**Міністерство освіти і науки України**

**ВСП «Львівський фаховий коледж харчової і переробної промисловості**

**Національного університету харчових технологій»**

**Розрахунково-пояснювальна записка курсової роботи**

з дисципліни **«Організації бази даних та знань»**

на тему: «Проектування бази даних інформаційної системи

продуктової гуртівні»

студента IV курсу групи ПК-4

спеціальності 122 «Комп’ютерні науки»

Гринчака В.О.

Керівник викладач Грицак В.А.

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(підпис) (прізвище та ініціали)

Львів – 2020 рік

**ЗМІСТ**

Арк.

ВСТУП 4

1. ВИЗНАЧЕННЯ ТА ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ 4

1.1. Опис предметної області бази даних 4

1.2.Специфікація вимог для користувачів 5

# 2. КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ 5

## 2.1 Визначення типів сутностей 6

## 2.2 Визначення типів зв'язків 7

## 2.3 Визначення атрибутів і зв'язування їх з типами сутностей і зв'язків 9

## 2.4 Визначення доменів атрибутів 10

## 2.5 Визначення атрибутів, що є потенційними і первинними ключами 11

## 2.6 Створення діаграми „сутність-зв'язок” 11

3. ЛОГІЧНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ 12

3.1. Аналіз реляційної схеми на коректність об’єднання атрибутів

в одному відношенні. 13

3.2. Створення логічної моделі бази даних на основі створеної ER-моделі. 13

3.3. Застосування послідовної нормалізації до відношень. 14

3.4. Обґрунтування вибору СУБД. 15

4. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ ЧЕРЕЗ СУБД 16

4.1. Створення бази даних засобами СУБД 16

4.2. Побудова даталогічної моделі бази даних. 17

4.3. Засоби автоматизації управління системою 18

4.4. Керівництво користувача 19

ВИСНОВКИ 20

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 21

ДОДАТКИ 22

**ВСТУП**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

**КР.122.ПК-4.005.ПЗ**

**ПК-4.011.ПЗ**

Відомі два підходи до організації інформації: файлова організація та організація у вигляді бази даних.

База даних ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) database) – сукупність [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96_(%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%82%D0%B5%D1%85%D0%BD%D1%96%D0%BA%D0%B0)), організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015[[1]](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85#cite_note-1)). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96). Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в [інформаційних системах](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). На даний час [застосунки](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) для роботи з базами даних є одними з найпоширеніших прикладних програм.

У сучасних [інформаційних системах](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) для забезпечення роботи з базами даних використовують [системи керування базами даних (СКБД)](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85). [Система керування базами даних](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) — це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних (за стандартом ISO/IEC 2382:2015[[2]](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85#cite_note-2)). Застосунки для роботи з базою даних можуть бути частиною СКБД або автономними. Найпопулярнішими [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) є [MySQL](https://uk.wikipedia.org/wiki/MySQL), [PostgreSQL](https://uk.wikipedia.org/wiki/PostgreSQL), [Microsoft SQL Server](https://uk.wikipedia.org/wiki/Microsoft_SQL_Server), [Oracle](https://uk.wikipedia.org/wiki/Oracle), [Sybase](https://uk.wikipedia.org/wiki/Sybase), [Interbase](https://uk.wikipedia.org/wiki/Interbase), [Firebird](https://uk.wikipedia.org/wiki/Firebird) та [IBM DB2](https://uk.wikipedia.org/wiki/IBM_DB2). [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) дозволяють ефективно працювати з базами даних, обсяг яких робить неможливим їх ручне опрацювання.

Через тісний зв'язок баз даних з [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) під терміном «база даних» інколи необґрунтовано та неточно мають на увазі систему керування базами даних. Але варто розрізняти базу даних — сховище даних, та [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) — засоби для роботи з базою даних. [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85" \o "Система керування базами даних) з інформаційної системи може бути видалена, але база даних продовжить існувати. І навпаки: [СКБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) може функціонувати без жодної бази даних.

Дана база даних призначена для зберігання інформації про товар який є на складі магазину і на полицях. Дає змогу зберігати інформацію про продаж товару.

1. **ВИЗНАЧЕННЯ ТА ОПИС ПРЕДМЕТНОЇ**

**ОБЛАСТІ**

* 1. **Опис предметної області бази даних**

Крамниці бувають різного типу, вони відрізняються за такими параметрами: площа торгової зали, кількість товарних позицій, рівень обслуговування покупців, технологія розміщення товару тощо.

Загалом крамниці можна умовно розділити на групи:

* [супермаркет](https://uk.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D0%BC%D0%B0%D1%80%D0%BA%D0%B5%D1%82)
* [гіпермаркет](https://uk.wikipedia.org/wiki/%D0%93%D1%96%D0%BF%D0%B5%D1%80%D0%BC%D0%B0%D1%80%D0%BA%D0%B5%D1%82)
* [Бутик](https://uk.wikipedia.org/wiki/%D0%91%D1%83%D1%82%D0%B8%D0%BA)
* [дискаунтер](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D0%B0%D1%83%D0%BD%D1%82%D0%B5%D1%80)
* [cash&carry](https://uk.wikipedia.org/w/index.php?title=Cash%26carry&action=edit&redlink=1)
* [універсам](https://uk.wikipedia.org/wiki/%D0%A3%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B0%D0%BC)
* [універмаг](https://uk.wikipedia.org/wiki/%D0%A3%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D0%BC%D0%B0%D0%B3)
* [міні-маркет](https://uk.wikipedia.org/w/index.php?title=%D0%9C%D1%96%D0%BD%D1%96-%D0%BC%D0%B0%D1%80%D0%BA%D0%B5%D1%82&action=edit&redlink=1)
* [Крамниця безмитної торгівлі](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD_%D0%B1%D0%B5%D0%B7%D0%BC%D0%B8%D1%82%D0%BD%D0%BE%D1%97_%D1%82%D0%BE%D1%80%D0%B3%D1%96%D0%B2%D0%BB%D1%96)
* Крамниця-демонстраційний зал, що торгує за каталогами — підприємство роздрібної торгівлі з широкою номенклатурою дорогих ходових марочних товарів за зниженими цінами.
* Крамниця біля дому
* Крамниця товарів повсякденного попиту — невелика крамниця, розташована поблизу житлового району, яка працює з ранку і допізна без вихідних і пропонує обмежений асортимент ходових товарів повсякденного попиту.

За кваліфікацією крамниці поділяються:

* ринок, вулична торгівля, кіоск, намет, павільйон
* спеціалізована крамниця
* крамниця [гуртової торгівлі](https://uk.wikipedia.org/wiki/%D0%93%D1%83%D1%80%D1%82%D0%BE%D0%B2%D0%B0_%D1%82%D0%BE%D1%80%D0%B3%D1%96%D0%B2%D0%BB%D1%8F)
* універсальна крамниця
* продуктова крамниця з обслуговуванням продавцем через прилавок
* невелика крамниця з самообслуговуванням (менше ніж 4 каси)
* дискаунтер (обмежений вибір товарів, низькі ціни, переважно гуртова торгівля)
* супермаркет, великий продуктова крамниця (від 4 до 15 кас)
* гіпермаркет (понад 15 кас)

# 1.2 Специфікація вимог для користувачів

Виконання фази збору й аналізу вимог користувача є першою в циклі розроблення програм роботи з базами даних.

Результатом виконання цієї програми є полегшення роботи продавця в магазині.

