Міністерство освіти і науки України

Черкаський національний університет імені Богдана Хмельницького

Факультет обчислювальної техніки, інтелектуальних та управляючих систем

Кафедра Програмного забезпечення та автоматизованих систем

КУРСОВА РОБОТА

**з дисципліни**

**ОРГАНІЗАЦІЯ БАЗ ДАНИХ І ЗНАНЬ**

на тему: «Інформаційна система фермерського господарства»

|  |  |  |  |
| --- | --- | --- | --- |
| Студента(ки) | 3 | курсу групи | КС-20 |
| спеціальності | «121 – Інженерія | | |
| програмного забезпечення» | | | |
| Царенко Р.М. | | | |
| (прізвище та ініціали) | | | |

|  |  |
| --- | --- |
| Керівник | доцент, к.т.н. |
| Гребенович Ю.Є. | |
| (посада, вчене звання, науковий ступінь, прізвище та ініціали ) | |

|  |  |  |  |
| --- | --- | --- | --- |
| Оцінка: | | |  |
| за університетською шкалою | | |  |
| за шкалою ECTS |  | |  |
| за національно шкалою | |  |  |

Члени комісії

|  |  |  |
| --- | --- | --- |
| (підпис) |  | (прізвище та ініціали) |
|  |  |  |
| (підпис) |  | (прізвище та ініціали) |
|  |  |  |
| (підпис) |  | (прізвище та ініціали) |

м. Черкаси – 2023 рік

Зміст

[Вступ 3](#_Toc138643083)

[1. Аналіз предметної області 4](#_Toc138643084)

[1.1. Огляд та порівняння аналогів об’єкта розробки. 4](#_Toc138643085)

[1.2. Постановка задач 6](#_Toc138643086)

[2. Проектування бази даних 8](#_Toc138643087)

[2.1. Інфологічне та даталогічне проектування 8](#_Toc138643088)

[2.1.1. Інфологічне проектування 8](#_Toc138643089)

[2.1.2. Даталогічне проектування 10](#_Toc138643090)

[2.2. Проектування серверної частини 16](#_Toc138643091)

[2.2.1. Таблиці та об’єкти бази даних 16](#_Toc138643092)

[2.2.2. Виконання запитів 24](#_Toc138643093)

[3. Опис клієнтського додатку 31](#_Toc138643094)

[Висновок 35](#_Toc138643095)

[Список інформаційних джерел 36](#_Toc138643096)

# **Вступ**

Інформаційні технології розвиваються дуже швидко. Вони допомагають в любій сфері життя людини. Починаючи від простого використання як бази знань і примітивних задач запису, запам’ятовування, і прискореного вирішення примітивних обрахунків, до, в нинішній час, обробки великої кількості інформації, обрахунку всіх можливих фізичних явищ природи, космічних тіл, а наданий момент нашумілу технологію нейронних мереж, яка майже миттєво генерує текст, картинку, музику, може керувати автомобілем, заводом, будь чим, дивлячись чому буде вона навчена.

Так само і сфера сільського господарства в нинішній час впроваджує нові технології, щоб полегшити собі роботу, оптимізувати процеси, мати потрібну інформацію в одному місці, і до якої є швидкий доступ, і зручний інтерфейс.

**Метою, а слідом і темою моєї курсової є**: «Інформаційна система фермерського господарства», потенційно, це програма, загального обліку фермерського господарства.

А отже під час виконання курсової роботи потрібно буде проаналізувати предметну область, пояснити, чому ця тема має актуальність, проявити навички, які здобув під час вивчення цього курсу, щоб побудувати згодом правильну БД.

Застосувати навички попередніх курсів, щоб мав логічність інтерфейс, і реалізувати повноцінну програму, з підтримкою БД.

# **1. Аналіз предметної області**

Отже першим пунктом є аналіз предметної області, в даному випадку в нас «Інформаційна система фермерського господарства». Для такої структури, потрібно враховувати велику кількість даних, оскільки фермерські господарства в нинішній час, це структури хоч і різного масштабу, але мають під собою різні підрозділи, структури, різний набір компаній партнерів. Це складна система з великою кількістю процесів.

У кожної ферми також і різні пріоритети. Якщо взяти в загальному, то фермерське господарство займається як насадженнями різної продукції, так і вирощуванням тварин, облік продукції яку має ферма для подальшого експорту, звітність цього експорту, що вона імпортувала до себе, дані з ким, співпрацюють.

Цю велику кількість даних, є потреба структурувати, щоб потім мати, майже завжди, актуальні дані, швидкий доступ до цих даних, і зручність керування цими даними.

## **1.1. Огляд та порівняння аналогів об’єкта розробки.**

Дивлячись на ситуацію, зазвичай фермерські господарства для зберігання даних використовували, великі зошити з розміром листа А4, або спеціальні альбоми, вже з зарані розкресленими, або роздрукованими, таблицями, це те, з чого розпочинали, і навіть місцями використовують і зараз, для локальних, невеликих господарств. Через певний час, передові ферми, з появою перших доступних комп’ютерів і інтуїтивного приємного інтерфейсу, почали користуватися звичайними текстовими документами, а згодом і таблицями. Це звісно спростило задачу деякого обрахунку, збереження даних, але не сильно, оскільки більшість даних могли бути в різних документах, що забирало час на пошук потрібної інформації, або її зміну в різних документах, їх можна було легко втратити.

Зараз основними програмами обліку так і залишаються програми типу Word, для збереження загальної інформації. Також Excel – це вже більш професійна програма, яка в загальному задовольняє потреби невеликих ферм (Приклади на рис. 1.1.1 та 1.1.2 ). Більші ж структури замовляють собі кастомне ПО, або більш «програмовану» Excel таблицю за вимогами ферми, або використовують 1С.

Дякуючи одному фермерському господарству, можна побачити, приблизно які дані, вони зберігають, для рослинництва, які фінансові дані вони зберігають. Справді існуючої програми, розглянути неможливо, оскільки кожна структура замовляє, таблицю, ПО, під свої потреби, а надавати детальний приклад якась структура навряд-чи буде, а якщо і буде, то нажаль виходів на неї немає, оскільки в більшості це може бути корпоративною таємницею.

А отже написання програми, і створення зручної БД, є актульною.

