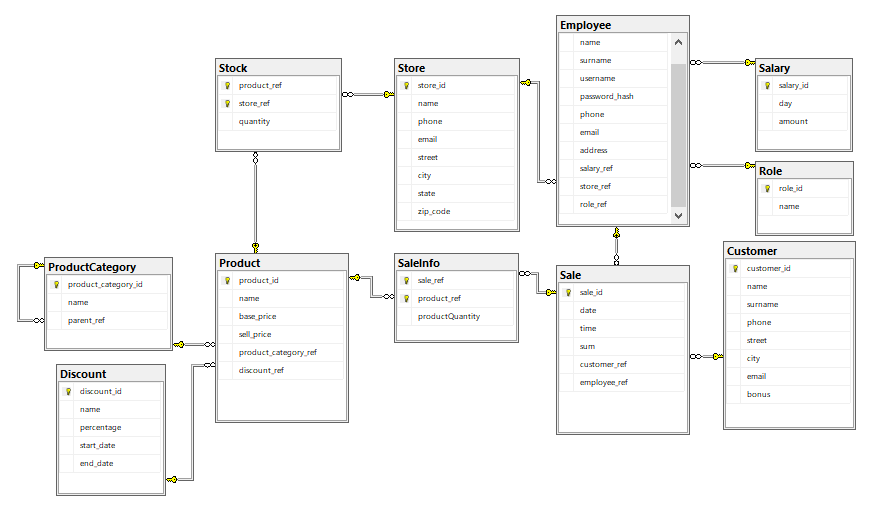


Рис.2.14. Діаграма побудована в середовищі SQL Server

Були додані наступні обмеження:

* Усі первинні ключі не можуть бути рівні NULL (Обмеження SQL) та менші 0.
* На усі атрибути з вторинними ключами було накладено обмеження з умовою більшою за 0.
* На усі атрибути, що позначають кількість компонентів було накладено обмеження з умовою більшою за 0.

## 2.3. Проектування типових запитів і транзакцій

В ході проектування системи управління базою даних було, спроектовано ряд запитів, а саме:

* Вибірка всіх наявних товарів (Таблиця «Product»).
* Запит на додавання нового товару пального (Таблиця «Product»).
* Запит на перевірку логіну і паролю
* Запит на створення нового клієнта (Таблиця «Customer»).
* Запит на створення нового працівника (Таблиця «Employee»).
* Запит на створення чеку (Таблиця «SaleInfo»).
* Запит на оновлення даних про товар (Таблиця «Product»).

Індекси в даній системі будуть використовуватися виключно за замовчуванням середовища SQL Server, через відсутність великих потоків даних для зберігання.

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

## 3.1. Реалізація доступу до бази даних

Для реалізації доступу до даних БД обрано Qt Framework. Він дозволяє використовував зручну Model-View архітектуру для роботи з таблицями.

Для зв‘язування Qt з MS SQL Server було використано драйвер DRIVER={ODBC Driver 17 for SQL Server}.

Щоб отримати доступ до бази даних потрібно просто створити об’єкт класу QSqlDatabase та виконати наступний код

qdb.setConnectOptions();

QString serverName = "{MyServerName}";

QString dbName = "PlumbingStore";

QString connectionString = QString("DRIVER={ODBC Driver 17 for SQL Server};Server=%1;Database=%2;Trusted\_Connection=Yes;").arg(serverName).arg(dbName);

## qdb.setDatabaseName(connectionString);

if (!qdb.open()) {…

## 3.2. Реалізація функціональних характеристик системи

### 3.2.1. Авторизація

Авторизація реалізована наступним чином:

За допомогою елементу input користувач вводить свій email і пароль. Якщо введені дані вірні та при спробі під'єднатися до SQL Server, не виникло виняткової ситуації, тоді відкривається головне вікно програми, при завантаженні якого форма авторизації закривається, якщо ж виникла помилка, або введені дані були хибними згенерується повідомлення про помилку і виведеться на форму.