# 2. КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИДАНИХ

Етап концептуального проектування полягає в описі і синтезі інформаційних вимог користувачів у початковий проект БД. Вихідними даними можуть бути сукупність документів користувача при класичному підході або алгоритми додатків (алгоритми бізнесу) при сучасному підході. Результатом цього етапу є високорівневе подання (у вигляді системи таблиць БД) інформаційних вимог користувачів на основі різних підходів.

Спочатку вибирається модель БД. Потім створюється структура БД, яка заповнюється даними за допомогою систем меню, екранних форм або в режимі перегляду таблиць БД. Тут же забезпечується захист і цілісність (у тому числі посилальна) даних за допомогою СУБД або шляхом побудови тригерів.

Концептуальне проектування

Концептуальне проектування бази даних - процес створення моделі використовуваної на підприємстві інформації, що не залежить від будь-яких фізичних аспектів її представлення. Перша фаза процесу проектування бази даних називається концептуальним проектуванням бази даних. Вона полягає в створенні концептуальної моделі даних для аналізованої частини підприємства. Ця модель даних створюється на основі інформації, записаної в специфікаціях вимог користувачів. Концептуальне проектування бази даних абсолютно не залежить від таких подробиць її реалізації, як тип обраної цільовий СКБД, набір створюваних прикладних програм, використовувані мови програмування, тип обраної обчислювальної платформи, а також від будь-яких інших особливостей фізичної реалізації. При розробці концептуальна модель даних постійно піддається тестуванню і перевірці на відповідність вимогам користувачів. Створена концептуальна модель дані підприємства є джерелом інформації для фази логічного проектування бази даних. Приступаючи до розроблення локальної концептуальної моделі даних для представлення користувача «Диспетчер» та «Головний інженер» у базі даних «Автотранспортне підприємство», насамперед, варто виявити різні компоненти цієї моделі, використовуючи наявні специфікації вимог користувача (далі - просто "специфікації"). У кожну створювану модель даних входять наступні компоненти:

* типи сутностей
* типи зв'язків
* атрибути
* домени атрибутів
* потенційні ключі
* первинні ключі

## 2.1 Визначення типів сутностей

Почнемо роботу з того, що визначимо основні типи сутностей, виходячи з наявних специфікацій. У специфікаціях сутності звичайно представлені як іменники або вирази, що містять іменники. Аналіз показує, що основними сутностями, що згадуються в специфікаціях, є наступні:

* Групи продуктів
* Продукти
* Закупочна ціна
* Ціна продажу
* Кількість товару
* Номер чеку
* Дата чеку
* **Таблиця 2.1 Відомості про типи сутностей, які поміщено в документацію бази даних «магазин продуктів»**

|  |  |  |  |
| --- | --- | --- | --- |
| Ім’я сутності | Опис | Псевдоніми | Особливості використання |
| GroupName | Назва групи товарів | Група | Є різні групи товарів |
| Name | Назва товару | Товар | Є різні групи товарів |
| Unit\_of\_measure | Кількість товарів на полиця і магазині | Кількість товару | Кількість товару в магазині по назві товару |
| Purchase\_price | Закупочна ціна | Ціна закупки товару | Кожен товар має різну ціну |
| Selling\_price | Ціна продажу товару | Ціна продажу | Кожен товар має різну ціну продажу |
| Count | Кількість товару на складі | Кількість | Кількість товару на складі |
| DateOfCheck | Дата чеку | Чек | Дата продажу товару |

**2.2 Визначення типів зв'язків**

Для встановлення зв'язків між сутностями і створення зовнішніх ключів ERWin надає можливість поділу типів зв'язків на кілька варіантів:

* • ідентифікує зв'язок - зв'язок, що визначає однозначна відповідність екземпляра однієї сутності єдиному примірнику пов'язаної сутності і, як правило, описує зв'язок 1: 1, але при реалізації зчепленого первинного ключа може реалізовувати зв'язок один - до - багатьох (1: JV);
* • неідентіфіцірующей зв'язок - зв'язок, що реалізує тип зв'язку один - до - багатьох (1 *: N),* представляючи зовнішній ключ в пов'язаної сутності як простого атрибута, на який можуть бути накладені певні додаткові обмеження в порівнянні зі звичайними інформаційними атрибутами;
* • множинна зв'язок - зв'язок, що реалізує тип зв'язку багато - до - багатьох (Л Г : М), представляється тільки на рівні логічної моделі, ілюструючи з'єднання між сутностями, але не створюючи зовнішніх ключів в пов'язаних сутностей;
* • категорізаціона - зв'язок, що забезпечує зв'язування сутності- спільності з сутностями-категоріями типом зв'язку один - до - одному (1: 1) і одночасно створює зовнішній первинний ключ в сутності-категоріях, пов'язаний з первинним ключем сутності-спільності.
* Якщо зведень, що утримуються в специфікаціях, не досить для однозначного визначення властивостей деяких зв'язків, для прояснення ситуації варто звернутися до користувачів.
* Результати аналізу представлені в табл. 2.2.
* **Таблиця 2.2. Зведення про типи зв'язків, поміщені в документацію**

|  |  |  |  |
| --- | --- | --- | --- |
| * Тип сутності | Тип зв'язку | Тип сутності | Кардинальність |
| Група товарів | Групи товарів в магазині | Група | Б:1 |
| Назва товару | Назва товару в магазині | Назва | 1:Б |
| Кількість товару в магазині | Кількість товару | Кількість | 1:Б |
| Закупочна ціна | Ціна за якою закупляють товари | Ціна | 1:Б |
| Ціна продажу товару | Ціна з націнкою на товар | Ціна | 1:Б |
| Дата чеку | Дата коли продали товар | Дата | 1:Б |

## 2.3 Визначення атрибутів і зв'язування їх з типами сутностей і зв'язків

Тепер нам необхідно виділити атрибути сутностей, що у специфікаціях також можуть бути представлені іменниками (або відповідними сполученнями). Атрибут описує деякий аспект визначеної сутності або зв'язку. При виконанні цього етапу варто звернути особливу увагу на ті випадки, коли визначений атрибут справляє враження, ніби він описує більше від одного типу сутності або зв'язку. Подібна ситуація виникає в одному з наступних випадків.

У документацію необхідно помістити докладні зведення про атрибути, перераховані у таблиці. Для кожного атрибута варто вказати загальний опис, тип даних і довжину значення, наявні обмеження, значення за замовчуванням (якщо таке є), псевдоніми (якщо такі існують), а також є атрибут складеним **або** похідним і чи припустиме для нього значення **NULL**. Фрагмент подібного документа наведений у кінці цього розділу.