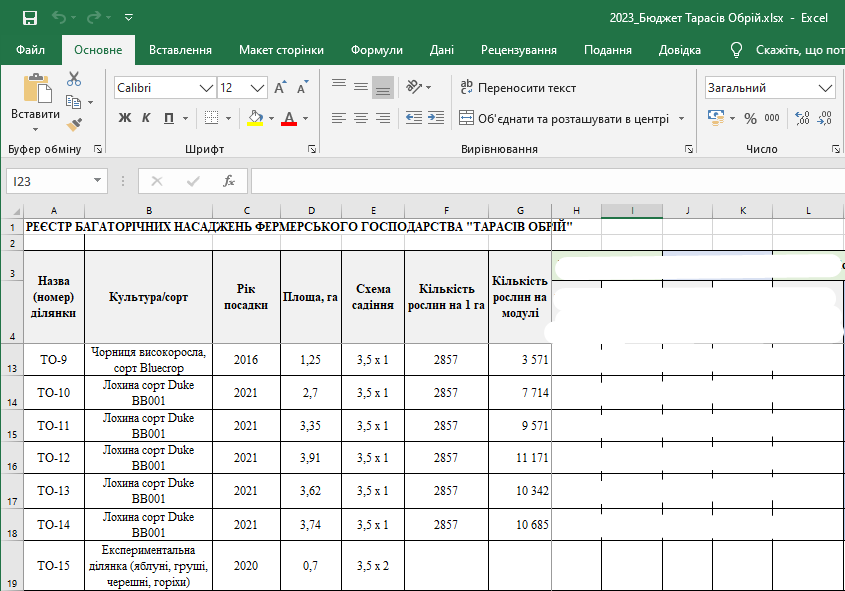


Рис 1.1.1 – приклад Excel таблиці фермерського господарства

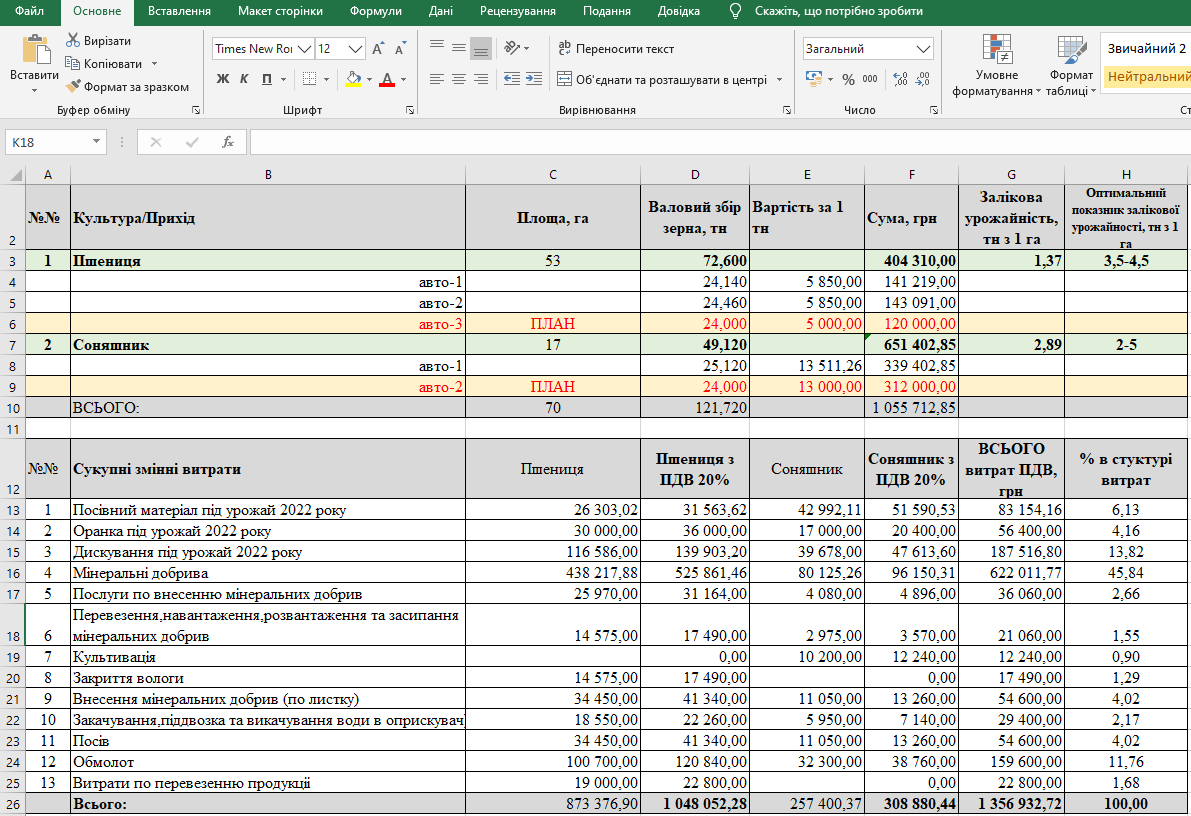


Рис 1.1.2 – приклад Excel таблиці фермерського господарства

## **1.2.** **Постановка задач**

І все ж, для початку, постає питання, що потрібно враховувати на фермерському господарстві? Фермерів які б проінструктували, нажаль немає, тому виокремлено загально потрібні розділи даних, яким потрібен облік, до яких потрібен швидкий доступ та структурність, ці дані будуть розбиті на декілька таблиць, а також для них буде розроблено графічний інтерфейс, відповідно до пунктів.

Розглянемо пункти:

1. Провівши аналіз предметної області можна виділити основні категорії, яким потрібен облік та БД: облік адрес компаній партнерів, та облік адрес підрозділів ферми, облік продуктів експорту та імпорту, робітників, процесу посіву вирощування і тд.

Виходячи з вище написаного, під наші цілі, підходить SQLite – оскільки це реляційна база даних, яку і потрібно використати, а також зберігає БД (дані) локально, в окремому файлі, що не потребує додаткових ресурсів людей які будуть використовувати програму, а також це спростить трішки розробку, щоб уникнути непотрібної тяганини з окремим сервером.

2. Розробити програмний додаток та його інтерфейс. Для його розробки буде використано мову програмування C#, та фреймворк WPF, обрані дані технології були по причині, що вони вже використовувалися і знайомі, що дозволить швидше виконати поставленні задачі, а також ці технології дадуть нам пов’язати БД з програмним додатком (інтерфейсом), і головний плюс, що це універсальні технології для написання додатків для Windows.

Далі перейдемо до більш детальнішого розгляду зовнішнього вигляду програмного додатку, потрібно буде реалізувати 3 вкладки:

2.1. Перша та друга вкладки повинні мати бокову панель, яка дозволить нам переключатися між таблицями, збоку від цієї панелі, блок, де будуть відображатися самі таблиці. Також за задумом, біля записів повинна розміщатися кнопка видалення запису. А також кнопка додавання запису, скоріш всього над таблицею. Чому дві вкладки, пояснення доволі просте, потрібно буде розділити таблиці, які стосуються тільки ферми, і таблиці які відносяться до фінансової частини ферми та співробітництва з компаніями партнерами.

2.2. Третя вкладка, це просто функціональна вкладка з рядком для користувацького запиту, якщо користувачу буде потрібно щось нестандартне.

На цьому етапі загальна постановка задач завершена, переходимо до більш поглибленого проектування БД і розширення задач, які буде потрібно виконати в наступному пункті.

# **2. Проектування бази даних**

## **2.1. Інфологічне та даталогічне проектування**

### **2.1.1. Інфологічне проектування**

Логічним завершенням аналізу предметної області, є те, що тепер ми можемо детально розглянути максимально потрібні сутності для майбутньої БД, для початку потрібно спроектувати ER-діаграму яка зобразить взаємодію сутностей, полів і допоможе нам створити БД.

Головні, а отже наймісткіші сутності можна виділити такі:

1. Тварини (Animals);

Ця сутність повинна вміщати тип тварини (наприклад корова, куриця, індик, і тп.), стать, кількість, де розміщені, споживання. З цього переліку, випливає, що нам потрібна ще одна сутність *Animal Type*, щоб потім, у випадку, коли ферма не буде працювати з певними типами тварин, просто видалити цей тип, і він автоматично видалився з основної сутності *Animals*.

2. Насадження (Plantation);

Ця сутність повинна вміщати, назву насадження, де посаджене, кількість насаджень, у випадку якщо це кущ або саджанець, а якщо порахувати не можливо, то вказувати площу насаджень, і дату посадки (посіву і тп.)

Тепер, що стосується розміщення, було прийнято рішення, зробити 4 сутності. Загальну, під назвою *Location Type*, для загальних назв територій:  магазин, оптова база, ділянка землі, склад. Оскільки всі вище перераховані назви можуть бути використані, як і в частинах землі які відносяться тільки до ферми, і можуть бути використані як для експорту так і для імпорту. Тому, щоб було зручно, і не було зайвої інформації в інших сутностях (таблицях), створено 3 сутності *Land Addresses, Export Addresses, Import Addresses*, зробивши це, ми уникаємо ситуації, коли всі адреси, були б записані в одній таблиці, щоб створило плутанину. У всіх трьох сутностях , зберігатиметься адреса, id загальної назви локації з таблиці *Location Type*. А тепер відмінності: в *Land Addresses* буде зберігатися назва підрозділу ферми, в *Export Addresses, Import Addresses*  будуть зберігатися назви компаній, магазині і тп., з якими співпрацює ферма.

3. Експорт (Export);

Ця сутність, повинна зберігати велику кількість інформації, я то лот який продано, тип експорту (в нашому випадку продукту нехай то, м'ясо, фрукти, овочі, ягоди і тп.), куди експортувалося, назва що продали (свинина, огірок, помідор, яловичина), кількість в кілограмах, ціна за весь лот, і дата продажу. Потенційно ці всі поля потрібні, щоб в майбутньому була можливість проаналізувати продажі за рік до прикладу, або підбити різні підсумки які захоче ферма самостійно. Також варто зазначити, що така сутність не повинна бути прив’язана до інших сутностей, оскільки вона є загальною історією (обліком), всього експорту ферми.

З описаного, випливає, що потрібно ще 2 сутності — це *Export Type* та *Export Warehouse*. Для першої сутності, причина створення така сама як і для сутності *Animal Type*, щоб у випадку, коли ферма перестане виробляти той чи інший тип продукції, її просто можна було б видалити, і видалення в одній таблиці видалить пов’язані записи з інших таблиць. Що до другої сутності, це загальний склад, який буде зберігати, всю можливу для продажу продукцію. Ця сутність буде зберігати майже всю ту саму інформацію, але з деякими поправками, ціна буде вказана за кілограм, щоб замовник, міг оцінювати, що йому вигідніше, кількість буде змінною, оскільки продукт може псуватися.