Логічне керування формами відповідно до ролі :

void MainWindow::on\_pb\_login\_clicked()

{

auto res = DAL::login(ui->lineEdit\_4->text(), ui->lineEdit\_3->text());

qDebug() << "id and role: " <<res.first << " " << res.second;

QString role = res.second;

if(!role.isEmpty())

{

if(role == "Admin")

loadAdmin();

if(role == "Manager")

loadManager();

if(role == "Casier")

loadCasier();

}

else {

ui->label\_error->setVisible(true);

}

}

Метод, що відповідає за перевірку логіну і паролю з бази даних, повертає ідентифікатор працівника та його роль у разі успіху.

std::pair<int, QString> DAL::login(QString username, QString password)

{

QSqlQuery query;

query.prepare("SELECT e.employee\_id, r.name FROM Employee e "

"JOIN Role r on e.role\_ref = r.role\_id "

"WHERE e.username = ? AND e.password\_hash = ?");

query.bindValue(0, username);

query.bindValue(1, password);

query.exec();

query.first();

return std::make\_pair(query.value(0).toInt(),

query.value(1).toString());

}

На рис.3.1 зображена форма авторизації:

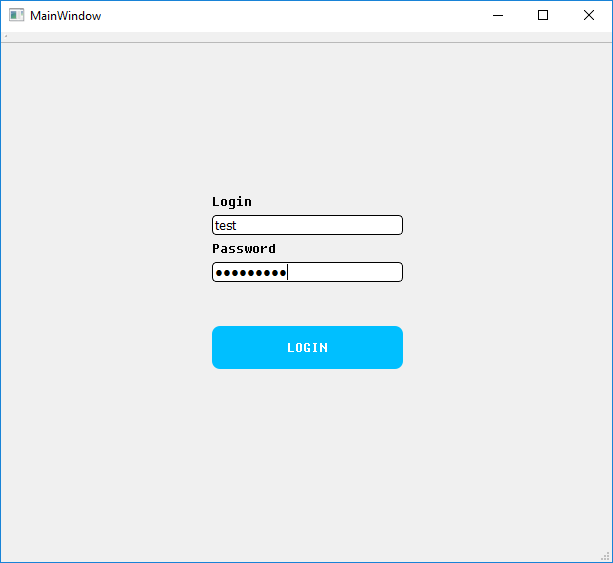


Рис.3.1 форма авторизації

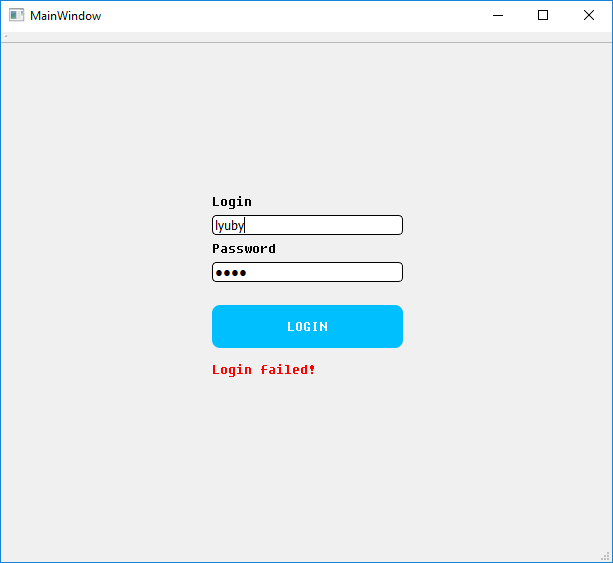


Рис.3.2 Невдала авторизація

### 3.2.2. Вибірка даних

Вибірка даних через модель:

void initializeModel(QSqlRelationalTableModel \*model)