**Таблиця 2.3. Зведення про атрибути, поміщені в документацію**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Тип сутності | Атрибут | Тип даних, довжина | Обмеження | Значення за замовчуванням | Псевдонім | Допустимість Null | Похідний |
| Product\_groups | GroupID | Ціле число | Первинний ключ |  | ID групи | Ні | Ні |
| GroupName | Символьний тип |  |  | Ім’я групи | Ні | Ні |
| Products | ProductID | Ціле число | Первинний ключ |  | ID пасажира | Ні | Ні |
| GroupID | Ціле число | Вторинний ключ |  | ID групи | Ні | Ні |
| Name | Символьний тип |  |  | Назва товару | Ні | Ні |
| Unit\_of\_measure | Символьний тип |  |  | Кількість товару на полицях | Ні | Ні |
| Purchase\_price | Символьний тип |  |  | Ціна закупки товару | Ні | Ні |
| Selling\_price | Символьний тип |  |  | Ціна продажу товару | Ні | Ні |
| Сonsumption | CheckID | Ціле число |  |  | Чек | Ні | Ні |
| ProductID | Ціле число |  |  | ID продукту | Ні | Ні |
| Count | Ціле число |  |  | Кількість товару якого купили | Ні | Ні |
| DateOfCheck | Дата |  |  | Дата чеку | Ні | Ні |
|  |  |  |  |  |  |  |  |
| Storage | Arrival\_CodeID | Ціле число | Первинний ключ |  | ID товару на складі | Ні | Ні |
| ProductID | Ціле число |  |  | ID продукту | Ні | Ні |
| Purchase\_price | Ціле число |  |  | Ціна закупки товару | Ні | Ні |
|  |  |  |  |  |  |  |
| Selling\_price | Ціле число |  |  | Ціна продажу товару | Ні | Ні |
| NumberOfSales | Символьний тип |  |  | Кількість проданого товару | Ні | Ні |

**2.4 Визначення доменів атрибутів**

Завдання цього етапу побудови локальної концептуальної моделі даних полягає у визначенні доменів атрибутів для всіх атрибутів, присутніх в моделі. Доменом називається деякий пул значень, елементи якого вибираються для присвоєння значень одному або більше атрибутів.

|  |  |  |
| --- | --- | --- |
| Ім'я домену | Характеристики домену | Зразки припустимих значень |
| GroupName | Символьний тип(1-30символів)  Назва групи товарів | Напої, бакалія, овочі, фрукти |
| Name | Символьний тип(1-30символів)  Назва товару | Огірки, Кока-Кола, помідори |
| Unit\_of\_measure | Символьний тип(1-30символів)  Кількість товару на полицях магазину | 12 шт, 5л, 10 шт |
| Purchase\_price | Ціна закупки товару | 40, 12, 51, 75, 10 |
| Selling\_price | Ціна продажу товару | 40, 12, 51, 75, 10 |
| Count | Кількість товару на складі | 12 шт, 5л, 10 шт |
| DateOfCheck | Символьний тип(1-30символів)  Дата чеку | 12.02.2020, 05.06.2020 |

**2.5 Визначення атрибутів, що є потенційними і первинними ключами**

На цьому етапі для кожної сутності встановлюється потенційний ключ (або ключі), після чого здійснюється вибір первинного ключа. Потенційним ключем називається атрибут або мінімальний набір атрибутів заданої суті, дозволяє

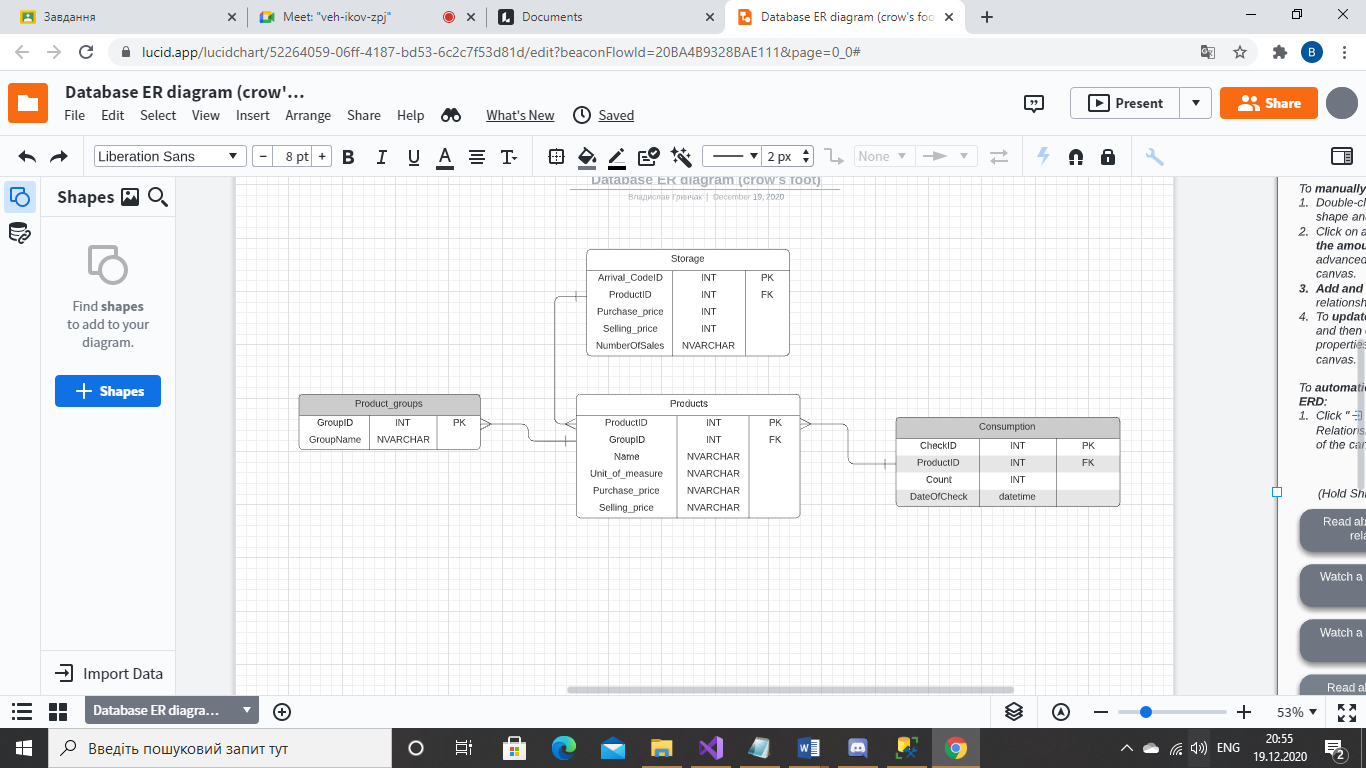
унікальним чином ідентифікувати кожен її примірник. Для деяких сутностей можлива наявність кількох потенційних ключів. У цьому випадку серед них потрібно вибрати один ключ, який буде називатися первинним ключем. Всі інші потенційні ключі будуть називатися альтернативними ключами.

Таблиця 2.5 Зведення про первинні та альтернативні ключі, поміщені в документацію

|  |  |  |
| --- | --- | --- |
| Сутність | Первинний ключ | Альтернативний ключ |
| Product\_groups | GroupID |  |
| Products | ProductID | GroupID |
| Сonsumption |  |  |
| Storage | Arrival\_CodeID | ProductID |

**2.6 Створення діаграми „сутність-зв'язок”**

З метою одержання наочного представлення основних сутностей і зв'язків, визначених у специфікаціях, ми побудували вихідну ER-діаграму. Ця ER-діаграма і підготовлена на етапі 1 документація (у сукупності) являють собою локальну концептуальну модель даних бази даних «Магазин продуктів».



# Розділ 3. Логічне проектування учбової бази даних

Концептуальне (інфологічне) проектування — побудова семантичної моделі предметної області, тобто інформаційної моделі найбільш високого рівня абстракції. Така модель створюється без орієнтації на якусь конкретну [СУБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) і модель даних. Терміни «семантична модель», «концептуальна модель» і «інфологічна модель» є синонімами.

Конкретний вид і зміст концептуальної моделі бази даних визначається обраним для цього формальним апаратом. Зазвичай використовуються графічні нотації, подібні [ER-діаграм](https://uk.wikipedia.org/wiki/ER-%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0).