4. Імпорт (Import);

Сутність імпорту, як і експорту має той самий сенс, зберігати дані, для подальшого звіту, аналізу, або що тому ця сутність теж відрізана від зав’язків. Поля ті самі, тип імпорту (деталі, інструменти, насіння, саджанці, яди, добрива і тп.), назва імпорту (лопата «Копатор 3000», мінеральне добриво N, і тп.), хто імпортував, скільки заплатили, і коли імпортовано. Також додатково створено *Import Type*, з тих самих причин що і *Export Type,* і для додаткової зручності.

5. Робітники (Employees);

І остання сутність, як же не вести облік тих, хто працює у тебе на фермі. Вона вміщує такі дані: повне ім’я, адреса проживання робітника, номер телефону, на якому підрозділі (ділянці) ферми працює, і також тип його роботи.

Як наслідок, має ще одну сутність під назвою Type Of Work, ця сутність матиме назву (тип, професія) роботи, а також буде вказуватися заробітна плата.

Виходячи з аналізу вище, маємо таку ER-діаграму або ж інфологічну модель БД, яка зображена на рисунку 2.1.1.1.

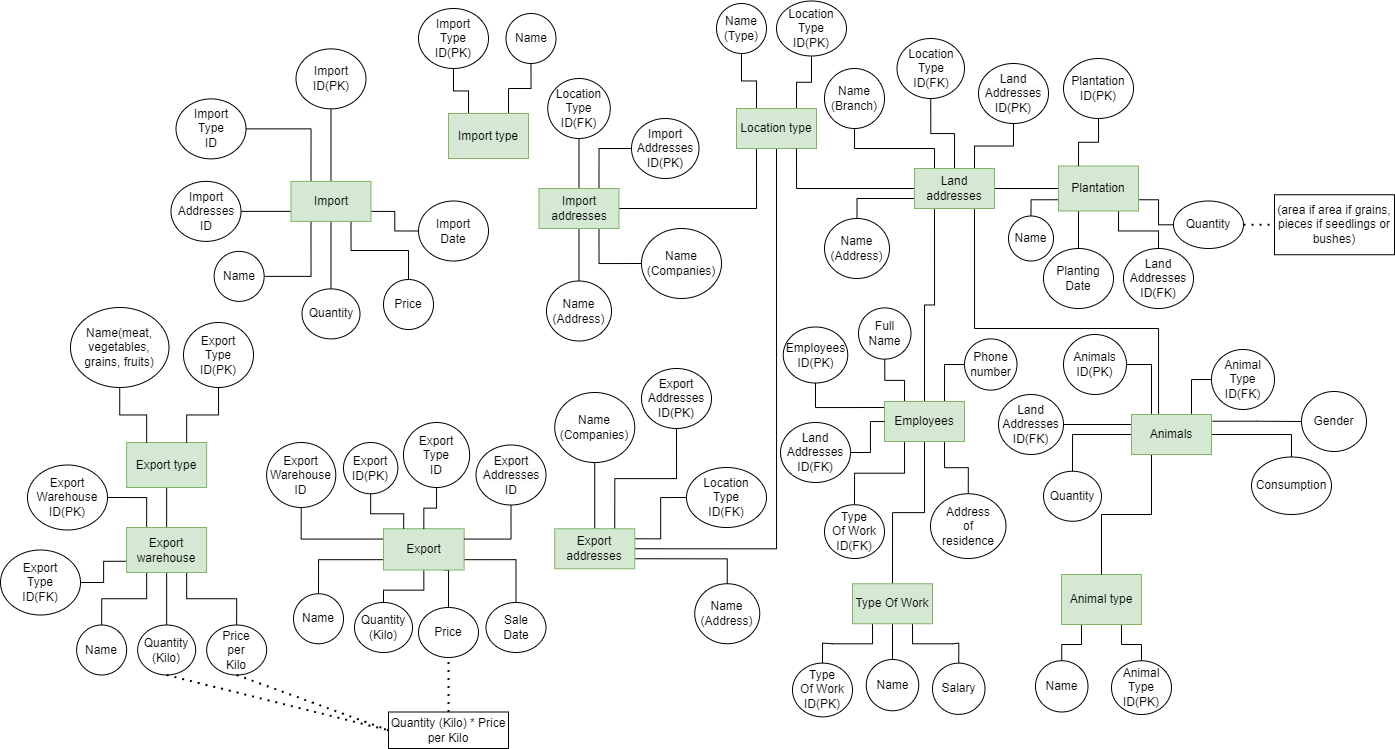


Рис 2.1.1.1 – ER- діаграма або інфологічна модель.

### **2.1.2. Даталогічне проектування**

Основну частину я описав при інфологічному проектуванні, тому під час даталогічної, в основному буде сухий опис, з усіма полями, типами даних для цих полів, і назвами самих таблиць відповідно.

1. Таблиця Location\_Type:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;

2. Таблиця Land\_Addresses:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле location\_type\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Location\_Type;
* Поле branch\_name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле address – тип даних TEXT, має додаткову умову NOT NULL;

3. Таблиця Type\_Of\_Work:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле salary – тип даних INTEGER, має додаткову умову NOT NULL;

4. Таблиця Employees:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле type\_of\_work\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Type\_Of\_Work;
* Поле land\_addresses\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Land\_Addresses;
* Поле full\_name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле address\_of\_residence – тип даних TEXT, має додаткову умову NOT NULL;
* Поле phone\_number – тип даних TEXT, має додаткову умову NOT NULL;

5. Таблиця Animal\_Type:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;

6. Таблиця Animals:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле land\_addresses\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Land\_Addresses;
* Поле animal\_type\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Animal\_Type;
* Поле gender – тип даних TEXT, має додаткову умову NOT NULL;
* Поле quantity – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле consumption – тип даних INTEGER, має додаткову умову NOT NULL;

7. Таблиця Plantation:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле land\_addresses\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Land\_Addresses;
* Поле plantation\_type\_name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле quantity – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле planting\_date – тип даних TEXT, має додаткову умову NOT NULL;

8. Таблиця Export\_Addresses:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле location\_type\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Location\_Type;
* Поле name\_companies – тип даних TEXT, має додаткову умову NOT NULL;
* Поле address – тип даних TEXT, має додаткову умову NOT NULL;

9. Таблиця Export\_Type:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;

10. Таблиця Export\_Warehouse

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле export\_type\_id – тип даних INTEGER, має додаткову умову NOT NULL зовнішній ключ (FK) пов’язаний з таблицею Export\_Type;
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле quantity – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле price\_per\_kilo – тип даних INTEGER, має додаткову умову NOT NULL;

11. Таблиця Export:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле export\_addresses\_id – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле export\_warehouse\_id – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле export\_type\_id – ип даних INTEGER, має додаткову умову NOT NULL;
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле quantity – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле price – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле sale\_date – тип даних TEXT, має додаткову умову NOT NULL;

12. Таблиця Import\_Addresses:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле location\_type\_id – тип даних INTEGER, має додаткову умову NOT NULL, зовнішній ключ (FK) пов’язаний з таблицею Location\_Type;
* Поле company\_name (різність в назвах між name\_companies ті цією, можливо через те, що створення самої БД було в різні дні) – тип даних TEXT, має додаткову умову NOT NULL;
* Поле address – тип даних TEXT, має додаткову умову NOT NULL;

13. Таблиця Import\_Type:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;

14. Таблиця Import:

* Поле id – тип даних INTEGER, має додаткову умову автоінкременту та NOT NULL, тобто поле не може бути пустим, первинний ключ (PK);
* Поле import\_addresses\_id – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле import\_type\_id – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле name – тип даних TEXT, має додаткову умову NOT NULL;
* Поле quantity – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле price – тип даних INTEGER, має додаткову умову NOT NULL;
* Поле import\_date – тип даних TEXT, має додаткову умову NOT NULL;

Розписавши всі пункти, які потрібні, тепер ми можемо поглянути на даталогічну модель виконання(Рис. 2.1.2.1.), і перейти до проектування БД.