{

model->setTable("Product");

model->setEditStrategy(QSqlTableModel::OnFieldChange);

model->setHeaderData(1, Qt::Horizontal, QObject::tr("Product"));

model->setHeaderData(2, Qt::Horizontal, QObject::tr("Price"));

model->setHeaderData(4, Qt::Horizontal, QObject::tr("Category"));

model->setHeaderData(5, Qt::Horizontal, QObject::tr("Discount %"));

model->setRelation(4, QSqlRelation("ProductCategory", "product\_category\_id", "name"));

model->setRelation(5, QSqlRelation("Discount", "discount\_id", "percentage"));

model->select();

}

На рис.3.3. зображена вибірка даних методом QSqlTableModel ::select()

менеджером для всесення змін

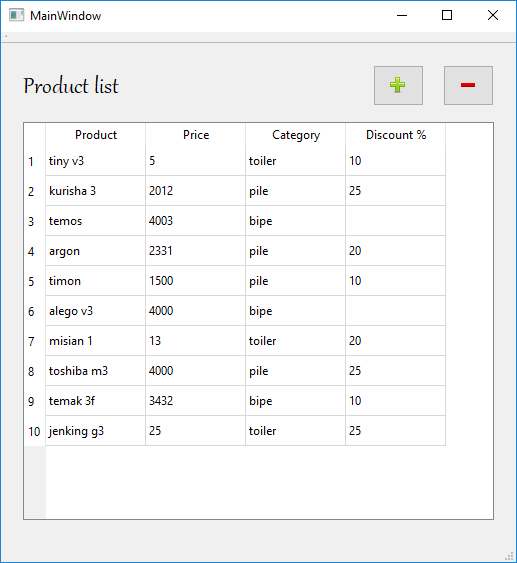


Рис.3.3. Вибірка даних з таблиці Product

### 3.2.3. Додавання даних

### bool DAL::insertIntoProduct(Product product)

### {

### QSqlQuery query;

### query.prepare("INSERT INTO Product "

### "VALUES (?, ?, ?, ?, ?)");

### query.bindValue(0, product.name);

### query.bindValue(1, product.base\_price);

### query.bindValue(2, product.sell\_price);

### query.bindValue(3, product.product\_category\_ref);

### query.bindValue(4, product.discount\_ref);

### return query.exec();

### }

### 3.2.4. Видалення даних

bool DAL::deleteFromProduct(int id)

{

QSqlQuery query;

query.prepare("DELETE FROM Product WHERE product\_id = ?");

query.bindValue(0, id);

return query.exec();

}

### Редагування даних

Редагування даних забезпечує модель, а саме встановулюємо стратегію редагування, яка спрацьовує після змін у комірці таблиці

model->setEditStrategy(QSqlTableModel::OnFieldChange);

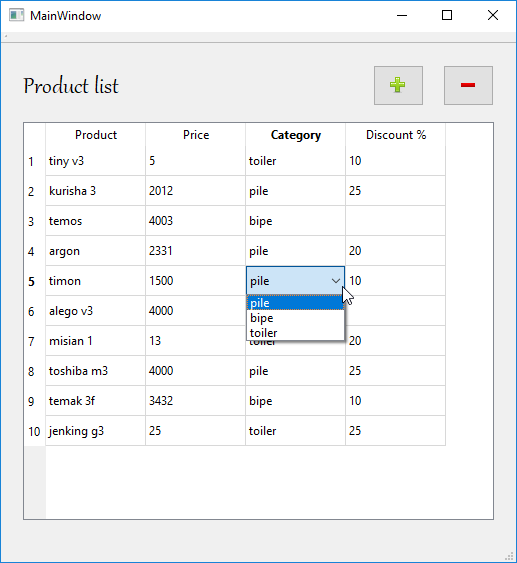


Рис.3.3. Редагування даних з таблиці Product

(зміна поля що є зовнішнім ключем)

# ВИСНОВКИ

Результатом даної курсової роботи є база даних для керування мережею магазинів сантехніки та програмне забезпечення для роботи з нею.