Найчастіше концептуальна модель бази даних включає в себе:

* опис інформаційних об'єктів або понять предметної області та зв'язків між ними.
* опис [обмежень цілісності](https://uk.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D0%BC%D0%B5%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D1%86%D1%96%D0%BB%D1%96%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%96&action=edit&redlink=1), тобто вимог до допустимих значень даних і до зв'язків між ними.

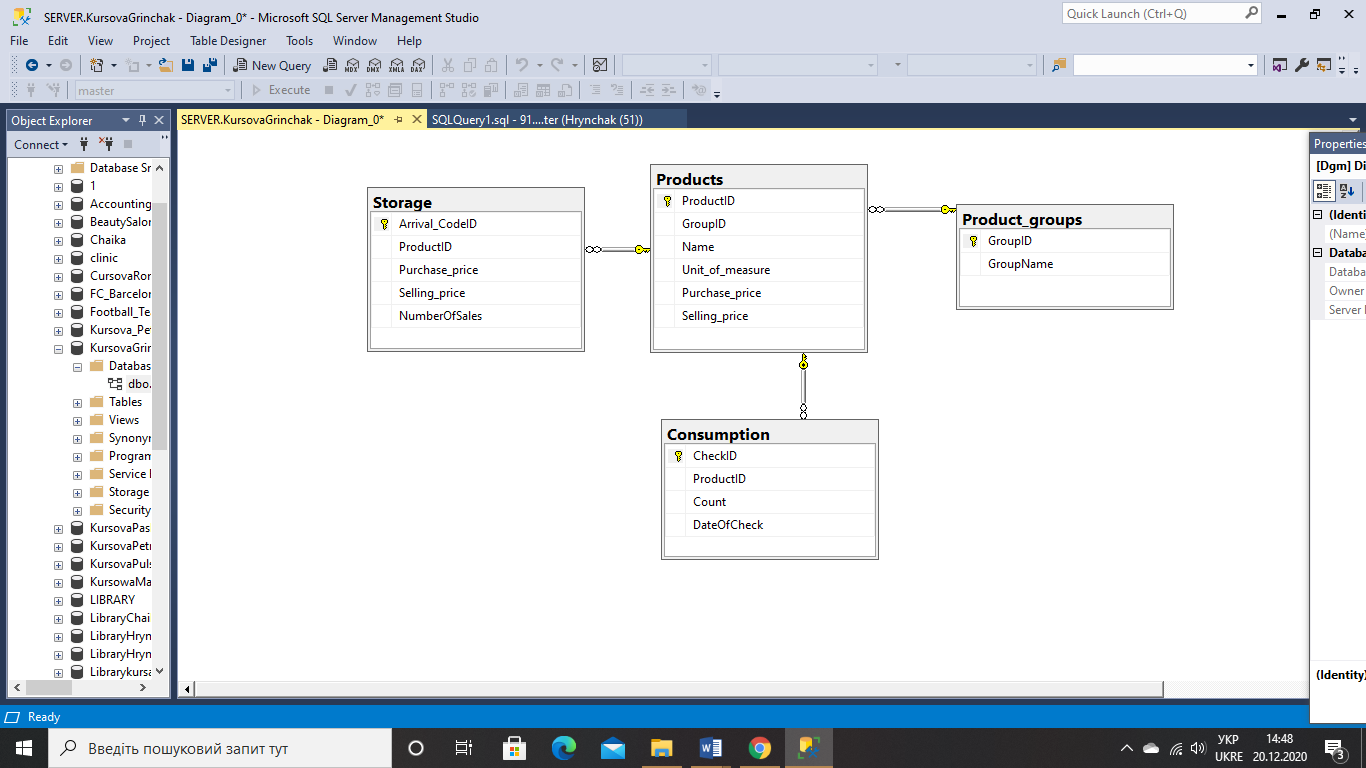
В результаті виконання даного етапу будуть створені коректні, повні та точні моделі уявлень користувачів. Це дасть нам міцну основу, необхідну для виконання наступного етапу, що полягає в об'єднанні окремих локальних логічних моделей даних в єдину глобальну модель даних всього підприємства.

**3.1** . **Аналіз реляційної схеми на коректність об’єднання атрибутів в одному відношенні.**

На цьому етапі ми займимося перетворенням концептуальної моделі даних з метою видалення з неї всіх структур, реалізація яких у СУБД реляційного типу скрутна. Бажаний результат може бути досягнуть за допомогою виконання таких дій, як:

1. Видалення зв'язків типу M : N.
2. Видалення складних зв'язків.
3. Видалення рекурсивних зв'язків.
4. Видалення зв'язків, що мають атрибути.
5. Видалення множинних атрибутів.
6. Повторний огляд зв'язків типу 1:1.
7. Видалення надлишкових зв'язків..

**3.2. Створення логічної моделі бази даних на основі створеної ER-моделі.**



**3.3. Застосування послідовної нормалізації до відношень.**

Нормалізація - це розбивка таблиці на дві або більш, які характеризуються кращими властивостями при доповненні , зміні і вилученні даних. Кінцева мета

нормалізації зводиться до отримання такого проекту бази даних, у котрому кожний факт з'являється лише в однім місці, тобто виключена надлишковість інформації. Це робиться не стільки з метою економії пам'яті, скільки для виключення можливої суперечливості збережених даних.

Кожна таблиця в реляційній БД задовольняє умові, у відповідності з якою у позиції на перетині кожного рядка і стовпця таблиці завжди знаходиться єдине атомарне значення і ніколи не може бути множини таких значень. Будь-яка

таблиця, що задовольняє цій умові, називається нормалізованою.

Кожній нормальній формі відповідає деякий визначений набір обмежень. Відношення знаходиться в деякій нормальній формі, якщо задовольняється властивий їй набір обмежень.

Кожна нормальна форма є більш обмеженою і більш бажаною, ніж попередня. Це зв'язано з тим, що в (N+1)-ій нормальній формі вилучаються деякі небажані властивості, які характерні N-ій нормальній формі. Теорія нормалізації грунтується на наявності тієї або іншої залежності між полями таблиці.

Основні властивості нормальних форм:

· кожна наступна нормальна форма в деякому змісті краще попередньої;

· при переході до наступної нормальної форми властивості попередніх нормальних властивостей зберігаються.

Найбільше важливі нормальні форми відношень грунтуються на фундаментальному в теорії реляційних баз даних понятті функціональної залежності.

Визначення 1. Функціональна залежність .

У відношенні R атрибут Y функціонально залежить від атрибута X (X і Y можуть бути складовими) у тому і тільки в тому випадку, якщо кожному значенню X відповідає в точності одне значення Y: X Y.

Визначення 2. Повна функціональна залежність.

Функціональна залежність X Y називається повною, якщо атрибут Y не залежить функціонально від будь-якої підмножини X.

Визначення 3. Транзитивна функціональна залежність .

Функціональна залежність називається транзитивною, якщо з функціональних залежностей X Y та Y Z випливає, що X Z.

**3.4. Обґрунтування вибору СУБД**

Кожна СУБД повинна розв’язувати такі задачі:

1. Створення баз даних – для цього необхідно визначити які дані будуть зберігатися і якого вони типу, потрібно описати структуру бази даних, тобто зв’язки між її елементами. Все це заноситься в пам’ять і СУБД звертається до цієї інформації по мірі необхідності.
2. Введення даних в пам’ять – здійснюється контроль за вводом СУБД, вона керує розміщенням даних у пам’яті.
3. Експлуатація бази даних – користувач може обновити базу даних, додати чи вилучити, вибрати з бази даних необхідну інформацію.
4. Захист даних – він необхідний у випадку раптового виключення живлення і якщо декілька користувачів одночасно здійснюють доступ до даних.
5. Обробка даних – найчастіше це сортування даних, математична обробка даних, об’єднання даних.
6. Вивід даних на екран або тверді копії.