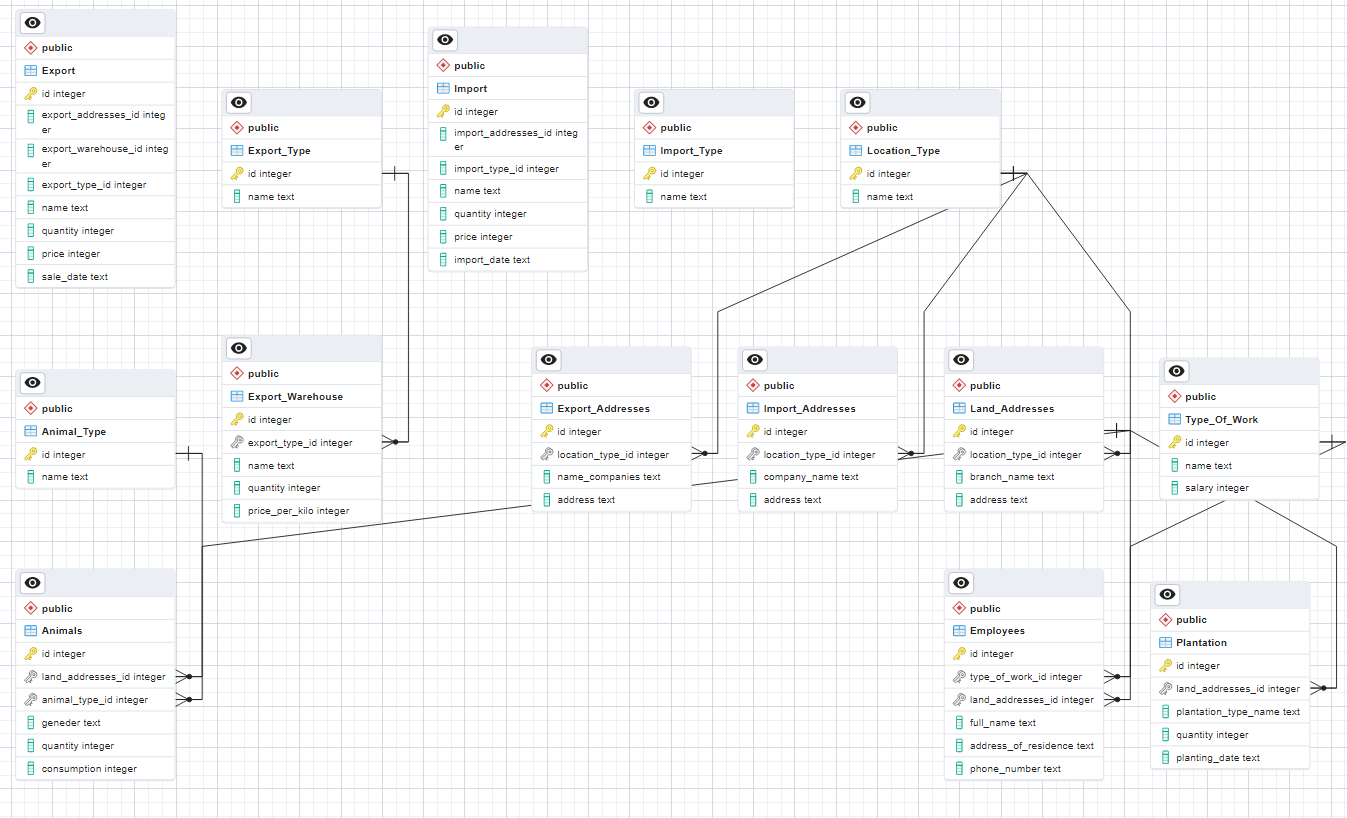


Рис. 2.1.2.1. Даталогічна модель

## **2.2. Проектування серверної частини**

В даному розділі, буде створимо БД по діаграмам вище, а також заповнено інформацією ці таблиці, щоб виконати запити, які вказані як одне із завдань до курсової роботи.

### **2.2.1. Таблиці та об’єкти бази даних**

Почну показувати таблиці в порядку, який був розписаний вище.