В ході роботи над проектом було використано такі програмні продукти як Qt Creator, Microsoft SQL Server Management Studio 2017. Qt Creator являє собою середовище розробки даного проекту. Сама ж база даних була реалізована за допомогою MS SQL Server. В якості технології зв’язку між додатком і базою даних використано технологію Qt Framework. Користувацький інтерфейс побудований на основі Qt Creator Widgets.

Виконуючи цю курсову роботу я оволодів навичками концептуального проектування баз даних, а також організації взаємодії бази даних з програмним забезпеченням за допомогою технології Qt Framework, навчився проектувати типові запити та транзакції.

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

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

# 

# СПИСОК ЛІТЕРАТУРИ

1. Дудзяний І. М. Об’єктно-орієнтоване моделювання програмних систем. Львів: Видавництво ЛНУ ім. Івана Франка, 2007. 107 с.
2. Шлеє М. Qt 5.10. Профессиональное программирование на C++.СПб: БХВ\_Петербург, 2016. 1072 с.
3. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ: Видавнича група BHV, 2006. 384 с.
4. Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих: пер. с англ. СПб.: БХВ-Петербург, 2013. 816 с.
5. Qt SQL. Qt Documentation. URL: https://doc.qt.io/qt-5/qtsql-index.html
6. SQL Server Documentation URL: https://docs.microsoft.com/ru-ru/sql/sql-server/sql-server-technical-documentation
7. Создание и удаление пользователей в Microsoft SQL Server.URL: https://info-comp.ru/obucheniest/541-creating-and-deleting-users-in-ms-sql-server.html

**Додаток А. Скрипт створення БД**

CREATE TABLE Customer

(

customer\_id int NOT NULL ,

name nvarchar(50) NOT NULL ,

surname nvarchar(50) NOT NULL ,

phone nvarchar(50) NOT NULL ,

street nvarchar(50) NULL ,

city nvarchar(50) NULL ,

email nvarchar(50) NULL ,

bonus int NULL

)

go

ALTER TABLE Customer

ADD CONSTRAINT XPKCustomer PRIMARY KEY CLUSTERED (customer\_id ASC)

go

CREATE TABLE Discount

(

discount\_id int NOT NULL ,

name nvarchar(50) NULL ,

percentage int NOT NULL ,

start\_date date NOT NULL ,

end\_date date NOT NULL

)

go

ALTER TABLE Discount

ADD CONSTRAINT XPKDiscount PRIMARY KEY CLUSTERED (discount\_id ASC)

go

CREATE TABLE Employee

(

employee\_id int NOT NULL ,

name nvarchar(50) NOT NULL ,

surname nvarchar(50) NOT NULL ,

username nvarchar(50) NOT NULL ,

password\_hash nvarchar(50) NOT NULL ,

phone nvarchar(50) NOT NULL ,

email nvarchar(50) NULL ,

address nvarchar(50) NOT NULL ,

salary\_ref int NOT NULL ,

role\_ref int NOT NULL ,

store\_ref int NOT NULL

)

go

ALTER TABLE Employee

ADD CONSTRAINT XPKEmployee PRIMARY KEY CLUSTERED (employee\_id ASC)

go

CREATE TABLE Product

(

product\_id int NOT NULL ,

base\_price decimal(7,2) NOT NULL ,

sell\_price decimal(7,2) NOT NULL ,

name nvarchar(50) NOT NULL ,

discount\_ref int NULL ,

product\_category\_ref int NULL

)

go

ALTER TABLE Product

ADD CONSTRAINT XPKProduct PRIMARY KEY CLUSTERED (product\_id ASC)

go

CREATE TABLE ProductCategory

(

product\_category\_id int NOT NULL ,

name nvarchar(50) NOT NULL ,

parent\_ref int NULL

)