Через зростаючі об'єми інформації зростає ймовірність помилок при розробці баз даних. Для надання кінцевому користувачеві динамічної інформації можна використовувати різні бази даних. Вибір залежить в основному від того, які

програмні продукти використовуються в даний час у організації, а також від можливостей бюджету організації. У цьому розділі будуть перераховані найпопулярніші бази даних, які широко застосовуються в прикладному програмуванні.

**4. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ ЧЕРЕЗ СУБД**

На цьому етапі відбувається створення і перевірка глобальної логічної моделі даних. Глобальна модель створюється шляхом злиття локальних логічних моделей для представлення користувача Диспетчер та Головний інженер. Глобальна логічна модель даних повинна відбивати особливості представлень обох користувачів

**4.1 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ ЧЕРЕЗ СУБД**

Таблиці складають основу бази даних – саме в них зберігаються всі дані. Перед усім, повинна бути спланована структура кожної таблиці. Структура

таблиць обумовлюється вмістом тих вихідних форм, запитів та звітів, які

повинні бути отримані при роботі з базою даних. При планування таблиць необхідно уникати повторення колонок в різних таблицях, тільки якщо

вони не слугують для визначення зав’язків між ними.

Таблиця складається з записів (рядків), кожний з яких описує одну сутність. Кожна колонка таблиці – це поле. Поле містить однотипну інформацію, яка визначає тип даних. Тип даних визначає вид і межі допустимих значень. Які можуть бути введені в поле, а також об’єм пам’яті, який виділяється для цього поля, що важливо при проектуванні великих БД.

Створимо таблицю Product\_groups з наступними атрибутами за допомогою SQL – команди Create

GO

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Product\_groups' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Product\_groups](

[GroupID] INT IDENTITY(1,1) NOT NULL,

[GroupName] NVARCHAR(50) NOT NULL

CONSTRAINT PK\_Group PRIMARY KEY CLUSTERED (GroupID)

)

END;

Далі створимо таблицю Products:

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Products' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Products](

[ProductID] INT IDENTITY(1,1) NOT NULL,

[GroupID] INT NOT NULL,

[Name] NVARCHAR(50) NOT NULL,

[Unit\_of\_measure] NVARCHAR(50) NOT NULL,

[Purchase\_price] NVARCHAR(50) NOT NULL,

[Selling\_price] NVARCHAR(50) NOT NULL

CONSTRAINT PK\_Product PRIMARY KEY CLUSTERED (ProductID)

CONSTRAINT FK\_Products\_Product\_groups FOREIGN KEY (GroupID) REFERENCES Product\_groups(GroupID)

)

END;

Далі створимо таблицю Storage:

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Storage' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Storage](

[Arrival\_CodeID] INT IDENTITY(1,1) NOT NULL,

[ProductID] INT NOT NULL,

[Purchase\_price] INT NOT NULL,

[Selling\_price] INT NOT NULL,

[NumberOfSales] NVARCHAR(50) NOT NULL

CONSTRAINT PK\_Arrival\_Code PRIMARY KEY CLUSTERED (Arrival\_CodeID),

CONSTRAINT FK\_Storage\_Products FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

)

END;

Далі створимо таблицю Сonsumption:

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Сonsumption' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Сonsumption](

[CheckID] INT NOT NULL,

[ProductID] INT NOT NULL,

[Count] INT NOT NULL,

[DateOfCheck] datetime

CONSTRAINT PK\_Consumption\_Code PRIMARY KEY CLUSTERED (CheckID),

CONSTRAINT FK\_Consumption\_Products FOREIGN

KEY (ProductID) REFERENCES Products(ProductID)

)

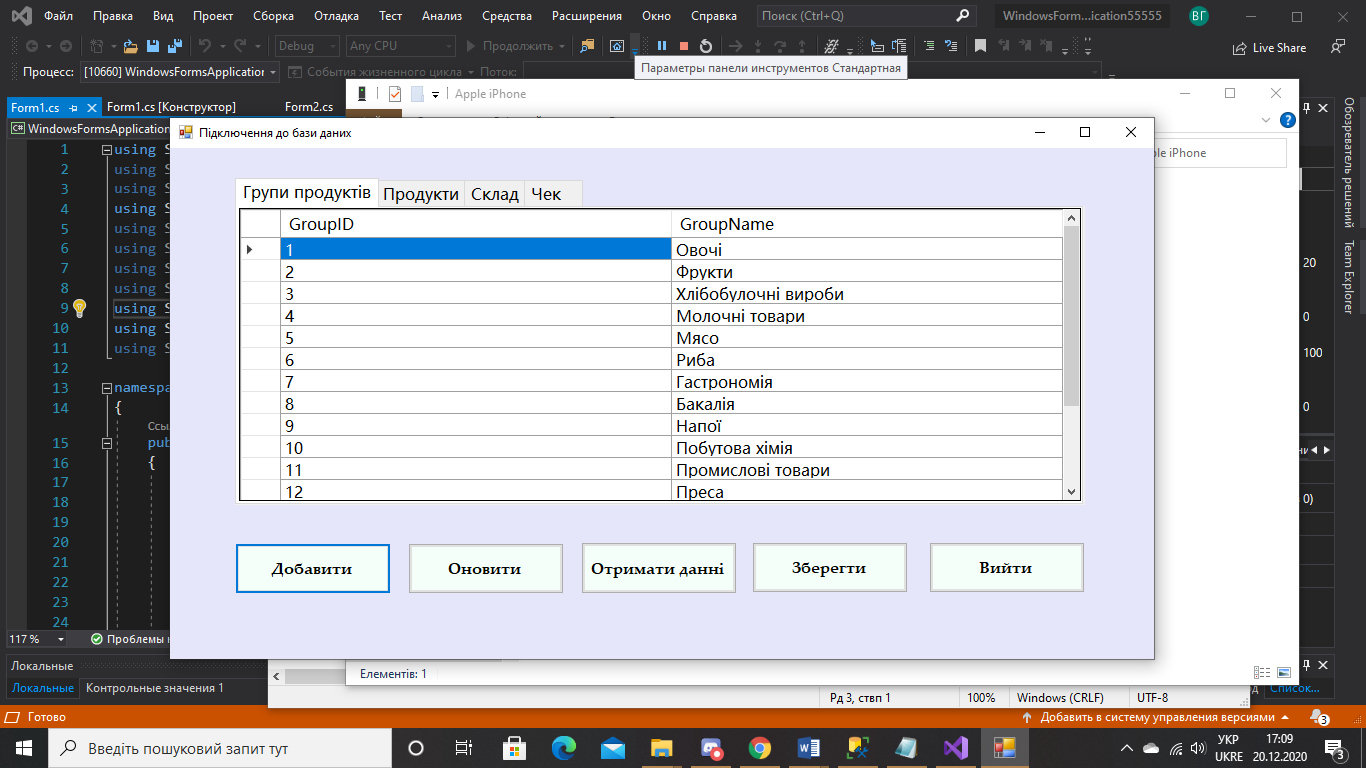
END;

**4.3 Засоби автоматизації управління системою**

База даних з предметною областю “Продуктовий магазин” повинна в зручному вигляді відображати інформацію про гравців, персонал, інформацію про команду. Також за допомогою бази даних ми повинні мати можливість редагувати і добавляти нові данні, а також дізнаватись детальну інформацію.