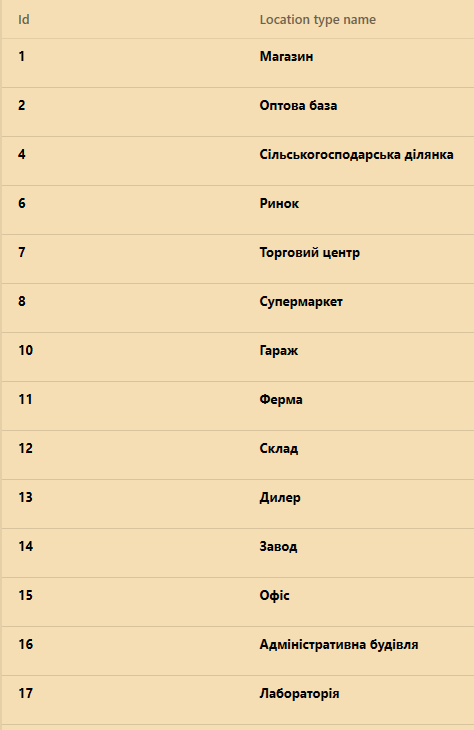


Рис. 2.2.1.1. Таблиця Location\_Type



Рис. 2.2.1.2. Таблиця Land\_Addresses



Рис. 2.2.1.3. Таблиця Type\_Of\_Work

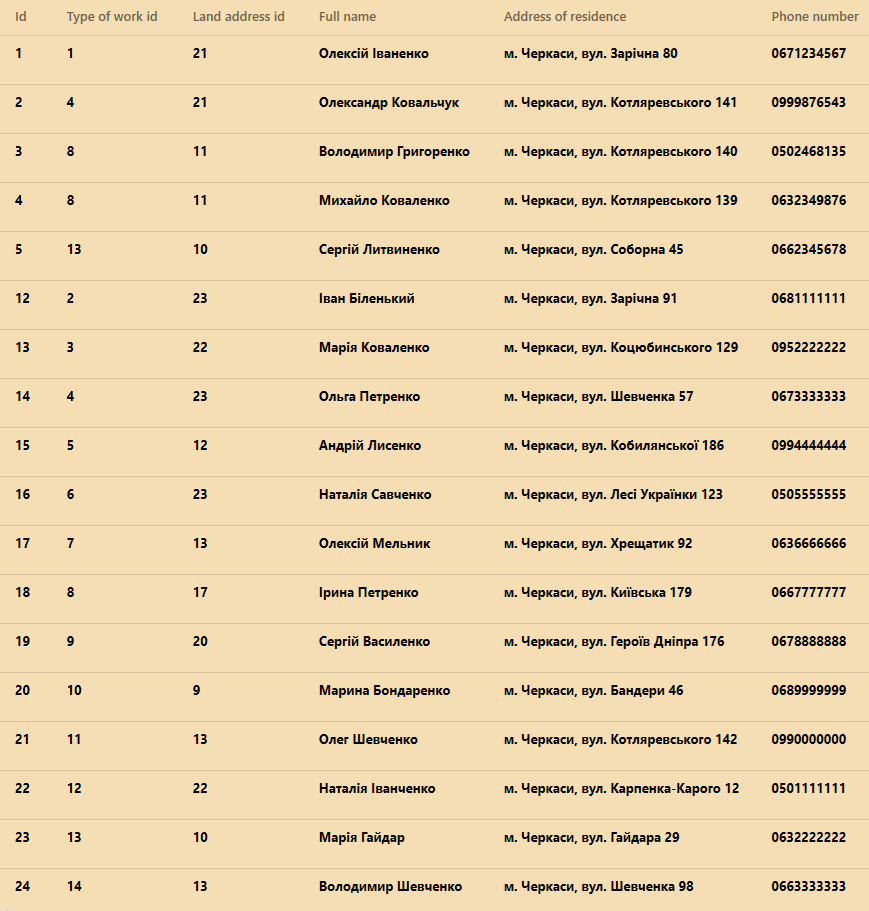


Рис. 2.2.1.4. Таблиця Employees



Рис. 2.2.1.5. Таблиця Animal\_Type

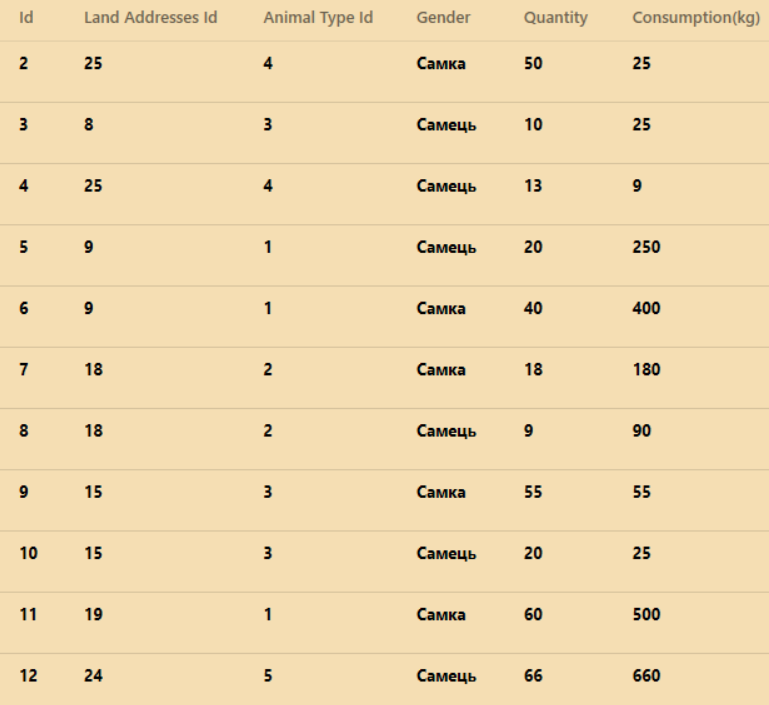


Рис. 2.2.1.6. Таблиця Animals



Рис. 2.2.1.7. Таблиця Plantation



Рис. 2.2.1.8. Таблиця Export\_Addresses

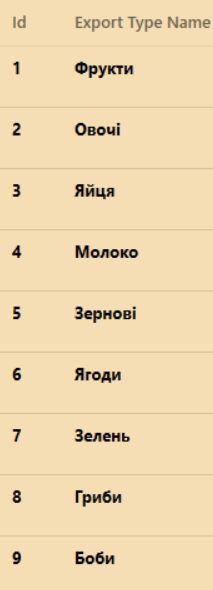


Рис. 2.2.1.9. Таблиця Export\_Type

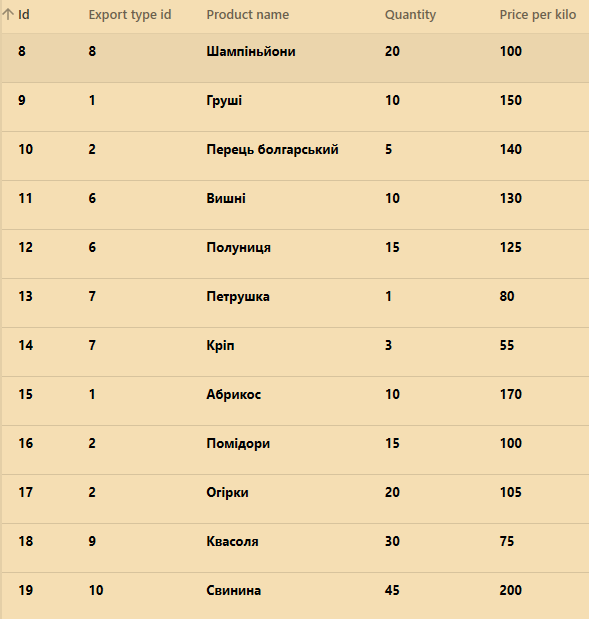


Рис. 2.2.1.10. Таблиця Export\_Warehouse



Рис. 2.2.1.11. Таблиця Export

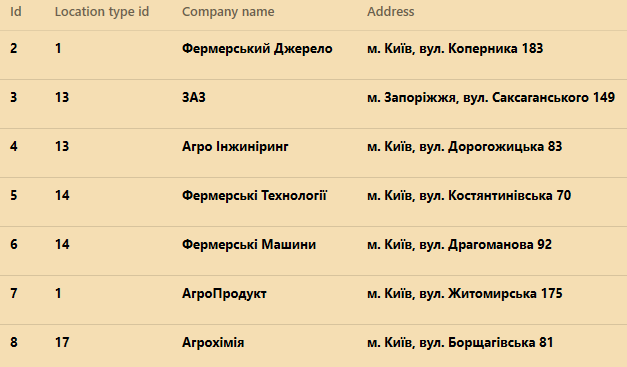


Рис. 2.2.1.12. Таблиця Import\_Addresses



Рис. 2.2.1.13. Таблиця Import\_Type



Рис. 2.2.1.14. Таблиця Import

### **2.2.2. Виконання запитів**

Перейдемо до виконання запитів. Вигляд буде як і в попередньому розділі, рисунок і результат виконання на скріншоті.

Запит який виведе всі дані таблиці Employees;

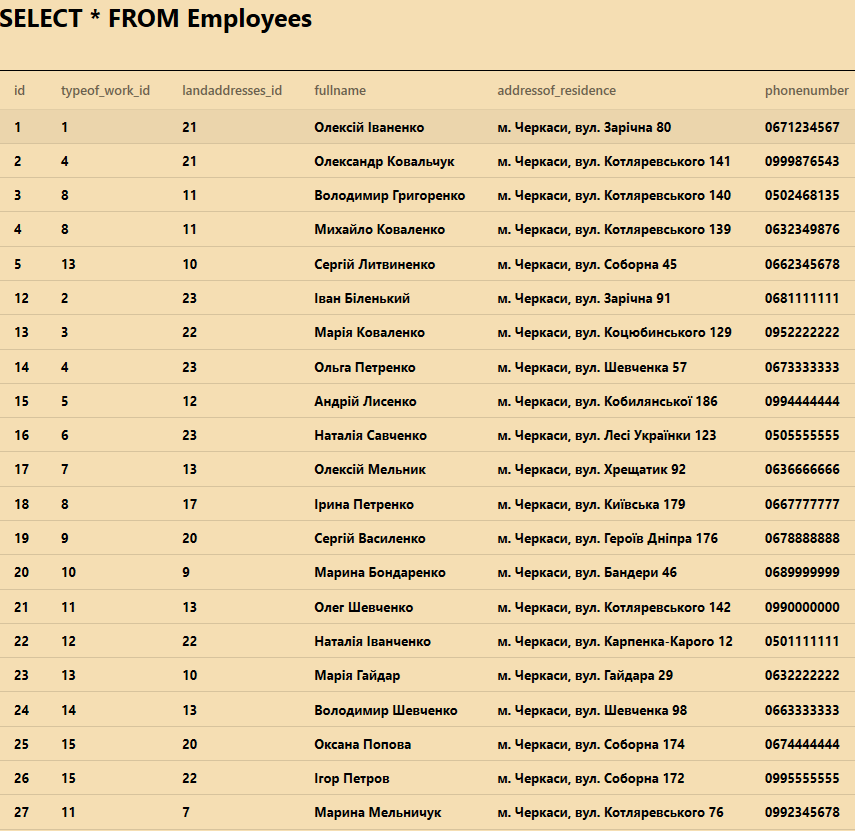


Рис. 2.2.2.1. Простий запит на вибірку

Цей запит вибирає всі професії, де заробітна плата знаходиться в діапазоні від 12000 до 15000.

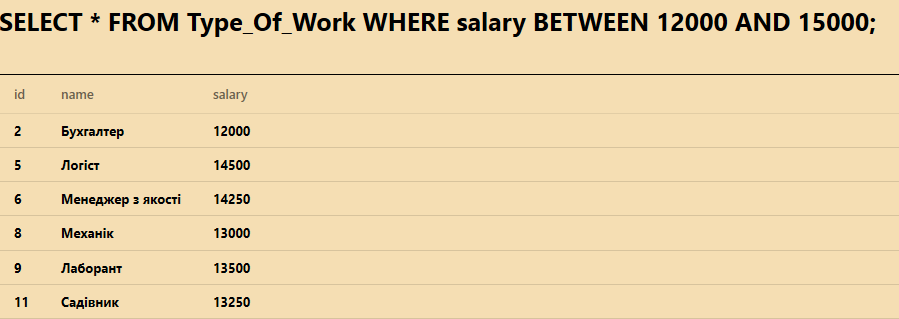


Рис. 2.2.2.2. Запит на вибірку з використанням «between....and»

Запит який виводить робітників, у яких type\_of\_work\_id 1,8,5.