go

ALTER TABLE ProductCategory

ADD CONSTRAINT XPKProductCategory PRIMARY KEY CLUSTERED (product\_category\_id ASC)

go

CREATE TABLE Role

(

role\_id int NOT NULL ,

name nvarchar(50) NULL

)

go

ALTER TABLE Role

ADD CONSTRAINT XPKRole PRIMARY KEY CLUSTERED (role\_id ASC)

go

CREATE TABLE Salary

(

salary\_id int NOT NULL ,

day int NOT NULL ,

amount decimal(7,2) NOT NULL

)

go

ALTER TABLE Salary

ADD CONSTRAINT XPKSalary PRIMARY KEY CLUSTERED (salary\_id ASC)

go

CREATE TABLE Sale

(

sale\_id int NOT NULL ,

date date NOT NULL ,

time time NOT NULL ,

sum decimal(7,2) NOT NULL ,

employee\_ref int NOT NULL ,

customer\_ref int NULL

)

go

ALTER TABLE Sale

ADD CONSTRAINT XPKSale PRIMARY KEY CLUSTERED (sale\_id ASC)

go

CREATE TABLE SaleInfo

(

product\_ref int NOT NULL ,

sale\_ref int NOT NULL ,

quantity int NOT NULL

)

go

ALTER TABLE SaleInfo

ADD CONSTRAINT XPKSaleInfo PRIMARY KEY CLUSTERED (product\_ref ASC,sale\_ref ASC)

go

CREATE TABLE Stock

(

store\_ref int NOT NULL ,

product\_ref int NOT NULL ,

quantity int NOT NULL

)

go

ALTER TABLE Stock

ADD CONSTRAINT XPKStock PRIMARY KEY CLUSTERED (store\_ref ASC,product\_ref ASC)

go

CREATE TABLE Store

(

store\_id int NOT NULL ,

name nvarchar(50) NOT NULL ,

phone nvarchar(50) NOT NULL ,

email nvarchar(50) NULL ,

street nvarchar(50) NOT NULL ,

city nvarchar(50) NOT NULL ,

state nvarchar(50) NOT NULL ,

zip\_code int NULL

)

go

ALTER TABLE Store

ADD CONSTRAINT XPKStore PRIMARY KEY CLUSTERED (store\_id ASC)

go

ALTER TABLE Employee

ADD CONSTRAINT R\_4 FOREIGN KEY (salary\_ref) REFERENCES Salary(salary\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Employee

ADD CONSTRAINT R\_5 FOREIGN KEY (role\_ref) REFERENCES Role(role\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Employee

ADD CONSTRAINT R\_7 FOREIGN KEY (store\_ref) REFERENCES Store(store\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Product

ADD CONSTRAINT R\_16 FOREIGN KEY (discount\_ref) REFERENCES Discount(discount\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Product

ADD CONSTRAINT R\_17 FOREIGN KEY (product\_category\_ref) REFERENCES ProductCategory(product\_category\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE ProductCategory

ADD CONSTRAINT R\_15 FOREIGN KEY (product\_category\_id) REFERENCES ProductCategory(product\_category\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Sale

ADD CONSTRAINT R\_2 FOREIGN KEY (employee\_ref) REFERENCES Employee(employee\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Sale

ADD CONSTRAINT R\_3 FOREIGN KEY (customer\_ref) REFERENCES Customer(customer\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE SaleInfo

ADD CONSTRAINT R\_8 FOREIGN KEY (product\_ref) REFERENCES Product(product\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE SaleInfo

ADD CONSTRAINT R\_10 FOREIGN KEY (sale\_ref) REFERENCES Sale(sale\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Stock

ADD CONSTRAINT R\_11 FOREIGN KEY (store\_ref) REFERENCES Store(store\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Stock

ADD CONSTRAINT R\_12 FOREIGN KEY (product\_ref) REFERENCES Product(product\_id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go