Ці завдання в базі даних реалізуються за допомогою звітів форм.

На рисунку зображено стартове вікно програми, яке реалізує авторизацію користувача для перегляду таблиць



**4.4. Керівництво користувача**

При запуску програми користувачу вибиває вікно для з’єднання з БД, куди сам користувач має ввести дані про назву бази даних, свій логін та пароль. Після входу в свою базу даних користувач може переглянути всі таблиці та їх данні. На самій формі буде розміщений TabPage, який буде переключатися на

потрібні користувачу таблиці0, що розміщені в DataGriew. Також для кожної таблиці будуть такі кнопки: Добавити, Оновити, Вийти, Зберегти. Щоб

добавити нові дані в таблицю потрібно натиснути на кнопку Добавити, після чого відкриється нова форма куди можна буде ввести данні до таблиць. Видалення данних реалізоване через кнопку Delete. На кнопку Вийти користувач може вийти з програми.

**Висновок:**

Таким чином, під час виконання курсової роботи я закріпив свої теоретичні знання про проектування баз даних.

Я навчився:

* Проектувати концептуальні моделі бази даних, а саме: визначати сутності, зв’язки між ними, атрибути сутностей, їх домени.
* Проектувати логічні моделі бази даних, а саме: на основі концептуальної моделі створювати відношення, визначати потенційні і зовнішні ключі.
* Створювати зовнішні представлення бази даних для різних користувачів.

У ході виконання курсової роботи я за допомогою програми Microsoft Visual Studio створив ORM source model та Database model diagram для бази даних продуктовий магазин.

# Література

1. Visio-Based Database Modeling in Visual Studio .NET Enterprise Architect: Part 3 [an electronic resource] URL: <http://msdn.microsoft.com/en-us/library/aa290382(v=VS.71).aspx> (date of treatment: 11.25.2011)
2. Методичні вказівки до виконання курсової роботи з дисципліни «Проектування баз даних» «Концептуальне, логічне та фізичне проектування навчальної бази даних». Укладач: канд. техн. наук, доцент Крещенко Л.Ф. Полтава: ПолтНТУ, 2003 р. – 54с.
3. Борис Леонтьев. MS Office Visio 2003 не для дилетантов. Построение проектов, диаграмм и бизнес-схем в операционной системе MS Windows XP. М: Новый Издательский Дом, 2005 г. – 384с.
4. Якушев Д.М. IT-проектирование в программе Microsoft Visio 2002 Professional. М: Познавательная книга Прес, 2004г. – 384с.

**Додаток**

**SQL code**

USE master;

GO

IF DB\_ID('KursovaGrinchak') IS NOT NULL

BEGIN

DROP DATABASE KursovaGrinchak;

END;

GO

IF DB\_ID('KursovaGrinchak') IS NULL

BEGIN

CREATE DATABASE KursovaGrinchak

CONTAINMENT = NONE

ON PRIMARY

(NAME = N'KursovaGrinchak', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLSERVER\MSSQL\DATA\KursovaGrinchak.mdf', SIZE=6096KB, MAXSIZE=UNLIMITED, FILEGROWTH=1024KB)

LOG ON

(NAME = N'KursovaGrinchak\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLSERVER\MSSQL\DATA\KursovaGrinchak\_log.ldf', SIZE=1024KB, MAXSIZE=2048GB, FILEGROWTH=10%)

COLLATE Ukrainian\_100\_CI\_AS;

END;

GO

USE KursovaGrinchak;

GO

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Product\_groups' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Product\_groups](

[GroupID] INT IDENTITY(1,1) NOT NULL,

[GroupName] NVARCHAR(50) NOT NULL

CONSTRAINT PK\_Group PRIMARY KEY CLUSTERED (GroupID)

)

END;

DELETE FROM Product\_groups

DBCC CHECKIDENT('Product\_groups', RESEED ,0)

GO

INSERT INTO Product\_groups(GroupName)

VALUES ('Овочі'), ('Фрукти'),('Хлібобулочні вироби'), ('Молочні товари'),('Мясо'), ('Риба'),

('Гастрономія'), ('Бакалія'),('Напої'), ('Побутова хімія'),('Промислові товари'), ('Преса'),

('Сигарети'),('Алкоголь'), ('Кондитерські вироби');

DELETE FROM Products

DBCC CHECKIDENT('Products', RESEED ,0)

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Products' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Products](

[ProductID] INT IDENTITY(1,1) NOT NULL,

[GroupID] INT NOT NULL,

[Name] NVARCHAR(50) NOT NULL,

[Unit\_of\_measure] NVARCHAR(50) NOT NULL,

[Purchase\_price] NVARCHAR(50) NOT NULL,

[Selling\_price] NVARCHAR(50) NOT NULL

CONSTRAINT PK\_Product PRIMARY KEY CLUSTERED (ProductID)

CONSTRAINT FK\_Products\_Product\_groups FOREIGN KEY (GroupID) REFERENCES Product\_groups(GroupID)

)

END;

GO

INSERT INTO Products(GroupID, [Name], Unit\_of\_measure, Purchase\_price, Selling\_price)

VALUES(3,'Батон Сихівський', '10 шт',11.00,16.00),

(3,'Хліб Стрілецький','10 шт', 10.00, 15.50),

(3,'Хліб Карпацький', '10 шт',14.00,18.00),

(1,'Помідори','15 кг', 25.00, 37.00),

(1,'Цибуля', '20 кг',3.00,5.00),

(1,'Огірки','40 кг', 27.00, 39.00),

(2,'Мандарини', '18 кг',30.00,41.00),

(2,'Лимони','30 кг', 29.00, 35.00),

(2,'Банани','15 кг', 26.00, 37.00),

(4,'Галичина 2.5%', '15шт',20.00,26.00),

(4,'Живинка (смак чорниця)','20 шт', 6.00, 12.00),

(4,'Кефір Галичина 400мг','30 шт', 9.00, 12.00),

(5,'Філе куряче', '30 кг',76.00,85.00),

(5,'Фарш асорті','25 кг', 71.00, 80.00),

(5,'Печінка індича','15 кг', 31.00, 45.00),

(6,'Короп живий', '20 кг',91.00,101.00),

(6,'Скумбрія хол. коп.','10 кг', 135.00, 150.00),

(6,'Оселедець слабосол.','12 кг', 49.00, 60.00),

(7,'Сосиски Філейні', '14 кг', 105.00,120.00),

(7,'Сердельки Молочні','20 кг', 117.00, 130.00),

(7,'Сир Сметанковий (Радимо)', '17 кг',124.00,140.00),

(7,'Ковбаса Дрогобицька (Алан)','7 кг', 167.00, 180.00),

(8,'Вермішель Чемпіон 500г', '20 шт',10.00,14.00),

(8,'Консерва сардина в ол. (Море)','15 шт', 24.00, 30.00),

(8,'Олія (Чумак) 900мг', '2 шт',32.00,41.00),

(9,'Кока-Кола 1.5л','25 шт', 18.00, 21.00),

(9,'Моршинська 1.5л','20 шт', 11.00, 15.00),

(9,'Сік (Садочок) 1л', '25 шт',18.00,23.00),

(10,'Порошок (Ушастий нянь) 4.5кг','15 шт', 163.00, 185.00),

(10,'Доместос 1л','12 шт', 63.00, 70.00),

(10,'Миючий засіб до посуду (Галла лимон)', '30 шт',18.00,26.00),

(11,'Колготки жіночі (Конте)','10 шт', 71.00, 80.00),

(11,'Шкарпетки чоловічі (Славія)','10 кг', 21.00, 28.00),

(11,'Чашка 200мг', '6 кг',27.00,32.00),

(12,'Газета (Високий замок)','12 шт', 8.00, 10.00),

(12,'Журнал (Колосок)','12 шт', 48.00, 55.00),

(12,'Газета (Експрес тиждневик)', '12 шт',9.00,11.00),

(13,'Парламент сільвер','10 шт', 51.00, 60.00),

(13,'Бонд компакт','10 шт', 43.00, 50.00),

(13,'Мальборо голд', '10 шт',48.00,60.00);