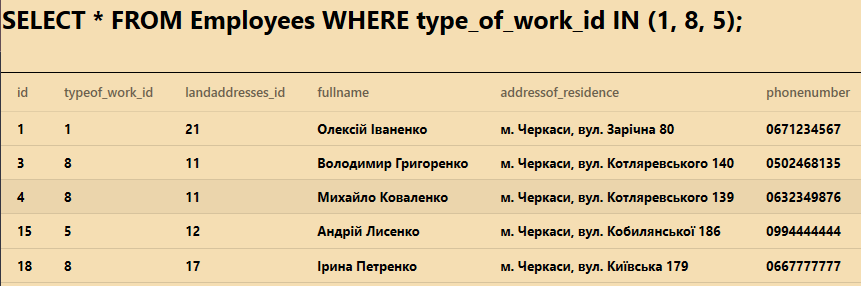


Рис. 2.2.2.3. Запит на вибірку з використанням «in»

Запит який виводить робітників, які проживають на вулиці «Котляревського»



Рис. 2.2.2.4. Запит на вибірку з використанням «like»

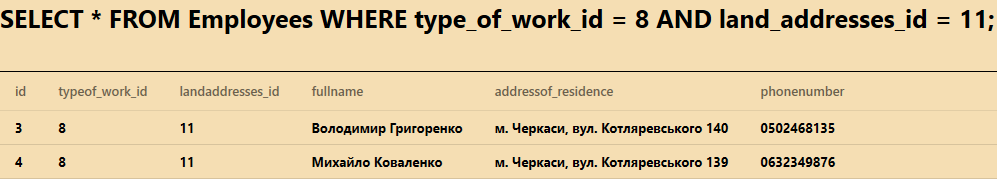


Рис. 2.2.2.5. Запит на вибірку з двома умовами через «and»

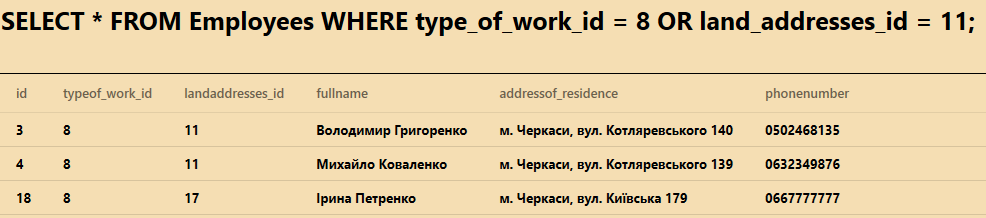


Рис. 2.2.2.6. Запит на вибірку з двома умовами через «or»

Цей запит виводить кількість унікальних значень land\_addresses\_id, в таблиці employees;

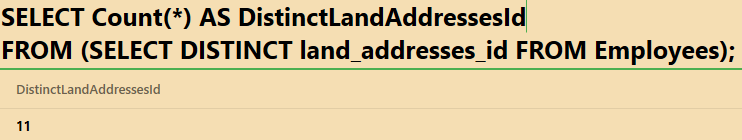


Рис. 2.2.2.7. Запит на вибірку з використанням «DISTINCT»

У цьому запиті ми об'єднали таблиці Land\_Addresses і Employees за допомогою умови Land\_Addresses.id = Employees.land\_addresses\_id. Потім ми вибрали поле branch\_name з таблиці Land\_Addresses і використали агрегатну функцію COUNT(\*), щоб підрахувати кількість співробітників для кожного branch\_name. Далі, за допомогою умови HAVING employees\_count > 2, ми обмежили результати вибірки, вибираючи тільки ті записи, для яких кількість співробітників більше 2.

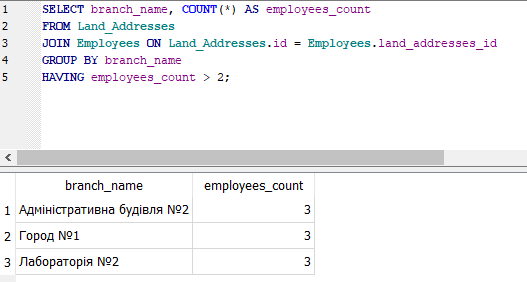


Рис. 2.2.2.8. Запит з функцією «count»

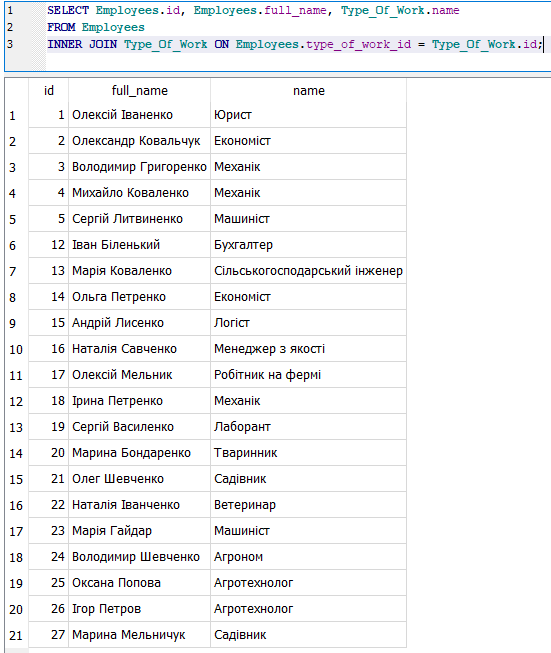


Рис. 2.2.2.9. Запит на вибірку з використанням агрегатної функції і умовою на вибірку поля

У цьому запиті вибираються імена працівників та назви типів робіт, які вони виконують. Використовується зв'язок INNER JOIN між таблицями "Employees" і "Type\_Of\_Work" на основі спільного поля "type\_of\_work\_id". Підзапит використовується для обмеження вибірки працівників, які працюють у "Адміністративна будівля №1" у таблиці "Land\_Addresses".

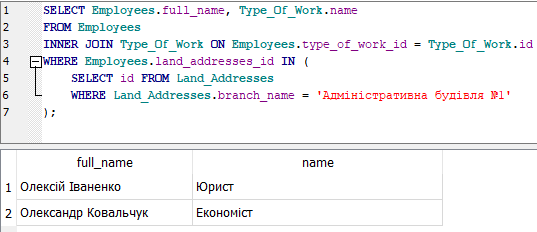


Рис. 2.2.2.10. Запит з використанням підзапита і зв’язку INNER JOIN

У цьому запиті вибираються всі записи з таблиці "Import", якщо значення поля "import\_addresses\_id" знаходиться серед значень поля "id" у таблиці "Import\_Addresses", де значення поля "location\_type\_id" рівне 17.

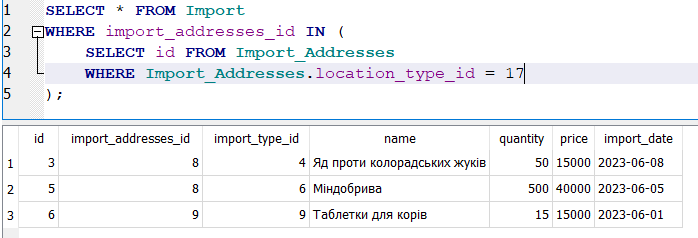


Рис. 2.2.2.11. Запит з використанням підзапита і зв’язку INNER JOIN

У цьому запиті вибираються всі записи з таблиці "Animals", для яких значення поля "quantity" більше ніж найбільше значення поля "quantity" для тварин з типом "animal\_type\_id" рівним 1.

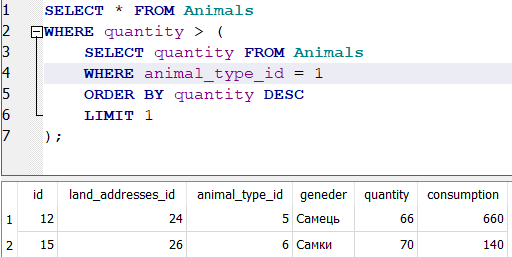


Рис. 2.2.2.12. Запит з використанням підзапита з використанням оператора (=, <, >):

У цьому запиті обирається середня ціна, для кожного типу експорту, де кількість більше 15, а також застосовується умова (HAVING), що обмежує результати, вибираючи лише ті записи, де середня ціна менше 5000. Результати сортуються за спаданням середньої ціни.