GO

DELETE FROM Сonsumption

DROP TABLE Сonsumption

DBCC CHECKIDENT('Сonsumption', RESEED ,0)

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Сonsumption' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Сonsumption](

[CheckID] INT NOT NULL,

[ProductID] INT NOT NULL,

[Count] INT NOT NULL,

[DateOfCheck] datetime

CONSTRAINT PK\_Consumption\_Code PRIMARY KEY CLUSTERED (CheckID),

CONSTRAINT FK\_Consumption\_Products FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

)

END;

INSERT INTO Сonsumption(CheckID,ProductID,[Count])

VALUES(1,2,3),

(2,3,1)

GO

DROP TABLE Storage

IF NOT EXISTS(SELECT 1 FROM sys.Tables WHERE Name=N'Storage' AND Type=N'U')

BEGIN

CREATE TABLE [dbo].[Storage](

[Arrival\_CodeID] INT IDENTITY(1,1) NOT NULL,

[ProductID] INT NOT NULL,

[Purchase\_price] INT NOT NULL,

[Selling\_price] INT NOT NULL,

[NumberOfSales] NVARCHAR(50) NOT NULL

CONSTRAINT PK\_Arrival\_Code PRIMARY KEY CLUSTERED (Arrival\_CodeID),

CONSTRAINT FK\_Storage\_Products FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

)

END;

DELETE FROM Storage

DBCC CHECKIDENT('Storage', RESEED ,0)

GO

INSERT INTO Storage(ProductID,Purchase\_price,Selling\_price, NumberOfSales)

VALUES(1, 11.00, 16.00,'10 шт'),

(2, 10.00,15.00,'10 шт'),

(3, 14.00, 18.00,'10 шт'),

(4, 25.00, 37.00,'15 кг'),

(5, 3.00, 5.00,'20 кг'),

(6, 27.00, 39.00,'40 кг'),

(7, 30.00, 41.00,'18 кг'),

(8, 29.00, 35.00,'30 кг'),

(9, 26.00, 37.00,'15 кг'),

(10, 20.00, 26.00,'15 шт'),

(11, 6.00, 12.00,'20 шт'),

(12, 9.00, 12.00,'30 шт'),

(13, 76.00, 86.00,'18 кг'),

(14, 71.00, 80.00,'25 кг'),

(15, 91.00, 101.00,'20 кг'),

(16, 76.00, 86.00,'20 кг'),

(17, 135.00, 150.00,'10 кг'),

(18, 49.00, 60.00,'15 кг'),

(19, 105.00, 120.00,'14 кг'),

(20, 117.00, 130.00,'20 кг'),

(21, 124.00, 140.00,'17 кг'),

(22, 167.00, 180.00,'7 кг'),

(23, 10.00, 14.00,'20 шт'),

(24, 24.00, 30.00,'20 шт'),

(25, 32.00, 41.00,'4 шт'),

(26, 6.00, 12.00,'20 шт'),

(27, 11.00, 15.00,'20 шт'),

(28, 18.00, 23.00,'25 шт'),

(29, 163.00, 185.00,'15 шт'),

(30, 63.00, 70.00,'12 шт'),

(31, 18.00, 26.00,'30 шт'),

(32, 71.00, 80.00,'10 шт'),

(33, 21.00, 28.00,'10 шт'),

(34, 27.00, 32.00,'6 шт'),

(35, 8.00, 10.00,'12 шт'),

(36, 48.00, 55.00,'12 шт'),

(37, 9.00, 11.00,'12 шт'),

(38, 51.00, 60.00,'10 шт'),

(39, 43.00, 50.00,'10 шт'),

(40, 48.00, 60.00,'10 шт')

**C# code**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Drawing.Printing;

namespace WindowsFormsApplication55555

{

public partial class Form1 : Form

{

DataSet ds;

SqlDataAdapter adapter;

public string connectionString;

string tablProduct\_groups = "SELECT \* FROM Product\_groups";

string tablProducts = "SELECT \* FROM Products";

string tablStorage = "SELECT \* FROM Storage";

string tablСonsumption = "SELECT \* FROM Сonsumption";

public Form1()

{

InitializeComponent();

Fill();

}

private void Form1\_Load(object sender, EventArgs e)

{

this.Text="Підключення до бази даних";

this.Width = 400;

this.Height = 300;

dataGridView1.Visible = false;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

dataGridView4.Visible = false;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView4.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

tabControl1.Visible = false;

label1.Text = "Для підключення до SQL Server введіть свій логін і пароль";

label2.Text = "Логін";

label3.Text = "Пароль";

label4.Text = "База даних магазину";

button1.Text = "Приєднатись";

textBox2.PasswordChar = '\*';

}

private void button1\_Click(object sender, EventArgs e)

{

connectionString = "Data Source=91.225.200.84,1433;Initial Catalog="+textBox3.Text+";User ID="+textBox1.Text+";Password="+textBox2.Text; connectionString = "Data Source=91.225.200.84,1433;Initial Catalog="+textBox3.Text+";User ID="+textBox1.Text+";Password="+textBox2.Text;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(tablProduct\_groups, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

adapter = new SqlDataAdapter(tablProducts, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView2.DataSource = ds.Tables[0];

adapter = new SqlDataAdapter(tablStorage, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView3.DataSource = ds.Tables[0];

adapter = new SqlDataAdapter(tablСonsumption, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView4.DataSource = ds.Tables[0];

}

this.Width = 1000;

this.Height = 550;

dataGridView1.Visible = true;

dataGridView2.Visible = true;

dataGridView3.Visible = true;

dataGridView4.Visible = true;

tabControl1.Visible = true;

label1.Visible = false;

label2.Visible = false;

label3.Visible = false;

label4.Visible = false;

textBox1.Visible = false;

textBox2.Visible = false;

textBox3.Visible = false;

button1.Visible = false;

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 fr2 = new Form2();

fr2.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

Fill();

}

private void Fill()

{

FillSource(tablProduct\_groups, 0);

FillSource(tablProducts, 1);

FillSource(tablStorage, 2);

FillSource(tablСonsumption, 3);

}

private void FillSource(string tabl, int a)

{

connectionString = "Data Source=91.225.200.84,1433;Initial Catalog = KursovaGrinchak ;User ID= Hrynchak ; Password= 123456\*";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(tabl, connection);

ds = new DataSet();

adapter.Fill(ds);

switch (a)

{

case 0:

dataGridView1.DataSource = ds.Tables[0];

break;

case 1:

dataGridView2.DataSource = ds.Tables[0];

break;

case 2:

dataGridView3.DataSource = ds.Tables[0];

break;

case 3:

dataGridView4.DataSource = ds.Tables[0];

break;

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

DialogResult res = MessageBox.Show("Чи бажаєте вийти з програми?", "Вийти з програми?", MessageBoxButtons.YesNoCancel);

if (res == DialogResult.Cancel) MessageBox.Show("Ви залишились в програмі");

if (res == DialogResult.Yes) Close();

}

static SqlConnection con = new SqlConnection("Data Source = 91.225.200.84,1433; Initial Catalog = KursovaGrinchak; Persist Security Info = True; User ID = Hrynchak; Password = 123456\*");

static SqlCommand cmd = new SqlCommand("select \* from Product\_groups ", con);

static SqlCommand cmd1 = new SqlCommand("select \* from Products ", con);

static SqlCommand cmd2 = new SqlCommand("select \* from Storage ", con);

static SqlCommand cmd3 = new SqlCommand("select \* from Сonsumption ", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

SqlDataAdapter da1 = new SqlDataAdapter(cmd1);

SqlDataAdapter da2 = new SqlDataAdapter(cmd2);

SqlDataAdapter da3 = new SqlDataAdapter(cmd3);

private void GetData()

{

dataGridView1.Visible = true;

dataGridView2.Visible = true;

dataGridView3.Visible = true;

dataGridView4.Visible = true;

da.Fill(ds, "Product\_groups");

da1.Fill(ds, "Products");

da2.Fill(ds, "Storage");

da3.Fill(ds, "Сonsumption");

dataGridView1.DataSource = ds.Tables["Product\_groups"];

dataGridView2.DataSource = ds.Tables["Products"];

dataGridView3.DataSource = ds.Tables["Storage"];

dataGridView4.DataSource = ds.Tables["Сonsumption"];

MessageBox.Show("Данні отримано");

}

private void button4\_Click(object sender, EventArgs e)

{

SqlCommandBuilder cmd = new SqlCommandBuilder(da);

SqlCommandBuilder cmd1 = new SqlCommandBuilder(da1);

SqlCommandBuilder cmd2 = new SqlCommandBuilder(da2);

SqlCommandBuilder cmd3 = new SqlCommandBuilder(da3);

try

{

da.Update(ds, "Product\_groups");

da1.Update(ds, "Products");

da2.Update(ds, "Storage");

da2.Update(ds, "Сonsumption");

MessageBox.Show("Данні збережено");

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

}

private void button6\_Click(object sender, EventArgs e)

{

GetData();

}

private void dataGridView1\_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)

{

DialogResult dr = MessageBox.Show("Видалити запис?", "Видалення", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button3);

if (dr == DialogResult.Cancel)

{

e.Cancel = true;

}

}

private void dataGridView2\_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)

{

DialogResult dr = MessageBox.Show("Видалити запис?", "Видалення", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button3);

if (dr == DialogResult.Cancel)

{

e.Cancel = true;

}

}

private void dataGridView3\_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)

{

DialogResult dr = MessageBox.Show("Видалити запис?", "Видалення", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button3);

if (dr == DialogResult.Cancel)

{

e.Cancel = true;

}

}

private void dataGridView4\_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)

{

DialogResult dr = MessageBox.Show("Видалити запис?", "Видалення", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button3);

if (dr == DialogResult.Cancel)

{

e.Cancel = true;

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApplication55555

{

public partial class Form2 : Form

{

public string connectionString = " Data Source=91.225.200.84,1433;Initial Catalog=KursovaGrinchak;User ID=Hrynchak; Password=123456\* ";

public Form2()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

InsertAuto(textBox2.Text);

}

public void InsertAuto(string groupname)

{

SqlConnection conn = new SqlConnection(connectionString);

conn.Open();

// Оператор SQL

string sql = string.Format("Insert Into Product\_groups" +

"(GroupName)" +

"Values (@GroupName)");

using (SqlCommand cmd = new SqlCommand(sql, conn))

{

cmd.Parameters.AddWithValue("@GroupName", groupname);

cmd.ExecuteNonQuery();

MessageBox.Show("Дані додано в таблицю");

}

conn.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

InsertAuto2(Convert.ToInt32(numericUpDown2.Value), textBox5.Text, Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown3.Value), Convert.ToInt32(numericUpDown4.Value));

}

public void InsertAuto2(int groupid, string name, int unit\_of\_measure, int purchase\_price, int selling\_price)

{

SqlConnection conn = new SqlConnection(connectionString);

conn.Open();

string sql = string.Format("Insert Into Products" +

"(GroupID,Name,Unit\_of\_measure, Purchase\_price, Selling\_price)" +

"Values (@GroupID, @Name, @Unit\_of\_measure, @Purchase\_price, @Selling\_price)");

using (SqlCommand cmd = new SqlCommand(sql, conn))

{

cmd.Parameters.AddWithValue("@GroupID", groupid);

cmd.Parameters.AddWithValue("@Name", name);

cmd.Parameters.AddWithValue("@Unit\_of\_measure", unit\_of\_measure);

cmd.Parameters.AddWithValue("@Purchase\_price", purchase\_price);

cmd.Parameters.AddWithValue("@Selling\_price", selling\_price);

cmd.ExecuteNonQuery();

MessageBox.Show("Дані додано в таблицю");

}

conn.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

InsertAuto3(Convert.ToInt32(numericUpDown5.Value),Convert.ToInt32(numericUpDown6.Value), Convert.ToInt32(numericUpDown7.Value),textBox12.Text);

}

public void InsertAuto3(int productid, int purchase\_price, int selling\_price, string numberofsales)

{

SqlConnection conn = new SqlConnection(connectionString);

conn.Open();

string sql = string.Format("Insert Into Storage" +

"(ProductID,Purchase\_price, Selling\_price, NumberOfSales)" +

"Values (@ProductID, @Purchase\_price, @Selling\_price, @NumberOfSales)");

using (SqlCommand cmd = new SqlCommand(sql, conn))

{

cmd.Parameters.AddWithValue("@ProductID", productid);

cmd.Parameters.AddWithValue("@Purchase\_price", purchase\_price);

cmd.Parameters.AddWithValue("@Selling\_price", selling\_price);

cmd.Parameters.AddWithValue("@NumberOfSales", numberofsales);

cmd.ExecuteNonQuery();

MessageBox.Show("Дані додано в таблицю");

}

conn.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

InsertAuto4(Convert.ToInt32(numericUpDown8.Value), Convert.ToInt32(numericUpDown9.Value), Convert.ToInt32(numericUpDown10.Value), dateTimePicker1.Value.ToString("dd-MM-yyyy"));

}

public void InsertAuto4(int checkid, int productid, int count, string dateofcheck)

{

SqlConnection conn = new SqlConnection(connectionString);

conn.Open();

string sql = string.Format("Insert Into Сonsumption" +

"(CheckID, ProductID, Count, DateOfCheck)" +

"Values (@CheckID, @ProductID, @Count, @DateOfCheck)");

using (SqlCommand cmd = new SqlCommand(sql, conn))

{

cmd.Parameters.AddWithValue("@CheckID", checkid);

cmd.Parameters.AddWithValue("@ProductID", productid);

cmd.Parameters.AddWithValue("@Count", count);

cmd.Parameters.AddWithValue("@DateOfCheck", dateofcheck);

cmd.ExecuteNonQuery();

MessageBox.Show("Дані додано в таблицю");

}

conn.Close();

}

}

}