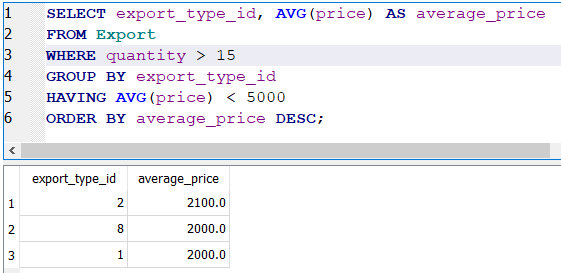


Рис. 2.2.2.13. Запит на вибірку з використанням агрегатної функції, умовою на агрегатну функцію, умовою на вибірку поля з сортуванням даних:

У цьому запиті вибирається максимальна кількість ("quantity") для кожного типу тварин ("animal\_type\_id") з таблиці "Animals". Застосовується умова (HAVING), що обмежує результати, вибираючи лише ті записи, де максимальна кількість перевищує 50.

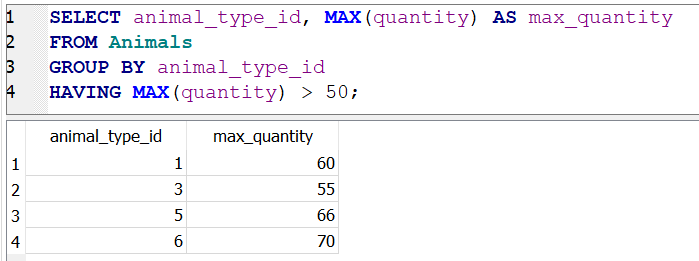


Рис. 2.2.2.14. Запит на вибірку з використанням агрегатної функції і умовою на агрегатну функцію

У цьому запиті використовується INNER JOIN для з'єднання таблиці "Export" з таблицею "Export\_Addresses" за допомогою умови E.export\_addresses\_id = EA.id. Запит вибирає поля E.id, E.name та EA.name\_companies. Умова WHERE EA.name\_companies LIKE '%АТБ%' фільтрує результати і вибирає лише ті записи, в яких поле EA.name\_companies містить підрядок "АТБ".

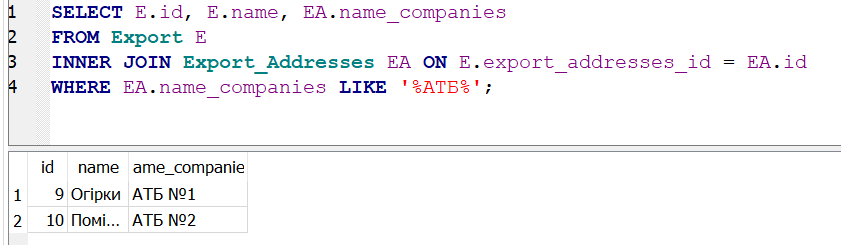


Рис. 2.2.2.15. Запит на вибірку з використанням агрегатної функції і умовою на агрегатну функцію

У цьому запиті використовується підзапит з оператором EXISTS. Зовнішній запит вибирає всі записи з таблиці "Export", для яких існує хоча б один запис в таблиці "Export\_Addresses", де поле export\_addresses\_id в таблиці "Export" співпадає з полем id в таблиці "Export\_Addresses" і поле name\_companies в таблиці "Export\_Addresses" має значення 'АТБ №1'. Таким чином, запит поверне всі записи з таблиці "Export", які мають принаймні один відповідний запис у таблиці "Export\_Addresses" зі значенням 'АТБ №1'.

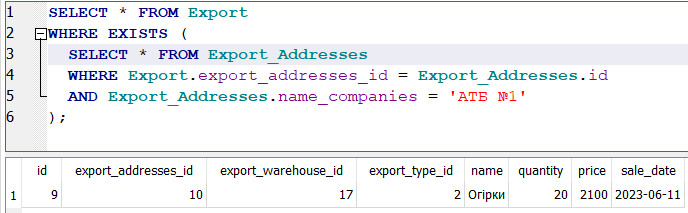


Рис. 2.2.2.16. Запит з використанням підзапита з використанням оператора EXIST.

# **3. Опис клієнтського додатку**

Основна частина опису, можна вважати була написана в **Розділі 1, пункті 1.2. Постановка задач**[**[5]**](#_1.2._Постановка_задач)**.** Відповідно в цьому пункті, будуть прикріплені скріншоти, і короткий опис до них.

Під час запуску, нас зустрічає головний екран (Рис. 3.1.). Як вже писалося, три вкладки, між якими можна переключатися та бокова панель, в перших 2 вкладках для перемикання, між таблицями.

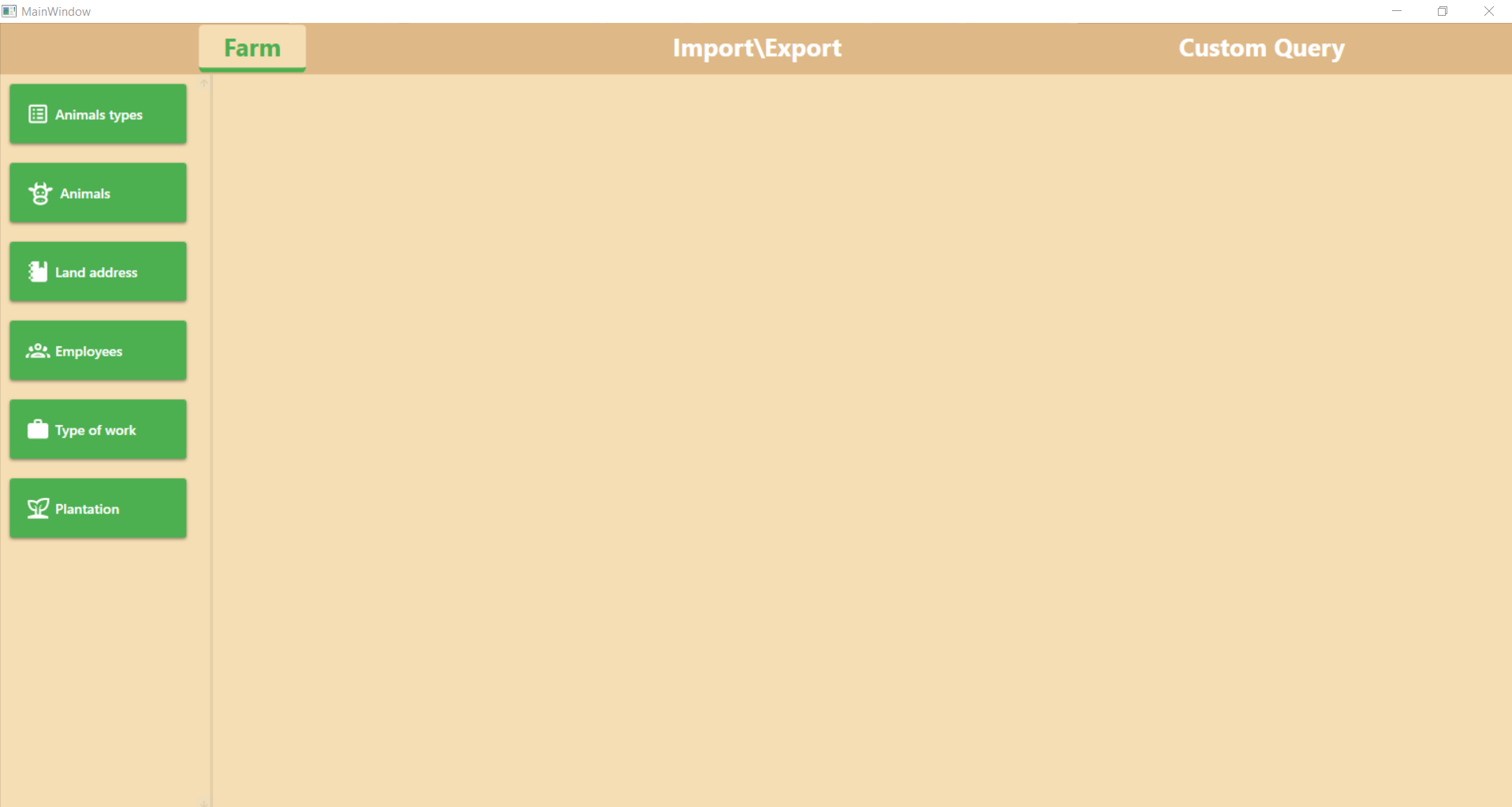


Рис 3.1. Головний екран

Такий вигляд має сторінка таблиці, в більшості, вони схожі, різниця тільки в кількості стовпчиків самої таблиці. А також зверху з’являється кнопка, для створення запису, для відповідної таблиці.



Рис 3.2. Вигляд таблиці Animals\_Type

Також варто згадати, що можна було бачити вигляд таблиць, раніше в **Розділі 2, пункті** **2.2.2. Таблиці та об’єкти бази даних**[**[6]**](#_2.2.1._Таблиці_та)



Рис 3.3. Вигляд таблиці Export

Меню створення запису, має поля для вводу (TextBox), а де є зовнішні ключі в таблицях, то під час створення є спадаючого меню (ComboBox), в те меню підтягуються дані, для зручності.

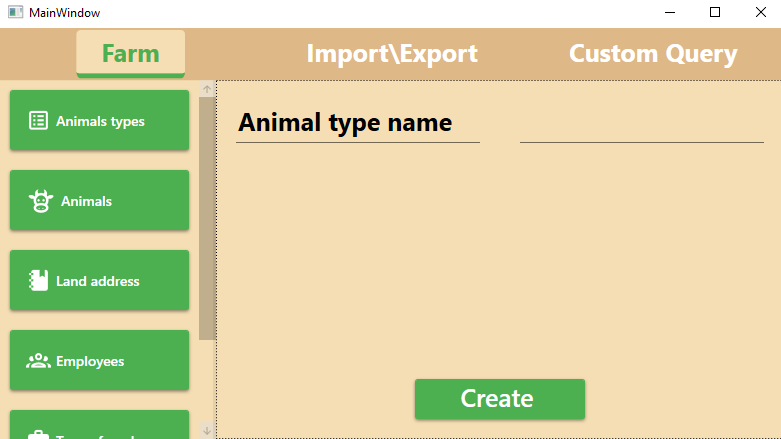


Рис 3.4. Вигляд сторінки створення запису для таблиці Animals\_Type

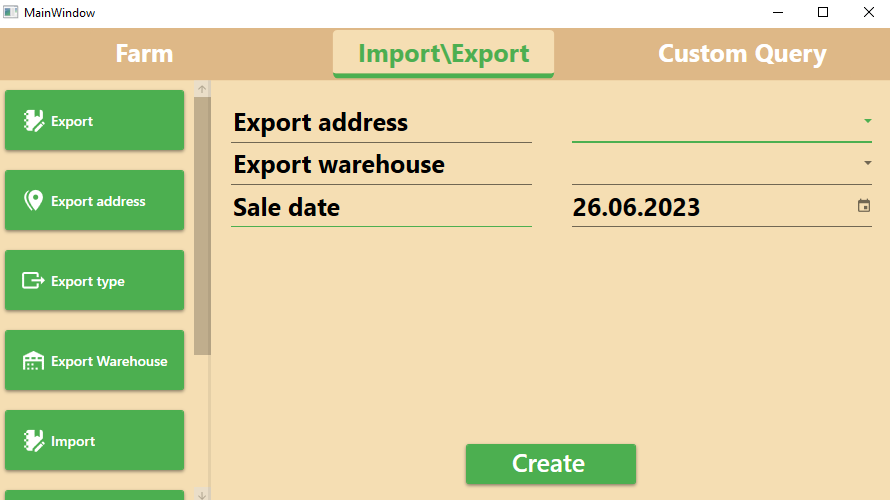


Рис 3.5. Вигляд сторінки створення запису для таблиці Export

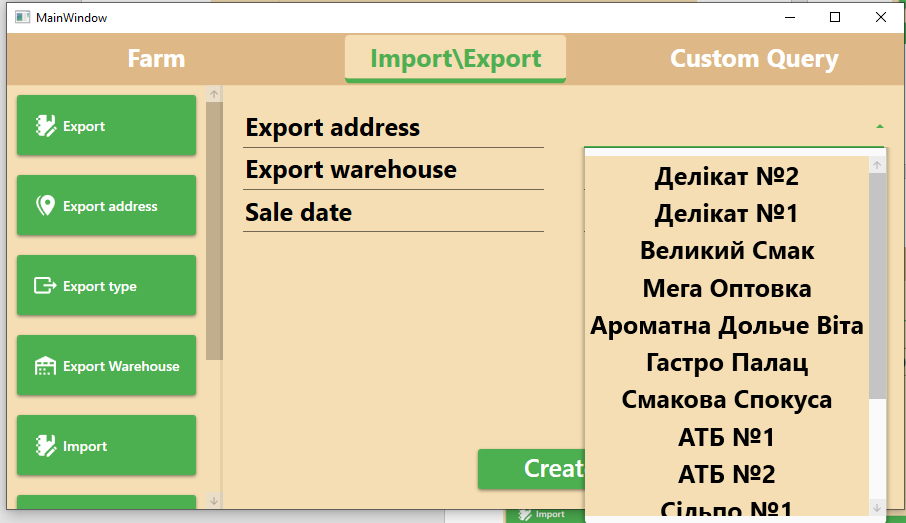


Рис 3.6. Вигляд спадаючого меню в таблиці Export, яке підтягує дані з таблиці Export\_Addresses

Ну і вигляд третьої вкладки, для користувацького запиту, яка має три частини, кнопка щоб виконати запит, поле куди вводити запит, і нижнє поле, де виведеться результат.



Рис. 3.7. Вигляд вкладки Custom Query



Рис. 3.8. Вигляд вкладки Custom Query з виконаним запитом

# **Висновок**

Роботу виконано, а отже потрібно підбити підсумки. Було поставлено мету, розширити, а в основному проявити навички, які були здобуті, під час проходження курсу «Організація баз даних і знань» на обрану тему курсової роботи «Інформаційна система фермерського господарства».

Наступним пунктом був пошук даних та аналогів, для аналізу цих аналогів та виявлення їх проблем. Ці проблеми можна, також, частково вважати актуальністю теми, оскільки їх потрібно було вирішити.

Після аналізу, виявлення всіх недоліків була постановка задач, а саме, що потрібно фермерському господарству та яке програмне забезпечення використовувати для розробки, також потрібно не забувати, що було описано загальний концепт майбутнього інтерфейсу та бази даних.

Далі був крок до поглибленої роботи з базою даних, були побудовані інфологічна та даталогічна модель. В першій було описано, більш детально які сутності будуть масивним, і які сутності потрібні додатково, для комфортної роботи і розробки, також визналися поля, які потрібні цим сутностям та які приблизно зв’язки будуть між сутностями. В даталогічній моделі, розглянули саме технічну частину, який тип даних для полів потрібен, і який саме зв'язок буде між сутностями в базі даних, і врешті решт було створено та заповнено базу даних. І фінальним кроком роботи з базою даних, це було виконання запитів, для проявлення знань, і виконання мети курсової роботи.

Ну і фінальною частиною всіє курсової роботи, є розробка інтерфейсу і програми, щоб взаємодіяти з базою даних, виконати тему курсової роботи та отримати готовий програмний продукт. Так, програма не ідеальна, дуже багато малих помилок, функціонал і зручність є куди розширяти, але це вже в майбутньому.

# **Список інформаційних джерел**

1. Авраменко А. С., Авраменко В. С., Розломій І. О. Організація баз даних і знань. Навчальний посібник. Черкаси: Черкаський національній університет імені Богдана Хмельницького, 2021. 414 с.

Перевірено: 26.06.2023.

2. Розломій І.О. Методичні вказівки до виконання курсової роботи з дисципліни «Організація баз даних та знань» для студентів спеціальностей 121 – «Інженерія програмного забезпечення», 122 – «Комп’ютерні науки», 123 – «Комп’ютерна інженерія», 124 – «Системний аналіз» денної та заочної форм навчання / Укл.: І.О. Розломій. Черкаси ЧНУ, 2022. 30 с.

Перевірено: 26.06.2023.

3. SQL Tutorial [Електронний документ]. Режим доступу: <https://www.w3schools.com/sql/>;

Перевірено: 26.06.2023.

4. SQL Tutorial [Електронний документ]. Режим доступу: <https://www.sqlitetutorial.net/what-is-sqlite/>;

Перевірено: 26.06.2023.

5. WPF Tutorial [Електронний документ]. Режим доступу:  <https://www.tutorialspoint.com/wpf/index.htm>;

Перевірено: 26.06.2023.

6. WPF Tutorial [Електронний документ]. Режим доступу: <https://wpf-tutorial.com/>;

Перевірено: 26.06.2023.